



# HFSS 3D Layout Scripting Guide



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

Release 2023 R2  
July 2023

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

## Copyright and Trademark Information

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

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

## Disclaimer Notice

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

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

## U.S. Government Rights

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

## Third-PartySoftware

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

# Table of Contents

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

---

Sub Procedures .....	1-10
Converting Between Data Types .....	1-10
Including Scripts .....	1-11
Aborting Scripts .....	1-11
Interacting with a Script .....	1-11
Recommended VBScript References .....	1-12
Sample HFSS Script .....	1-12
Sample Q3D Extractor Script .....	1-14
Introduction to IronPython .....	1-16
Scope .....	1-16
Python compatibility .....	1-16
Advantages of IronPython .....	1-16
IronPython Mini Cookbook .....	1-17
Comments .....	1-18
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 .....	1-25
Primitive Method Arguments .....	1-25
Named Array Arguments .....	1-25
Named Array Values with All Key Value Pairs .....	1-25
Named Arrays with Nested Named Arrays .....	1-26
Function Blocks .....	1-27
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 .....	1-29
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 .....	1-34
Creating an External Script .....	1-34
Example Script .....	1-35
IronPython Samples .....	1-36
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 .....	1-42
Constants .....	1-42
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 .....	1-45
Developing a UDM/UDP .....	1-46
Creation .....	1-46
Location .....	1-46
Organize .....	1-46
Edit/Reload .....	1-47
UDPExtension .....	1-47
Import .....	1-47
Main class: UDPExtension .....	1-47
IUDPExtension methods .....	1-47
Mandatory methods .....	1-47
GetLengthParameterUnits() .....	1-47
GetPrimitiveTypeInfo() .....	1-47
GetPrimitiveParametersDefinition2() .....	1-47
AreParameterValuesValid2(errorMsg, udpParams) .....	1-48
CreatePrimitive2(funcLib, udpParams) .....	1-48
Optional methods .....	1-48
GetPrimitiveParameters() .....	1-48
GetRegisteredFaceNames() .....	1-48
GetRegisteredEdgeNames() .....	1-48
GetRegisteredVertexNames() .....	1-48
MapParametersDefinitionVersions2(oldVersion, oldUDPPParams) .....	1-49
GetOldPrimitiveParametersDefinition2(version ) .....	1-49

---

Example UDP .....	1-49
UDMExtension .....	1-49
Import .....	1-49
Main class: UDMExtension .....	1-50
IUDMExtension methods .....	1-50
Mandatory methods .....	1-50
GetInfo() .....	1-50
IsAttachedToExternalEditor() .....	1-50
CreateInstance(funcLib) .....	1-50
GetUnits(instanceld) .....	1-50
ReleaseInstance(instanceld) .....	1-51
GetAttribNameForEntityId() .....	1-51
GetAttribNameForPartId() .....	1-51
Optional methods .....	1-51
GetInstanceSourceInfo(instanceld) .....	1-52
ShouldAttachDefinitionFilesToProject() .....	1-52
Example UDM .....	1-52
UDMFunctionLibrary .....	1-53
Functions list: .....	1-54
UDM/UDP Functions .....	1-55
Return Values for Each UDM and UDP Function .....	1-55
UDP/UDM Structures and Constants .....	1-58
UDP/UDM Structures .....	1-58
List of structures .....	1-59
UDP/UDM Constants .....	1-65
Enum constants: .....	1-65
UDP Python Example .....	1-67
Introduction to CPython (Beta) .....	1-72
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 .....	1-80
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 .....	1-84
Executing a Script from Within a Script .....	1-85
Electronics Desktop Scripting Conventions .....	1-85
Named Arguments .....	1-86
VBscript Example .....	1-86
IronPython Example .....	1-87
Setting Numerical Values .....	1-88
Layout Scripts and the Active Layer .....	1-89
Scripts and Locked Layers .....	1-89
Event Callback Scripting .....	1-89
PyAEDT (Beta) .....	1-90
<b>2 - Application Object Script Commands</b> .....	<b>2-1</b>
GetAppDesktop .....	2-2
<b>3 - Desktop Object Script Commands</b> .....	<b>3-1</b>
AddMessage .....	3-6

---

AreThereSimulationsRunning .....	3-7
ClearMessages .....	3-7
CloseAllWindows .....	3-8
CloseProject .....	3-9
CloseProjectNoForce .....	3-10
Count .....	3-11
DeleteProject .....	3-12
DownloadJobResults .....	3-13
DeleteRegistryEntry .....	3-14
DoesRegistryValueExist .....	3-15
EnableAutoSave .....	3-16
ExportOptionsFiles .....	3-17
GetActiveProject .....	3-17
GetActiveScheduler .....	3-18
GetActiveSchedulerInfo .....	3-19
GetAutoSaveEnabled .....	3-20
GetBuildDateTimeString .....	3-21
GetCustomMenuSet .....	3-21
GetDefaultUnit .....	3-22
GetDesktopConfiguration .....	3-24
GetDistributedAnalysisMachines .....	3-25
GetDistributedAnalysisMachinesForDesignType .....	3-26
GetExeDir .....	3-27
GetGDIObjectCount .....	3-27
GetLibraryDirectory .....	3-28
GetLocalizationHelper .....	3-29
GetMessages .....	3-30
GetMonitorData .....	3-31
GetPersonalLibDirectory .....	3-32
GetPPELicensingEnabled .....	3-33

---

GetProcessID .....	3-33
GetProjectDirectory .....	3-34
GetProjectList .....	3-35
GetProjects .....	3-35
GetRegistryInt .....	3-36
GetRegistryString .....	3-37
GetRunningInstancesMgr .....	3-38
GetSchematicEnvironment .....	3-39
GetScriptingToolsHelper .....	3-40
GetSysLibDirectory .....	3-40
GetTempDirectory .....	3-41
GetUserLibDirectory .....	3-42
GetVersion .....	3-42
IsFeatureEnabled .....	3-43
KeepDesktopResponsive .....	3-45
LaunchJobMonitor .....	3-45
NewProject .....	3-46
OpenAndConvertProject .....	3-47
OpenMultipleProjects .....	3-48
OpenProject .....	3-48
OpenProjectWithConversion .....	3-49
PageSetup (Layout Editor) .....	3-50
PauseRecording .....	3-50
PauseScript .....	3-51
Print .....	3-52
QuitApplication .....	3-53
RefreshJobMonitor .....	3-53
ResetLogging .....	3-54
RestoreProjectArchive .....	3-55
RestoreWindow .....	3-56

---

ResumeRecording .....	3-57
RunACTWizardScript .....	3-57
RunProgram .....	3-58
RunScript .....	3-59
RunScriptWithArguments .....	3-60
SelectScheduler .....	3-62
SetActiveProject .....	3-63
SetActiveProjectByPath .....	3-64
SetCustomMenuSet .....	3-65
SetDesktopConfiguration .....	3-66
SetLibraryDirectory .....	3-67
SetProjectDirectory .....	3-67
SetRegistryFromFile .....	3-68
SetRegistryInt .....	3-69
SetRegistryString .....	3-70
SetSchematicEnvironment .....	3-71
SetTempDirectory .....	3-72
ShowDockingWindow .....	3-72
Sleep .....	3-73
StopSimulations .....	3-74
SubmitJob .....	3-75
TileWindows .....	3-76
Desktop Commands For Registry Values .....	3-77
DeleteRegistryEntry .....	3-78
DoesRegistryValueExist .....	3-78
GetRegistryInt .....	3-79
GetRegistryString .....	3-80
SetRegistryFromFile .....	3-81
SetRegistryInt .....	3-82
SetRegistryString .....	3-83

---

ImportExport Tool Commands .....	3-84
ImportANF .....	3-84
ImportANFv2 .....	3-86
ImportAutoCAD .....	3-88
ImportAWRMicrowaveOffice .....	3-88
ImportEDB .....	3-90
ImportExtracta .....	3-91
ImportGDSII .....	3-92
ImportGerber .....	3-93
ImportIDF .....	3-94
ImportIDFandMerge .....	3-97
ImportIDX .....	3-98
ImportIPC .....	3-101
ImportODB .....	3-102
ImportXFL .....	3-103
<b>4 - Running Instances Manager Script Commands .....</b>	<b>4-1</b>
GetAllRunningInstances .....	4-1
GetRunningInstanceByProcessID .....	4-2
GetRunningInstancesMgr .....	4-2
<b>5 - Project Object Script Commands .....</b>	<b>5-1</b>
AddDataset .....	5-4
AddMaterial .....	5-7
AnalyzeAll [project] .....	5-11
ChangeProperty (3D Modeler Layout Component Visualisation) .....	5-12
ChangeProperty (Symbol Editor) .....	5-16
ClearMessages .....	5-29
CloneMaterial .....	5-30
Close .....	5-31
CopyDesign .....	5-32
CutDesign .....	5-32

---

DeleteDataset .....	5-33
DeleteDesign .....	5-34
DeleteToolObject .....	5-35
EditDataset .....	5-36
EditMaterial .....	5-38
ExportDataset .....	5-47
ExportMaterial .....	5-48
GetActiveDesign .....	5-49
GetArrayVariables .....	5-50
GetChildNames [Project] .....	5-51
GetChildObject [Project] .....	5-51
GetChildTypes [Project] .....	5-53
GetConfigurableData (Project) .....	5-53
GetDefinitionManager .....	5-54
GetDependentFiles .....	5-54
GetDesign .....	5-55
GetDesigns .....	5-56
GetEDBHandle .....	5-57
GetLegacyName .....	5-57
GetName [Project] .....	5-58
GetObjPath [Project] .....	5-59
GetPath .....	5-59
GetPropNames [Project] .....	5-60
GetPropValue [Project] .....	5-61
GetProperties .....	5-62
GetPropertyValue .....	5-63
GetTopDesignList .....	5-65
GetVariableValue .....	5-66
GetVariables .....	5-66
ImportDataset .....	5-67

---

Note About File Types:	5-68
InsertDesign	5-69
InsertDesign (Layout Editor)	5-70
InsertDesignWithWorkflow	5-71
InsertToolObject	5-72
Paste (Project Object)	5-73
Redo [Project Level]	5-73
RemoveAllUnusedDefinitions	5-74
RemoveMaterial	5-75
RemoveUnusedDefinitions	5-76
Rename	5-77
Save	5-78
SaveAs	5-79
SaveAsStandAloneProject	5-81
SaveProjectArchive	5-82
SetActiveDefinitionEditor	5-83
SetActiveDesign	5-84
SetPropValue [Project]	5-84
SetPropertyValue	5-85
SetVariableValue	5-87
SimulateAll	5-88
Undo [Project]	5-89
UpdateDefinitions	5-89
<b>6 - Object Oriented Property Scripting</b>	<b>6-1</b>
Object-Oriented Scripting	6-1
Material Properties and Examples	6-9
Body Properties and Modification	6-11
Retrieving Variables	6-11
Retrieve Datasets and Values	6-12
GetSolutionData API	6-14

---

Summary .....	6-14
Additional Details Specific to AEDT Solvers .....	6-18
Additional details on Boundaries/Excitations .....	6-20
3D component encapsulation .....	6-21
Additional details on Solve setup .....	6-22
Materials Scripting Support .....	6-23
Object Oriented Scripting for Materials .....	6-27
Examples showing change to material property type: .....	6-28
Examples showing change to a vector component value .....	6-28
Change choice property value .....	6-29
Change choice property value .....	6-30
<b>7 - Property Script Commands .....</b>	<b>7-1</b>
Object Script Property Function Summary .....	7-3
Object Path .....	7-3
Property Object .....	7-4
Project Object .....	7-5
Design Object .....	7-6
3D Modeler Object .....	7-7
Variable Object .....	7-7
Optimetrics Module Object: .....	7-9
Optimetrics Setup Object .....	7-9
ReportSetup(Results) Module Object: .....	7-10
ReportSetup(Results) Module Child Objects: .....	7-11
Radiation Module Object: .....	7-12
Radiation Module Child Objects: .....	7-12
Conventions Used in this Chapter .....	7-13
Callback Scripting Using PropHost Object .....	7-16
ChangeProperty (Schematic Editor and Layout Editor) .....	7-19
PropHost Functions .....	7-28
AddMenuProp .....	7-28

---

AddMenuProp2 .....	7-30
AddProp .....	7-31
AddProp2 .....	7-35
ExecuteScript .....	7-37
GetApplication .....	7-38
GetCallback .....	7-38
GetChangedProperty .....	7-39
GetDescription .....	7-40
GetDesign .....	7-41
GetEditor .....	7-41
GetEvaluatedText .....	7-42
GetFileName .....	7-42
GetHidden .....	7-43
GetProgress .....	7-44
GetPropHost .....	7-44
GetPropServers .....	7-45
GetPropTabType .....	7-46
GetReadOnly .....	7-46
GetRunStatus .....	7-47
GetTabTypeName .....	7-48
GetText .....	7-48
GetValue .....	7-49
IsValueConstant .....	7-50
PropertyExists .....	7-50
RemoveProp .....	7-51
SetCallback .....	7-52
SetDescription .....	7-53
SetHidden .....	7-53
SetReadOnly .....	7-54
SetText .....	7-55

---

SetValue .....	7-55
GetArrayVariables .....	7-66
GetProperties .....	7-67
GetPropertyValue .....	7-68
GetVariables .....	7-70
GetVariableValue .....	7-70
SetPropertyValue .....	7-71
SetVariableValue .....	7-73
Example Use of Record Script and Edit Properties .....	7-74
Additional Property Scripting Examples .....	7-75
<b>8 - Dataset Script Commands .....</b>	<b>8-1</b>
AddDataset .....	8-1
DeleteDataset .....	8-4
EditDataset .....	8-5
ExportDataset .....	8-7
HasDataset .....	8-8
ImportDataset .....	8-9
Note About File Types: .....	8-10
<b>9 - Design Object Script Commands .....</b>	<b>9-1</b>
AddDataset .....	9-7
AddDesignVariablesForDynamicLink .....	9-10
AddDynamicLink .....	9-11
AddModelingProperties .....	9-12
Analyze .....	9-13
AnalyzeAll [design] .....	9-14
AnalyzeAllNominal .....	9-14
AnalyzeDistributed .....	9-15
ApplyMeshOps .....	9-16
AssignDCThickness .....	9-17
ChangeProperty[ReportSetup] .....	9-19

---

ConstructVariationString .....	9-23
CopyItemCommand .....	9-24
CreateReport .....	9-24
DeleteDataset .....	9-39
DeleteFieldVariation .....	9-40
DeleteFullVariation .....	9-41
DeleteLinkedDataVariation .....	9-42
DeleteOutputVariable .....	9-42
DeleteSolutionVariation .....	9-43
DeleteOutputVariable .....	9-44
DistributeAnalysis .....	9-45
EditCoSimulationOptions .....	9-46
EditImportData .....	9-47
EditInfiniteArray .....	9-48
Edit (Layout Editor) .....	9-48
EditLayoutForLayoutComponent .....	9-50
EditNotes .....	9-50
EditNotes (Layout Editor) .....	9-51
EditOptions (Layout Editor) .....	9-51
EMDesignOptions .....	9-52
ExportConvergence .....	9-54
ExportDataset .....	9-55
ExportForHSpice .....	9-56
ExportForHSpice (Layout Editor) .....	9-57
ExportForSpice (Layout Editor) .....	9-59
ExportMatrixData .....	9-61
ExportMeshStats .....	9-64
ExportNetworkData .....	9-65
ExportNetworkData .....	9-67
ExportNMFDATA .....	9-70

---

ExportProfile .....	9-75
Generate Mesh .....	9-76
GetActiveEditor (Layout Editor) .....	9-77
GetChildNames [Design] .....	9-77
GetChildObject [Design] .....	9-78
GetChildTypes [Design] .....	9-79
GetConfigurableData .....	9-79
GetData .....	9-80
GetDesignID .....	9-81
GetDesignName .....	9-81
GetDesignType .....	9-82
GetDesignValidationInfo .....	9-83
GetEdgePositionAtNormalizedParameter .....	9-84
GetEditor .....	9-85
GetEditor (Layout Editor) .....	9-85
GetGeometryIdsForNetLayerCombinations .....	9-86
GetGeometryIdsForAllNetLayerCombinations .....	9-87
GetManagedFilesPath .....	9-88
GetModule .....	9-89
GetModule (Layout Editor) .....	9-90
GetModule (RadiationSetupMgr) .....	9-91
GetName .....	9-92
GetName (Layout Editor) .....	9-93
GetNominalVariation .....	9-93
GetNoteText .....	9-94
GetObjPath [Design] .....	9-94
GetOutputVariableValue .....	9-95
GetOutputVariables .....	9-97
GetPostProcessingVariables .....	9-97
GetPropHost .....	9-98

---

GetProperties .....	9-98
GetPropertyValue .....	9-100
GetPropNames [Design] .....	9-101
GetPropValue [Design] .....	9-102
GetSelections [Design] .....	9-103
GetSolutionType .....	9-103
GetSolveInsideThreshold .....	9-104
GetVariables .....	9-105
GetVariableValue .....	9-105
GetVariationVariableValue .....	9-106
ImportDataset .....	9-107
Note About File Types: .....	9-108
InsertDesign .....	9-109
OverlayCurrents (Layout Editor) .....	9-110
OverlayFarField (Layout Editor) .....	9-111
OverlayMesh (Layout Editor) .....	9-111
OverlayNearField (Layout Editor) .....	9-112
PasteDesign (Design Object) .....	9-112
PasteItemCommand (Layout Editor) .....	9-114
PushExcitations .....	9-114
Redo [Design] .....	9-117
Redo (Layout Editor) .....	9-118
RemoveImportData (Layout Editor) .....	9-118
RemoveModelingProperties (Layout Editor) .....	9-119
RenameDesignInstance .....	9-119
RenameDesignInstance (Layout Editor) .....	9-120
RenameImportData (Layout Editor) .....	9-120
RenameSource [Interface Source] .....	9-121
ReportTemplates (Layout Editor) .....	9-121
RunToolkit .....	9-122

---

SARSetup .....	9-123
SetActiveEditor .....	9-123
SetActiveEditor (Layout Editor) .....	9-124
SetAllowMaterialOverride .....	9-125
SetBackgroundMaterial .....	9-125
SetDesignMode .....	9-126
SetDesignSettings .....	9-127
SetDoMeshAssembly .....	9-131
SetFastTransformationForLayoutComponent .....	9-134
SetLengthSettings .....	9-134
SetObjectAttributesForLayoutComponent .....	9-135
SetPropValue [Design] .....	9-138
SetPropertyValue .....	9-139
SetShowLayoutForLayoutComponent .....	9-140
SetSolutionType .....	9-141
SetSolveInsideThreshold .....	9-143
SetSourceContexts .....	9-144
SetVariableValue .....	9-145
Solve .....	9-145
StartAnalysis (Layout Editor) .....	9-146
StopSimLink .....	9-146
Undo [Design] .....	9-147
Undo (Layout Editor) .....	9-148
ValidateCircuit (Layout Editor) .....	9-148
ValidateLink .....	9-149
ValidateLink .....	9-149
<b>10 - 3D Modeler Editor Script Commands .....</b>	<b>10-1</b>
Conventions Used in this Chapter: .....	10-1
<AttributesArray> .....	10-1
<SelectionsArray> .....	10-2

---

Draw Menu Commands .....	10-4
Create3DComponent .....	10-6
CreateBondwire .....	10-9
CreateBox .....	10-13
CreateCircle .....	10-16
CreateCone .....	10-19
CreateCutplane .....	10-21
CreateCylinder .....	10-23
CreateEllipse .....	10-26
CreateEquationCurve .....	10-29
CreateEquationSurface .....	10-32
CreateHelix .....	10-35
CreatePoint .....	10-37
CreatePolyline .....	10-39
CreateRectangle .....	10-44
CreateRegion .....	10-47
CreateRegularPolygon .....	10-51
CreateRegularPolyhedron .....	10-54
CreateSphere .....	10-57
CreateSpiral .....	10-59
CreateTorus .....	10-61
CreateUserDefinedModel .....	10-64
CreateUserDefinedPart .....	10-66
Edit3DComponent .....	10-74
Edit3DComponentDefinition .....	10-76
EditPolyline .....	10-79
Get3DComponentDefinitionNames .....	10-83
Get3DComponentInstanceNames .....	10-84
Get3DComponentMaterialNames .....	10-84
Get3DComponentMaterialProperties .....	10-85

---

Get3DComponentParameters .....	10-86
Get3DComponentPartNames .....	10-86
Insert3DComponent .....	10-87
InsertComponent .....	10-88
InsertPolylineSegment .....	10-90
SweepAlongPath .....	10-92
SweepAlongVector .....	10-94
SweepAroundAxis .....	10-96
SweepFacesAlongNormal .....	10-98
SweepFacesAlongNormalWithAttributes .....	10-100
UpdateComponentDefinition .....	10-103
<b>Edit Menu Commands .....</b>	<b>10-104</b>
Copy .....	10-105
DeletePolylinePoint .....	10-105
DuplicateAlongLine .....	10-107
DuplicateAroundAxis .....	10-109
DuplicateMirror .....	10-112
Mirror .....	10-115
Move .....	10-117
OffsetFaces .....	10-119
Paste (Model Editor) .....	10-120
Rotate .....	10-121
Scale .....	10-122
<b>Modeler Menu Commands .....</b>	<b>10-124</b>
AlignFaces .....	10-126
AssignMaterial .....	10-127
Chamfer .....	10-130
CleanUpModel .....	10-132
Connect .....	10-133
CoverLines .....	10-134

---

CoverSurfaces .....	10-135
CreateEntityList .....	10-136
CreateFaceCS .....	10-137
CreateGroup .....	10-142
CreateObjectCS .....	10-143
CreateObjectFromEdges .....	10-150
CreateObjectFromFace .....	10-152
CreateObjectFromFaces .....	10-154
CreateRelativeCS .....	10-155
DeleteEmptyGroups .....	10-157
DeleteLastOperation .....	10-158
DeleteOperation .....	10-159
DetachEdges .....	10-161
DetachFaces .....	10-162
EditEntityList .....	10-164
EditFaceCS .....	10-165
EditObjectCS .....	10-171
EditRelativeCS .....	10-178
Export .....	10-180
ExportModelImageToFile .....	10-182
ExportModelMeshToFile .....	10-185
Fillet .....	10-186
FlattenGroup .....	10-188
GenerateHistory .....	10-189
GetActiveCoordinateSystem .....	10-190
GetActiveCoordinateSystemTransform .....	10-190
GetCoordinateSystems .....	10-191
HealObject .....	10-192
Import .....	10-196
ImportDXF [Modeler] .....	10-200

---

ImportFromClipboard .....	10-204
ImportGDSII [Modeler] .....	10-205
Imprint .....	10-208
ImprintProjection .....	10-209
Intersect .....	10-211
MoveCStoEnd .....	10-213
MoveEntityToGroup .....	10-214
MoveFaces .....	10-215
ProjectSheet .....	10-217
PurgeHistory .....	10-218
ReplaceWith3DComponent .....	10-219
Section .....	10-222
SeparateBody .....	10-224
SetModelUnits .....	10-225
SetWCS .....	10-226
ShowWindow .....	10-227
Simplify .....	10-228
Split .....	10-231
Stitch .....	10-233
Subtract .....	10-235
SweepFacesAlongNormal .....	10-236
ThickenSheet .....	10-238
UncoverFaces .....	10-240
Unite .....	10-241
Ungroup .....	10-242
WrapSheet .....	10-243
Other oEditor Commands .....	10-244
AddDefinitionFromBlock .....	10-247
AddDefinitionFromLibFile .....	10-252
AssignSurfaceMaterial .....	10-256

---

BreakUDMConnection .....	10-259
ChangeProperty .....	10-260
CloseAllWindows[Editor] .....	10-263
Defeature .....	10-264
Delete .....	10-265
FitAll .....	10-266
GenerateAllUserDefinedModels .....	10-266
GenerateUserDefinedModel .....	10-267
GeometryCheckAndAutofix .....	10-268
GetBodyNamesByPosition .....	10-270
GetChildNames [Modeler] .....	10-271
GetChildObject [Modeler] .....	10-275
GetChildTypes [Modeler] .....	10-278
GetEdgeByPosition .....	10-280
GetEdgeIDFromNameForFirstOperation .....	10-282
GetEdgeIDsFromFace .....	10-282
GetEdgeIDsFromObject .....	10-283
GetEdgeLength .....	10-284
GetEntityListIDByName .....	10-284
GetExtendedDefinitionObject .....	10-285
GetFaceArea .....	10-286
GetFaceCenter .....	10-287
GetFaceByPosition .....	10-287
GetFaceIDFromNameForFirstOperation .....	10-289
GetFaceIDs .....	10-290
GetFaceIDsOfSheet .....	10-290
GetMatchedObjectName .....	10-291
GetModelBoundingBox .....	10-292
GetModelUnits .....	10-292
GetNumObjects .....	10-293

---

GetObjectIDByName .....	10-294
GetObjectName .....	10-294
GetObjectNameByEdgeID .....	10-295
GetObjectNameByFaceID .....	10-296
GetObjectNameByID .....	10-296
GetObjectNameByVertexID .....	10-297
GetObjectsByMaterial .....	10-298
GetObjectShapeType .....	10-298
GetObjectsInGroup .....	10-299
GetObjectVolume .....	10-300
GetObjPath [Editor] .....	10-301
GetPartsForUserDefinedModel .....	10-301
GetPoints [3D Modeler Editor] .....	10-302
GetPropEvaluatedValue .....	10-303
GetPropertyValue .....	10-304
GetPropNames [Modeler] .....	10-305
GetPropSIValue .....	10-307
GetPropValue [Modeler] .....	10-308
GetRelativeCoordinateSystems .....	10-309
GetSelections [Model Editor] .....	10-309
GetSubGroupsInGroup .....	10-310
GetUserPosition .....	10-311
GetVertexIDFromNameForFirstOperation .....	10-312
GetVertexIDsFromEdge .....	10-312
GetVertexIDsFromFace .....	10-313
GetVertexIDsFromObject .....	10-314
GetVertexPosition .....	10-314
GetWireBodyNames .....	10-315
OpenExternalEditor .....	10-316
PageSetup .....	10-316

---

RemoveBadEdges .....	10-318
RemoveBadFaces .....	10-319
RemoveBadVertices .....	10-320
RenamePart .....	10-320
SetPropValue [Modeler] .....	10-322
SetTopDownViewDirectionForActiveView .....	10-323
SetTopDownViewDirectionForAllViews .....	10-323
UpdatePriorityList .....	10-324
UpgradeVersion .....	10-325
Validate3DComponent .....	10-326
WriteHistoryTreeLayoutForTest .....	10-327
<b>Cable Modeling Commands .....</b>	<b>10-328</b>
AddCableToBundle .....	10-330
CreateCableBundle .....	10-331
CreateCableHarness .....	10-332
CreateClockSource .....	10-335
CreatePWLSource .....	10-337
CreateStraightWireCable .....	10-338
CreateTwistedPairCable .....	10-340
ExportCableLibrary .....	10-342
ImportCableLibrary .....	10-342
RemoveCable .....	10-343
UpdateCableHarness .....	10-344
<b>11 - Output Variable Script Commands .....</b>	<b>11-1</b>
CreateOutputVariable .....	11-1
DeleteOutputVariable .....	11-3
DoesOutputVariableExist .....	11-4
EditOutputVariable .....	11-5
ExportOutputVariables .....	11-6
GetOutputVariables .....	11-7

---

GetOutputVariableValue .....	11-8
ImportOutputVariables .....	11-9
SimValueContext .....	11-10
<b>12 - Reporter Editor Script Commands .....</b>	<b>12-1</b>
AddAllEyeMeasurements .....	12-6
AddCartesianLimitLine .....	12-6
AddCartesianLimitLineFromCurve .....	12-8
AddCartesianLimitLineFromEquation .....	12-10
AddCartesianXMarker .....	12-12
AddCartesianYMarker .....	12-12
AddCartesianYMarkerToStack .....	12-13
AddDeltaMarker .....	12-14
AddMarker .....	12-16
AddNote .....	12-17
AddTraceCharacteristics .....	12-19
AddTraces .....	12-20
ApplyReportTemplate .....	12-23
ChangeProperty[ReportSetup] .....	12-23
ClearAllMarkers .....	12-27
ClearAllTraceCharacteristics .....	12-28
CloneReportsFromDatasetSolution .....	12-29
CopyPlotSettings .....	12-29
CopyReportDefinitions .....	12-30
CopyReportsData .....	12-31
CopyTraceDefinitions .....	12-32
CopyTracesData .....	12-32
CreateReport .....	12-33
CreateReport [Designer] .....	12-48
CreateReportFromTemplate .....	12-54
CreateReportOfAllQuantities .....	12-55

---

DeleteMarker .....	12-56
DeleteAllReports .....	12-57
DeleteReports .....	12-57
DeleteTraceCharacteristics .....	12-58
DeleteTraces .....	12-59
DoesSupportTraceCharacteristics .....	12-60
DumpAllReportsData .....	12-61
EditCartesianXMarker .....	12-62
EditCartesianYMarker .....	12-62
EditMarker .....	12-63
ExportEyeMaskViolation .....	12-64
ExportImageToFile [Reporter] .....	12-65
ExportModelImageToFile .....	12-66
ExportModelMeshToFile .....	12-69
ExportPlot3DToFile [Reporter] .....	12-70
ExportReport .....	12-71
ExportReportDataToFile .....	12-73
ExportTableToFile .....	12-73
ExportToFile [Reporter] .....	12-74
ExportUniformPointsToFile .....	12-75
GetAllCategories .....	12-77
GetAllQuantities .....	12-78
GetAllReportNames .....	12-79
GetAvailableDisplayTypes .....	12-80
GetAvailableReportTypes .....	12-81
GetAvailableSolutions .....	12-81
GetChildNames [Report Setup] .....	12-82
GetChildObject [Report Setup] .....	12-83
GetChildTypes [ReportSetup] .....	12-84
GetCurvePropServerName .....	12-84

---

GetDisplayType .....	12-85
GetDynLinkIntrinsicVariables .....	12-86
GetDynLinkQtyValueState .....	12-86
GetDynLinkTraces .....	12-87
GetDynLinkVariableValues .....	12-88
GetName .....	12-89
GetObjPath [Design] .....	12-89
GetPropertyValue .....	12-90
GetPropNames [Reporter] .....	12-92
GetPropValue [Report Setup] .....	12-92
GetQtyExpressionsForSourceTrace .....	12-93
GetReportTraceNames .....	12-94
GetReportSummaryForRegressionTesting .....	12-94
GetSolutionContexts .....	12-95
GetSolutionDataPerVariation .....	12-96
GetDataExpressions .....	12-99
GetDataUnits .....	12-100
GetDesignVariableNames .....	12-100
GetDesignVariableUnits .....	12-101
GetDesignVariableValue .....	12-102
GetDesignVariationKey .....	12-103
GetImgDataValues .....	12-104
GetPerQuantityPrimarySweepValues .....	12-105
GetRealDataValues .....	12-106
GetSweepNames .....	12-107
GetSweepUnits .....	12-108
GetSweepValues .....	12-109
IsDataComplex .....	12-110
IsPerQuantityPrimarySweep .....	12-110
Release Data .....	12-111

---

GroupPlotCurvesByGroupingStrategy .....	12-112
ImportIntoReport .....	12-113
ImportReportDataIntoReport .....	12-114
MovePlotCurvesToGroup .....	12-114
MovePlotCurvesToNewGroup .....	12-115
OpenWindowForAllReports .....	12-116
OpenWindowForReports .....	12-117
PastePlotSettings .....	12-118
PasteReports .....	12-118
PasteReportsWithLegacyNames .....	12-119
PasteTraces .....	12-120
PasteTracesWithLegacyNames .....	12-120
RenameReport .....	12-121
RenameTrace .....	12-122
ResetPlotSettings .....	12-123
SavePlotSettingsAsDefault .....	12-123
SetLinkOutputTraces .....	12-124
SetPropValue [Report Setup] .....	12-125
UnGroupPlotCurvesInGroup .....	12-126
UpdateAllReports .....	12-127
UpdateReports .....	12-127
UpdateTraces .....	12-128
UpdateTracesContextAndSweeps .....	12-131
<b>13 - Boundary and Excitation Module Script Commands .....</b>	<b>13-1</b>
Script Commands for Creating and Modifying PMLs .....	13-2
CreatePML .....	13-2
ModifyPMLGroup .....	13-5
PMLGroupCreated .....	13-7
PMLGroupModified .....	13-8
RecalculatePMLMaterials .....	13-10

---

<b>14 - Optimetrics Module Script Commands .....</b>	<b>14-1</b>
CopySetup .....	14-7
DeleteSetups [Optimetrics] .....	14-8
DistributedAnalyzeSetup .....	14-9
EditSetup .....	14-9
EnableSetup .....	14-22
ExportDXConfigFile .....	14-22
ExportOptimetricsProfile .....	14-23
ExportOptimetricsResult .....	14-24
ExportParametricResults .....	14-25
ExportParametricSetupTable .....	14-26
ExportRespSurfaceMinMaxTable .....	14-27
ExportRespSurfaceRefinePoints .....	14-28
ExportRespSurfaceResponsePoints .....	14-29
ExportRespSurfaceVerificationPoints .....	14-30
GenerateVariationData [Parametric] .....	14-31
GetChildNames [Optimetrics] .....	14-31
GetChildObject [Optimetrics] .....	14-32
GetChildTypes [Optimetrics] .....	14-32
GetName .....	14-33
GetObjPath [Design] .....	14-34
GetOptimetricResult .....	14-34
GetOptimetricsResult .....	14-35
GetPropNames [Optimetrics] .....	14-36
GetPropValue [Optimetrics] .....	14-37
GetSetupNames [Optimetrics] .....	14-38
GetSetupNamesByType [Optimetrics] .....	14-39
ImportSetup .....	14-39
InsertSetup .....	14-41
PasteSetup [Optimetrics] .....	14-76

---

RenameSetup [Optimetrics] .....	14-77
SetPropValue [Optimetrics] .....	14-77
SolveAllSetup .....	14-78
SolveSetup [Optimetrics] .....	14-79
ValidateSetup [Parametric] .....	14-80
General Commands Recognized by the Optimetrics Module .....	14-80
CopySetup .....	14-82
DeleteSetups [Optimetrics] .....	14-83
DistributedAnalyzeSetup .....	14-83
EditSetup .....	14-84
EnableSetup .....	14-96
ExportDXConfigFile .....	14-97
ExportOptimetricsProfile .....	14-98
ExportOptimetricsResult .....	14-99
ExportParametricResults .....	14-100
ExportParametricSetupTable .....	14-101
ExportRespSurfaceMinMaxTable .....	14-102
ExportRespSurfaceRefinePoints .....	14-102
ExportRespSurfaceResponsePoints .....	14-103
ExportRespSurfaceVerificationPoints .....	14-104
GenerateVariationData [Parametric] .....	14-105
GetChildNames [Optimetrics] .....	14-106
GetChildObject [Optimetrics] .....	14-106
GetChildTypes [Optimetrics] .....	14-107
GetName .....	14-107
GetObjPath [Design] .....	14-108
GetOptimetricResult .....	14-109
GetPropNames [Optimetrics] .....	14-110
GetPropValue [Optimetrics] .....	14-110
GetSetupNames [Optimetrics] .....	14-111

---

GetSetupNamesByType [Optimetrics] .....	14-112
ImportSetup .....	14-113
InsertSetup .....	14-114
PasteSetup [Optimetrics] .....	14-149
RenameSetup [Optimetrics] .....	14-150
SetPropValue [Optimetrics] .....	14-151
SolveAllSetup .....	14-152
SolveSetup [Optimetrics] .....	14-152
ValidateSetup [Parametric] .....	14-153
Parametric Script Commands .....	14-154
EditSetup [Parametric] .....	14-154
ExportParametricSetupTable .....	14-155
GenerateVariationData [Parametric] .....	14-156
InsertSetup [Parametric] .....	14-156
ValidateSetup [Parametric] .....	14-161
Optimization Script Commands .....	14-162
EditSetup [Optimization] .....	14-162
InsertSetup [Optimization] .....	14-167
Sensitivity Script Commands .....	14-178
EditSetup [Sensitivity] .....	14-179
InsertSetup [Sensitivity] .....	14-186
Statistical Script Commands .....	14-190
EditSetup [Statistical] .....	14-190
InsertSetup [Statistical] .....	14-193
<b>15 - Solutions Module Script Commands .....</b>	<b>15-1</b>
DeleteImportData .....	15-1
DeleteSolutionVariation .....	15-2
EditSources .....	15-3
ExportEigenmodes .....	15-5
ExportForHSpice .....	15-6

---

ExportNetworkData .....	15-8
ExportNMFDATA [HFSS] .....	15-10
ExportTransientData .....	15-11
FFTOnReport .....	15-12
GetAdaptiveFreq .....	15-13
GetAdaptiveSettings .....	15-14
GetAllSourceMagnitudes .....	15-15
GetAllSourceModes .....	15-15
GetAllSourcePhases .....	15-16
GetAllSources .....	15-17
GetAntennaParameters .....	15-17
GetAvailableVariations .....	15-18
GetExcitationScaling .....	15-19
GetFieldType .....	15-19
GetIncludePortPostProcessing .....	15-20
GetMultipactionBreakdown .....	15-21
GetNetworkDataSolution .....	15-21
GetNetworkDataSolutionDefinition .....	15-22
GetNetworkPostprocSetup .....	15-23
GetISolutionVersionID .....	15-23
GetSolveRangeInfo .....	15-24
GetSourceContexts .....	15-25
GetSourceData (Layout Editor) .....	15-25
GetTerminalExcitationType .....	15-26
GetTransientSolveTimes .....	15-26
GetValidISolutionList .....	15-27
HasFields .....	15-28
HasMatrixData .....	15-28
HasMesh .....	15-29
ImportSolution .....	15-30

---

ImportTable .....	15-31
IsFieldAvailableAt .....	15-32
ListMatchingVariations .....	15-33
ListValuesOfVariable .....	15-34
ListVariations .....	15-35
SetExternalExcitations .....	15-36
SetSourceContexts .....	15-39
TDROnReport .....	15-40
<b>16 - Field Overlays Module Script Commands .....</b>	<b>16-1</b>
AddMarker[Fields Reporter] .....	16-4
AddMarkerToPlot .....	16-5
AddNamedExpr .....	16-6
AddNamedExpression .....	16-7
CalcOp .....	16-8
CalcStack .....	16-8
CalcWrite .....	16-9
CalculatorRead .....	16-10
CalculatorWrite .....	16-11
ChangeGeomSettings .....	16-12
ClcEval .....	16-13
ClcMaterial .....	16-14
ClcMaterialValue .....	16-14
ClearAllMarkers[Fields Reporter] .....	16-15
ClearAllNamedExpr .....	16-16
CopyNamedExprToStack .....	16-16
CreateFieldPlot .....	16-17
Maxwell Field Line Trace Plot Examples .....	16-39
DeleteFieldPlot .....	16-49
DeleteMarker[Fields Reporter] .....	16-49
DeleteNamedExpr .....	16-50

---

DeleteUneditablePlot .....	16-51
DoesNamedExpressionExists .....	16-51
EditSurfaceMeshSummaryData .....	16-52
EnterComplex .....	16-56
EnterComplexVector .....	16-57
EnterCoord .....	16-58
EnterEdge .....	16-59
EnterLine .....	16-59
EnterOutputVar .....	16-60
EnterPoint .....	16-61
EnterQty .....	16-61
EnterScalar .....	16-62
EnterScalarFunc .....	16-63
EnterSurf .....	16-64
EnterVector .....	16-64
EnterVectorFunc .....	16-65
EnterVol .....	16-66
ExportFieldPlot .....	16-66
ExportMarkerTable .....	16-67
ExportOnGrid [Field Overlays] .....	16-68
ExportPlotImageToFile .....	16-71
ExportPlotImageWithViewToFile [Reporter] .....	16-72
ExportSurfaceMeshSummary .....	16-73
ExportToFile .....	16-75
GetFieldFolderNames .....	16-76
GetFieldPlotNames .....	16-77
GetFieldPlotQuantityName .....	16-77
GetMeshPlotNames .....	16-78
GetTopEntryValue .....	16-79
HideAntennaParametersOverlay .....	16-80

---

HidePolarPlot .....	16-80
HideRadiatedPlotOverlay .....	16-81
LoadNamedExpressions .....	16-82
ModifyFieldPlot .....	16-83
ReassignFieldPlot .....	16-85
RenameFieldPlot .....	16-88
RenamePlotFolder .....	16-89
SaveFieldsPlots .....	16-90
SaveNamedExpressions .....	16-91
SetFieldPlotSettings .....	16-92
SetPlotFolderSettings .....	16-95
ShowAntennaParameterOverlay .....	16-100
UpdateAllFieldsPlots .....	16-101
UpdateQuantityFieldsPlots .....	16-101
<b>17 - Fields Calculator Script Commands .....</b>	<b>17-1</b>
AddNamedExpression .....	17-2
AddNamedExpr .....	17-3
CalcOp .....	17-4
CalcRead(deprecated) .....	17-4
CalculatorRead .....	17-5
CalcStack .....	17-6
CalculatorWrite .....	17-7
CalcWrite .....	17-8
ChangeGeomSettings .....	17-9
ClcEval .....	17-10
ClcMaterial .....	17-11
ClcMaterialValue .....	17-11
ClearAllNamedExpr .....	17-12
CopyNamedExprToStack .....	17-13
DeleteNamedExpr .....	17-13

---

DoesNamedExpressionExists .....	17-14
EnterComplex .....	17-15
EnterComplexVector .....	17-16
EnterCoord .....	17-17
EnterEdge .....	17-17
EnterLine .....	17-18
EnterOutputVar .....	17-19
EnterPoint .....	17-19
EnterQty .....	17-20
EnterScalar .....	17-21
EnterScalarFunc .....	17-22
EnterSurf .....	17-22
EnterVector .....	17-23
EnterVectorFunc .....	17-24
EnterVol .....	17-24
ExportOnGrid [Fields Calculator] .....	17-25
ExportToFile [Fields Calculator] .....	17-28
GetTopEntryValue .....	17-31
LoadNamedExpressions .....	17-32
SaveNamedExpressions .....	17-33
<b>18 - Radiation Module Script Commands .....</b>	<b>18-1</b>
General Commands Recognized by the Radiation Module .....	18-2
CopyRadFieldSetup .....	18-2
DeleteFarFieldSetup .....	18-3
DeleteNearFieldSetup .....	18-3
DeleteSetup .....	18-4
EditRadiatedPowerCalculationMethod .....	18-5
GetChildNames [Radiation] .....	18-5
GetChildObject [Radiation] .....	18-6
GetChildTypes [Radiation] .....	18-6

---

GetPropNames [Radiation] .....	18-7
GetPropValue [Radiation] .....	18-7
GetSetupNames .....	18-8
PasteRadFieldSetup .....	18-8
RenameSetup [Radiation] .....	18-9
SetPropValue [Radiation] .....	18-10
Script Commands for Creating and Modifying Radiation Setups .....	18-11
AddAntennaOverlay .....	18-11
AddRadFieldSourceGroup .....	18-13
EditAntennaOverlay .....	18-14
EditBoxSetup .....	18-16
EditFarFieldSphereSetup .....	18-18
EditInfiniteSphereSetup .....	18-21
EditLineSetup .....	18-23
EditNearFieldBoxSetup .....	18-24
EditNearFieldLineSetup .....	18-26
EditNearFieldRectangleSetup .....	18-28
EditNearFieldSphereSetup .....	18-29
EditRadFieldSourceGroup .....	18-31
EditRectangleSetup .....	18-33
EditSphereSetup .....	18-34
InsertBoxSetup .....	18-37
InsertFarFieldSphereSetup .....	18-39
InsertInfiniteSphereSetup .....	18-41
InsertLineSetup .....	18-43
InsertNearFieldBoxSetup .....	18-45
InsertNearFieldLineSetup .....	18-47
InsertNearFieldRectangleSetup .....	18-48
InsertNearFieldSphereSetup .....	18-50
InsertRectangleSetup .....	18-52

---

InsertSphereSetup .....	18-53
Script Commands for Modifying Antenna Array Setups .....	18-55
EditAntennaArraySetup .....	18-56
Script Commands for Exporting Antenna Parameters and Max Field Parameters .....	18-61
ExportElementPatternToFile .....	18-62
ExportFieldsToFile .....	18-63
ExportParametersToFile .....	18-65
ExportRadiationFieldsToFile .....	18-67
ExportRadiationParametersToFile .....	18-69
<b>19 - Radiation Setup Manager Module Script Commands .....</b>	<b>19-1</b>
Add (Infinite Sphere) .....	19-1
Edit (RadiationSetupMgr for Infinite Sphere) .....	19-5
Add (Near Field Line) .....	19-8
Edit (RadiationSetupMgr for Near Field Line) .....	19-10
Add (Near Field Plane) .....	19-12
Edit (RadiationSetupMgr for Near Field Plane) .....	19-15
Add (Near Field Sphere) .....	19-18
Edit (RadiationSetupMgr for Near Field Sphere) .....	19-22
<b>20 - User Defined Document Script Commands .....</b>	<b>20-1</b>
AddDocument .....	20-1
DeleteAllDocuments .....	20-4
DeleteDocument .....	20-5
EditDocument .....	20-5
GetDocumentDefinitionNames .....	20-7
GetDocumentNames .....	20-8
RenameDocument .....	20-8
SaveHtmlDocumentAs .....	20-9
SavePdfDocumentAs .....	20-10
UpdateAllDocuments .....	20-10
UpdateDocument .....	20-11

---

ViewHtmlDocument .....	20-12
ViewPdfDocument .....	20-13
Explication of a Sample UDD Script .....	20-13
Example Python Script: Defining a Document .....	20-15
<b>21 - User Defined Solutions Commands .....</b>	<b>21-1</b>
CreateUserDefinedSolution .....	21-1
DeleteUserDefinedSolutions .....	21-3
EditUserDefinedSolution .....	21-4
GetUserDefinedSolutionNames .....	21-6
GetUserDefinedSolutionProperties .....	21-7
<b>22 - Network Data Explorer Script Commands .....</b>	<b>22-1</b>
AddDiffPair .....	22-4
Cascade (SPISim) .....	22-5
ClearDiffPairs .....	22-6
Clone .....	22-7
Close .....	22-8
Combine (SPISim) .....	22-9
Deembed (SPISim) .....	22-10
DeembedBack (SPISim) .....	22-11
DeembedFront (SPISim) .....	22-13
DisableDiffPairs .....	22-14
EnableDiffPairs .....	22-15
ExportCitiFile .....	22-16
ExportFullWaveSpice .....	22-18
ExportMatlab .....	22-19
ExportNMFDATA .....	22-21
ExportNetworkData .....	22-26
ExportSpreadsheet .....	22-29
ExportTouchstone .....	22-31
ExportTouchstone2 .....	22-34

---

Extract (SPISim) .....	22-36
GetFrequencies .....	22-37
GetFrequencyCount .....	22-38
GetName .....	22-39
GetPortCount .....	22-40
GetPortNumber .....	22-41
GetPostProcSettings .....	22-42
GetSolutionVariation .....	22-43
GetVariation .....	22-44
HasSameData .....	22-45
LoadSolution .....	22-47
Open .....	22-48
Rename (SPISim) .....	22-49
Renormalize (SPISim) .....	22-50
Reorder .....	22-51
Reorder (SPISim) .....	22-52
Reset .....	22-53
SetAllPortImpedances .....	22-54
SetPortDeembedDistance .....	22-56
SetPortImpedance .....	22-57
SetPostProcSettings .....	22-58
Smooth .....	22-59
Stretch (SPISim) .....	22-60
Terminate .....	22-61
<b>23 - ComplInstance Script Commands .....</b>	<b>(multiple)</b>
Callback Scripting Using ComplInstance Object .....	(multiple)
ComplInstance Functions .....	64
GetComponentName .....	64
GetInstanceID [Component Instance] .....	65
GetInstanceName [Component Instance] .....	66

---

GetParentDesign .....	66
GetPropHost .....	67
GetPropServerName .....	67
<b>24 - Simulation Setup Commands .....</b>	<b>24-1</b>
AddSetup (Layout Editor) .....	24-2
AddMeshOperation .....	24-2
AddSweep (Layout Editor) .....	24-7
Analyze (Layout Editor) .....	24-8
AnalyzeSweep (Layout Editor) .....	24-9
Delete (Layout Editor) .....	24-9
DeleteSweep (Layout Editor) .....	24-9
DynamicMeshOverlays (Layout Editor) .....	24-9
Edit (Layout Editor) .....	24-10
EditSweep (Layout Editor) .....	24-10
ExportToHFSS (Layout Editor) .....	24-10
ExportToQ3D (Layout Editor) .....	24-11
GetAllSolutionNames (Layout Editor) .....	24-11
GetMeshOperations .....	24-11
GetSetupData (Layout Editor) .....	24-12
LayoutMeshOverlay (Layout Editor) .....	24-13
Layout3DMeshOverlay (Layout Editor) .....	24-13
ListVariations (Layout Editor) .....	24-13
RefreshMeshOverlays (Layout Editor) .....	24-13
RenameSweep (Layout Editor) .....	24-13
<b>25 - Excitations Commands .....</b>	<b>25-1</b>
Add [Excitation HFSS 3D Layout] .....	25-2
AddRefPort (Layout Editor) .....	25-2
AddRefPortUsingEdges (Layout Editor) .....	25-3
ConvertPlaneToTrace (Layout Editor) .....	25-4
ConvertTraceToPlane (Layout Editor) .....	25-5

---

CoupleEdgePorts .....	25-5
CreateGroundPortComponent (Layout Editor) .....	25-5
CreateInterfaceGround (Layout Editor) .....	25-6
CreateInterfacePort (Layout Editor) .....	25-6
CreateInterfacePortComponent (Layout Editor) .....	25-7
DecoupleEdgePorts .....	25-7
Delete[Excitation HFSS 3D Layout] .....	25-7
Delete [Interface Port] .....	25-7
Delete[Excitation HFSS 3D Layout] .....	25-8
DeleteProbePortAndVia (Layout Editor) .....	25-8
DeleteSource [Interface Source] .....	25-8
Edit (Layout Editor) .....	25-8
EditCircuitExcitations (Layout Editor) .....	25-9
EditCircuitPort [Interface Port] .....	25-9
EditExcitations (Layout Editor) .....	25-10
GetAllBoundariesList (Layout Editor) .....	25-11
GetAllPortsList (Layout Editor) .....	25-12
Rename [Interface Port] .....	25-12
RenameSource [Interface Source] .....	25-12
Rename (Layout Editor) .....	25-13
RemoveRefPort (Layout Editor) .....	25-13
SelectInLayout (Layout Editor) .....	25-13
<b>26 - 2.5D Via Commands .....</b>	<b>26-1</b>
ConvertPrimitives (Layout Editor) .....	26-1
Delete [Via HFSS 3D Layout] .....	26-1
Edit [Via HFSS 3D Layout] .....	26-2
MultipleEdit [multiple via HFSS 3D Layout] .....	26-2
Rename [Via HFSS 3D Layout] .....	26-2
SelectInLayout [Via HFSS 3D Layout] .....	26-3
<b>27 - Cavity Commands .....</b>	<b>27-1</b>

---

Add [Cavity HFSS 3D Layout] .....	27-1
Delete [Cavity HFSS 3D Layout] .....	27-1
Edit [Cavity HFSS 3D Layout] .....	27-2
Rename [Cavity HFSS 3D Layout] .....	27-2
SelectInLayout [Cavity HFSS 3D Layout] .....	27-2
<b>28 - Layout Scripting .....</b>	<b>28-1</b>
Object Identifiers and Script Recording .....	28-2
Create Primitives .....	28-2
CreateCircle (Layout Editor) .....	28-3
CreateLine (Layout Editor) .....	28-4
CreateRectangle (Layout Editor) .....	28-7
CreatePolygon (Layout Editor) .....	28-8
CreateText (Layout Editor) .....	28-9
Create Voids in Primitives .....	28-12
CreateCircleVoid (Layout Editor) .....	28-13
CreateLineVoid (Layout Editor) .....	28-14
CreatePolygonVoid (Layout Editor) .....	28-15
CreateRectangleVoid (Layout Editor) .....	28-16
Other Creation Methods .....	28-17
CreateComponent (Layout Editor) .....	28-18
CreateHole (Layout Editor) .....	28-19
CreateMeasure (Layout Editor) .....	28-21
CreatePin (Layout Editor) .....	28-23
CreatePinGroup (Layout Editor) .....	28-24
CreatePinGroupPort (Layout Editor) .....	28-24
CreateTrace (Layout Editor) .....	28-24
CreateVia (Layout Editor) .....	28-30
DeletePinGroup (Layout Editor) .....	28-31
Object Movement and Modification Methods .....	28-31
Connect (Layout Editor) .....	28-32

---

Copy (Layout Editor) .....	28-32
Delete (Layout Editor) .....	28-33
Disconnect (Layout Editor) .....	28-33
Edit (Layout Editor) .....	28-34
Expand (Layout Editor) .....	28-35
ExpandWithPorts (Layout Editor) .....	28-36
FlipHorizontal (Layout Editor) .....	28-37
FlipVertical (Layout Editor) .....	28-38
Move (Layout Editor) .....	28-38
Paste (Layout Editor) .....	28-38
Rotate (Layout Editor) .....	28-39
Activation and Deactivation Methods .....	28-39
Activate (Layout Editor) .....	28-39
DeactivateOpen (Layout Editor) .....	28-40
DeactivateShort (Layout Editor) .....	28-40
Delete (Layout Editor) .....	28-41
Layout and Geometry Interrogation Methods .....	28-41
Layout Interrogation (Layout Editor) .....	28-41
Point (Layout Editor) .....	28-42
Polygon (Layout Editor) .....	28-42
FindObjects (Layout Editor) .....	28-43
FilterObjectList (Layout Editor) .....	28-44
GetCSOObjects (Layout Editor) .....	28-45
GetPolygon (Layout Editor) .....	28-45
GetPolygonDef (Layout Editor) .....	28-46
GetBBox (Layout Editor) .....	28-46
FindObjectsByPolygon (Layout Editor) .....	28-46
FindObjectsByPoint (Layout Editor) .....	28-47
CreateObjectFromPolygon (Layout Editor) .....	28-47
CreateLineFromPolygon (Layout Editor) .....	28-48

---

Point Object (Layout Editor) .....	28-48
Set (Layout Editor) .....	28-49
SetX (Layout Editor) .....	28-49
GetX (Layout Editor) .....	28-50
SetY (Layout Editor) .....	28-50
GetY (Layout Editor) .....	28-50
SetArc (Layout Editor) .....	28-51
IsArc (Layout Editor) .....	28-51
IsEqual (Layout Editor) .....	28-51
Mag (Layout Editor) .....	28-52
Distance (Layout Editor) .....	28-52
Cross (Layout Editor) .....	28-52
Move (Layout Editor) .....	28-52
Rotate (Layout Editor) .....	28-53
Normalize (Layout Editor) .....	28-53
DistanceFromLine (Layout Editor) .....	28-53
ClosestPointOnLine (Layout Editor) .....	28-53
Polygon Object (Layout Editor) .....	28-54
AddPoint (Layout Editor) .....	28-56
SetClosed (Layout Editor) .....	28-56
IsClosed (Layout Editor) .....	28-56
Move (Layout Editor) .....	28-57
Rotate (Layout Editor) .....	28-57
Scale (Layout Editor) .....	28-57
MirrorX (Layout Editor) .....	28-58
GetPoints (Layout Editor) .....	28-58
AddHole (Layout Editor) .....	28-58
HasHoles (Layout Editor) .....	28-59
GetHoles (Layout Editor) .....	28-59
HasArcs (Layout Editor) .....	28-59

---

HasSelfIntersections (Layout Editor) .....	28-60
BBoxLL (Layout Editor) .....	28-60
BBoxUR (Layout Editor) .....	28-60
IsParametric (Layout Editor) .....	28-60
IsConvex (Layout Editor) .....	28-61
IsPoint (Layout Editor) .....	28-61
IsSegment (Layout Editor) .....	28-61
IsArc (Layout Editor) .....	28-61
IsBox (Layout Editor) .....	28-62
IsCircle (Layout Editor) .....	28-62
GetBoundingCircleCenter (Layout Editor) .....	28-62
GetBoundingCircleRadius (Layout Editor) .....	28-62
Area (Layout Editor) .....	28-63
PointInPolygon (Layout Editor) .....	28-63
CircleIntersectsPolygon (Layout Editor) .....	28-63
GetIntersectionType (Layout Editor) .....	28-63
GetClosestPoint (Layout Editor) .....	28-64
GetClosestPoints (Layout Editor) .....	28-64
Unite (Layout Editor) .....	28-65
Intersect (Layout Editor) .....	28-65
Subtract (Layout Editor) .....	28-65
Xor (Layout Editor) .....	28-66
<b>Boolean Operations on Primitives</b> .....	<b>28-66</b>
Unite (Layout Editor) .....	28-67
Intersect (Layout Editor) .....	28-67
SplitRegion .....	28-67
Subtract (Layout Editor) .....	28-68
<b>Coordinate System Methods</b> .....	<b>28-69</b>
CreateCS (Layout Editor) .....	28-69
ClearRelative (Layout Editor) .....	28-73

---

Create3DStructure (Layout Editor) .....	28-74
Group (Layout Editor) .....	28-74
CreateGroupSelected (Layout Editor) .....	28-74
PositionRelative (Layout Editor) .....	28-75
GetCSObjects (Layout Editor) .....	28-78
SetCS (Layout Editor) .....	28-78
Ungroup (Layout Editor) .....	28-78
IC Editor Script Commands .....	28-79
IC Mode Script Commands .....	28-85
ChangeActiveLayout (IC Editor) .....	28-86
CreateViaGroupsOnLayer (IC Editor) .....	28-87
ReconstructArcs (IC Editor) .....	28-89
ReconstructArcsOnLayer (IC Editor) .....	28-91
RemoveSmallHoles (IC Editor) .....	28-91
RemoveSmallLines (IC Editor) .....	28-93
RemoveSmallMetallIslands (IC Editor) .....	28-93
SnapPrimitives (IC Editor) .....	28-94
SnapPrimitivesOnLayer (IC Editor) .....	28-96
SnapVias (IC Editor) .....	28-97
SnapViasOnLayer (IC Editor) .....	28-99
UnitePrimitivesOnLayer (IC Editor) .....	28-101
WrapGeometries (IC Editor) .....	28-101
WrapGeometry (IC Editor) .....	28-103
IC Mode Commands Common to General Mode .....	28-104
AddObject (Layout Editor and IC Editor) .....	28-108
AddPinGroupRefPort (Layout Editor) .....	28-110
AddPortsToNet (Layout Editor) .....	28-111
AddRefPort (Layout Editor) .....	28-112
AssignRefPort (Layout Editor) .....	28-114
ChangeLayer (Layout Editor) .....	28-117

---

ChangeLayers (Layout Editor) .....	28-121
ChangeOptions (Layout Editor) .....	28-128
ChangeProperty (Schematic Editor and Layout Editor) .....	28-134
ClearRefPort (Layout Editor) .....	28-143
ClipPlane (Layout Editor) .....	28-144
ConvertPrimitives (Layout Editor) .....	28-144
ConvertPrimitivesToVias (Layout Editor) .....	28-145
Copy (Layout Editor) .....	28-146
AddObject ("3D Line") (Layout Editor) .....	28-146
CreateCircle (Layout Editor) .....	28-150
CreateCircuitPort (Layout Editor) .....	28-151
CreateComponent (Layout Editor) .....	28-154
CreateDifferentialPair (Layout Editor) .....	28-155
CreateEdgePort (Layout Editor) .....	28-156
CreateLine (Layout Editor) .....	28-158
CreateLineFromPolygon (Layout Editor) .....	28-161
CreateNetClass .....	28-161
CreateObjectFromPolygon (Layout Editor) .....	28-161
CreatePinGroupPort (Layout Editor) .....	28-162
CreatePinGroups (Layout Editor) .....	28-162
CreatePolygon (Layout Editor) .....	28-163
CreatePortsOnComponents (Layout Editor) .....	28-165
CreatePortsOnComponentsByNet (Layout Editor) .....	28-165
CreateRectangle (Layout Editor) .....	28-167
CreateViaGroups (Layout Editor) .....	28-169
CutOutSubDesign (Layout Editor) .....	28-170
Delete (Layout Editor) .....	28-175
DeleteNets (Layout Editor) .....	28-175
DeletePinGroup (Layout Editor) .....	28-176
DelNetClass .....	28-177

---

DissolveComponents (Layout Editor) .....	28-177
Duplicate (Layout Editor) .....	28-178
DuplicateAcrossLyrs (Layout Editor) .....	28-179
Edit (Layout Editor) .....	28-179
EditComponent (Layout Editor) .....	28-181
EditPinGroup (Layout Editor) .....	28-182
EnableComponents (Layout Editor) .....	28-184
ExportStackupXML (Layout Editor) .....	28-185
FilterObjectList (Layout Editor) .....	28-186
FindObjects (Layout Editor) .....	28-187
FindObjectsByPoint (Layout Editor) .....	28-188
FindObjectsByPolygon (Layout Editor) .....	28-189
GetActiveUnits (Layout Editor) .....	28-189
GetAllLayerNames (Layout Editor) .....	28-190
GetBBox (Layout Editor) .....	28-190
GetCSObjects (Layout Editor) .....	28-191
GetEditorName (Layout Editor) .....	28-191
GetLayerInfo (Layout Editor) .....	28-191
GetMaterialList (Layout Editor) .....	28-194
GetNetClasses .....	28-194
GetNetClassNets .....	28-195
GetNets (Layout Editor) .....	28-195
GetPolygon (Layout Editor) .....	28-196
GetPolygonDef (Layout Editor) .....	28-196
GetPolygonVoids (Layout Editor) .....	28-196
GetPortInfo (Layout Editor) .....	28-197
GetProperties (Layout Editor) .....	28-200
GetPropertyValue (Layout Editor) .....	28-200
GetSelections (Layout Editor) .....	28-201
GetStackupLayerNames (Layout Editor) .....	28-201

---

HighlightNet (Layout Editor) .....	28-202
ImportStackupXML (Layout Editor) .....	28-203
Intersect (Layout Editor) .....	28-204
ModifyDifferentialPair (Layout Editor) .....	28-204
ModifyNetClass .....	28-205
Move (Layout Editor) .....	28-206
Paste (Layout Editor) .....	28-206
Point (Layout Editor) .....	28-206
Polygon (Layout Editor) .....	28-207
RemoveLayer (Layout Editor) .....	28-207
General Mode .....	28-207
IC Mode (IC Editor) .....	28-208
RemovePort (Symbol Editor) .....	28-209
RemovePortsFromAllNets (Layout Editor) .....	28-210
RemovePortsOnComponents (Layout Editor) .....	28-210
ResetDifferentialpairs (Layout Editor) .....	28-211
Select (Layout Editor) .....	28-214
SelectAll (Layout Editor) .....	28-214
SelectNetConnected (Layout Editor) .....	28-215
SelectPhysicallyConnected (Layout Editor) .....	28-216
SetActiveUnits (Layout Editor) .....	28-217
SetCS (Layout Editor) .....	28-218
SetHfssExtentsVisible (Layout Editor) .....	28-218
SetHfssPortsVisible (Layout Editor) .....	28-219
SetNetColor (Layout Editor) .....	28-219
SetNetVisible (Layout Editor) .....	28-221
SetPropertyValue (Layout Editor) .....	28-223
Subtract (Layout Editor) .....	28-224
ToggleNetHighlight (Layout Editor) .....	28-224
ToggleViaPin (Symbol Editor) .....	28-225

---

Ungroup (Layout Editor) .....	28-226
Unite (Layout Editor) .....	28-226
UnselectAll (Layout Editor) .....	28-227
Layers Methods .....	28-227
AddObject (Layout Editor and IC Editor) .....	28-228
AddLayer (Layout Editor) .....	28-230
AddStackupLayer (Layout Editor) .....	28-233
ChangeLayers (Layout Editor) .....	28-238
ClearLayerMappings (Layout Editor) .....	28-245
DuplicateAcrossLyrs (Layout Editor) .....	28-245
GetAllLayerNames (Layout Editor) .....	28-245
GetLayerInfo (Layout Editor) .....	28-246
GetStackupLayerNames (Layout Editor) .....	28-248
ImportStackupXML (Layout Editor) .....	28-248
RemoveLayer (Layout Editor) .....	28-249
SetLayerMapping (Layout Editor) .....	28-250
SyncRigidFlexSandbox (Layout Editor) .....	28-251
Net and NetClass Methods .....	28-252
CreateNetClass .....	28-253
DeleteNets (Layout Editor) .....	28-253
DelNetClass .....	28-254
GetNetClasses .....	28-255
GetNetClassNets .....	28-255
GetNetConnections (Layout Editor) .....	28-255
HighlightNet (Layout Editor) .....	28-256
ModifyNetClass .....	28-257
SetNetVisible (Layout Editor) .....	28-258
Port Methods .....	28-260
AddCircuitRefPort (Layout Editor) .....	28-261
AddPortsToAllNets (Layout Editor) .....	28-263

---

AddPortsToNet (Layout Editor) .....	28-263
AddRefPort (Layout Editor) .....	28-264
AlignPorts (Layout Editor) .....	28-266
AssignCircuitRefPort (Layout Editor) .....	28-266
AssignRefPort (Layout Editor) .....	28-269
ClearRefPort (Layout Editor) .....	28-271
CreateCircuitPort (Layout Editor) .....	28-272
CreateEdgePort (Layout Editor) .....	28-274
CreateNPort (Layout Editor) .....	28-277
CreatePortInstancePorts (Layout Editor) .....	28-279
CreatePortsOnComponents (Layout Editor) .....	28-279
GetPortInfo (Layout Editor) .....	28-280
RemovePortsFromAllNets (Layout Editor) .....	28-283
RemovePortsFromNet (Layout Editor) .....	28-283
RemovePortsOnComponents (Layout Editor) .....	28-284
Miscellaneous Methods .....	28-285
AlignObjects (Layout Editor) .....	28-287
ChangeOptions (Layout Editor) .....	28-288
ClipPlane (Layout Editor) .....	28-294
CopyToHFSS 3D Layout (Layout Editor) .....	28-294
CutOutSubDesign (Layout Editor) .....	28-295
Defeature Objects (Layout Editor) .....	28-300
Duplicate (Layout Editor) .....	28-301
Edit3DComponentDefinition .....	28-302
EraseMeasurements (Layout Editor) .....	28-303
ExportDXF (Layout Editor) .....	28-303
ExportGDSII (Layout Editor) .....	28-306
ExportGerber (Layout Editor) .....	28-308
ExportNCDrill (Layout Editor) .....	28-311
GetComponentInfo (Layout Editor) .....	28-313

---

GetCompInstanceFromRefDes (Layout Editor) .....	28-314
GetComponentPins (Layout Editor) .....	28-315
GetComponentPinInfo (Layout Editor) .....	28-316
GetEditorName (Layout Editor) .....	28-317
GetMaterialList (Layout Editor) .....	28-317
GetProperties (Layout Editor) .....	28-318
GetPropertyValue (Layout Editor) .....	28-318
GetSelections (Layout Editor) .....	28-318
Heal (Layout Editor) .....	28-319
PageSetup (Layout Editor) .....	28-322
PushExcitations .....	28-323
SelectAll (Layout Editor) .....	28-326
SetPropertyValue (Layout Editor) .....	28-327
StitchLines (Layout Editor) .....	28-327
UnselectAll (Layout Editor) .....	28-328
ZoomToFit (Layout Editor) .....	28-328
<b>29 - Definition Manager Script Commands .....</b>	<b>29-1</b>
Add [component manager] .....	29-3
AddDataset .....	29-22
AddDefinitionFromBlock .....	29-25
AddMaterial .....	29-30
Add [padstack manager] .....	29-34
Add [symbol manager] .....	29-40
AreMaterialPropertiesEqual .....	29-51
AreSurfaceMaterialPropertiesEqual .....	29-52
ChangeProperty (Symbol Editor) .....	29-53
CloneMaterial .....	29-66
DeleteDataset .....	29-67
DoesMaterialExist .....	29-68
Edit [component manager] .....	29-69

---

EditDataset .....	29-95
EditMaterial .....	29-98
Edit [padstack manager] .....	29-107
ExportDataset .....	29-114
Export [footprint manager] .....	29-115
ExportMaterial .....	29-116
Export [padstack manager] .....	29-117
ExportScript .....	29-118
Export [symbol manager] .....	29-119
GetProjectMaterialNames .....	29-120
GetPropertyValue .....	29-121
ImportDataset .....	29-122
Note About File Types: .....	29-124
Remove [component manager] .....	29-125
Remove [footprint manager] .....	29-127
RemoveMaterial .....	29-128
Remove [padstack manager] .....	29-129
RemoveScript .....	29-130
Remove [symbol manager] .....	29-131
RemoveUnusedDefinitions .....	29-132
SetPropertyValue .....	29-134
UpdateDefFromBlock .....	29-135
UpdateDefFromBlockEx .....	29-137
UpdateDefinitions .....	29-139
Component Manager Script Commands .....	29-140
Add [component manager] .....	29-141
AddDynamicNPortData [component manager] .....	29-160
AddNPortData [component manager] .....	29-163
AddSolverOnDemandModel .....	29-169
ClearSolutionCache [component manager] .....	29-169

---

Edit [component manager] .....	29-169
EditSolverOnDemandModel .....	29-196
EditWithComps [component manager] .....	29-196
Export [component manager] .....	29-208
GetData [component manager] .....	29-209
GetNames [component manager] .....	29-210
GetNPortData [component manager] .....	29-211
GetSolverOnDemandData .....	29-214
GetSolverOnDemandModelList .....	29-214
IsUsed [component manager] .....	29-215
Remove [component manager] .....	29-215
RemoveSolverOnDemandModel .....	29-217
RemoveUnused [component manager] .....	29-217
Update Dynamic Link [component manager] .....	29-218
Footprint Manager Script Commands .....	29-219
Add [footprint manager] .....	29-219
Edit [footprint manager] .....	29-242
EditWithComps [footprint manager] .....	29-242
Export [footprint manager] .....	29-260
GetData [footprint manager] .....	29-261
GetNames [footprint manager] .....	29-262
IsUsed [footprint manager] .....	29-262
Remove [footprint manager] .....	29-263
RemoveUnused [footprint manager] .....	29-264
Material Manager Script Commands .....	29-265
GetData [material manager] .....	29-265
GetNames [material manager] .....	29-266
GetProperties [material manager] .....	29-268
IsUsed [material manager] .....	29-268
RemoveUnused [material manager] .....	29-269

---

Model Manager Script Commands .....	29-270
Add [model manager] .....	29-270
ConvertToDynamic .....	29-281
ConvertToParametric .....	29-281
Edit [deprecated] .....	29-282
EditWithComps [model manager] .....	29-282
Export [model manager] .....	29-293
GetData [model manager] .....	29-295
GetNames [model manager] .....	29-296
IsUsed [model manager] .....	29-296
Remove [model manager] .....	29-297
RemoveUnused [model manager] .....	29-298
Network Data Explorer Manager Script Commands .....	29-299
ExportFullWaveSpice .....	29-300
ExportNetworkData .....	29-301
ExportNMFDATA .....	29-304
Padstack Manager Script Commands .....	29-309
Add [padstack manager] .....	29-310
Edit [padstack manager] .....	29-316
EditWithComps [padstack manager] .....	29-323
Export [padstack manager] .....	29-331
GetData [padstack manager] .....	29-332
GetNames [padstack manager] .....	29-333
IsUsed [padstack manager] .....	29-333
Remove [padstack manager] .....	29-334
RemoveUnused [padstack manager] .....	29-335
Script and Library Scripts .....	29-336
AddScript .....	29-336
EditScript .....	29-337
ExportScript .....	29-338

---

ModifyLibraries .....	29-339
RemoveScript .....	29-340
Symbol Manager Script Commands .....	29-341
Add [symbol manager] .....	29-342
BringToFront [symbol manager] .....	29-353
Edit [deprecated] .....	29-353
EditWithComps [symbol manager] .....	29-353
Export [symbol manager] .....	29-365
GetData [symbol manager] .....	29-366
GetNames [symbol manager] .....	29-367
IsUsed [symbol manager] .....	29-367
Remove [symbol manager] .....	29-368
RemoveUnused [symbol manager] .....	29-369
<b>30 - Definition Editor Script Commands .....</b>	<b>30-1</b>
Activate (Layout Editor) .....	30-7
AddCircuitRefPort (Layout Editor) .....	30-8
AddLayer (Layout Editor) .....	30-10
AddPortsToAllNets (Layout Editor) .....	30-14
AddPortsToNet (Layout Editor) .....	30-14
AddRefPort (Layout Editor) .....	30-14
AlignObjects (Layout Editor) .....	30-17
AlignPorts (Layout Editor) .....	30-18
AssignCircuitRefPort (Layout Editor) .....	30-19
AssignRefPort (Layout Editor) .....	30-21
ChangeProperty .....	30-24
ClearLayerMappings (Layout Editor) .....	30-27
ClearRefPort (Layout Editor) .....	30-27
ClearRelative (Layout Editor) .....	30-28
Connect (Layout Editor) .....	30-28
ConvertPlaneToTrace (Layout Editor) .....	30-28

---

ConvertTraceToPlane (Layout Editor) .....	30-29
Copy (Layout Editor) .....	30-29
CopyToHFSS 3D Layout (Layout Editor) .....	30-29
Create3DStructure (Layout Editor) .....	30-30
CreateCS (Layout Editor) .....	30-30
CreateCircle (Layout Editor) .....	30-35
CreateCircleVoid (Layout Editor) .....	30-36
CreateCircuitPort (Layout Editor) .....	30-37
CreateComponent (Layout Editor) .....	30-40
CreateEdgePort (Layout Editor) .....	30-41
CreateGroundPortComponent (Layout Editor) .....	30-43
CreateGroupSelected (Layout Editor) .....	30-44
CreateHole (Layout Editor) .....	30-44
CreateInterfaceGround (Layout Editor) .....	30-46
CreateInterfacePort (Layout Editor) .....	30-46
CreateInterfacePortComponent (Layout Editor) .....	30-46
CreateLine (Layout Editor) .....	30-47
CreateLineFromPolygon (Layout Editor) .....	30-49
CreateLineVoid (Layout Editor) .....	30-50
CreateMeasure (Layout Editor) .....	30-51
CreateNportCircuitElements .....	30-53
CreateNetClass .....	30-55
CreateObjectFromPolygon (Layout Editor) .....	30-56
CreatePin (Layout Editor) .....	30-56
CreatePinGroup (Layout Editor) .....	30-57
CreatePinGroupPort (Layout Editor) .....	30-57
CreatePolygon (Layout Editor) .....	30-57
CreatePolygonVoid (Layout Editor) .....	30-59
CreatePortInstancePorts (Layout Editor) .....	30-60
CreatePortsOnComponents (Layout Editor) .....	30-60

---

CreateRectangle (Layout Editor) .....	30-61
CreateRectangleVoid (Layout Editor) .....	30-62
CreateText (Layout Editor) .....	30-64
CreateTrace (Layout Editor) .....	30-67
CreateVia (Layout Editor) .....	30-72
CutOutSubDesign (Layout Editor) .....	30-73
DeactivateOpen (Layout Editor) .....	30-78
DeactivateShort (Layout Editor) .....	30-79
Defeature Objects (Layout Editor) .....	30-79
DelNetClass .....	30-80
Delete (Layout Editor) .....	30-81
DeleteNets (Layout Editor) .....	30-81
DeletePinGroup (Layout Editor) .....	30-82
Disconnect (Layout Editor) .....	30-82
Duplicate (Layout Editor) .....	30-83
DuplicateAcrossLyrs (Layout Editor) .....	30-83
Edit (Layout Editor) .....	30-84
Edit3DComponentDefinition .....	30-85
EraseMeasurements (Layout Editor) .....	30-87
Expand (Layout Editor) .....	30-87
ExpandWithPorts (Layout Editor) .....	30-88
ExportDXF (Layout Editor) .....	30-89
ExportGDSII (Layout Editor) .....	30-92
ExportGerber (Layout Editor) .....	30-94
ExportNCDrill (Layout Editor) .....	30-97
ExportToHFSS (Layout Editor) .....	30-99
ExportToQ3D (Layout Editor) .....	30-100
FilterObjectList (Layout Editor) .....	30-100
FindObjectsByPoint (Layout Editor) .....	30-101
FindObjectsByPolygon (Layout Editor) .....	30-102

---

FlipHorizontal (Layout Editor) .....	30-102
FlipVertical (Layout Editor) .....	30-103
GeometryCheckAndAutofix .....	30-103
GetAllLayerNames (Layout Editor) .....	30-105
GetBBox (Layout Editor) .....	30-105
GetCSObjects (Layout Editor) .....	30-106
GetCompInstanceFromRefDes (Layout Editor) .....	30-106
GetComponentInfo (Layout Editor) .....	30-106
GetComponentPinInfo (Layout Editor) .....	30-107
GetComponentPins (Layout Editor) .....	30-108
GetEditorName (Layout Editor) .....	30-109
GetMaterialList (Layout Editor) .....	30-110
GetNetClassNets .....	30-110
GetNetClasses .....	30-110
GetNetConnections (Layout Editor) .....	30-111
GetPolygon (Layout Editor) .....	30-112
GetPolygonDef (Layout Editor) .....	30-112
GetPortInfo (Layout Editor) .....	30-112
GetProperties (Layout Editor) .....	30-115
GetPropertyValue (Layout Editor) .....	30-116
GetSelections [Design] .....	30-116
Group (Layout Editor) .....	30-116
Heal (Layout Editor) .....	30-117
HighlightNet (Layout Editor) .....	30-120
ImportDXF [Modeler] .....	30-121
ImportGDSII .....	30-125
Intersect (Layout Editor) .....	30-126
ModifyNetClass .....	30-127
Move (Layout Editor) .....	30-127
PageSetup (Layout Editor) .....	30-127

---

Point (Layout Editor) .....	30-128
Polygon (Layout Editor) .....	30-129
PositionRelative (Layout Editor) .....	30-129
PushExcitations .....	30-132
RemoveLayer (Layout Editor) .....	30-135
RemovePortsFromAllNets (Layout Editor) .....	30-136
RemovePortsFromNet (Layout Editor) .....	30-136
RemovePortsOnComponents (Layout Editor) .....	30-138
Rotate (Layout Editor) .....	30-139
SelectAll (Layout Editor) .....	30-139
SetCS (Layout Editor) .....	30-139
SetLayerMapping (Layout Editor) .....	30-140
SetNetVisible (Layout Editor) .....	30-140
StitchLines (Layout Editor) .....	30-142
Subtract (Layout Editor) .....	30-143
ToggleViaPin (Footprint Editor) .....	30-143
Ungroup (Layout Editor) .....	30-144
Unite (Layout Editor) .....	30-144
UnselectAll (Layout Editor) .....	30-144
ZoomToFit (Layout Editor) .....	30-145
Symbol Editor Scripts .....	30-145
AddLevel (Symbol Editor) .....	30-146
AlignHorizontal (Symbol Editor) .....	30-147
AlignVertical (Symbol Editor) .....	30-148
BringToFront (Symbol Editor) .....	30-148
ChangeProperty (Symbol Editor) .....	30-149
CloseEditor (Symbol Editor) .....	30-162
Copy (Symbol Editor) .....	30-162
CreateArc (Symbol Editor) .....	30-163
CreateCircle (Symbol Editor) .....	30-164

---

CreateImage (Symbol Editor) .....	30-165
CreateLine (Symbol Editor) .....	30-166
CreatePin (Symbol Editor) .....	30-167
CreatePolygon (Symbol Editor) .....	30-168
CreateRectangle (Symbol Editor) .....	30-168
CreateText (Symbol Editor) .....	30-169
Cut (Symbol Editor) .....	30-170
Delete (Symbol Editor) .....	30-171
DisplayPinNames (Symbol Editor) .....	30-171
ExportFile (Symbol Editor) .....	30-172
FlipHorizontal (Symbol Editor) .....	30-173
FlipVertical (Symbol Editor) .....	30-173
GetProperties (Symbol Editor) .....	30-174
GetPropertyValue (Symbol Editor) .....	30-174
GetVisibleLevels (Symbol Editor) .....	30-175
GridSetup (Symbol Editor) .....	30-175
Move (Symbol Editor) .....	30-176
Pan (Symbol Editor) .....	30-177
Paste (Symbol Editor) .....	30-178
Redo (Symbol Editor) .....	30-179
RemoveLevels (Symbol Editor) .....	30-179
RemovePort (Symbol Editor) .....	30-180
Rotate (Symbol Editor) .....	30-181
Save (Symbol Editor) .....	30-181
SelectAll (Symbol Editor) .....	30-182
SendToBack (Symbol Editor) .....	30-182
SetActiveLevel (Symbol Editor) .....	30-183
SetInitialLevels (Symbol Editor) .....	30-184
SetPropertyValue (Symbol Editor) .....	30-185
SetVisibleLevels (Symbol Editor) .....	30-185

---

ToggleLevel (Symbol Editor) .....	30-186
ToggleViaPin (Symbol Editor) .....	30-187
Undo (Symbol Editor) .....	30-187
ZoomArea (Symbol Editor) .....	30-188
ZoomIn (Symbol Editor) .....	30-189
ZoomOut (Symbol Editor) .....	30-189
ZoomPrevious (Symbol Editor) .....	30-190
ZoomToFit (Symbol Editor) .....	30-190
Footprint Editor Scripts .....	30-191
AddLayer (Footprint Editor) .....	30-193
AddStackupLayer (Footprint Editor) .....	30-196
ChangeLayer (Footprint Editor) .....	30-201
ChangeLayers (Footprint Editor) .....	30-205
ChangeOptions (Footprint Editor) .....	30-212
CloseEditor (Footprint Editor) .....	30-218
CreateCircle (Footprint Editor) .....	30-218
CreateCircleVoid (Footprint Editor) .....	30-219
CreateEdgePort (Footprint Editor) .....	30-220
CreateLine (Footprint Editor) .....	30-222
CreateLineVoid (Footprint Editor) .....	30-223
CreateMeasure (Footprint Editor) .....	30-224
CreatePolygonVoid (Footprint Editor) .....	30-226
CreatePin (Footprint Editor) .....	30-227
CreatePolygon (Footprint Editor) .....	30-228
CreateRectangle (Footprint Editor) .....	30-228
CreateText (Footprint Editor) .....	30-229
CreateVia (Footprint Editor) .....	30-230
Duplicate (Footprint Editor) .....	30-230
Edit (Footprint Editor) .....	30-231
EraseMeasurements (Footprint Editor) .....	30-232

---

FlipHorizontal (Footprint Editor) .....	30-232
FlipVertical (Footprint Editor) .....	30-232
GetAllLayerNames (Footprint Editor) .....	30-233
GetLayerInfo (Footprint Editor) .....	30-233
GetProperties (Footprint Editor) .....	30-236
GetStackupLayerNames (Footprint Editor) .....	30-236
Intersect (Footprint Editor) .....	30-236
Move (Footprint Editor) .....	30-237
PageSetup (Footprint Editor) .....	30-237
RemoveLayer (Footprint Editor) .....	30-238
RemovePort (Footprint Editor) .....	30-239
Rotate [Footprint Editor] .....	30-239
Save (Footprint Editor) .....	30-240
SetActiveDefinitionEditor (Footprint Editor) .....	30-240
SetPropertyValue (Footprint Editor) .....	30-241
Subtract (Footprint Editor) .....	30-241
ToggleViaPin (Footprint Editor) .....	30-242
Unite (Footprint Editor) .....	30-242
ZoomToFit (Footprint Editor) .....	30-243
<b>31 - Design Verification Script Commands .....</b>	<b>31-1</b>
AddRuleSet .....	31-1
AddRun .....	31-2
DeleteRuleSet .....	31-5
DeleteRun .....	31-5
EditRuleSet .....	31-5
EditRun .....	31-7
RenameRuleSet .....	31-9
RenameRun .....	31-10
RunAllDV .....	31-10
RunAllRuleSetDV .....	31-11

---

RunDV .....	31-11
<b>32 - Core Global Script Context Commands .....</b>	<b>32-1</b>
AddErrorMessage .....	32-1
AddFatalMessage .....	32-2
AddInfoMessage .....	32-2
AddWarningMessage .....	32-3
LogDebug .....	32-3
.LogError .....	32-4
<b>33 - Example Scripts .....</b>	<b>33-1</b>
VBScript Example Scripts .....	33-1
Data Export Script .....	33-1
Variable Helix Script .....	33-4
GetPropNames and GetPropValues for Layered Impedance Boundary Script .....	33-8
IronPython Example Scripts .....	33-11
BH Coordinates Python Script .....	33-11
Script Contents .....	33-11
Posco_BH_Curve.tab Contents .....	33-13
Equation Based Curve Python Script .....	33-16
HFSS Waveguide Array Python Script .....	33-18
<b>Index .....</b>	<b>Index-1</b>



# 1 - Introduction to Scripting

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

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

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

The following sections contain more information about scripting:

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

## Scripting Help Conventions

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

### [ScriptName]

[Description of script use.]

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

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

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

## Variable Types

The following data types are used throughout the help:

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

## Introduction to VBScript

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

This chapter provides an overview of key VBScript components.

[Simple and Composite Names](#)

[VBScript Variables](#)

[VBScript Operators](#)

[Controlling Program Execution](#)

[Looping Through Code](#)

[VBScript Procedures](#)

[Converting Between Data Types](#)

[Including Scripts](#)

[Aborting Scripts](#)

[Interacting with a Script](#)

[Recommended VBScript References](#)

[Sample HFSS Script](#)

[Sample Circuit Script](#)

[Sample Q3D Script](#)

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

## Simple and Composite Names

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

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

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

## VBScript Variables

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

## Declaring Variables

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

```
Dim box_xsize
```

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

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

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

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

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

### Declaring Variables in Python

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

You can directly assign information to it. For example:

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

### Variable Naming Conventions

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

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

### Scope and Lifetime of Variables

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

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

### Array Variables

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

```
Dim Primitives(2)
```

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

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

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

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

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

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

#### Note:

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

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

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

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

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

## VBScript Operators

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

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

## Operator Precedence

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

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

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

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

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

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

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

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

### Arithmetic Operators

Following is a list of VBScript's arithmetic operators:

Symbol	Description
^	Exponentiation
-	Unary negation
*	Multiplication
/	Division

---

\	Integer division
Mod	Modulus arithmetic
+	Addition
-	Subtraction

**Note:**

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

**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:

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

	Converts variablename to a double precision number. For example, it can be used to convert the string "2.5" to the number 2.5.
<b>CInt</b>	Syntax: CInt(variablename). Converts variablename to an integer.

## Including Scripts

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

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

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

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

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

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

## Aborting Scripts

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

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

## Interacting with a Script

VBScript provides two functions that enable you to interact with a script while it is running: the InputBox function and the MsgBox function.

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

```
Dim users_string
```

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

The last two arguments to the function are optional.

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

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

## Recommended VBScript References

Microsoft Corporation. *VBScript User's Guide*.

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

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

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

## 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:=", __  
0.40000005960464, "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/>).

### Advantages of IronPython

The advantages that IronPython use provides are significant:

- Python has a large eco-system with plenty of supporting libraries, Visual IDEs and debuggers. It is actively developed and enhanced.
- IronPython, in addition, has access to the entire .NET eco system. This allows us, for instance, to create a modern GUI using the **System.Windows.Forms** assembly from IronPython code and call any other .NET assembly for that matter.

- The use of IronPython's technologies enables the ability to interactively script Desktop (feature in development). This allows better discovery of the scripting APIs as well as directly programming to the scripting API in python, a language more tractable and platform independent compared with VBScript.
- The Python syntax of dictionaries is somewhat easier to read and write when supplying arguments to the scripting methods.

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

---

 IronPython Command Window

---

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

---

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

[Introduction to IronPython](#)

[IronPython Mini Cookbook](#)

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

[Scripting Using Iron Python](#)

[Standalone IronPython and Desktop IronPython](#)

[IronPython Examples](#)

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

## IronPython Mini Cookbook

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

consult Python documentation for additional information.

### Comments

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

### Assigning/Creating Variables

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

**Boolean Values**

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

**Converting Numbers to Strings and Vice Versa**

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

**String Formatting/Concatenation**

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

**Looping over Lists**

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

**Looping over a Range**

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

**Related Topics:**

[Indentation in IronPython](#)

[Methods in IronPython](#)

[Introduction to IronPython](#)

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

[Scripting Using Iron Python](#)

[IronPython Samples](#)

### Indentation in IronPython

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

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

**Note:**

Python recommends the use of spaces over tabs.

#### Indenting Functions

Define a function that starts at 0 indentation:

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

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

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

#### Indenting If Conditions

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

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

## Methods in IronPython

### Finding Methods

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

```
import string
```

Then get the directory listing:

```
dir(string)
```

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

### Help

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

## Related Topics

[Introduction to IronPython](#)

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

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

[IronPython Samples](#)

## Translating Script Commands from VBScript to IronPython

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

## Related Topics

[Script Method Argument](#)

[VBscript Method Call Types](#)

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

[Introduction to IronPython](#)

[IronPython Mini Cookbook](#)

[Scripting Using Iron Python](#)

[IronPython Samples](#)

## Script Method Argument

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

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

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

- Primitive Types
- Named Arrays
- Named Functions

### Primitive Types

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

### Named Arrays

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

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

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

### Named Functions

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

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

## Related Topics

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

### VBScript Method Call Types

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

### VBScript Functions

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

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

### Note:

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

## VBScript Sub-Routines

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

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

## Related Topics

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

### Converting VBScript Function calls to IronPython Syntax

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

So:

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

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

## Return Values

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

## Primitive Method Arguments

Replace each VBScript primitive with an equivalent IronPython primitive.

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

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

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

## Named Array Arguments

Formatting helps readability immensely but is not required.

All that *must* be done is:

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

## Named Array Values with All Key Value Pairs

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

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

**Named Arrays with Nested Named Arrays**

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

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

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

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

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

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

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

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

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

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

### Function Blocks

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

#### Important:

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

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

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

## Scripting Using Iron Python

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

### Translating a script in VBScript to IronPython

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

### Writing an IronPython script from scratch

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

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

[IronPython Script Execution Environment](#)

[Scripting using Embedded VBScript or JavaScript](#)

[Scripting with IronPython](#)

[Standalone IronPython and Desktop IronPython](#)

**Related Topics**

[Introduction to IronPython](#)

[IronPython Mini-cookbook](#)

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

[Appendix: IronPython Samples](#)

**IronPython Script Execution Environment**

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

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

---



IronPython Command Window

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

---

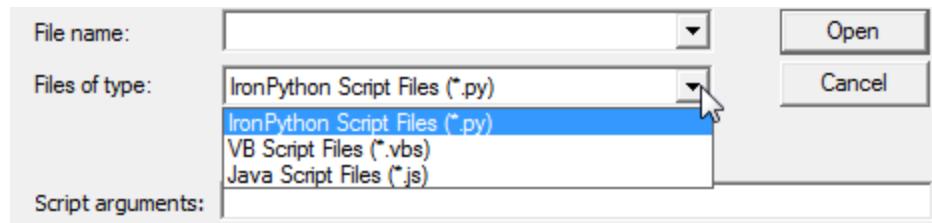
- **Tools > Run Script** menu item, select "IronPython" from the file type drop-down list.
- Launch the product with a script argument.
  - HFSS -runscript someScript.py to keep HFSS GUI open after completing script execution.
  - HFSS -features=beta -ng -runscriptandexit someScript.py to run HFSS in 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\AnsysEM\v232\Win64\common\IronPython\ipy64.exe"
"<filePath>\myScript.py"
```

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

### Using a Recorded Script

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

### Creating an External Script

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

- ```
import sys
```

  
# Imports the sys module containing system-specific functions native to IronPython.
- ```
sys.path.append("<InstallationPath>")
```

  
# Adds the Electronics Desktop installation path to the list of directories Python searches for modules and files.
- ```
sys.path.append("<InstallationPath>/PythonFiles/DesktopPlugin")
```

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

  
# This imports ScriptEnv.py from the installation path specified above. ScriptEnv.py performs an operating system check and defines functions used in Electronics Desktop scripts. See the annotations in the ScriptEnv.py file for more information.
- ```
ScriptEnv.Initialize("Ansoft.ElectronicsDesktop")
```

  
**or**

```
ScriptEnv.InitializeNew(NonGraphical=True)
```

  
# Initialize and InitializeNew are functions within ScriptEnv.py. The first option launches Electronics Desktop. The second allows you to run a script without launching Electronics Desktop. See the annotations in the ScriptEnv.py file for more information.

You must end the script with:

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

### Example Script

```
import sys
sys.path.append(r"C:\Program Files\AnsysEM\v232\Win64")
sys.path.append(r"C:\Program Files\An-
sysEM\v232\Win64\PythonFiles\DesktopPlugin")

import ScriptEnv
ScriptEnv.Initialize("Ansoft.ElectronicsDesktop")
oDesktop.RestoreWindow()
oProject = oDesktop.NewProject()
oProject.InsertDesign("HFSS", "HFSSDesign1", "DrivenModal", "")
oDesign = oProject.SetActiveDesign("HFSSDesign1")
oEditor = oDesign.SetActiveEditor("3D Modeler")
oEditor.CreateRectangle(
[
    "NAME:RectangleParameters",
    "IsCovered:= ", True,
    "XStart:= ", "-0.2mm",
    "YStart:= ", "-3mm",
    "ZStart:= ", "0mm",
    "Width:= ", "0.8mm",
    "Height:= ", "1.2mm",
    "WhichAxis:= ", "Z"
],
[
    "NAME:Attributes",
    "Name:= ", "Rectangle1",
]
```

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

## IronPython Samples

### Change property

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

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

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

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

```
])
```

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

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

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

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

#### Create a Cone using IronPython

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

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

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

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

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

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

```
oProject = oDesktop.GetActiveProject()

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

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

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

---

```
coneBotRad = "1.5mm"

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

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

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

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

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

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

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

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

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

#### **Related Topics**

[Introduction to IronPython](#)

[IronPython Mini-cookbook](#)

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

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

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

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

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

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

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

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

#### **Structures**

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

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

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

### Return Values for UDM and UDP Functions

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

### Constants

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

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

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

### Methods

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

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

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

### Output Parameters

The rule for the output parameters is as follows:

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

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

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

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

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

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

#### Comparison with C function:

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

#### 'List Size' Parameters

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

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

Example:

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

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

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

Comparison with C function:

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

---

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

**Added Parameters**

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

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

Example:

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

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

Here cloneIDsSize is a new integer parameter.

Comparison with C function:

| C                                                                                                                            | Python                                                                                                                                                                   |
|------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| long duplicateAlongLine(<br>long partId,<br>UDPVector transVector,<br>int nClones,<br>long* nClones,<br>void* callbackData); | (lret, cloneIDs) = udmFunctionLibrary.DuplicateAlongLine(partId,<br>transVec, nClones, cloneIDsSize)<br><br>(callbackData is omitted<br>cloneIDsSize is a new parameter) |
| long duplicateAroundAxis(<br>long partId,<br>UDPCoordinateSystemAxis<br>axis,                                                | (lret, cloneIDs) = udmFunctionLibrary.DuplicateAroundAxis<br>(partId, axis, angle, nClones, cloneIDsSize)                                                                |

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

## Developing a UDM/UDP

### Creation

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

### Location

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

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

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

### Organize

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

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

For UDM:extension:

```
from UDM import *
```

For UDP:extension:

```
from UDP import *
```

## Edit/Reload

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

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

## UDPExtension

### Import

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

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

```
from UDP import *
```

### Main class: UDPExtension

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

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

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

### IUDPExtension methods

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

#### Mandatory methods.

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

##### **GetLengthParameterUnits()**

- returns string.

##### **GetPrimitiveTypeInfo()**

- returns UDPPrimitiveTypeInfo.

##### **GetPrimitiveParametersDefinition2()**

- returns a list of UDPPrimitiveParameterDefinition2 or None on failure

**AreParameterValuesValid2(errorMsg, udpParams)**

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

**CreatePrimitive2(funcLib, udpParams)**

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

**Optional methods**

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

**GetPrimitiveParameters()**

- returns Python list of strings or NULL

**GetRegisteredFaceNames()**

- returns Python list of strings or NULL

**GetRegisteredEdgeNames()**

- returns Python list of strings or NULL

**GetRegisteredVertexNames()**

- returns Python list of strings or NULL

**ProjectParametersOnToValidPlane2(currentUDPPParams, projectedUDPPParams)**

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

**MapParametersDefinitionVersions2(oldVersion, oldUDPPParams)**

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

**GetOldPrimitiveParametersDefinition2(version )**

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

**Example UDP**

```
import sys

class UDPExtension(IUDPExtension):

    def GetLengthParameterUnits(self):
        return "mm"

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

        return typeInfo
    ...

    ...
```

**UDMExtension****Import**

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

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

---

```
from UDM import *
```

**Main class: UDMExtension**

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

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

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

**IUDMExtension methods**

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

**Mandatory methods.**

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

**GetInfo()**

- returns UDMInfo object populated with appropriate UDM information.

**IsAttachedToExternalEditor()**

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

**CreateInstance(funcLib)**

- funcLib is UDMFunctionLibrary
- returns UDMPARAMETERS.

**GetUnits(instanceId)**

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

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

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

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

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

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

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

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

returns True on success, False on failure.

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

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

#### **GetAttribNameForEntityId()**

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

#### **GetAttribNameForPartId()**

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

#### **Optional methods**

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

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

---

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

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

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

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

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

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

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

#### **ShouldAttachDefinitionFilesToProject()**

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

#### **Example UDM**

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

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

    return udmInfo
...
...
```

## UDMFunctionLibrary

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

- CreateInstance
- Refresh

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

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

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

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

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

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

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

**Functions list:**

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

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

**Example:**

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

## UDM/UDP Functions

Return Values for Each UDM and UDP Function

*ID* – *ID of created Object*

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

**Functions list:**

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

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

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

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

## UDP/UDM Structures and Constants

The following sections describe:

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

### UDP/UDM Structures

Differences compared to C API

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

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

Example (creating UDMParameter list):

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

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

- **UDPPParam**
- **UDPPParamData**

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

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

Example:

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

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

- **UDP3x3Matrix**

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

Example:

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

### List of structures

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

Example:

```

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

```

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

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

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

| Structure            | Construction                                                                                                                                                                       | Members                                                                                                                                                                                                                                                                                                                                                                      |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| UDPHelixDefinition   | UDPHelixDefinition(<br>int profileID,<br>UDPPosition ptOnAxis,<br>UDPPosition axisDir,<br>double noOfTurns,<br>bool isRightHanded,<br>double radiusChangePerTurn,<br>double pitch) | int ProfileID<br>UDPPosition PtOnAxis<br>UDPPosition AxisDir<br>double NoOfTurns<br>bool IsRightHanded<br>double RadiusChangePerTurn<br>double Pitch                                                                                                                                                                                                                         |
| UDPSpiralDefinition  | UDPSpiralDefinition(<br>int profileID,<br>UDPPosition ptOnAxis,<br>UDPPosition axisDir,<br>double noOfTurns,<br>bool isRightHanded,<br>double radiusChangePerTurn)                 | int ProfileID<br>UDPPosition PtOnAxis<br>UDPPosition AxisDir<br>double NoOfTurns<br>bool IsRightHanded<br>double RadiusChangePerTurn                                                                                                                                                                                                                                         |
| UDPBLNDElements      | UDPBLNDElements(<br>int partID,<br>int noOfEdges;<br>int* listOfEdges; )<br><br>UDPBLNDElements(<br>int partID,<br>int noOfVertices;<br>int* listOfVertices;<br>)                  | 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.<br><br>int PartID /* part to be blended i.e. filleted/chamfered */<br>int noOfEdges;<br>int* listOfEdges; /* edges to be blended */<br>int noOfVertices;<br>int* listOfVertices; /* vertices to be blended */ |
| UDPBLNDFilletOptions | UDPBLNDFilletOptions(                                                                                                                                                              | bool SupressFillet /* Reserved for future */                                                                                                                                                                                                                                                                                                                                 |

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

| Structure    | Construction                                                                                                                                      | Members                                                                                                                  |
|--------------|---------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------|
|              | UDParam value,<br>ParamPropType propType,<br>ParamPropFlag propFlag)                                                                              |                                                                                                                          |
| UDMOption    | UDMOption()<br><br>UDMOption(<br>string name,<br>UDParam value,<br>ParamPropType propType,<br>ParamPropFlag propFlag)                             | string OptName<br><br>UDPParam OptValue<br><br>ParamPropType PropType<br>ParamPropFlag PropFlag                          |
| UDMParameter | UDMParameter()<br><br>UDMParameter(<br>string name,<br>UDParam value,<br>UnitType unitType,<br>ParamPropType propType,<br>ParamPropFlag propFlag) | string ParamName<br><br>UDPParam ParamValue<br><br>UnitType UnitType<br>ParamPropType PropType<br>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<br>LengthUnit<br>AngleUnit |
| ParamDataType | Int                               |

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

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

### UDP Python Example

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

```
import sys

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

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

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

length_units = "mm"

#####
```

```
# Class Implementation

#####
class UDPExtension(IUDPExtension):

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

        param_dict = self.get_param_dict(param_values)

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

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

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

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

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

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

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

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

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

def GetPrimitiveTypeInfo(self):
    return primitive_info
```

```
def GetLengthParameterUnits(self):
    return length_units

def GetPrimitiveParametersDefinition2(self):
    return primitive_param_definitions

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

# Custom Functions

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

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

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

## Introduction to CPython (Beta)

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

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

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

### Important:

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

### Important:

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

## Creating an External Script

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

---

```

import sys

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

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

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

```

Those lines are followed by:

```

import ScriptEnv

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

```

Follow that with:

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

## Launching Electronics Desktop

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

```

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

```

Where:

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

### Note:

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

## Connecting with a Running Instance of Electronics Desktop

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

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

Where:

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

## Closing Electronics Desktop/Ending the Script

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

```
ScriptEnv.Shutdown()
```

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

### -grpcsrv Flag

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

On Windows:

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

On Linux:

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

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

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

### Related Topics:

[Standalone Scripting in Iron Python](#)

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](#)

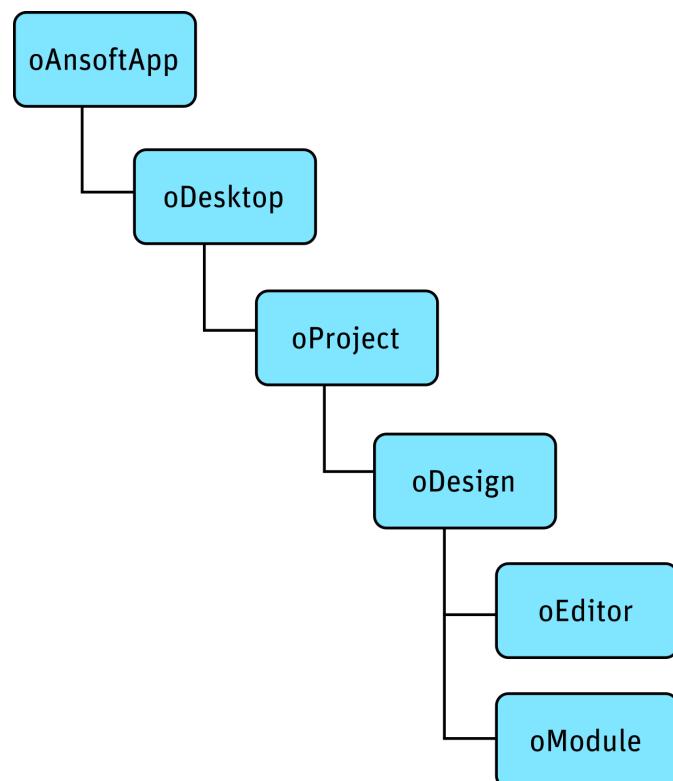
[Working with Project Scripts](#)

[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()
```

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

## **oProject**

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

In Iron Python, for example:

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

In VBscript, for example:

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

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

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

## **oDesign**

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

In Iron Python, for example:

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

In VBscript, for example:

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

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

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

## **oEditor**

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

In Iron Python, for example:

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

In VBscript, for example:

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

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

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

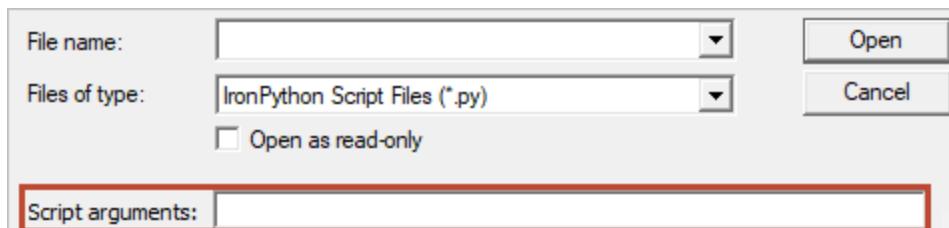
To run scripts in Electronics Desktop:

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:

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

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

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

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

For more information about running a script from the command line, consult the 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 col-
lection

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

```

## Recording a Script

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

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

### Important:

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

## Recording a Script to File

To record a script to file:

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

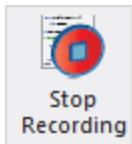


A **Save As** file browser appears.

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

5. Click **Save**.

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



6. Perform the steps you want to record.

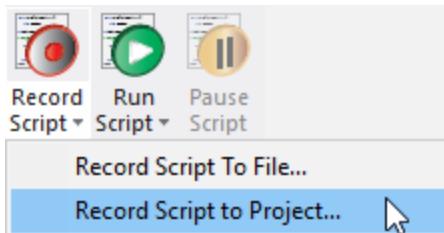
7. When you have finished recording the script, click **Stop Recording**, or select **Tools > Stop Script Recording**.

The recorded script is saved to the folder you specified.

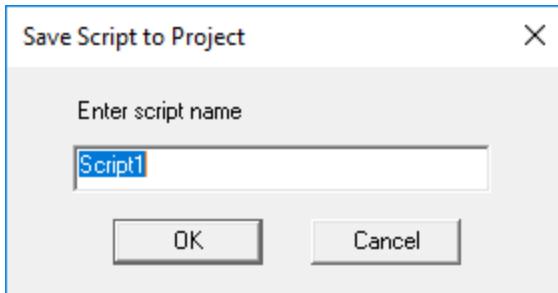
### Recording a Script to a Project

To record a script to a project:

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



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



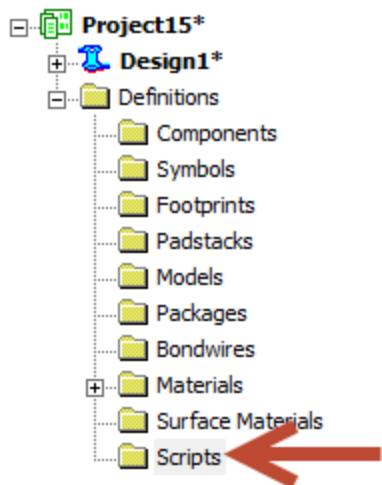
2. Enter a name for the script in the text box, then click **OK**.
3. Perform the steps you want to record.
4. When you have finished, click **Stop Recording**, or select **Tools > Stop Script Recording**.

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

## Working with Project Scripts

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

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



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



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

**Note:**

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

## Executing a Script from Within a Script

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

In IronPython:

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

In VBscript:

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

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

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

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

## Electronics Desktop Scripting Conventions

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

Consult the following topics:

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

- [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 ComplInstance Object](#).

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

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

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.

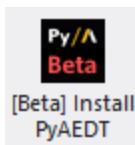
Documentation for PyAEDT can be found online at: <https://aedt-docs.pyansys.com/version/stable/>

Installing PyAEDT adds three items to the **Tools > Toolkit** 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.

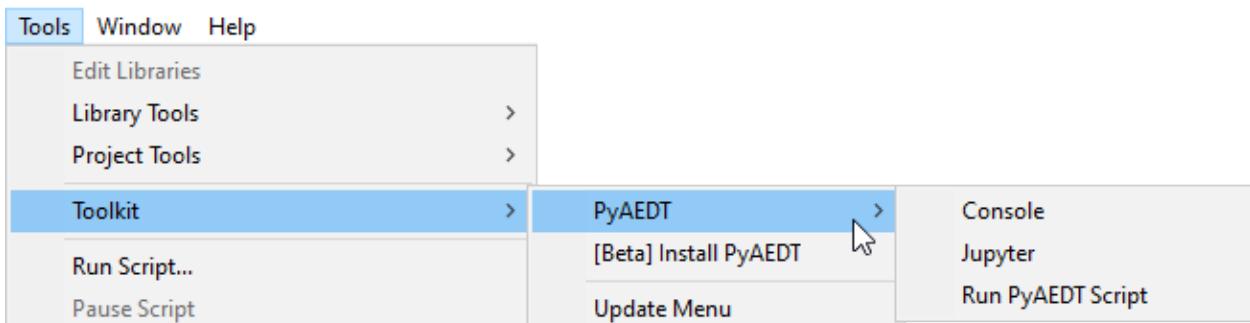
To install PyAEDT:

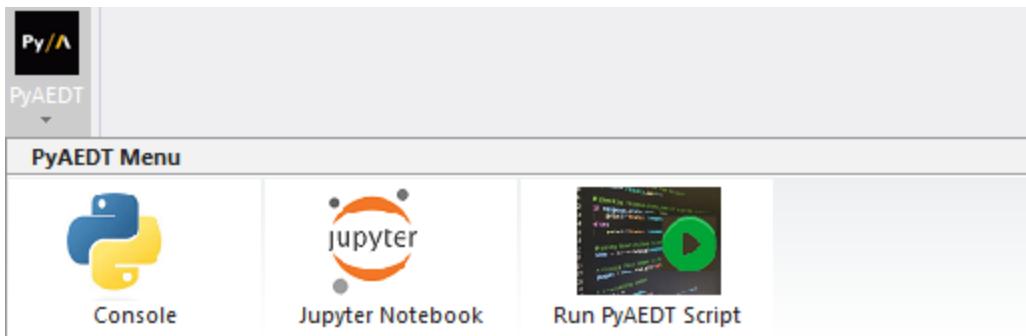
- From the **Automation** tab, click **[Beta] Install PyAEDT**.



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

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





This page intentionally  
left blank.

## 2 - Application Object Script Commands

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

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

### General Application Script Commands

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

- [GetAppDesktop](#)

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

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

- SetNumberOfProcessors (deprecated)
- SetUseHPCForMP (deprecated)

## GetAppDesktop

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

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

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

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

## 3 - Desktop Object Script Commands

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

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

[AddMessage](#)

[AreSimulationsRunning](#)

[ClearMessages](#)

[CloseAllWindows](#)

[CloseProject](#)

[CloseProjectNoForce](#)

[DeleteProject](#)

[DownloadJobResults](#)

[EnableAutoSave](#)

[ExportOptionsFiles](#)

[GetActiveProject](#)

[GetActiveScheduler](#)

[GetActiveSchedulerInfo](#)

[GetAutoSaveEnabled](#)

[GetBuildDateTimeString](#)

[GetCustomMenuSet](#)

[GetDefaultUnit](#)

[GetDesktopConfiguration](#)

[GetDistributedAnalysisMachines](#)

[GetDistributedAnalysisMachinesForDesignType](#)

[GetExeDir](#)

[GetGDIObjectCount](#)

[GetLibraryDirectory](#)

[GetLocalizationHelper](#)

[GetMessages](#)

[GetPersonalLibDirectory](#)

[GetPPELicensingEnabled](#)

[GetProcessID](#)

[GetProjectDirectory](#)

[GetProjectList](#)

[GetProjects](#)

[GetRegistryInt](#)

[GetRegistryString](#)

[GetRunningInstancesMgr](#)

[GetSchematicEnvironment](#)

[GetScriptingToolsHelper](#)

[GetSysLibDirectory](#)

[GetTempDirectory](#)

[GetUserLibDirectory](#)

[GetVersion](#)

[IsFeatureEnabled](#)

[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](#)

[Tile Windows](#)

**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"<br>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 | oDesktop.AddMessage("Project1", "HFSS 3D Layout", 0, "This is a test message", "") |

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

## AreThereSimulationsRunning

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

| <b>UI Access</b>        | NA                                                                                                                                                                                                                                                                                 |                                                                       |  |      |      |             |                         |         |                                                                       |
|-------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------|--|------|------|-------------|-------------------------|---------|-----------------------------------------------------------------------|
| <b>Parameters</b>       | <table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;bool&gt;<br/>&lt;AlsoPending&gt;</td> <td>Integer</td> <td>Whether to report if simulations are running, or running and pending.</td> </tr> </tbody> </table> |                                                                       |  | Name | Type | Description | <bool><br><AlsoPending> | Integer | Whether to report if simulations are running, or running and pending. |
| Name                    | Type                                                                                                                                                                                                                                                                               | Description                                                           |  |      |      |             |                         |         |                                                                       |
| <bool><br><AlsoPending> | Integer                                                                                                                                                                                                                                                                            | Whether to report if simulations are running, or running and pending. |  |      |      |             |                         |         |                                                                       |
| <b>Return Value</b>     | A human readable status string specifying what happened.                                                                                                                                                                                                                           |                                                                       |  |      |      |             |                         |         |                                                                       |

|                       |                                                              |
|-----------------------|--------------------------------------------------------------|
| <b>Python Syntax</b>  | AreThereSimulationsRunning (<bool>, clean)                   |
| <b>Python Example</b> | <code>oDesktop.AreThereSimulationsRunning(bool clean)</code> |

## ClearMessages

Clears messages, optionally specifying severity and design.

| <b>UI Access</b>  | In Message Manager, right-click the project name and click <b>Clear messages for Project#</b>            |             |  |      |      |             |
|-------------------|----------------------------------------------------------------------------------------------------------|-------------|--|------|------|-------------|
| <b>Parameters</b> | <table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> </table> |             |  | Name | Type | Description |
| Name              | Type                                                                                                     | Description |  |      |      |             |

|                     |                 |         |                                                                                                                                                                                                   |
|---------------------|-----------------|---------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                     | < projectName > | String  | Project name, an empty string clears all project messages.                                                                                                                                        |
|                     | < designName >  | String  | Design name; ignored if project name is empty; an empty string clears messages in all designs.                                                                                                    |
|                     | < severity >    | Integer | 0 - clear all info messages<br>1 - clear all info and warning messages<br>2 - clear all info, warning and error messages<br>3 - clear all messages included info, warning, error, and fatal-error |
| <b>Return Value</b> | None            |         |                                                                                                                                                                                                   |

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

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

## CloseAllWindows

Closes all MDI child windows on the desktop.

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

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

|                   |                                       |
|-------------------|---------------------------------------|
| <b>VB Syntax</b>  | CloseAllWindows                       |
| <b>VB Example</b> | <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.

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

|                      |                              |
|----------------------|------------------------------|
| <b>Python Syntax</b> | CloseProject (<ProjectName>) |
|----------------------|------------------------------|

**Python Example**

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

**VB Syntax**

```
CloseProject <ProjectName>
```

**VB Example**

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

## CloseProjectNoForce

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

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

**Python Syntax**

```
CloseProjectNoForce (<ProjectName>)
```

**Python Example**

```
oDesktop.CloseProjectNoForce ("MyProject")
```

|                   |                                                       |
|-------------------|-------------------------------------------------------|
| <b>VB Syntax</b>  | CloseProjectNoForce <ProjectName>                     |
| <b>VB Example</b> | <code>oDesktop.CloseProjectNoForce "MyProject"</code> |

## Count

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

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

|                   |                                                                                                                                                                   |
|-------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>VB Syntax</b>  | Count                                                                                                                                                             |
| <b>VB Example</b> | <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.

|                     |                                                                                                                                                                   |                     |  |      |      |             |               |        |                     |
|---------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------|--|------|------|-------------|---------------|--------|---------------------|
| <b>UI Access</b>    | <b>Edit &gt; Delete</b>                                                                                                                                           |                     |  |      |      |             |               |        |                     |
| <b>Parameters</b>   | <table border="1"><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;ProjectName&gt;</td><td>String</td><td>Name of the project</td></tr></table> |                     |  | Name | Type | Description | <ProjectName> | String | Name of the project |
| Name                | Type                                                                                                                                                              | Description         |  |      |      |             |               |        |                     |
| <ProjectName>       | String                                                                                                                                                            | Name of the project |  |      |      |             |               |        |                     |
| <b>Return Value</b> | None                                                                                                                                                              |                     |  |      |      |             |               |        |                     |

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

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

## 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.

|                     |                                                 |        |                                                                                                                                                                                                                                                        |
|---------------------|-------------------------------------------------|--------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>UI Access</b>    | Select Scheduler                                |        |                                                                                                                                                                                                                                                        |
| <b>Parameters</b>   | 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.                                                                                                                                                                                           |
| <b>Return Value</b> | <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.                                                                    |
|                     | A Boolean result about download complete or not |        |                                                                                                                                                                                                                                                        |

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

|                   |                                                                                                                                                                         |
|-------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>VB Syntax</b>  | DownloadJobResults( <i>jobID</i> , <i>projectPath</i> , <i>resultsPath</i> , <i>filters</i> )                                                                           |
| <b>VB Example</b> | <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.

|                     |                                                                                                                                                                          |                         |      |             |                  |        |                         |
|---------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------|------|-------------|------------------|--------|-------------------------|
| <b>UI Access</b>    | N/A                                                                                                                                                                      |                         |      |             |                  |        |                         |
| <b>Parameters</b>   | <table border="1"><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;pathToRegistry&gt;</td><td>String</td><td>Path to Registry entry.</td></tr></table> | Name                    | Type | Description | <pathToRegistry> | String | Path to Registry entry. |
| Name                | Type                                                                                                                                                                     | Description             |      |             |                  |        |                         |
| <pathToRegistry>    | String                                                                                                                                                                   | Path to Registry entry. |      |             |                  |        |                         |
| <b>Return Value</b> | Boolean: <ul style="list-style-type: none"><li>• <b>True</b> – Key has been deleted.</li><li>• <b>False</b> – Key does not exist.</li></ul>                              |                         |      |             |                  |        |                         |

|                      |                                       |
|----------------------|---------------------------------------|
| <b>Python Syntax</b> | DeleteRegistryEntry(<pathToRegistry>) |
|----------------------|---------------------------------------|

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

|                   |                                                                     |
|-------------------|---------------------------------------------------------------------|
| <b>VB Syntax</b>  | <code>DeleteRegistryEntry &lt;pathToRegistry&gt;</code>             |
| <b>VB Example</b> | <pre>res = oDesktop.DeleteRegistryEntry "Desktop/ColorScheme"</pre> |

## DoesRegistryValueExist

Determines whether a registry value exists.

| <b>UI Access</b>             | N/A                                                                                                                                                                                                                                            |                                            |      |             |                              |        |                                            |
|------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------|------|-------------|------------------------------|--------|--------------------------------------------|
| <b>Parameters</b>            | <table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><code>&lt;KeyName&gt;</code></td> <td>String</td> <td>Full name of registry key, including path.</td> </tr> </tbody> </table> | Name                                       | Type | Description | <code>&lt;KeyName&gt;</code> | String | Full name of registry key, including path. |
| Name                         | Type                                                                                                                                                                                                                                           | Description                                |      |             |                              |        |                                            |
| <code>&lt;KeyName&gt;</code> | String                                                                                                                                                                                                                                         | Full name of registry key, including path. |      |             |                              |        |                                            |
| <b>Return Value</b>          | <p>Boolean:</p> <ul style="list-style-type: none"> <li>• <b>True</b> – Key exists.</li> <li>• <b>False</b> – Key does not exist.</li> </ul>                                                                                                    |                                            |      |             |                              |        |                                            |

|                       |                                                                                                      |
|-----------------------|------------------------------------------------------------------------------------------------------|
| <b>Python Syntax</b>  | <code>DoesRegistryValueExist(&lt;KeyName&gt;)</code>                                                 |
| <b>Python Example</b> | <pre>Exist = oDesktop.DoesRegistryValueExist('Desktop/ActiveDSOConfigurations/HFSS 3D Layout')</pre> |

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

## EnableAutoSave

Enable or disable autosave feature.

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

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

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

## ExportOptionsFiles

Copies the options config files to the *DestinationDirectory*.

|                     |                                                         |                |                                                       |
|---------------------|---------------------------------------------------------|----------------|-------------------------------------------------------|
| <b>UI Access</b>    | <b>Tools &gt; Options &gt; Export Options Files ...</b> |                |                                                       |
| <b>Parameters</b>   | Name<br><DestinationDirectory>                          | Type<br>String | Description<br>The path to the destination directory. |
| <b>Return Value</b> | None                                                    |                |                                                       |

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

|                   |                                               |
|-------------------|-----------------------------------------------|
| <b>VB Syntax</b>  | ExportOptionsFiles <DestinationDirectory>     |
| <b>VB Example</b> | 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.

|                  |     |
|------------------|-----|
| <b>UI Access</b> | N/A |
|------------------|-----|

|                     |                                                              |
|---------------------|--------------------------------------------------------------|
| <b>Parameters</b>   | None.                                                        |
| <b>Return Value</b> | Object: the project that is currently active in the desktop. |

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

|                   |                                                     |
|-------------------|-----------------------------------------------------|
| <b>VB Syntax</b>  | GetActiveProject                                    |
| <b>VB Example</b> | <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.

|                     |                                                                                                                                                                                                            |
|---------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>UI Access</b>    | N/A                                                                                                                                                                                                        |
| <b>Parameters</b>   | None.                                                                                                                                                                                                      |
| <b>Return Value</b> | 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.) |

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

|                   |                                                             |
|-------------------|-------------------------------------------------------------|
| <b>VB Syntax</b>  | GetActiveScheduler                                          |
| <b>VB Example</b> | <pre>Set oSchedulerName = oDesktop.GetActiveScheduler</pre> |

## GetActiveSchedulerInfo

Obtains info for the scheduler active in the Desktop.

**Note:**

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

|                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|---------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>UI Access</b>    | N/A                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| <b>Parameters</b>   | None.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| <b>Return Value</b> | 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 <code>{"Selected Scheduler": "RSM", "Scheduler Name": "RSM", "Server": ""}</code> . For a remote scheduler, the return format resembles <code>{"Selected Scheduler": "Remote RSM", "Scheduler Name": "SLURM", "Server": "&lt;server&gt;"}</code> |

|                       |                                                               |
|-----------------------|---------------------------------------------------------------|
| <b>Python Syntax</b>  | GetActiveSchedulerInfo()                                      |
| <b>Python Example</b> | <pre>oSchedulerInfo = oDesktop.GetActiveSchedulerInfo()</pre> |

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

## GetAutoSaveEnabled

Checks whether the autosave feature is enabled.

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

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

---

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

## GetBuildDateTimeString

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

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

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

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

## GetCustomMenuSet

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

|                  |     |
|------------------|-----|
| <b>UI Access</b> | N/A |
|------------------|-----|

|                     |                                                                                   |
|---------------------|-----------------------------------------------------------------------------------|
| <b>Parameters</b>   | None.                                                                             |
| <b>Return Value</b> | String containing current menu set. For example, 'Default', 'EM', 'Twin Builder'. |

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

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

## GetDefaultUnit

Returns the default unit for a physical quantity.

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

- "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)

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

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

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

## GetDesktopConfiguration

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

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

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

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

## GetDistributedAnalysisMachines

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

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

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

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

## GetDistributedAnalysisMachinesForDesignType

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

|                     |                                                                                                                                                                                                                                                                                                                                    |                                                                                                                                                                                   |  |      |      |             |                  |        |                                                                                                                                                                                   |
|---------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--|------|------|-------------|------------------|--------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>UI Access</b>    | NA                                                                                                                                                                                                                                                                                                                                 |                                                                                                                                                                                   |  |      |      |             |                  |        |                                                                                                                                                                                   |
| <b>Parameters</b>   | <table border="1"><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;designTypeName&gt;</td><td>String</td><td>The name of the type of design, such as "Twin Builder", "HFSS", HFSS-IE", "Maxwell 3D", "Maxwell 2D", "RMxprt", "EM Design", "Circuit", "System", "Q3D Extractor", "2D Extractor"</td></tr></table> |                                                                                                                                                                                   |  | Name | Type | Description | <designTypeName> | String | The name of the type of design, such as "Twin Builder", "HFSS", HFSS-IE", "Maxwell 3D", "Maxwell 2D", "RMxprt", "EM Design", "Circuit", "System", "Q3D Extractor", "2D Extractor" |
| Name                | Type                                                                                                                                                                                                                                                                                                                               | Description                                                                                                                                                                       |  |      |      |             |                  |        |                                                                                                                                                                                   |
| <designTypeName>    | String                                                                                                                                                                                                                                                                                                                             | The name of the type of design, such as "Twin Builder", "HFSS", HFSS-IE", "Maxwell 3D", "Maxwell 2D", "RMxprt", "EM Design", "Circuit", "System", "Q3D Extractor", "2D Extractor" |  |      |      |             |                  |        |                                                                                                                                                                                   |
| <b>Return Value</b> | Object; returns a collection of machine names.                                                                                                                                                                                                                                                                                     |                                                                                                                                                                                   |  |      |      |             |                  |        |                                                                                                                                                                                   |

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

|                  |                                                              |
|------------------|--------------------------------------------------------------|
| <b>VB Syntax</b> | GetDistributedAnalysisMachinesForDesignType <designTypeName> |
|------------------|--------------------------------------------------------------|

|                   |                                                                                                       |
|-------------------|-------------------------------------------------------------------------------------------------------|
| <b>VB Example</b> | <pre>Set machineNames =oDesktop. GetDistributedAnalysisMachinesForDesignType ("HFSS 3D Layout")</pre> |
|-------------------|-------------------------------------------------------------------------------------------------------|

## GetExeDir

Returns the path where the executable is located.

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

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

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

## GetGDIObjectCount

### Note:

This command is for internal Ansys use only.

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

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

## GetLibraryDirectory

Get the path to the SysLib directory.

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

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

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

*VB Example:*

---

message box returns the path in this example

---

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

## GetLocalizationHelper

**Note:**

This command is for internal Ansys use only.

Returns the object for the localization helper.

|                     |                                                                                        |      |             |
|---------------------|----------------------------------------------------------------------------------------|------|-------------|
| <b>UI Access</b>    | NA                                                                                     |      |             |
| <b>Parameters</b>   | Name                                                                                   | Type | Description |
| <b>Return Value</b> | None<br>Object<br>Localization helper object, such as "IDispatch(ILocalizationHelper)" |      |             |

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

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

## GetMessages

Get the messages from a specified project and design.

|                   |                                    |                |                                                                                                                                                               |
|-------------------|------------------------------------|----------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>UI Access</b>  | NA                                 |                |                                                                                                                                                               |
| <b>Parameters</b> | Name<br><i>&lt;ProjectName&gt;</i> | Type<br>String | Description<br>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: <ul style="list-style-type: none"><li>• 0 – info and above</li><li>• 1 – warning and above</li><li>• 2 – error and fatal</li><li>• 3 – fatal only (rarely used)</li></ul> |
| <b>Return Value</b> | Array of string messages. |                                                                                                                                                                                                                                                         |

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

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

## GetMonitorData

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

|                  |    |
|------------------|----|
| <b>UI Access</b> | NA |
|------------------|----|

| Parameters   | Name                             | Type   | Description                                            |
|--------------|----------------------------------|--------|--------------------------------------------------------|
|              | <Request>                        | String | a json string describing what monitor data is desired. |
| Return Value | a json string with the results.. |        |                                                        |

|                |                                              |
|----------------|----------------------------------------------|
| Python Syntax  | GetMonitorData (Request>)                    |
| Python Example | Messages = oDesktop.GetMonitorData (request) |

## GetPersonalLibDirectory

Get the path to the PersonalLib directory.

|              |                                           |
|--------------|-------------------------------------------|
| UI Access    | N/A                                       |
| Parameters   | None.                                     |
| Return Value | String path to the PersonalLib directory. |

|                |                                     |
|----------------|-------------------------------------|
| Python Syntax  | GetPersonalLibDirectory()           |
| Python Example | oDesktop.GetPersonalLibDirectory () |

|           |                         |
|-----------|-------------------------|
| VB Syntax | GetPersonalLibDirectory |
|-----------|-------------------------|

|                   |                                               |
|-------------------|-----------------------------------------------|
| <b>VB Example</b> | <code>oDesktop.GetPersonalLibDirectory</code> |
|-------------------|-----------------------------------------------|

## GetPPELicensingEnabled

Returns whether the PPE licensing is enabled.

|                     |                                                                                                                                                                     |
|---------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>UI Access</b>    | N/A                                                                                                                                                                 |
| <b>Parameters</b>   | None.                                                                                                                                                               |
| <b>Return Value</b> | <p>Boolean:</p> <ul style="list-style-type: none"> <li>• <b>True</b> – PPE licensing is enabled.</li> <li>• <b>False</b> – PPE licensing is not enabled.</li> </ul> |

|                       |                                                |
|-----------------------|------------------------------------------------|
| <b>Python Syntax</b>  | <code>GetPPELicensingEnabled()</code>          |
| <b>Python Example</b> | <code>oDesktop.GetPPELicensingEnabled()</code> |

|                   |                                              |
|-------------------|----------------------------------------------|
| <b>VB Syntax</b>  | <code>GetPPELicensingEnabled</code>          |
| <b>VB Example</b> | <code>oDesktop.GetPPELicensingEnabled</code> |

## GetProcessID

Returns the process ID of `ansysedt.exe`.

|                  |     |
|------------------|-----|
| <b>UI Access</b> | N/A |
|------------------|-----|

|                     |                                                         |
|---------------------|---------------------------------------------------------|
| <b>Parameters</b>   | None.                                                   |
| <b>Return Value</b> | Integer process ID of ansysedt.exe. For example, 12716. |

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

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

## GetProjectDirectory

Gets the path to the Project directory.

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

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

---

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

## GetProjectList

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

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

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

|                   |                                                         |
|-------------------|---------------------------------------------------------|
| <b>VB Syntax</b>  | GetProjectList                                          |
| <b>VB Example</b> | <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.

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

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

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

## GetRegistryInt

Obtains registry key integer value.

|                              |                                                                                                                                                                                                   |                                            |  |      |      |             |                              |        |                                            |
|------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------|--|------|------|-------------|------------------------------|--------|--------------------------------------------|
| <b>UI Access</b>             | N/A                                                                                                                                                                                               |                                            |  |      |      |             |                              |        |                                            |
| <b>Parameters</b>            | <table border="1"><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><code>&lt;KeyName&gt;</code></td><td>String</td><td>Full name of registry key, including path.</td></tr></table> |                                            |  | Name | Type | Description | <code>&lt;KeyName&gt;</code> | String | Full name of registry key, including path. |
| Name                         | Type                                                                                                                                                                                              | Description                                |  |      |      |             |                              |        |                                            |
| <code>&lt;KeyName&gt;</code> | String                                                                                                                                                                                            | Full name of registry key, including path. |  |      |      |             |                              |        |                                            |
| <b>Return Value</b>          | Integer if the integer value is found. Return as Bad-Argument-Value if registry key does not exist or it is not an integer value.                                                                 |                                            |  |      |      |             |                              |        |                                            |

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

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

## GetRegistryString

Obtains registry key string value.

|                     |                                                                                                                               |                |                                                           |
|---------------------|-------------------------------------------------------------------------------------------------------------------------------|----------------|-----------------------------------------------------------|
| <b>UI Access</b>    | N/A                                                                                                                           |                |                                                           |
| <b>Parameters</b>   | Name<br><KeyName>                                                                                                             | Type<br>String | Description<br>Full name of registry key, including path. |
| <b>Return Value</b> | String if the string value is found. Return as Bad-Argument-Value if registry key does not exist or it is not a string value. |                |                                                           |

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

|  |  |
|--|--|
|  |  |
|--|--|

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

## GetRunningInstancesMgr

Returns the object of the Running Instances Manager.

| <b>UI Access</b>    | N/A                                                                                                                                                     |             |  |      |      |             |      |  |  |
|---------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------|-------------|--|------|------|-------------|------|--|--|
| <b>Parameters</b>   | <table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>None</td><td></td><td></td></tr></tbody></table> |             |  | Name | Type | Description | None |  |  |
| Name                | Type                                                                                                                                                    | Description |  |      |      |             |      |  |  |
| None                |                                                                                                                                                         |             |  |      |      |             |      |  |  |
| <b>Return Value</b> | <p>Object<br/>Running instances manager object</p>                                                                                                      |             |  |      |      |             |      |  |  |

|                       |                                                       |
|-----------------------|-------------------------------------------------------|
| <b>Python Syntax</b>  | GetRunningInstancesMgr()                              |
| <b>Python Example</b> | oRunningInstances = oDesktop.GetRunningInstancesMgr() |

|                  |                          |
|------------------|--------------------------|
| <b>VB Syntax</b> | 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**.

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

**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.

|                     |                                                |      |             |
|---------------------|------------------------------------------------|------|-------------|
| <b>UI Access</b>    | NA                                             |      |             |
| <b>Parameters</b>   | Name                                           | Type | Description |
| <b>Return Value</b> | <p>Object<br/>ScriptingTools helper object</p> |      |             |

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

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

## GetSysLibDirectory

Get the path to the SysLib directory.

---

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

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

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

## GetTempDirectory

Gets the path to the Temp directory.

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

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

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

## GetUserLibDirectory

Gets the path to the UserLib directory.

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

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

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

## GetVersion

Returns a string representing the version.

---

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

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

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

## IsFeatureEnabled

Returns a Boolean for whether a queried feature is enabled.

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

|                      |                                 |
|----------------------|---------------------------------|
| <b>Python Syntax</b> | <code>IsFeatureEnabled()</code> |
|----------------------|---------------------------------|

|                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|-----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Python Example</b> | <pre> import ScriptEnv ScriptEnv.Initialize("Ansoft.ElectronicsDesktop") oDesktop.RestoreWindow()  feature_strs = ["SF3519", "F195709_EXPORT_TO_EMIT", "F362235_EMIT_RESULTS_WINDOW_IMPROVEMENTS", _<br/>     "F136736_SBR_Rough_Surface", "F353006_VOLUMETRIC_SBR", "F359673_SBR_MULTISTATE_ARRAY",<br/>     "F393115_SWE_ANTENNA", _<br/>     "S196592_SBR_Directivity", "F432541_VSBR_IMPROVEMENTS", "F353007_VRT_FILTERS_ENHANCE",<br/>     "S540337_SBR_REGION_LOSS_DIRECTIVITY", _<br/>     "F11941_VRT_CURRENT_DENSITY", "S544593_SBR_3D_COMPONENT_ARRAY", "F539850_SBR_GO_BLOCKAGE", "F540275_SBR_RAYSTATS"]<br/> <br/>     results = [oDesktop.IsEnabled(str) for str in feature_strs]<br/> <br/>     result_txt = open("C:/Users/MyResults/Downloads/results.txt", "w")<br/> <br/>         for i in range(len(feature_strs)):<br/> <br/>             result_txt.write('%s : %s\n' % (feature_strs[i], results[i]))<br/> <br/>     result_txt.close() </pre> |
|-----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

|                   |                                      |
|-------------------|--------------------------------------|
| <b>VB Syntax</b>  | GetVersion()                         |
| <b>VB Example</b> | results = oDesktop.IsEnabled(SF3519) |

## KeepDesktopResponsive

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

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

|                       |                                                             |
|-----------------------|-------------------------------------------------------------|
| <b>Python Syntax</b>  | KeepDesktopResponsive ( <i>&lt;TimeInMilliseconds&gt;</i> ) |
| <b>Python Example</b> | <code>oDesktop.KeepDesktopResponsive(10000)</code>          |

|                   |                                                         |
|-------------------|---------------------------------------------------------|
| <b>VB Syntax</b>  | KeepDesktopResponsive <i>&lt;TimeInMilliseconds&gt;</i> |
| <b>VB Example</b> | <code>oDesktop.KeepDesktopResponsive 10000</code>       |

## LaunchJobMonitor

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

|                     |                                    |                |                                                         |
|---------------------|------------------------------------|----------------|---------------------------------------------------------|
| <b>UI Access</b>    | Launch Job Monitor                 |                |                                                         |
| <b>Parameters</b>   | Name<br><i>&lt;projectPath&gt;</i> | Type<br>String | Description<br>Path to the project file to be monitored |
| <b>Return Value</b> | None                               |                |                                                         |

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

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

## NewProject

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

|                     |                                    |
|---------------------|------------------------------------|
| <b>UI Access</b>    | File > New.                        |
| <b>Parameters</b>   | None.                              |
| <b>Return Value</b> | Object, the project that is added. |

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

|                  |            |
|------------------|------------|
| <b>VB Syntax</b> | NewProject |
|------------------|------------|

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

## OpenAndConvertProject

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

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

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

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

|                  |                                                            |
|------------------|------------------------------------------------------------|
| <b>VB Syntax</b> | <code>OpenAndConvertProject(filePath, legacyChoice)</code> |
|------------------|------------------------------------------------------------|

**VB Example**

```
Set oProject = oDesktop.OpenAndConvertProject("c:\files\optimtee.hfss", 1)
```

**Note:** optimtee.hfss is gone after this code executes

## OpenMultipleProjects

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

|                     |             |        |                       |
|---------------------|-------------|--------|-----------------------|
| <b>UI Access</b>    | N/A         |        |                       |
| <b>Parameters</b>   | Name        | Type   | Description           |
|                     | <directory> | String | Path to the projects. |
| <b>Return Value</b> | None.       |        |                       |

**Python Syntax**

```
OpenAndConvertProject(<filePath>, <legacyChoice>)
```

**Python Example**

```
oProject = oDesktop.OpenAndConvertProject("c:\files\optimtee.", "*.aedt")
```

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

## OpenProject

Opens a specified project.

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

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

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

## OpenProjectWithConversion

**Note:**

This command is for internal Ansys use only.

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

## 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 |
|-----------|----|

|                     |      |      |             |
|---------------------|------|------|-------------|
| <b>Parameters</b>   | Name | Type | Description |
|                     | None |      |             |
| <b>Return Value</b> | None |      |             |

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

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

## PauseScript

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

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

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

|  |  |
|--|--|
|  |  |
|--|--|

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

## Print

Prints the contents of the active view window.

|                     |               |
|---------------------|---------------|
| <b>UI Access</b>    | File > Print. |
| <b>Parameters</b>   | None.         |
| <b>Return Value</b> | None.         |

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

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

## QuitApplication

Exits the desktop.

|                     |              |
|---------------------|--------------|
| <b>UI Access</b>    | File > Exit. |
| <b>Parameters</b>   | None.        |
| <b>Return Value</b> | None.        |

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

|                   |                                       |
|-------------------|---------------------------------------|
| <b>VB Syntax</b>  | QuitApplication                       |
| <b>VB Example</b> | <code>oDesktop.QuitApplication</code> |

## RefreshJobMonitor

For use in monitoring a job.

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

- |  |                                                                                                                                                                                                                                               |
|--|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|  | <ul style="list-style-type: none"><li>• "Monitor Not Visible"</li><li>• "Queued"</li><li>• "Running"</li><li>• "Shutting Down"</li><li>• "Unknown"</li><li>• "Completed"</li><li>• "Not Monitoring"</li><li>• "Starting Monitoring"</li></ul> |
|--|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

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

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

|                   |                                         |
|-------------------|-----------------------------------------|
| <b>VB Syntax</b>  | RefreshJobMonitor                       |
| <b>VB Example</b> | <code>oDesktop.RefreshJobMonitor</code> |

## ResetLogging

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

|                  |     |
|------------------|-----|
| <b>UI Access</b> | N/A |
|------------------|-----|

|                     |            |         |                      |
|---------------------|------------|---------|----------------------|
| <b>Parameters</b>   | Name       | Type    | Description          |
|                     | <logFile>  | String  | Path to log file.    |
|                     | <logLevel> | Integer | Specified log level. |
| <b>Return Value</b> | None.      |         |                      |

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

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

## RestoreProjectArchive

Restores a previously archived project to a specified path.

|                     |                           |         |                                                                  |
|---------------------|---------------------------|---------|------------------------------------------------------------------|
| <b>UI Access</b>    | File > Restore Archive.   |         |                                                                  |
| <b>Parameters</b>   | 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.       |
| <b>Return Value</b> | None.                     |         |                                                                  |

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

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

## RestoreWindow

Restores a minimized Desktop window.

|                     |       |
|---------------------|-------|
| <b>UI Access</b>    | N/A   |
| <b>Parameters</b>   | None. |
| <b>Return Value</b> | None. |

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

|                   |                                     |
|-------------------|-------------------------------------|
| <b>VB Syntax</b>  | RestoreWindow                       |
| <b>VB Example</b> | <code>oDesktop.RestoreWindow</code> |

## ResumeRecording

Resume recording a script.

|                     |       |
|---------------------|-------|
| <b>UI Access</b>    | N/A   |
| <b>Parameters</b>   | None. |
| <b>Return Value</b> | None  |

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

|                   |                                       |
|-------------------|---------------------------------------|
| <b>VB Syntax</b>  | ResumeRecording                       |
| <b>VB Example</b> | <code>oDesktop.ResumeRecording</code> |

## RunACTWizardScript

### Note:

This command is for internal Ansys use only.

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

## RunProgram

Runs an external program.

| <b>UI Access</b>              | NA                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |                                                                              |      |             |                               |        |                             |                               |        |                                                                          |                               |        |                                                |                               |                  |                                                                              |
|-------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------|------|-------------|-------------------------------|--------|-----------------------------|-------------------------------|--------|--------------------------------------------------------------------------|-------------------------------|--------|------------------------------------------------|-------------------------------|------------------|------------------------------------------------------------------------------|
| <b>Parameters</b>             | <table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td><code>&lt;ProgName&gt;</code></td><td>String</td><td>Name of the program to run.</td></tr><tr><td><code>&lt;ProgPath&gt;</code></td><td>String</td><td>Location of the program. Pass in an empty string to use the system path.</td></tr><tr><td><code>&lt;WorkPath&gt;</code></td><td>String</td><td>Working directory in which program will start.</td></tr><tr><td><code>&lt;ArgArray&gt;</code></td><td>Array of Strings</td><td>Arguments to pass to the program. If no arguments, pass in <code>None</code></td></tr></tbody></table> | Name                                                                         | Type | Description | <code>&lt;ProgName&gt;</code> | String | Name of the program to run. | <code>&lt;ProgPath&gt;</code> | String | Location of the program. Pass in an empty string to use the system path. | <code>&lt;WorkPath&gt;</code> | String | Working directory in which program will start. | <code>&lt;ArgArray&gt;</code> | Array of Strings | Arguments to pass to the program. If no arguments, pass in <code>None</code> |
| Name                          | Type                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               | Description                                                                  |      |             |                               |        |                             |                               |        |                                                                          |                               |        |                                                |                               |                  |                                                                              |
| <code>&lt;ProgName&gt;</code> | String                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             | Name of the program to run.                                                  |      |             |                               |        |                             |                               |        |                                                                          |                               |        |                                                |                               |                  |                                                                              |
| <code>&lt;ProgPath&gt;</code> | String                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             | Location of the program. Pass in an empty string to use the system path.     |      |             |                               |        |                             |                               |        |                                                                          |                               |        |                                                |                               |                  |                                                                              |
| <code>&lt;WorkPath&gt;</code> | String                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             | Working directory in which program will start.                               |      |             |                               |        |                             |                               |        |                                                                          |                               |        |                                                |                               |                  |                                                                              |
| <code>&lt;ArgArray&gt;</code> | Array of Strings                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   | Arguments to pass to the program. If no arguments, pass in <code>None</code> |      |             |                               |        |                             |                               |        |                                                                          |                               |        |                                                |                               |                  |                                                                              |
| <b>Return Value</b>           | None                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |                                                                              |      |             |                               |        |                             |                               |        |                                                                          |                               |        |                                                |                               |                  |                                                                              |

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

|                   |                                                                                                           |
|-------------------|-----------------------------------------------------------------------------------------------------------|
| <b>VB Syntax</b>  | RunProgram <ProgName>, <ProgPath>, <WorkPath>, <ArgArray>                                                 |
| <b>VB Example</b> | <pre> oDesktop.RunProgram "winword.exe", _ "C:\Program Files\Microsoft Office\Office10",_ "", None </pre> |

## RunScript

Launches another script from within the script currently being executed.

| UI Access  | Tools>Run Script |        |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|------------|------------------|--------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Parameters | Name             | Type   | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|            | <SCriptPath>     | String | <p>Name or full path of the script to execute.</p> <p>If the full path to the script is not specified, Twin Builder searches for the specified script in the following locations, in this order:</p> <ol style="list-style-type: none"> <li>1. Personal library directory.</li> </ol> <p>This is the <b>PersonalLib</b> subdirectory in the project directory. The project directory can be specified in the <b>General Options</b> dialog box (click <b>Tools&gt;Options&gt;General Options</b> to open this dialog box) under the <b>Project Options</b> tab.</p> <ol style="list-style-type: none"> <li>2. User library directory.</li> </ol> <p>This is the <b>userlib</b> subdirectory in the library directory. The library directory can be specified in the <b>General Options</b> dialog box (click <b>Tools&gt;Options&gt;General Options</b> to open this dialog box) under the <b>Project Options</b> tab.</p> <ol style="list-style-type: none"> <li>3. System library directory.</li> </ol> <p>This is the <b>syslib</b> subdirectory in the library directory. The library directory can</p> |

|                     |                                                    |  |                                                                                                                                                                                                               |
|---------------------|----------------------------------------------------|--|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                     |                                                    |  | be specified in the <b>General Options</b> dialog box (click <b>Tools&gt;Options&gt;General Options</b> to open this dialog box) under the <b>Project Options</b> tab.<br><br>4. HFSS installation directory. |
| <b>Return Value</b> | Long<br><br>the return code for the script method. |  |                                                                                                                                                                                                               |

|                       |                                                          |
|-----------------------|----------------------------------------------------------|
| <b>Python Syntax</b>  | RunScript (< <i>ScriptPath</i> >)                        |
| <b>Python Example</b> | <code>oDesktop.RunScript ("C:/Project/test1.vbs")</code> |

|                   |                                                          |
|-------------------|----------------------------------------------------------|
| <b>VB Syntax</b>  | RunScript < <i>ScriptPath</i> >                          |
| <b>VB Example</b> | <code>oDesktop.RunScript ("C:/Project/test1.vbs")</code> |

## RunScriptWithArguments

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

|                   |                               |                |                                                                                                                                                                                   |
|-------------------|-------------------------------|----------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>UI Access</b>  | NA                            |                |                                                                                                                                                                                   |
| <b>Parameters</b> | Name<br>< <i>ScriptPath</i> > | Type<br>String | Description<br><br>The name or full path of the script to execute. If the full path to the script is not specified, the software looks for the script in the following locations: |

|                     |                                                           |        |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|---------------------|-----------------------------------------------------------|--------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                     |                                                           |        | <ul style="list-style-type: none"> <li>• Personal library directory: "PersonalLib". The PersonalLib directory can be specified in Tools&gt;Options&gt;General Options on the 'Project Options' tab.</li> <li>• User library directory: directory "userlib". The UserLib directory can be specified in Tools&gt;Options&gt;General Options on the 'Project Options' tab.</li> <li>• System library directory: directory "syslib". The SysLib directory can be specified in Tools&gt;Options&gt;General Options on the 'Project Options' tab.</li> <li>• Software installation directory</li> </ul> |
|                     | <Arguments>                                               | String | The arguments to supply to the specified script.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| <b>Return Value</b> | <p>Long</p> <p>the return code for the script method.</p> |        |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |

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

|                   |                                                                              |
|-------------------|------------------------------------------------------------------------------|
| <b>VB Syntax</b>  | RunScriptWithArguments <ScriptPath>, <Arguments>                             |
| <b>VB Example</b> | <pre>oDesktop.RunScriptWithArguments<br/>"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                                       | Type   | Description                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|            | <option>                                   | String | One of the following options (not case sensitive): <ul style="list-style-type: none"><li>• Empty string for remote RSM service</li><li>• "RSM" for local RSM</li><li>• "Windows HPC" for Windows HPC</li><li>• "LSF" for Load-Sharing Facility</li><li>• "SGE" for Grid Engine (GE, OGE, SGE, UGE, etc.)</li><li>• "PBS" for Portable Batch Scheduler/System (PBSPro, Torque, Maui, etc.)</li><li>• "Ansys Cloud" for Ansys Cloud</li></ul> |
|            | <address (optional)>                       | String | String specifying the IP address or hostname of the head node or for the remote host running the RSM service.                                                                                                                                                                                                                                                                                                                               |
|            | <username (optional)>                      | String | Username string to use for remote RSM service (or blank to use username stored in current submission host user settings). If the (non-blank) username doesn't match the username stored in current submission host user settings, then the Select Scheduler dialog is displayed to allow for password entry prior to job submission.                                                                                                        |
|            | <forcePasswordEntry (optional)>            | String | Boolean used to force display of the Select Scheduler GUI to allow for password entry prior to job submission.                                                                                                                                                                                                                                                                                                                              |

|                     |                                                                                                                                                      |
|---------------------|------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Return Value</b> | The selected scheduler (if selection was successful, this string should match the input option string, although it could differ in upper/lowercase). |
|---------------------|------------------------------------------------------------------------------------------------------------------------------------------------------|

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

|                   |                                                                                        |
|-------------------|----------------------------------------------------------------------------------------|
| <b>VB Syntax</b>  | Select Scheduler <option>, <address>, <username>, <forcePasswordEntry>                 |
| <b>VB Example</b> | <pre>result = oDesktop.SelectScheduler "Windows HPC", "headnode.win.example.com"</pre> |

## SetActiveProject

Specifies the name of the project that should become active in the desktop. Returns that project.

| <b>UI Access</b>    | N/A                                                                                                                                                                                                                                                                |                                                                         |      |      |             |               |        |                                                                         |
|---------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------|------|------|-------------|---------------|--------|-------------------------------------------------------------------------|
| <b>Parameters</b>   | <table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;ProjectName&gt;</td> <td>String</td> <td>The name of the project already in the Desktop that is to be activated.</td> </tr> </tbody> </table> |                                                                         | Name | Type | Description | <ProjectName> | String | The name of the project already in the Desktop that is to be activated. |
| Name                | Type                                                                                                                                                                                                                                                               | Description                                                             |      |      |             |               |        |                                                                         |
| <ProjectName>       | String                                                                                                                                                                                                                                                             | The name of the project already in the Desktop that is to be activated. |      |      |             |               |        |                                                                         |
| <b>Return Value</b> | Object, the project that is activated.                                                                                                                                                                                                                             |                                                                         |      |      |             |               |        |                                                                         |

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

|  |  |
|--|--|
|  |  |
|--|--|

|                   |                                                                |
|-------------------|----------------------------------------------------------------|
| <b>VB Syntax</b>  | SetActiveProject <ProjectName>                                 |
| <b>VB Example</b> | <pre>Set oProject = oDesktop.SetActiveProject "Project1"</pre> |

## 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 is selected). This command permits unambiguous specification of the active project.

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

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

|                  |                                      |
|------------------|--------------------------------------|
| <b>VB Syntax</b> | SetActiveProjectByPath <ProjectName> |
|------------------|--------------------------------------|

|                   |                                                                             |
|-------------------|-----------------------------------------------------------------------------|
| <b>VB Example</b> | Set oProject = oDesktop.SetActiveProjectByPath "c:\Projects\MyProject.aedt" |
|-------------------|-----------------------------------------------------------------------------|

## SetCustomMenuSet

Sets the custom menu set for Electronics Desktop.

|                     |                                                                                                                                                                              |                |                                                                                                                                                                                                                                                |
|---------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>UI Access</b>    | Navigate to <b>Tools &gt; Options &gt; General Options &gt; General &gt; Desktop Configuration</b> . Select a configuration using the <b>Custom Menu Set</b> drop-down menu. |                |                                                                                                                                                                                                                                                |
| <b>Parameters</b>   | Name<br><customMenuSet>                                                                                                                                                      | Type<br>String | Description<br>Name of desired menu set. Can be one of: <ul style="list-style-type: none"><li>• 'Default'</li><li>• 'EM'</li><li>• 'RF'</li><li>• 'RF.0'</li><li>• 'SI'</li><li>• 'SI1.0'</li><li>• 'SI2.0'</li><li>• 'Twin Builder'</li></ul> |
| <b>Return Value</b> | None                                                                                                                                                                         |                |                                                                                                                                                                                                                                                |

|                       |                                      |
|-----------------------|--------------------------------------|
| <b>Python Syntax</b>  | SetCustomMenuSet(<customMenuSet>)    |
| <b>Python Example</b> | oDesktop.SetCustomMenuSet('Default') |

|                   |                                               |
|-------------------|-----------------------------------------------|
| <b>VB Syntax</b>  | SetCustomMenuSet(<customMenuSet>)             |
| <b>VB Example</b> | Set oProject = oDesktop.SetCustomMenuSet "EM" |

## SetDesktopConfiguration

Sets the desktop configuration.

| <b>UI Access</b>    | Navigate to <b>Tools &gt; Options &gt; General Options &gt; General &gt; Desktop Configuration</b> . Select a configuration using the <b>Set targeted configuration</b> drop-down menu.                                                                                                                                                             |                                                                                                                                                                           |  |      |      |             |              |        |                                                                                                                                                                           |
|---------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--|------|------|-------------|--------------|--------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Parameters</b>   | <table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>&lt;configName&gt;</td><td>String</td><td>Name of desired Desktop configuration. Can be one of:<ul style="list-style-type: none"><li>'All'</li><li>'EM'</li><li>'RF'</li><li>'SI'</li><li>'Twin Builder'</li></ul></td></tr></tbody></table> |                                                                                                                                                                           |  | Name | Type | Description | <configName> | String | Name of desired Desktop configuration. Can be one of: <ul style="list-style-type: none"><li>'All'</li><li>'EM'</li><li>'RF'</li><li>'SI'</li><li>'Twin Builder'</li></ul> |
| Name                | Type                                                                                                                                                                                                                                                                                                                                                | Description                                                                                                                                                               |  |      |      |             |              |        |                                                                                                                                                                           |
| <configName>        | String                                                                                                                                                                                                                                                                                                                                              | Name of desired Desktop configuration. Can be one of: <ul style="list-style-type: none"><li>'All'</li><li>'EM'</li><li>'RF'</li><li>'SI'</li><li>'Twin Builder'</li></ul> |  |      |      |             |              |        |                                                                                                                                                                           |
| <b>Return Value</b> | None                                                                                                                                                                                                                                                                                                                                                |                                                                                                                                                                           |  |      |      |             |              |        |                                                                                                                                                                           |

|                       |                                         |
|-----------------------|-----------------------------------------|
| <b>Python Syntax</b>  | SetDesktopConfiguration (<configName>)  |
| <b>Python Example</b> | oDesktop.SetDesktopConfiguration ('RF') |

---

|                   |                                                      |
|-------------------|------------------------------------------------------|
| <b>VB Syntax</b>  | SetDesktopConfiguration(<configName>)                |
| <b>VB Example</b> | Set oProject = oDesktop.SetDesktopConfiguration "SI" |

## SetLibraryDirectory

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

| <b>UI Access</b>    | NA                                                                                                                                                                                          |                                  |      |             |                 |        |                                  |
|---------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------|------|-------------|-----------------|--------|----------------------------------|
| <b>Parameters</b>   | <table border="1"> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> <tr> <td>&lt;DirectoryPath&gt;</td> <td>String</td> <td>The path to the SysLib Directory</td> </tr> </table> | Name                             | Type | Description | <DirectoryPath> | String | The path to the SysLib Directory |
| Name                | Type                                                                                                                                                                                        | Description                      |      |             |                 |        |                                  |
| <DirectoryPath>     | String                                                                                                                                                                                      | The path to the SysLib Directory |      |             |                 |        |                                  |
| <b>Return Value</b> | None                                                                                                                                                                                        |                                  |      |             |                 |        |                                  |

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

|                   |                                            |
|-------------------|--------------------------------------------|
| <b>VB Syntax</b>  | SetLibraryDirectory <DirectoryPath>        |
| <b>VB Example</b> | oDesktop.SetLibraryDirectory"c:\libraries" |

## SetProjectDirectory

Sets the project directory path.

|                  |     |
|------------------|-----|
| <b>UI Access</b> | N/A |
|------------------|-----|

|                     |                                      |                |                                                                                         |
|---------------------|--------------------------------------|----------------|-----------------------------------------------------------------------------------------|
| <b>Parameters</b>   | Name<br><i>&lt;DirectoryPath&gt;</i> | Type<br>String | Description<br>The path to the project directory. This should be writeable by the user. |
| <b>Return Value</b> | None.                                |                |                                                                                         |

|                       |                                                           |
|-----------------------|-----------------------------------------------------------|
| <b>Python Syntax</b>  | SetProjectDirectory ( <i>&lt;DirectoryPath&gt;</i> )      |
| <b>Python Example</b> | <code>oDesktop.SetProjectDirectory ("c:\projects")</code> |

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

## SetRegistryFromFile

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

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

|                       |                                                               |
|-----------------------|---------------------------------------------------------------|
| <b>Python Syntax</b>  | <code>SetRegistryFromFile(&lt;filePath&gt;)</code>            |
| <b>Python Example</b> | <code>oDesktop.SetRegistryFromFile('c:/temp/test.acf')</code> |

|                   |                                                              |
|-------------------|--------------------------------------------------------------|
| <b>VB Syntax</b>  | <code>SetRegistryFromFile &lt;filePath&gt;</code>            |
| <b>VB Example</b> | <code>oDesktop.SetRegistryFromFile "c:/temp/test.acf"</code> |

## SetRegistryInt

Sets registry key to an integer value.

|                     |                              |                                                                                                                            |
|---------------------|------------------------------|----------------------------------------------------------------------------------------------------------------------------|
| <b>UI Access</b>    | N/A                          |                                                                                                                            |
| <b>Parameters</b>   | Name                         | Type                                                                                                                       |
|                     | <code>&lt;KeyName&gt;</code> | String                                                                                                                     |
| <b>Return Value</b> |                              | Success if the key is defined as an integer. Bad argument value if a key is not defined, or if the value is a text string. |

|                       |                                                                                                                           |
|-----------------------|---------------------------------------------------------------------------------------------------------------------------|
| <b>Python Syntax</b>  | <code>SetRegistryInt(&lt;KeyName&gt;, &lt;int&gt;)</code>                                                                 |
| <b>Python Example</b> | <code>oDesktop.SetRegistryInt('Desktop/Settings/ProjectOptions/HFSS 3D Layout/UpdateReportsDynamicallyOnEdits', 0)</code> |

|                   |                                                                                                                          |
|-------------------|--------------------------------------------------------------------------------------------------------------------------|
| <b>VB Syntax</b>  | <code>SetRegistryInt &lt;KeyName&gt; &lt;int&gt;</code>                                                                  |
| <b>VB Example</b> | <code>oDesktop.SetRegistryInt "Desktop/Settings/ProjectOptions/HFSS 3D Layout/UpdateReportsDynamicallyOnEdits", 0</code> |

## SetRegistryString

Sets registry key to a string value.

| <b>UI Access</b>             | N/A                                                                                                                                                                                                                                                                                                                                             |                                              |      |             |                              |        |                                            |                            |        |                                              |
|------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------|------|-------------|------------------------------|--------|--------------------------------------------|----------------------------|--------|----------------------------------------------|
| <b>Parameters</b>            | <table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td><code>&lt;KeyName&gt;</code></td><td>String</td><td>Full name of registry key, including path.</td></tr><tr><td><code>&lt;value&gt;</code></td><td>String</td><td>String value to be assigned to registry key.</td></tr></tbody></table> | Name                                         | Type | Description | <code>&lt;KeyName&gt;</code> | String | Full name of registry key, including path. | <code>&lt;value&gt;</code> | String | String value to be assigned to registry key. |
| Name                         | Type                                                                                                                                                                                                                                                                                                                                            | Description                                  |      |             |                              |        |                                            |                            |        |                                              |
| <code>&lt;KeyName&gt;</code> | String                                                                                                                                                                                                                                                                                                                                          | Full name of registry key, including path.   |      |             |                              |        |                                            |                            |        |                                              |
| <code>&lt;value&gt;</code>   | String                                                                                                                                                                                                                                                                                                                                          | String value to be assigned to registry key. |      |             |                              |        |                                            |                            |        |                                              |
| <b>Return Value</b>          | Success if the key is defined as a text string. Bad argument value if the key is not defined or requires an integer value.                                                                                                                                                                                                                      |                                              |      |             |                              |        |                                            |                            |        |                                              |

|                       |                                                                                                    |
|-----------------------|----------------------------------------------------------------------------------------------------|
| <b>Python Syntax</b>  | <code>SetRegistryString(&lt;KeyName&gt;, &lt;value&gt;)</code>                                     |
| <b>Python Example</b> | <code>oDesktop.SetRegistryString('Desktop/ActiveDSOConfigurations/HFSS 3D Layout', 'Local')</code> |

|                  |                                                                                                   |
|------------------|---------------------------------------------------------------------------------------------------|
| <b>VB Syntax</b> | <code>SetRegistryString &lt;KeyName&gt;, &lt;value&gt;</code>                                     |
| <b>VB</b>        | <code>oDesktop.SetRegistryString "Desktop/ActiveDSOConfigurations/HFSS 3D Layout", "Local"</code> |

**Example**

## SetSchematicEnvironment

Sets the schematic environment for Electronics Desktop.

|                     |                                                                                                                                                                                      |                 |                                                                                                                                                                                     |
|---------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>UI Access</b>    | Navigate to <b>Tools &gt; Options &gt; General Options &gt; General &gt; Desktop Configuration</b> . Select a schematic environment using the <b>Custom Menu Set</b> drop-down menu. |                 |                                                                                                                                                                                     |
| <b>Parameters</b>   | Name<br><schEnv>                                                                                                                                                                     | Type<br>Integer | Description<br>Desired schematic environment. Can be one of: <ul style="list-style-type: none"><li>• 0 (Circuit)</li><li>• 1 (Twin Builder)</li><li>• 2 (Maxwell Circuit)</li></ul> |
| <b>Return Value</b> | None                                                                                                                                                                                 |                 |                                                                                                                                                                                     |

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

|                   |                                                                |
|-------------------|----------------------------------------------------------------|
| <b>VB Syntax</b>  | SetSchematicEnvironment(<schEnv>)                              |
| <b>VB Example</b> | <code>Set oProject = oDesktop.SetSchematicEnvironment 2</code> |

## SetTempDirectory

Sets the temp directory path. The directory will be automatically created if it does not already exist.

|                     |                                      |                |                                                                                      |
|---------------------|--------------------------------------|----------------|--------------------------------------------------------------------------------------|
| <b>UI Access</b>    | N/A                                  |                |                                                                                      |
| <b>Parameters</b>   | Name<br><i>&lt;DirectoryPath&gt;</i> | Type<br>String | Description<br>The path to the Temp directory. This should be writeable by the user. |
| <b>Return Value</b> | None.                                |                |                                                                                      |

|                       |                                                    |
|-----------------------|----------------------------------------------------|
| <b>Python Syntax</b>  | SetTempDirectory ( <i>&lt;DirectoryPath&gt;</i> )  |
| <b>Python Example</b> | <code>oDesktop.SetTempDirectory ("c:\\tmp")</code> |

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

## ShowDockingWindow

Shows or hides a docking window.

|                   |                                         |                |                                                                                        |
|-------------------|-----------------------------------------|----------------|----------------------------------------------------------------------------------------|
| <b>UI Access</b>  | Right click docking window > Show/Hide. |                |                                                                                        |
| <b>Parameters</b> | Name<br><i>&lt;windowName&gt;</i>       | Type<br>String | Description<br>The window name (for example, "Message Manager", "Component Libraries", |

|                     |        |                                         |
|---------------------|--------|-----------------------------------------|
|                     |        | "Properties")                           |
|                     | <show> | Boolean<br>True to show; False to hide. |
| <b>Return Value</b> | None.  |                                         |

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

|                   |                                                                 |
|-------------------|-----------------------------------------------------------------|
| <b>VB Syntax</b>  | ShowDockingWindow <windowName> <show>                           |
| <b>VB Example</b> | <code>oDesktop.ShowDockingWindow "Message Manager" False</code> |

## Sleep

Suspends execution of HFSS for the specified number of milliseconds, up to 60,000 milliseconds (1 minute).

| <b>UI Access</b>     | NA                                                                                                                                                                                                                                                                 |                                                                 |      |             |                      |         |                                                                 |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------|------|-------------|----------------------|---------|-----------------------------------------------------------------|
| <b>Parameters</b>    | <table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;TimeInMilliseconds&gt;</td> <td>Integer</td> <td>The time that the execution should be suspended in milliseconds</td> </tr> </tbody> </table> | Name                                                            | Type | Description | <TimeInMilliseconds> | Integer | The time that the execution should be suspended in milliseconds |
| Name                 | Type                                                                                                                                                                                                                                                               | Description                                                     |      |             |                      |         |                                                                 |
| <TimeInMilliseconds> | Integer                                                                                                                                                                                                                                                            | The time that the execution should be suspended in milliseconds |      |             |                      |         |                                                                 |
| <b>Return Value</b>  | None                                                                                                                                                                                                                                                               |                                                                 |      |             |                      |         |                                                                 |

|                      |                              |
|----------------------|------------------------------|
| <b>Python Syntax</b> | Sleep (<TimeInMilliseconds>) |
|----------------------|------------------------------|

|                       |                                   |
|-----------------------|-----------------------------------|
| <b>Python Example</b> | <code>oDesktop.Sleep(1000)</code> |
|-----------------------|-----------------------------------|

|                   |                                               |
|-------------------|-----------------------------------------------|
| <b>VB Syntax</b>  | <code>Sleep &lt;TimeInMilliseconds&gt;</code> |
| <b>VB Example</b> | <code>oDesktop.Sleep 1000</code>              |

## StopSimulations

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

| <b>UI Access</b>           | NA                                                                                                                                                                                                                                                                                                                                   |                                                                                     |      |             |                           |         |                                                                                     |                            |  |  |
|----------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------|------|-------------|---------------------------|---------|-------------------------------------------------------------------------------------|----------------------------|--|--|
| <b>Parameters</b>          | <table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td><code>&lt;bool&gt;</code></td><td>Integer</td><td>If true, clean stop for all running and pending simulations. If false, aborts them.</td></tr><tr><td><code>&lt;Clean&gt;</code></td><td></td><td></td></tr></tbody></table> | Name                                                                                | Type | Description | <code>&lt;bool&gt;</code> | Integer | If true, clean stop for all running and pending simulations. If false, aborts them. | <code>&lt;Clean&gt;</code> |  |  |
| Name                       | Type                                                                                                                                                                                                                                                                                                                                 | Description                                                                         |      |             |                           |         |                                                                                     |                            |  |  |
| <code>&lt;bool&gt;</code>  | Integer                                                                                                                                                                                                                                                                                                                              | If true, clean stop for all running and pending simulations. If false, aborts them. |      |             |                           |         |                                                                                     |                            |  |  |
| <code>&lt;Clean&gt;</code> |                                                                                                                                                                                                                                                                                                                                      |                                                                                     |      |             |                           |         |                                                                                     |                            |  |  |
| <b>Return Value</b>        | A human readable status string specifying what happened.                                                                                                                                                                                                                                                                             |                                                                                     |      |             |                           |         |                                                                                     |                            |  |  |

|                       |                                                   |
|-----------------------|---------------------------------------------------|
| <b>Python Syntax</b>  | <code>StopSimulations(&lt;bool&gt;, clean)</code> |
| <b>Python Example</b> | <code>oDesktop.StopSimulations(bool clean)</code> |

## SubmitJob

Submits a batch job to a scheduler. When submitting the same project file multiple times, you should have the script wait for each job (or jobs for multi-step) to finish, which can be done via the monitoring functions LaunchJobMonitor() and RefreshJobMonitor(), checking the result of RefreshJobMonitor() in a loop until it returns completed ("Monitor Not Visible") status.

| UI Access           | Tools > Job Management > Submit Job.                  |        |                                                                                                                     |
|---------------------|-------------------------------------------------------|--------|---------------------------------------------------------------------------------------------------------------------|
| <b>Parameters</b>   | 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.                                                                                |
| <b>Return Value</b> | Array of job ID strings (empty if no jobs submitted). |        |                                                                                                                     |

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

|                  |                                                            |
|------------------|------------------------------------------------------------|
| <b>VB Syntax</b> | SubmitJob <settingsPath>, <projectPath>, <design>, <setup> |
|------------------|------------------------------------------------------------|

|                   |                                                                                                                                                                                                                                              |
|-------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>VB Example</b> | <pre>jobIDs = oDesktop.SubmitJob "C:\\hpc-settings\\Submit_ Job_Settings.areg", "C:\\projects\\basic.aedt" moreIDs = oDesktop.SubmitJob "C:\\hpc-settings\\Submit_ Job_Settings.areg", "C:\\projects\\basic.aedt", "Design1", "Setup1"</pre> |
|-------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

## TileWindows

Arrange all open windows in a tiled format.

| <b>UI Access</b>          | From main menu, <b>Window &gt;Tile Horizontally</b> or <b>Window &gt;Tile Vertically</b> .                                                                                                                                                                                                 |                                                                                                         |  |      |      |             |                           |         |                                                                                                         |
|---------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------|--|------|------|-------------|---------------------------|---------|---------------------------------------------------------------------------------------------------------|
| <b>Parameters</b>         | <table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td><i>&lt;TilingFlag&gt;</i></td><td>Integer</td><td><ul style="list-style-type: none"><li>• 0 – Tile vertically.</li><li>• 1 – Tile horizontally.</li></ul></td></tr></tbody></table> |                                                                                                         |  | Name | Type | Description | <i>&lt;TilingFlag&gt;</i> | Integer | <ul style="list-style-type: none"><li>• 0 – Tile vertically.</li><li>• 1 – Tile horizontally.</li></ul> |
| Name                      | Type                                                                                                                                                                                                                                                                                       | Description                                                                                             |  |      |      |             |                           |         |                                                                                                         |
| <i>&lt;TilingFlag&gt;</i> | Integer                                                                                                                                                                                                                                                                                    | <ul style="list-style-type: none"><li>• 0 – Tile vertically.</li><li>• 1 – Tile horizontally.</li></ul> |  |      |      |             |                           |         |                                                                                                         |
| <b>Return Value</b>       | None.                                                                                                                                                                                                                                                                                      |                                                                                                         |  |      |      |             |                           |         |                                                                                                         |

|                       |                                          |
|-----------------------|------------------------------------------|
| <b>Python Syntax</b>  | TileWindows( <i>&lt;TilingFlag&gt;</i> ) |
| <b>Python Example</b> | <code>oDesktop.TileWindows (0)</code>    |

|                   |                                       |
|-------------------|---------------------------------------|
| <b>VB Syntax</b>  | TileWindows <i>&lt;TilingFlag&gt;</i> |
| <b>VB Example</b> | <code>oDesktop.TileWindows 0</code>   |

## Desktop Commands For Registry Values

The Ansys Registry is stored as XML format file. By default it is located at C:\User-s\<UserName>\Documents\Ansoft\<AnsysProductName>\config\<PC\_NAME>\_user.XML. Most of the Ansys product configuration information is stored in this XML file. These methods allow you to change the product configuration in VB or Python.

For example, to set the DSO & HPC analysis setup for HFSS using a Python script:

1. Start 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.

|                     |                                                                                                                                                                          |                         |  |      |      |             |                  |        |                         |
|---------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------|--|------|------|-------------|------------------|--------|-------------------------|
| <b>UI Access</b>    | N/A                                                                                                                                                                      |                         |  |      |      |             |                  |        |                         |
| <b>Parameters</b>   | <table border="1"><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;pathToRegistry&gt;</td><td>String</td><td>Path to Registry entry.</td></tr></table> |                         |  | Name | Type | Description | <pathToRegistry> | String | Path to Registry entry. |
| Name                | Type                                                                                                                                                                     | Description             |  |      |      |             |                  |        |                         |
| <pathToRegistry>    | String                                                                                                                                                                   | Path to Registry entry. |  |      |      |             |                  |        |                         |
| <b>Return Value</b> | <p>Boolean:</p> <ul style="list-style-type: none"><li>• <b>True</b> – Key has been deleted.</li><li>• <b>False</b> – Key does not exist.</li></ul>                       |                         |  |      |      |             |                  |        |                         |

|                       |                                                            |
|-----------------------|------------------------------------------------------------|
| <b>Python Syntax</b>  | DeleteRegistryEntry(<pathToRegistry>)                      |
| <b>Python Example</b> | res = oDesktop.DeleteRegistryEntry ("Desktop/ColorScheme") |

|                   |                                                          |
|-------------------|----------------------------------------------------------|
| <b>VB Syntax</b>  | DeleteRegistryEntry <pathToRegistry>                     |
| <b>VB Example</b> | res = oDesktop.DeleteRegistryEntry "Desktop/ColorScheme" |

**DoesRegistryValueExist**

Determines whether a registry value exists.

| <b>UI Access</b>    | N/A                                                                                                                                                                                                                               |                                            |      |             |           |        |                                            |
|---------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------|------|-------------|-----------|--------|--------------------------------------------|
| <b>Parameters</b>   | <table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;KeyName&gt;</td> <td>String</td> <td>Full name of registry key, including path.</td> </tr> </tbody> </table> | Name                                       | Type | Description | <KeyName> | String | Full name of registry key, including path. |
| Name                | Type                                                                                                                                                                                                                              | Description                                |      |             |           |        |                                            |
| <KeyName>           | String                                                                                                                                                                                                                            | Full name of registry key, including path. |      |             |           |        |                                            |
| <b>Return Value</b> | <p>Boolean:</p> <ul style="list-style-type: none"> <li>• <b>True</b> – Key exists.</li> <li>• <b>False</b> – Key does not exist.</li> </ul>                                                                                       |                                            |      |             |           |        |                                            |

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

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

## GetRegistryInt

Obtains registry key integer value.

| <b>UI Access</b>  | N/A                                                                                                      |             |      |             |
|-------------------|----------------------------------------------------------------------------------------------------------|-------------|------|-------------|
| <b>Parameters</b> | <table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> </table> | Name        | Type | Description |
| Name              | Type                                                                                                     | Description |      |             |

|                     |                                                                                                                                   |        |                                            |
|---------------------|-----------------------------------------------------------------------------------------------------------------------------------|--------|--------------------------------------------|
|                     | <code>&lt;KeyName&gt;</code>                                                                                                      | String | Full name of registry key, including path. |
| <b>Return Value</b> | Integer if the integer value is found. Return as Bad-Argument-Value if registry key does not exist or it is not an integer value. |        |                                            |

|                       |                                                                                                                               |
|-----------------------|-------------------------------------------------------------------------------------------------------------------------------|
| <b>Python Syntax</b>  | <code>GetRegistryInt(&lt;KeyName&gt;)</code>                                                                                  |
| <b>Python Example</b> | <code>num = oDesktop.GetRegistryInt ('Desktop/Settings/ProjectOptions/HFSS 3D Layout/UpdateReportsDynamicallyOnEdits')</code> |

|                   |                                                                                                                               |
|-------------------|-------------------------------------------------------------------------------------------------------------------------------|
| <b>VB Syntax</b>  | <code>GetRegistryInt(&lt;KeyName&gt;)</code>                                                                                  |
| <b>VB Example</b> | <code>num = oDesktop.GetRegistryInt ("Desktop/Settings/ProjectOptions/HFSS 3D Layout/UpdateReportsDynamicallyOnEdits")</code> |

## GetRegistryString

Obtains registry key string value.

| <b>UI Access</b>             | N/A                                                                                                                                                                                                                             |                                            |  |      |      |             |                              |        |                                            |
|------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------|--|------|------|-------------|------------------------------|--------|--------------------------------------------|
| <b>Parameters</b>            | <table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td><code>&lt;KeyName&gt;</code></td><td>String</td><td>Full name of registry key, including path.</td></tr></tbody></table> |                                            |  | Name | Type | Description | <code>&lt;KeyName&gt;</code> | String | Full name of registry key, including path. |
| Name                         | Type                                                                                                                                                                                                                            | Description                                |  |      |      |             |                              |        |                                            |
| <code>&lt;KeyName&gt;</code> | String                                                                                                                                                                                                                          | Full name of registry key, including path. |  |      |      |             |                              |        |                                            |
| <b>Return Value</b>          | String if the string value is found. Return as Bad-Argument-Value if registry key does not exist or it is not a string                                                                                                          |                                            |  |      |      |             |                              |        |                                            |

|  |        |
|--|--------|
|  | value. |
|--|--------|

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

|                   |                                                                                           |
|-------------------|-------------------------------------------------------------------------------------------|
| <b>VB Syntax</b>  | GetRegistryString(<KeyName>)                                                              |
| <b>VB Example</b> | 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.

| <b>UI Access</b>    | N/A                                                                                                                                                                                                                      |                                  |      |      |             |            |        |                                  |
|---------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------|------|------|-------------|------------|--------|----------------------------------|
| <b>Parameters</b>   | <table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;filePath&gt;</td> <td>String</td> <td>Full file path of registry file.</td> </tr> </tbody> </table> |                                  | Name | Type | Description | <filePath> | String | Full file path of registry file. |
| Name                | Type                                                                                                                                                                                                                     | Description                      |      |      |             |            |        |                                  |
| <filePath>          | String                                                                                                                                                                                                                   | Full file path of registry file. |      |      |             |            |        |                                  |
| <b>Return Value</b> | Success if configuration is imported. Bad argument value if the file is not found or does not contain valid analysis configuration data.                                                                                 |                                  |      |      |             |            |        |                                  |

|                      |                                 |
|----------------------|---------------------------------|
| <b>Python Syntax</b> | SetRegistryFromFile(<filePath>) |
|----------------------|---------------------------------|

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

|                   |                                                              |
|-------------------|--------------------------------------------------------------|
| <b>VB Syntax</b>  | <code>SetRegistryFromFile &lt;filePath&gt;</code>            |
| <b>VB Example</b> | <code>oDesktop.SetRegistryFromFile "c:/temp/test.acf"</code> |

## SetRegistryInt

Sets registry key to an integer value.

| <b>UI Access</b>             | N/A                                                                                                                                                                                                                                                                                                                                             |                                               |      |             |                              |        |                                            |                          |         |                                               |
|------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------|------|-------------|------------------------------|--------|--------------------------------------------|--------------------------|---------|-----------------------------------------------|
| <b>Parameters</b>            | <table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td><code>&lt;KeyName&gt;</code></td><td>String</td><td>Full name of registry key, including path.</td></tr><tr><td><code>&lt;int&gt;</code></td><td>Integer</td><td>Integer value to be assigned to registry key.</td></tr></tbody></table> | Name                                          | Type | Description | <code>&lt;KeyName&gt;</code> | String | Full name of registry key, including path. | <code>&lt;int&gt;</code> | Integer | Integer value to be assigned to registry key. |
| Name                         | Type                                                                                                                                                                                                                                                                                                                                            | Description                                   |      |             |                              |        |                                            |                          |         |                                               |
| <code>&lt;KeyName&gt;</code> | String                                                                                                                                                                                                                                                                                                                                          | Full name of registry key, including path.    |      |             |                              |        |                                            |                          |         |                                               |
| <code>&lt;int&gt;</code>     | Integer                                                                                                                                                                                                                                                                                                                                         | Integer value to be assigned to registry key. |      |             |                              |        |                                            |                          |         |                                               |
| <b>Return Value</b>          | Success if the key is defined as an integer. Bad argument value if a key is not defined, or if the value is a text string.                                                                                                                                                                                                                      |                                               |      |             |                              |        |                                            |                          |         |                                               |

|                       |                                                                                                                           |
|-----------------------|---------------------------------------------------------------------------------------------------------------------------|
| <b>Python Syntax</b>  | <code>SetRegistryInt(&lt;KeyName&gt;, &lt;int&gt;)</code>                                                                 |
| <b>Python Example</b> | <code>oDesktop.SetRegistryInt('Desktop/Settings/ProjectOptions/HFSS 3D Layout/UpdateReportsDynamicallyOnEdits', 0)</code> |

|                   |                                                                                                                          |
|-------------------|--------------------------------------------------------------------------------------------------------------------------|
| <b>VB Syntax</b>  | <code>SetRegistryInt &lt;KeyName&gt; &lt;int&gt;</code>                                                                  |
| <b>VB Example</b> | <code>oDesktop.SetRegistryInt "Desktop/Settings/ProjectOptions/HFSS 3D Layout/UpdateReportsDynamicallyOnEdits", 0</code> |

## SetRegistryString

Sets registry key to a string value.

| <b>UI Access</b>             | N/A                                                                                                                                                                                                                                                                                                                                                                 |                                              |      |             |                              |        |                                            |                            |        |                                              |
|------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------|------|-------------|------------------------------|--------|--------------------------------------------|----------------------------|--------|----------------------------------------------|
| <b>Parameters</b>            | <table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><code>&lt;KeyName&gt;</code></td> <td>String</td> <td>Full name of registry key, including path.</td> </tr> <tr> <td><code>&lt;value&gt;</code></td> <td>String</td> <td>String value to be assigned to registry key.</td> </tr> </tbody> </table> | Name                                         | Type | Description | <code>&lt;KeyName&gt;</code> | String | Full name of registry key, including path. | <code>&lt;value&gt;</code> | String | String value to be assigned to registry key. |
| Name                         | Type                                                                                                                                                                                                                                                                                                                                                                | Description                                  |      |             |                              |        |                                            |                            |        |                                              |
| <code>&lt;KeyName&gt;</code> | String                                                                                                                                                                                                                                                                                                                                                              | Full name of registry key, including path.   |      |             |                              |        |                                            |                            |        |                                              |
| <code>&lt;value&gt;</code>   | String                                                                                                                                                                                                                                                                                                                                                              | String value to be assigned to registry key. |      |             |                              |        |                                            |                            |        |                                              |
| <b>Return Value</b>          | Success if the key is defined as a text string. Bad argument value if the key is not defined or requires an integer value.                                                                                                                                                                                                                                          |                                              |      |             |                              |        |                                            |                            |        |                                              |

|                       |                                                                                                    |
|-----------------------|----------------------------------------------------------------------------------------------------|
| <b>Python Syntax</b>  | <code>SetRegistryString(&lt;KeyName&gt;, &lt;value&gt;)</code>                                     |
| <b>Python Example</b> | <code>oDesktop.SetRegistryString('Desktop/ActiveDSOConfigurations/HFSS 3D Layout', 'Local')</code> |

|                   |                                                                                                   |
|-------------------|---------------------------------------------------------------------------------------------------|
| <b>VB Syntax</b>  | <code>SetRegistryString &lt;KeyName&gt;, &lt;value&gt;</code>                                     |
| <b>VB Example</b> | <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](#).

| <b>UI Access</b>    | <b>File &gt; Import &gt; ANF.</b>                                                                                                                                                                                 |                        |      |             |               |        |                        |
|---------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------|------|-------------|---------------|--------|------------------------|
| <b>Parameters</b>   | <table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;ANFfilename&gt;</td> <td>String</td> <td>Full path of ANF file.</td> </tr> </tbody> </table> | Name                   | Type | Description | <ANFfilename> | String | Full path of ANF file. |
| Name                | Type                                                                                                                                                                                                              | Description            |      |             |               |        |                        |
| <ANFfilename>       | String                                                                                                                                                                                                            | Full path of ANF file. |      |             |               |        |                        |
| <b>Return Value</b> | None.                                                                                                                                                                                                             |                        |      |             |               |        |                        |

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

|                   |                                                                                                                                |
|-------------------|--------------------------------------------------------------------------------------------------------------------------------|
| <b>VB Syntax</b>  | ImportANF <ANFfilename>                                                                                                        |
| <b>VB Example</b> | <pre> Dim oAnsoftApp Dim oDesktop Dim oProject Dim oDesign Dim oEditor Dim oModule Dim oProjects Dim omachine Dim oTool </pre> |

```

Set oAnsoftApp = CreateObject("Ansoft.ElectronicsDesktop")
Set oDesktop = oAnsoftApp.GetAppDesktop()
oDesktop.RestoreWindow
Set oTool = oDesktop.GetTool("ImportExport")
oTool.ImportANF("%UserProfile%\Documents\HFSS Examples\package.anf")

```

## ImportANFv2

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

| UI Access           | File > Import > ANF. |        |                                                   |
|---------------------|----------------------|--------|---------------------------------------------------|
| <b>Parameters</b>   | 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.                    |
| <b>Return Value</b> | <cmpFileName>        | String | Full path of CMP file. Pass empty string if none. |
|                     | None.                |        |                                                   |

|                       |                                                                                                                                                                                                                               |
|-----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Python Syntax</b>  | ImportANFv2 (<ANFfilename>, <outputPathName>, <controlFileName>, <cmpFileName>)                                                                                                                                               |
| <b>Python Example</b> | <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", </pre> |

|  |                                                    |
|--|----------------------------------------------------|
|  | "C:/Users/jdoe/Documents/Ansoft/my_model.xml", "") |
|--|----------------------------------------------------|

| VB Syntax  | ImportANFv2 <ANFfilename>, <outputPathName>, <controlFileName>, <cmpFileName>                                                                                                                                                                                                                                                                                                                                                                                                        |
|------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 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.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> |

## ImportAutoCAD

Imports an AutoCAD file into a new project.

| UI Access         | File > Import > AutoCAD. |                                                           |                                                |
|-------------------|--------------------------|-----------------------------------------------------------|------------------------------------------------|
| Parameters        | Name                     | Type                                                      | Description                                    |
|                   | <dxFileName>             | 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  | ImportAutoCAD (<dxFileName>, <outputPathFileName>, <controlFileName>)                                                                                       |
| 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  | ImportAutoCAD <dxFileName>, <outputPathFileName>, <controlFileName>                                                                                        |
| 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.

| <b>UI Access</b>    | <b>File &gt; Import &gt; AWRMicrowaveOffice.</b>                                                                                                                                                                                                                                                                                                                                                        |                           |      |             |               |        |                        |                  |        |                           |               |        |                        |
|---------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------|------|-------------|---------------|--------|------------------------|------------------|--------|---------------------------|---------------|--------|------------------------|
| <b>Parameters</b>   | <table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;xmlFileName&gt;</td> <td>String</td> <td>Full path of XML file.</td> </tr> <tr> <td>&lt;outputPathName&gt;</td> <td>String</td> <td>Full path of output file.</td> </tr> <tr> <td>&lt;logFileName&gt;</td> <td>String</td> <td>Full path of log file.</td> </tr> </tbody> </table> | Name                      | Type | Description | <xmlFileName> | String | Full path of XML file. | <outputPathName> | String | Full path of output file. | <logFileName> | String | Full path of log file. |
| Name                | Type                                                                                                                                                                                                                                                                                                                                                                                                    | Description               |      |             |               |        |                        |                  |        |                           |               |        |                        |
| <xmlFileName>       | String                                                                                                                                                                                                                                                                                                                                                                                                  | Full path of XML file.    |      |             |               |        |                        |                  |        |                           |               |        |                        |
| <outputPathName>    | String                                                                                                                                                                                                                                                                                                                                                                                                  | Full path of output file. |      |             |               |        |                        |                  |        |                           |               |        |                        |
| <logFileName>       | String                                                                                                                                                                                                                                                                                                                                                                                                  | Full path of log file.    |      |             |               |        |                        |                  |        |                           |               |        |                        |
| <b>Return Value</b> | None.                                                                                                                                                                                                                                                                                                                                                                                                   |                           |      |             |               |        |                        |                  |        |                           |               |        |                        |

|                       |                                                                                                                                                                                                       |
|-----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Python Syntax</b>  | ImportAWRMicrowaveOffice (<xmlFileName>, <outputPathName>, <logFileName>)                                                                                                                             |
| <b>Python Example</b> | <pre> oDesktop.RestoreWindow()  Set oTool = oDesktop.GetTool('ImportExport')  oTool.ImportAWRMicrowaveOffice('C:/MyFiles/package4.xml', 'C:/MyFiles/package4.aedb', 'C:/MyFiles/package4.log') </pre> |

|                   |                                                                               |
|-------------------|-------------------------------------------------------------------------------|
| <b>VB Syntax</b>  | ImportAWRMicrowaveOffice <xmlFileName>, <outputPathName>, <logFileName>       |
| <b>VB Example</b> | <pre> Dim oAnsoftApp Dim oDesktop Dim oProject Dim oDesign Dim oEditor </pre> |

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

## ImportEDB

Imports an EDB file into a new project.

|                     |                                                                                                                                                                      |                        |      |             |               |        |                        |
|---------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------|------|-------------|---------------|--------|------------------------|
| <b>UI Access</b>    | <b>File &gt; Import &gt; EDB.</b>                                                                                                                                    |                        |      |             |               |        |                        |
| <b>Parameters</b>   | <table border="1"><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;edbFileName&gt;</td><td>String</td><td>Full path of EDB file.</td></tr></table> | Name                   | Type | Description | <edbFileName> | String | Full path of EDB file. |
| Name                | Type                                                                                                                                                                 | Description            |      |             |               |        |                        |
| <edbFileName>       | String                                                                                                                                                               | Full path of EDB file. |      |             |               |        |                        |
| <b>Return Value</b> | None.                                                                                                                                                                |                        |      |             |               |        |                        |

|                      |                           |
|----------------------|---------------------------|
| <b>Python Syntax</b> | ImportEDB (<edbFileName>) |
|----------------------|---------------------------|

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

|                   |                                                                                                                     |
|-------------------|---------------------------------------------------------------------------------------------------------------------|
| <b>VB Syntax</b>  | ImportEDB <edbFileName>                                                                                             |
| <b>VB Example</b> | <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.

|                     |                                                           |        |                                         |
|---------------------|-----------------------------------------------------------|--------|-----------------------------------------|
| <b>UI Access</b>    | <b>File &gt; Import &gt; Cadence APD / Allegro / SiP.</b> |        |                                         |
| <b>Parameters</b>   | Name                                                      | Type   | Description                             |
|                     | <extractaFileName>                                        | String | Full path of Extracta file.             |
|                     | <outputPathName>                                          | String | Full path of output file to be created. |
| <b>Return Value</b> | None.                                                     |        |                                         |

|               |                                                                          |
|---------------|--------------------------------------------------------------------------|
| <b>Python</b> | ImportExtracta (<extractaFileName>, <outputPathName>, <controlFileName>) |
|---------------|--------------------------------------------------------------------------|

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

|                   |                                                                                                                                               |
|-------------------|-----------------------------------------------------------------------------------------------------------------------------------------------|
| <b>VB Syntax</b>  | ImportExtracta <extractaFileName>, <outputPathName>, <controlFileName>                                                                        |
| <b>VB Example</b> | <pre>Set oTool = oDesktop.GetTool("ImportExport") oTool.ImportExtracta "C:/MyFiles/projectforimport.brd", "C:/MyFiles/project.edb", "")</pre> |

## ImportGDSII

Imports a GDSII file into a new project.

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

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

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

## ImportGerber

Imports a GERBER file into a new project.

|                     |                                      |        |                                                                     |
|---------------------|--------------------------------------|--------|---------------------------------------------------------------------|
| <b>UI Access</b>    | <b>File &gt; Import &gt; GERBER.</b> |        |                                                                     |
| <b>Parameters</b>   | 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. |
| <b>Return Value</b> | None.                                |        |                                                                     |

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

|                   |                                                                                                                                                    |
|-------------------|----------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>VB Syntax</b>  | ImportGerber <gerberFileName>, <outputPathName>, <controlFileName>                                                                                 |
| <b>VB Example</b> | <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](#).

|                   |                            |        |                                                        |
|-------------------|----------------------------|--------|--------------------------------------------------------|
| <b>UI Access</b>  | placeholder                |        |                                                        |
| <b>Parameters</b> | 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                                          |

|                     |                          |        |                                       |
|---------------------|--------------------------|--------|---------------------------------------|
|                     | <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                         |
|                     | <ReplaceDevices>         | bool   | True or False                         |
| <b>Return Value</b> | None.                    |        |                                       |

|                       |                                                                                                                                                                                                                                                                                                                                                                                                                    |
|-----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Python Syntax</b>  | 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>) |
| <b>Python Example</b> | oDesign.ImportIDF(<br>[<br>]                                                                                                                                                                                                                                                                                                                                                                                       |

```
        "NAME:Settings",
        "Board:=" , "C:\\\\Users\\\\Model Files\\\\brd_board.emn",
        "Library:=" , "C:\\\\Users\\\\Model Files\\\\brd_board.emp",
        "Control:=" , "C:\\\\Users\\\\Model Files\\\\brd_board.xml",
        "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:=" , True,
        "ModelDeviceAsRect:=" , False,
        "Cutoff:=" , False,
        "IncludeDrilledHoles:=" , True,
        "ReplaceDevices:=" , False
    )
)
```

|                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|-------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>VB Syntax</b>  | ImportIDF < <i>idfFileName</i> >, < <i>outputPathName</i> >, < <i>controlFileName</i> >, < <i>libFileName</i> >                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| <b>VB Example</b> | <pre> oDesign.ImportIDF Array("NAME:Settings", "Board:=", _ "C:\Users\Model Files" "\brd_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](#).

| <b>UI Access</b>       | <b>File &gt; Import &gt; IDF.</b> Enable <b>Merge with existing</b> and select a project/design.                                                                                                                                                                                                                                                                                                                    |                                                                 |      |             |                        |        |                           |                        |        |                                                                 |
|------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------|------|-------------|------------------------|--------|---------------------------|------------------------|--------|-----------------------------------------------------------------|
| <b>Parameters</b>      | <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;<i>idfFileName</i>&gt;</td> <td>String</td> <td>Full path of GERBER file.</td> </tr> <tr> <td>&lt;<i>libFileName</i>&gt;</td> <td>String</td> <td>Optional. Full path of library file. Pass empty string if none.</td> </tr> </tbody> </table> | Name                                                            | Type | Description | < <i>idfFileName</i> > | String | Full path of GERBER file. | < <i>libFileName</i> > | String | Optional. Full path of library file. Pass empty string if none. |
| Name                   | Type                                                                                                                                                                                                                                                                                                                                                                                                                | Description                                                     |      |             |                        |        |                           |                        |        |                                                                 |
| < <i>idfFileName</i> > | String                                                                                                                                                                                                                                                                                                                                                                                                              | Full path of GERBER file.                                       |      |             |                        |        |                           |                        |        |                                                                 |
| < <i>libFileName</i> > | String                                                                                                                                                                                                                                                                                                                                                                                                              | Optional. Full path of library file. Pass empty string if none. |      |             |                        |        |                           |                        |        |                                                                 |

|                     |                                      |        |                                                                     |
|---------------------|--------------------------------------|--------|---------------------------------------------------------------------|
|                     | <code>&lt;controlFileName&gt;</code> | String | Optional. Full path of XML control file. Pass empty string if none. |
|                     | <code>&lt;project&gt;</code>         | String | Name of project to merge with.                                      |
|                     | <code>&lt;design&gt;</code>          | String | Name of design to merge with.                                       |
| <b>Return Value</b> | None.                                |        |                                                                     |

|                       |                                                                                                                                                                                                        |
|-----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Python Syntax</b>  | <code>ImportIDFandMerge(&lt;idfFileName&gt;, &lt;libPathName&gt;, &lt;controlFileName&gt;, &lt;project&gt;, &lt;design&gt;)</code>                                                                     |
| <b>Python Example</b> | <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> |

|                   |                                                                                                                                                                          |
|-------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>VB Syntax</b>  | <code>ImportIDFandMerge&lt;idfFileName&gt;, &lt;libPathName&gt;, &lt;controlFileName&gt;, &lt;project&gt;, &lt;design&gt;</code>                                         |
| <b>VB Example</b> | <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.

|                  |                               |
|------------------|-------------------------------|
| <b>UI Access</b> | <b>Icepak &gt; Import IDX</b> |
|------------------|-------------------------------|

|              | Name                       | Type   | Description                                     |
|--------------|----------------------------|--------|-------------------------------------------------|
| Parameters   | <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(
    [
        "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
    ]
)
```

|                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|-------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>VB Syn-tax</b> | ImportIDX (<NAME>, <Board>, <Library>, <Control>, <Filters>, <CreateFilteredAsNonModel>, <NAME>, <NAME>, <HighSurfThickness>, <LowSurfThickness>, <InternalLayerThickness>, <NumInternalLayer>, <HighSurfaceCopper>, <LowSurfaceCopper>, <InternalLayerCopper>, <TraceMaterial>, <SubstrateMaterial>, <CreateBoard>, <ModelBoardAsRect>, <ModelDeviceAsRect>, <Cutoff>, <ReplaceDevices>)                                                                                                                                                                                                                                                                                                                                   |
| <b>VB Example</b> | <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                                                                                                                                                                            | <table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;ipcFileName&gt;</td> <td>String</td> <td>Full path of IPC2581 file.</td> </tr> <tr> <td>&lt;outputPathName&gt;</td> <td>String</td> <td>Full path of EDB file to create during import.</td> </tr> </tbody> </table> |                                                                     |                   | Name   | Type                                                                | Description | <ipcFileName> | String | Full path of IPC2581 file. | <outputPathName> | String | Full path of EDB file to create during import. |
| Name                                                                                                                                                                                  | Type                                                                                                                                                                                                                                                                                                                                     | Description                                                         |                   |        |                                                                     |             |               |        |                            |                  |        |                                                |
| <ipcFileName>                                                                                                                                                                         | String                                                                                                                                                                                                                                                                                                                                   | Full path of IPC2581 file.                                          |                   |        |                                                                     |             |               |        |                            |                  |        |                                                |
| <outputPathName>                                                                                                                                                                      | String                                                                                                                                                                                                                                                                                                                                   | Full path of EDB file to create during import.                      |                   |        |                                                                     |             |               |        |                            |                  |        |                                                |
| <table border="1"> <tbody> <tr> <td>&lt;controlFileName&gt;</td> <td>String</td> <td>Optional. Full path of XML control file. Pass empty string if none.</td> </tr> </tbody> </table> |                                                                                                                                                                                                                                                                                                                                          |                                                                     | <controlFileName> | String | Optional. Full path of XML control file. Pass empty string if none. |             |               |        |                            |                  |        |                                                |
| <controlFileName>                                                                                                                                                                     | String                                                                                                                                                                                                                                                                                                                                   | Optional. Full path of XML control file. Pass empty string if none. |                   |        |                                                                     |             |               |        |                            |                  |        |                                                |
| Return Value                                                                                                                                                                          |                                                                                                                                                                                                                                                                                                                                          |                                                                     |                   |        |                                                                     |             |               |        |                            |                  |        |                                                |

|                       |                                                                                                                                                                                         |
|-----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Python Syntax</b>  | ImportIPC(<ipcFileName>, <outputPathName>, <controlFileName>)                                                                                                                           |
| <b>Python Example</b> | <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> |

|                   |                                                                                                                                         |
|-------------------|-----------------------------------------------------------------------------------------------------------------------------------------|
| <b>VB Syntax</b>  | ImportIPC <ipcFileName>, <outputPathName>, <controlFileName>                                                                            |
| <b>VB Example</b> | <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.

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

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

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

## ImportXFL

Imports an XFL file into a new project.

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

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

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

## 4 - Running Instances Manager Script Commands

The Running Instances Manager is a scripting object that lets you identify and connect to all running instances of Electronics Desktop. oDesktop objects that are returned provide full scripting functionality. Running Instances Manager commands should be executed by the oDesktop object. For example:

```
Set oRunningInstances = oDesktop.GetRunningInstancesMgr()
```

[GetAllRunningInstances](#)

[GetRunningInstanceByProcessID](#)

[GetRunningInstanceByProject](#)

### GetAllRunningInstances

Returns a list of running instances of Ansys Electronics Desktop.

|                     |                                                               |
|---------------------|---------------------------------------------------------------|
| <b>UI Access</b>    | N/A                                                           |
| <b>Parameters</b>   | None.                                                         |
| <b>Return Value</b> | Array containing list of Ansys Electronics Desktop instances. |

|                       |                                                  |
|-----------------------|--------------------------------------------------|
| <b>Python Syntax</b>  | GetAllRunningInstances()                         |
| <b>Python Example</b> | obj = oRunningInstances.GetAllRunningInstances() |

|                   |                                                      |
|-------------------|------------------------------------------------------|
| <b>VB Syntax</b>  | GetAllRunningInstances                               |
| <b>VB Example</b> | set obj = oRunningInstances.GetAllRunningInstances() |

## GetRunningInstanceByProcessID

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

|                     |                                                                                                                                                         |             |  |      |      |             |             |         |            |
|---------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------|-------------|--|------|------|-------------|-------------|---------|------------|
| <b>UI Access</b>    | N/A                                                                                                                                                     |             |  |      |      |             |             |         |            |
| <b>Parameters</b>   | <table border="1"><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;processID&gt;</td><td>Integer</td><td>Process ID</td></tr></table> |             |  | Name | Type | Description | <processID> | Integer | Process ID |
| Name                | Type                                                                                                                                                    | Description |  |      |      |             |             |         |            |
| <processID>         | Integer                                                                                                                                                 | Process ID  |  |      |      |             |             |         |            |
| <b>Return Value</b> | String of the returned instance.                                                                                                                        |             |  |      |      |             |             |         |            |

|                       |                                                              |
|-----------------------|--------------------------------------------------------------|
| <b>Python Syntax</b>  | GetRunningInstanceByProcessID(<processID>)                   |
| <b>Python Example</b> | obj = oRunningInstances.GetRunningInstanceByProcessID(12345) |

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

## GetRunningInstancesMgr

Returns the object of the Running Instances Manager.

|                   |                                                                                                                           |             |  |      |      |             |      |  |  |
|-------------------|---------------------------------------------------------------------------------------------------------------------------|-------------|--|------|------|-------------|------|--|--|
| <b>UI Access</b>  | N/A                                                                                                                       |             |  |      |      |             |      |  |  |
| <b>Parameters</b> | <table border="1"><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>None</td><td></td><td></td></tr></table> |             |  | Name | Type | Description | None |  |  |
| Name              | Type                                                                                                                      | Description |  |      |      |             |      |  |  |
| None              |                                                                                                                           |             |  |      |      |             |      |  |  |

|                     |                                            |
|---------------------|--------------------------------------------|
| <b>Return Value</b> | Object<br>Running instances manager object |
|---------------------|--------------------------------------------|

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

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

This page intentionally  
left blank.

## 5 - Project Object Script Commands

Project commands should be executed by the oProject object. One example of accessing this object is:

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

[AddDataset](#)

[AddMaterial](#)

[AnalyzeAll](#)

[ChangeProperty](#)

[ClearMessages](#)

[CloneMaterial](#)

[Close](#)

[CopyDesign](#)

[CutDesign](#)

[DeleteDataset](#)

[DeleteDesign](#)

[DeleteToolObject](#)

[EditDataset](#)

[EditMaterial](#)

[ExportDataset](#)

[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](#)  
[ImportDataset](#)  
[InsertDesign](#)  
[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<br>Array("NAME:<DatasetName>",<br>Array("NAME:Coordinates", <CoordinateArray>,<br><CoordinateArray>, ...)) |
|              | <<br><i>DatasetdataArray</i> > | Array  |                                                                                                                        |
|              | < <i>DatasetName</i> >         | String | Name of the dataset.                                                                                                   |
| Return Value | None.                          |        |                                                                                                                        |

|                |                                        |
|----------------|----------------------------------------|
| Python Syntax  | AddDataset < <i>DatasetdataArray</i> > |
| Python Example | oProject.AddDataset (<br>[             |

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

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

|                   |                                                                                                                                                                                                          |
|-------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>VB Syntax</b>  | AddDataset <DatasetdataArray>                                                                                                                                                                            |
| <b>VB Example</b> | <pre>oProject.AddDatasetArray("NAME:ds1",<br/>    Array("NAME:Coordinates",<br/>        Array("NAME:Coordinate", "X:=", 1, "Y:=", 2,<br/>            Array("NAME:Coordinate", "X:=", 3, "Y:=", 4),</pre> |

```

        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)))
)

```

## AddMaterial

Adds a local material.

| UI Access         | Add Material in the material editor. |       |                                                                                            |
|-------------------|--------------------------------------|-------|--------------------------------------------------------------------------------------------|
| <b>Parameters</b> | Name                                 | Type  | Description                                                                                |
|                   | <MaterialParams>                     | Array | [<br>"NAME: <name of the material to be added>",<br><MatProperty>, <MatProperty>, ...<br>] |
|                   | <MatProperty>                        | Array | For simple material:<br>"<PropertyName>:=", <value>"                                       |

|                |        |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|----------------|--------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                |        | <p>For anisotropic material:</p> <pre>[<br/>  "NAME:&lt;PropertyName&gt;",<br/>  "property_type:=", "AnisoProperty",<br/>  "unit:=", &lt;Unit&gt;,<br/>  "component1:=", &lt;value&gt;,<br/>  "component2:=", &lt;value&gt;,<br/>  "component3:=", &lt;value&gt;))<br/>]</pre>                                                                                                                                                                                                                                                                                                                                              |
| <PropertyName> | String | <p>Should be one of the following (depending on the material, design, and solution types):</p> <p>Electromagnetic (Maxwell-exclusive material properties omitted, see Maxwell Scripting help):</p> <pre>"permittivity", "permeability", "conductivity", "dielectric_loss_tangent",<br/>"magnetic_loss_tangent", "electric_coercivity", "magnetic_coercivity",<br/>"saturation_mag", "lande_g_factor", "delta_H", "delta_h_freq",<br/>"mass_density"</pre> <p>Thermal (including solids, Icepak fluid flow, and Mechanical rotating fluid modeling):</p> <pre>"thermal_conductivity", "mass_density", "specific_heat",</pre> |

|                     |        |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|---------------------|--------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                     |        | <pre>"thermal_expansion_coefficient", "thermal_material_type", "viscosity", "diffusivity", "molecular_mass", "clarity_type"  Structural:  "mass_density", "youngs_modulus", "poissons_ratio", "thermal_expansion_coefficient"</pre>                                                                                                                                                                                                                                                                                                                                                      |
| <Unit>              | String | <p>Possible values (Maxwell-exclusive properties omitted, see Maxwell Scripting Help; other missing entries are unitless):</p> <p>conductivity: "siemens/m"</p> <p>saturation_mag: "uTesla", "mTesla", "tesla", "kTesla", "uGauss", "mGauss", "gauss", "kGauss"</p> <p>delta_H: "A_per_meter", "kA_per_meter", "Oe", "kOe"</p> <p>delta_h_frequency: "Hz", "kHz", "MHz", "GHz", "THz", "rps", "per_sec"</p> <p>mass_density: "kg/m^3"</p> <p>thermal_conductivity: "W/m-C"</p> <p>specific_heat: "J/kg-C"</p> <p>youngs_modulus: "N/m^2"</p> <p>thermal_expansion_coefficient: "1/C"</p> |
| <b>Return Value</b> | None   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |

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

|                   |                                                                             |
|-------------------|-----------------------------------------------------------------------------|
| <b>VB Syntax</b>  | AddMaterial Array("NAME:<MaterialName>","<MatProperty>, <MatProperty>, ...) |
| <b>VB Example</b> | <pre>oDefinitionManager.AddMaterial Array(</pre>                            |

```

"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")

```

## AnalyzeAll [project]

Runs the project-level script command from the script, which simulates all solution setups and Optimetrics setups for all design instances in the project. The UI waits until simulation is finished before continuing with the script.

|                     |                        |
|---------------------|------------------------|
| <b>UI Access</b>    | Project > Analyze All. |
| <b>Parameters</b>   | None.                  |
| <b>Return Value</b> | None.                  |

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

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

## ChangeProperty (3D Modeler Layout Component Visualisation)

Changes the visualization properties of a a 3D Layout Component object HFSS Modeler window in the history tree.

|              |                                                                          |               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|--------------|--------------------------------------------------------------------------|---------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| UI Access    | Right-click an object in the History Tree and select <b>Properties</b> . |               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| Parameters   | Name<br><code>&lt;propertyArgs&gt;</code>                                | Type<br>Array | Description<br>Structured array. The properties vary depending on the object. Due to the number of potential configurations, it is recommended that you generate this script using the UI's <b>Automation</b> tab.<br><br>Below is an example script to set the visualization attributes of a layout component. The arguments for "DisplayMode" are 0 (for layout mode), 1 (for net mode), and 2 (for object mode). The three arguments for each object in "ObjectAttributesInLayerMode", "ObjectAttributesInNetMode", or "ObjectAttributesInObjectMode" represents the option in the "Show", "Wire Frame", "and "Transparency" column of the "Object Attributes" dialog. |
| Return Value | None.                                                                    |               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |

|               |                                                     |
|---------------|-----------------------------------------------------|
| Python Syntax | ChangeProperty( <code>&lt;propertyArgs&gt;</code> ) |
|---------------|-----------------------------------------------------|

**Python Example**

```
oProject = oDesktop.SetActiveProject("MyTest")
oDesign = oProject.SetActiveDesign("HFSSDesign1")
oEditor = oDesign.SetActiveEditor("3D Modeler")
oEditor.ChangeProperty(
    [
        "NAME:AllTabs",
        [
            [
                "NAME:Visualization",
                [
                    [
                        "NAME:PropServers",
                        "LC1_1"
                    ],
                    [
                        [
                            "NAME:ChangedProps",
                            [
                                [
                                    "NAME>Show Layout",
                                    "Value:=" , True
                                ]
                            ]
                        ]
                    ]
                ]
            ]
        ]
    ]
)
```

```
oEditor.ChangeProperty(
    [
        "NAME:AllTabs",
        [
            "NAME:Visualization",
            [
                "NAME:PropServers",
                "LC1_1"
            ],
            [
                "NAME:ChangedProps",
                [
                    "NAME:Object Attributes",
                    "ShowDielectric:=" , False,
                    "DisplayMode:=" , 0,
                    [
                        "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,0],  
"VCC1_L7:=" , [True,True,0],  
"VCC2_L8:=" , [True,True,0]  
,  
[  
    "NAME:ObjectAttributesInNetMode",  
    "GND:=" , [True,True,52],  
    "VCC:=" , [True,True,52],  
    "neg:=" , [True,True,52],  
    "pos:=" , [True,True,52]  
,  
[  
    "NAME:ObjectAttributesInObjectMode",  
    "line_1:=" , [True,True,20],  
    "line_2:=" , [True,True,20],  
    "line_3:=" , [True,True,20],  
    "line_4:=" , [True,True,53],  
    "rect_6:=" , [True,False,30],
```

```
        "rect_17:=" , [True,True,53],
        "rect_18:=" , [True,True,53],
        "rect_19:=" , [True,False,53],
        "rect_20:=" , [True,False,53],
        "rect_21:=" , [True,False,30],
        "rect_22:=" , [True,False,30],
        "via_0:=" , [False,False,30],
        "via_1:=" , [False,False,30],
        "via_2:=" , [False,False,30],
        "via_3:=" , [False,False,30]
    ]
}
]
]
)
```

## ChangeProperty (Symbol Editor)

**Use:** Changes to properties are scripted using the **ChangeProperty** command. This command can be executed by the **oEditor** to change editor properties, by the **oDesign** to change design level properties, and by the **oProject** to change project level properties.

The command can be used to create, edit, and/or remove properties. In Circuit, only Variable and Separator properties can be deleted.

*Command:* None

*Syntax:* ChangeProperty Array("Name:AllTabs", <PropTabArray>, <PropTabArray>, ...)

*Return Value:* None

*Parameters:* <PropTabArray>

```
    Array("Name:<PropTab>",
          <PropServersArray>,
          <NewPropsArray>,
          <ChangedPropsArray>,
          <DeletedPropsArray>)
```

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

```
<NewPropsArray>
    Array("Name:NewProps", <PropdataArray>,
          <PropdataArray>, ...)
```

```
<ChangedPropsArray>
    Array("Name:ChangedProps", <PropdataArray>,
```

<PropdataArray>, ...)

<DeletedPropsArray>

Array("Name:DeletedProps", <PropName>,

<PropName>, ...)

OR (for PropDisplay deletions only)

Array("Name:DeletedProps", <PropdataArray>,

<PropdataArray>, ...)

<PropdataArray>

Array("NAME:<PropName>",

"PropType:=", <PropType>,

"NewName:=", <string>,

"Description:=", <string>,

"Callback:=", <string>,

"NewRowPosition:=", <int>,

"ReadOnly:=", <bool>,

"Hidden:=", <bool>,

<PropTypeSpecificArgs>) OR (for PropDisplays only)

```
Array("Name:<PropName>",<PropDisplayData>)
```

```
<PropDisplayData>
```

for adding, changing, deleting PropDisplays

```
<PropDisplayAttributes>
```

for changing PropDisplays only

```
<PropDisplayNewAttributes>
```

```
<PropDisplayAttributes>
```

Layer & Location only used for PropDisplays in layout

For adding PropDisplays, this will add a single PropDisplay with attributes as shown; if an attribute is missing, a default value will be assigned. Adding PropDisplay to schematic with attributes that are identical to one already existing there will fail without an error message.

For deleting PropDisplays, these attributes are used to identify an existing PropDisplay to delete. If there doesn't exist a PropDisplay that matches the given attributes, then nothing will be deleted. If multiple PropDisplays match the given attributes, then all of them will be deleted. If an attribute is missing, then all PropDisplays match that missing attribute. For example, if Layer is missing, then PropDisplays on all layers that match the remaining given attributes will be deleted.

For changing PropDisplays, these attributes are used to identify an existing PropDisplay to change. If no PropDisplay matching the attributes is found, no changes will be made. If multiple PropDisplays match the attributes, all of them will be changed. If an attribute is missing, it matches all PropDisplays. For example, to change the format of PropDisplays that are on the bottom, but have any layer, style or format to show the name only, this command should have Location set to "Bottom" and all other attributes omitted.

```
"Format:=", <PropDisplayType>,
```

```
"Location:=", <PropDisplayLocation>,
```

```
"Layer:=", <string>,  
"Style:=", <string>
```

```
<PropDisplayNewAttributes>
```

NewLayer & NewLocation only used for PropDisplays in layout

For changing PropDisplays, these attributes are used to identify which attributes to change and what the new value is. If the attribute should not be changed, the corresponding entry should be omitted.

```
"NewName:=", <string>,  
"NewFormat:=", <PropDisplayType>,  
"NewLocation:=", <PropDisplayLocation>,  
"NewLayer:=", <string>,  
"NewStyle:=", <string>
```

```
<PropDisplayType>
```

Type: string

Identifies the format of PropDisplay.

```
"Name"  
"Value"  
"NameAndValue"  
"EvaluatedValue"
```

"NameAndEvaluatedValue"

<PropDisplayLocation>

Type: string

Identifies where PropDisplay is located with respect to object

"Left"

"Top"

"Right"

"Bottom"

"Custom"

<PropType>

Type: string

Identifies the type of property when a new property is added. In Circuit, only separator properties and variable properties can be added.

"SeparatorProp"

"VariableProp"

"TextProp"

"NumberProp"

"ValueProp"

"CheckboxProp"

"MenuProp"

"PointProp"

"VPointProp"

"ButtonProp"

#### NewName

Specify the new name of a property if the property's name is being edited. In Circuit, the name can only be changed for separators and variables.

#### Description

Specify a description of the property. In Circuit, the description can only be changed for separators and variables.

#### Callback

Specify the name of the script callback to be run when the property value is changed.

#### NewRowPosition

Used to reorder rows in the **Property** dialog box. In Circuit, this only applies to the **Project>Project Variables** panel and the **Designer>DesignProperties** panel. Specify the new zero-based row index of the variable or separator.

#### ReadOnly

Used to mark a property as "read only" so it can not be modified. In Circuit, this flag can only be set for variables and separators.

## Hidden

Used to hide a property so it can not be viewed outside of the **Property** dialog box. In Circuit, this flag can only be set for variables and separators.

<PropTypeSpecificArgs>

**SeparatorProp:** no arguments

TextProp: "Value:=", <string>

NumberProp: "Value:=", <double>

ValueProp: "Value:=", <value>

CheckboxProp: "Value:=", <bool>

MenuProp: "Value:=", <string>

PointProp "X:=", <double>, "Y:=", <double>

VPointProp: "X:=", <value>, "Y:=", <value>

Material Button: "Material:=", <string>

Color Button: "R:="",<int>,"G:="",<int>,"B:="",<int>

Transparency Button: "Value:=", <double>

<PropTypeSpecificArgs> for MenuProps

Syntax for NewProps array: "AllChoices:=",

<"choice1,choice2,..."> or <Array("choice1" "choice2", ... )>,

"Value:=", <string>

Syntax for ChangedProps array: "Value:=", <string>

<PropTypeSpecificArgs> for VariableProps

Syntax:

```
"Value:=", <value>, <OptimizationFlagsArray>,  
<TuningFlagsArray>, <SensitivityFlagsArray>,  
<StatisticsFlagsArray>
```

Parameters:

```
<OptimizationFlagsArray>  
Array("NAME:Optimization",  
"Included:=", <bool>,  
"Min:=", <value>,  
"Max:=", <value>)
```

```
<TuningFlagsArray>  
Array("NAME:Tuning",  
"Included:=", <bool>,  
"Step:=", <value>,  
"Min:=", <value>,  
"Max:=", <value>)
```

```
<SensitivityFlagsArray>
  Array("NAME:Sensitivity",
    "Included:=", <bool>,
    "Min:=", <value>,
    "Max:=", <value>,
    "IDisp:=", <value> )  
  
<StatisticsFlagsArray>
  Array("NAME:Statistical",
    "Included:=", <bool>,
    "Dist:=", <Distribution>,
    "StdD:=", <value>,
    "Min:=", <value>,
    "Max:=", <value>,
    "Tol:=", <string>)  
  
<Distribution>
  Type: string
  Value should be "Gaussian" or "Uniform"
```

StdD

Standard deviation.

Min

Low cut-off for the distribution.

Max

High cut-off for the distribution.

Tol

Tolerance for uniform distributions. Format is "<int>%".

Example: "20%".

*VB Example:*

Adding a new project level variable "\$width":

```
oProject.ChangeProperty Array("NAME:AllTabs", _  
    Array("NAME:ProjectVariableTab", _  
        Array("NAME:PropServers", "ProjectVariables"), _  
        Array("NAME:NewProps", _  
            Array("NAME:$width", _
```

```
"PropType:=", "VariableProp",_
"Value:=", "3mm",_
"Description:=", "my new variable")))
```

*VB Example:* Deleting the design level variable "height":

```
oDesign.ChangeProperty Array("NAME:AllTabs",_
Array("NAME:LocalVariableTab",_
Array("NAME:PropServers", "DefinitionParameters"),_
Array("NAME:DeletedProps", "height"))
```

Changing a property's value. If the following command were executed, then the value of the property "XSize" of the PropServer

"Box1>CreateBox:1" on the "Geometry3DCmdTab" tab would be changed. (oEditor is the Geometry3D editor in Circuit.)

```
oEditor.ChangeProperty Array("NAME:AllTabs",_
Array("NAME:Geometry3DCmdTab",_
Array("NAME:PropServers","Box1>CreateBox:1"),_
Array("NAME:ChangedProps",_
Array("NAME:XSize", "Value:=", "1.4mil"))))
```

*VB Example:* Changing a property's value. If the following command were executed, then the values of Callback and L on the PassedParameterTab would be changed.

```
oEditor.ChangeProperty Array("NAME:AllTabs", _  
    Array("NAME:PassedParameterTab", _  
        Array("NAME:PropServers", "CHOKE2"), _  
        Array("NAME:ChangedProps", _  
            Array("NAME:L", "Callback:=", "ac", "OverridingDef:=", true), _  
            Array("NAME:L", "Value:=", "1nH"))))
```

*VB Example:* Changing the Company Name, Design Name, the background color, and the Axis scaling in a Report.

```
Set oProject = oDesktop.SetActiveProject("wgcombiner")  
Set oDesign = oProject.SetActiveDesign("CircuitDesign2")  
Set oModule = oDesign.GetModule("ReportSetup")  
oModule.ChangeProperty Array("NAME:AllTabs", Array("NAME:Header", _ Array("NAME:PropServers", "XY  
Plot1:Header"), _  
    Array("NAME:ChangedProps", Array("NAME:Company Name", _  
        "Value:=", "My Company"))))  
oModule.ChangeProperty Array("NAME:AllTabs", Array("NAME:Header", _ Array("NAME:PropServers", "XY  
Plot1:Header"), _  
    Array("NAME:ChangedProps", Array("NAME:Design Name", _  
        "Value:=", "WG Combiner"))))
```

```

oModule.ChangeProperty Array("NAME:AllTabs", Array("NAME:General", _ Array("NAME:PropServers",
"XY Plot1:General"), _
Array("NAME:ChangedProps", Array("NAME:Back Color", _
"R:=", 128, "G:=", 255, "B:=", 255))))
oModule.ChangeProperty Array("NAME:AllTabs", Array("NAME:Axis", _ Array("NAME:PropServers", "XY
Plot1:AxisX"), _
Array("NAME:ChangedProps", Array("NAME:Axis Scaling", _
"Value:=", "Log"))))

```

*VB Example:* Changing a property's value. Note that the AllChoices parameter is only used when the MenuProp is being added. Also note that either a string of choices separated by commas or an Array("choice1", "choice2", "choice3") works for the AllChoices parameter.

```

Set oEditor = oDesign.SetActiveEditor("SchematicEditor")
oEditor.ChangeProperty Array("NAME:AllTabs", Array("NAME:PassedParameterTab", Array
("NAME:PropServers", _
"CompInst@CAP_2"), Array("NAME>NewProps", Array("NAME:xxxx", "PropType:=", _
"MenuProp", "AllChoices:=", Array("aa", "bb", "cc", "dd"), "Value:=", "bb"))))

```

## ClearMessages

Clears information in the **Messages** window.

UI Access	N/A
-----------	-----

<b>Parameters</b>	None.
<b>Return Value</b>	None.

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

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

## CloneMaterial

Clones a local material.

<b>UI Access</b>	N/A									
<b>Parameters</b>	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td><code>&lt;matName&gt;</code></td><td>String</td><td>Name of existing material.</td></tr><tr><td><code>&lt;newName&gt;</code></td><td>String</td><td>Name for newly cloned material.</td></tr></tbody></table>	Name	Type	Description	<code>&lt;matName&gt;</code>	String	Name of existing material.	<code>&lt;newName&gt;</code>	String	Name for newly cloned material.
Name	Type	Description								
<code>&lt;matName&gt;</code>	String	Name of existing material.								
<code>&lt;newName&gt;</code>	String	Name for newly cloned material.								
<b>Return Value</b>	Boolean: <ul style="list-style-type: none"><li>• 1 - Material is cloned.</li><li>• 0 - Existing material not found or a conflict with the new material name.</li></ul>									

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

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

## Close

Closes the active project.

**Warning:**

Unsaved changes will be lost.

<b>UI Access</b>	N/A
<b>Parameters</b>	None.
<b>Return Value</b>	None.

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

<b>VB Syntax</b>	Close
------------------	-------

**VB Example**

```
oProject.Close
```

## CopyDesign

Copies a specified design.

<b>UI Access</b>	Edit > Copy.								
<b>Parameters</b>	<table border="1"><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;DesignName&gt;</td><td>String</td><td>Name of the design to copy from.</td></tr></table>			Name	Type	Description	<DesignName>	String	Name of the design to copy from.
Name	Type	Description							
<DesignName>	String	Name of the design to copy from.							
<b>Return Value</b>	None.								

**Python Syntax**

```
CopyDesign (<DesignName>)
```

**Python Example**

```
oProject.CopyDesign ("HFSSDesign1")
```

<b>VB Syntax</b>	CopyDesign <DesignName>
<b>VB Example</b>	<pre>oProject.CopyDesign "HFSSDesign1"</pre>

## CutDesign

Cuts a design from the active project. The design is stored in memory and can be pasted.

**Warning:**

This is a legacy command that is no longer supported and should not be used as it may have unintended effects on solved designs.

<b>UI Access</b>	Edit > Cut.						
<b>Parameters</b>	<table border="1"> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> <tr> <td>&lt;DesignName&gt;</td> <td>String</td> <td>Name of the design.</td> </tr> </table>	Name	Type	Description	<DesignName>	String	Name of the design.
Name	Type	Description					
<DesignName>	String	Name of the design.					
<b>Return Value</b>	None.						

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

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

## DeleteDataset

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

<b>UI Access</b>	Project > Datasets > Remove.					
<b>Parameters</b>	<table border="1"> <tr> <td>Name</td> <td>Type</td> <td>Description</td> </tr> </table>			Name	Type	Description
Name	Type	Description				

	<i>&lt;DatasetName&gt;</i>	String	Name of the dataset found in the project.
<b>Return Value</b>	None.		

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

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

## DeleteDesign

Deletes a specified design in the project.

<b>UI Access</b>	Edit > Delete, or Delete in the ribbon.								
<b>Parameters</b>	<table border="1"><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><i>&lt;DesignName&gt;</i></td><td>String</td><td>Name of the design.</td></tr></table>			Name	Type	Description	<i>&lt;DesignName&gt;</i>	String	Name of the design.
Name	Type	Description							
<i>&lt;DesignName&gt;</i>	String	Name of the design.							
<b>Return Value</b>	None.								

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

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

## DeleteToolObject

**Note:**

This command is for internal Ansys use only.

<b>UI Access</b>	N/A						
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;ObjectName&gt;</td> <td>String</td> <td>Name of the tool object.</td> </tr> </tbody> </table>	Name	Type	Description	<ObjectName>	String	Name of the tool object.
Name	Type	Description					
<ObjectName>	String	Name of the tool object.					
<b>Return Value</b>	None.						

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

<b>VB Syntax</b>	DeleteToolObject <ObjectName>
------------------	-------------------------------

**VB Example**

```
oProject.DeleteToolObject "object1"
```

## EditDataset

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

UI Access	Project > Datasets > Edit.		
<b>Parameters</b>	Name <i>&lt;OriginalName&gt;</i>	Type String	Description Name of the original dataset.
	<i>&lt;DatasetdataArray&gt;</i>	Array	Data for the modified dataset.
<b>Return Value</b>	None.		

**Python Syntax**

```
EditDataset (<OriginalName> <DatasetdataArray>)
```

**Python Example**

```
oProject.EditDataset ("ds1"
    [ "NAME:ds2",
        [ "NAME:Coordinates",
            [
                "NAME:Coordinate",
                "X:=", 1, "Y:=", 2
            ],
            [
                "NAME:Coordinate",
```

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

**VB Syntax**

```
EditDataset <OriginalName> <DatasetdataArray>
```

**VB Example**

```

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)))

```

**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>", "CoordinateSystemType:=", <string>, "BulkOrSurfaceType:=" , <integer>, [ "NAME:PhysicsTypes", ...     ] ]

		<pre>         "set:=" , &lt;array containing string physics types&gt; ], &lt;Optional ModifierDataArray&gt;, "permeability:=" , &lt;string containing value&gt;, "conductivity:=" , &lt;string containing value&gt;, "thermal_conductivity:=" , &lt;string containing value&gt;, "mass_density:=" , &lt;string containing value&gt;, "specific_heat:=" , &lt;string containing value&gt;, "youngs_modulus:=" , &lt;string containing value&gt;, "poissons_ratio:=" , &lt;string containing value&gt;, "thermal_expansion_coefficient:=" , &lt;string con- taining value&gt; ] </pre>
<Modi- fierdataArray>	Array	<p>Optional structured array containing thermal or spatial modifiers:</p> <pre> [     "NAME:ModifierData",     [         "NAME:&lt;ThermalModifierData or SpatialModifierData&gt;",         "modifier_data:=" , &lt;"thermal_modifier_data" or "spa-         tial_modifier_data"&gt;,         [             "NAME:&lt;all_thermal_modifiers or all_spatial_mod- </pre>

		<pre>ifiers", [     "NAME:&lt;modifierName&gt;",     "Property::=" , &lt;string property being modified&gt;,     "Index::=" , &lt;integer&gt;,     "prop_modifier::=" , &lt;"thermal_modifier" or "spatial_modifier"&gt;,     "use_free_form::=" , &lt;Boolean&gt;,     "free_form_value::=" , &lt;string modifier value&gt;, ] ]</pre>
<b>Return Value</b>	None.	

<b>Python Syntax</b>	EditMaterial (<OriginalName>, <MatProperties>)
<b>Python Example</b>	<b>Without Modifiers:</b> 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"  
]  
)
```

### 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:=" , "12000000000",
"poissons_ratio:=" , "0.38",
"thermal_expansion_coefficient:=" , "1.77e-05"
])

```

### Transient Solve, Non-linear Drude Data Plasma

```
# -----
# Script Recorded by Ansys Electronics Desktop Version 2023.2.0
# 14:23:56 Jun 17, 2022
# -----
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",
        "PlasmaMaintainFrequency:=", "2.31174231183139GHz",
        "NeutralDensity:="      , 2.65164580488373E+20,
        "ElectronDensity:="      , 2.65164580488373E+17,
        "CollisionRateConstant:=", 2.05499505485618E-15
    ]
]
])

```

<b>VB Syntax</b>	EditMaterial <OriginalName>, <MatProperties>
<b>VB Example</b>	<p><b>Without Modifiers:</b></p> <pre> oDefinitionManager&gt;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_coeffcient:=", "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::=" , "12000000000",
"poissons_ratio::=" , "0.38",
"thermal_expansion_coefficient::=" , "1.77e-05"
)
```

## ExportDataset

Exports a dataset to a named file. This can be executed by the oProject, or oDesign variables.

<b>UI Access</b>	<b>Project &gt; Datasets &gt; Export.</b>		
<b>Parameters</b>	Name <code>&lt;datasetFilePath&gt;</code>	Type String	Description The full path to the file.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	<code>ExportDataset (&lt;datasetFilePath&gt;)</code>
<b>Python Example</b>	<code>oProject.ExportDataset ('e:/tmp/dsdata.txt')</code> <code>oDesign.ExportDataset ('e:/tmp/dsdata.txt')</code>

<b>VB Syntax</b>	<code>ExportDataset &lt;datasetFilePath&gt;</code>
<b>VB Example</b>	<code>oProject.ExportDataset "e:/tmp/dsdata.txt"</code> <code>oDesign.ExportDataset "e:/tmp/dsdata.txt"</code>

## ExportMaterial

Exports a local material to a library.

<b>UI Access</b>	<b>Export to Library</b> command in the material editor.		
<b>Parameters</b>	Name <code>&lt;ExportData&gt;</code>	Type Array	Description ["NAME:<LibraryName>", <MaterialName>, <MaterialName>, ...]

	<code>&lt;LibraryName&gt;</code>	String	Name of the exported library.
	<code>&lt;MaterialName&gt;</code>	String	Name of the material to be exported.
	<code>&lt;LibraryLocation&gt;</code>	String	Location to save the library. Only "PersonalLib" and "UserLib" are allowed.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	<code>ExportMaterial (&lt;ExportData&gt;, &lt;LibraryLocation&gt;)</code>
<b>Python Example</b>	<code>oDefinitionManager.ExportMaterial ([ "NAME:mylib", _ "Material1", "Material2", "Material3"], "PersonalLib")</code>

<b>VB Syntax</b>	<code>ExportMaterial &lt;ExportData&gt;, &lt;LibraryLocation&gt;</code>
<b>VB Example</b>	<code>oDefinitionManager.ExportMaterial Array("NAME:mylib", _ "Material1", "Material2", "Material3"), "PersonalLib"</code>

## GetActiveDesign

Returns the design in the active project

**Note:** GetActiveDesign will return normally if there are no active objects.

<b>UI Access</b>	N/A
<b>Parameters</b>	None.
<b>Return Value</b>	Object of the active design.

<b>Python Syntax</b>	GetActiveDesign()
<b>Python Example</b>	<pre>oDesign = oProject.GetActiveDesign()</pre>

<b>VB Syntax</b>	GetActiveDesign
<b>VB Example</b>	<pre>Set oDesign = oProject.GetActiveDesign</pre>

## GetArrayVariables

Returns a list of array variables. To get a list of indexed project variables, execute with oProject. To get a list of indexed local variables, use oDesign.

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

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

<b>VB Syntax</b>	GetArrayVariables
<b>VB Example</b>	<pre>oProject.GetArrayVariables</pre>

	<code>oDesign.GetArrayVariables</code>
--	----------------------------------------

## GetChildNames [Project]

Returns the names of the project's child objects.

<b>UI Access</b>	N/A		
<b>Parameters</b>	Name <code>&lt;type&gt;</code>	Type String	Description (Optional) Default is "design"; Any value returned by <a href="#">GetChildTypes()</a> can be used.
<b>Return Value</b>	Array of names of children for the queried object.		

<b>Python Syntax</b>	<code>GetChildNames (&lt;type&gt;)</code>
<b>Python Example</b>	<pre>arrDesignNames = oProject.GetChildNames() arrVarbleNames = oProject.GetChildNames("Variable")</pre>

<b>VB Syntax</b>	<code>GetChildNames (&lt;type&gt;)</code>
<b>VB Example</b>	<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.

<b>UI Access</b>	N/A		
<b>Parameters</b>	Name <code>&lt;path&gt;</code>	Type String	Description The path may include multiple generations (for example, "designObject/moduleObj/SetupObject"). See <a href="#">Object Path</a> .
<b>Return Value</b>	Object of a found child.		

<b>Python Syntax</b>	<code>GetChildObject(&lt;path&gt;)</code>
<b>Python Example</b>	<pre>oProject = oDesktop.GetActiveProject() oDesign = oProject.GetChildObject("TeeModel") oVariable = oProject.GetChildObject("VariableName") oReport = oProject.GetChildObject("TeeModel/Results/S Parameter Plot 1")</pre>

<b>VB Syntax</b>	<code>GetChildObject(&lt;path&gt;)</code>
<b>VB Example</b>	<pre>Set oProject = oDesktop.GetActiveproject() Set oDesign = oProject.GetChildObject("TeeModel") Set oVariable = oProject.GetChildObject("VariableName")</pre>

```
Set oReport = oProject.GetChildObject("TeeModel/Results/S Parameter Plot 1")
```

## GetChildTypes [Project]

Returns the types of the project's child objects.

<b>UI Access</b>	N/A
<b>Parameters</b>	None.
<b>Return Value</b>	Array of string represents types of the child objects.

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

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

## GetConfigurableData (Project)

### Note:

This command is for internal Ansys use only.

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

## GetDefinitionManager

Gets the `DefinitionManager` object.

<b>UI Access</b>	N/A
<b>Parameters</b>	None.
<b>Return Value</b>	<code>DefinitionManager</code> object.

<b>Python Syntax</b>	GetDefinitionManager()
<b>Python Example</b>	<code>oDefinitionManager = oProject.GetDefinitionManager()</code>

<b>VB Syntax</b>	GetDefinitionManager
<b>VB Example</b>	<code>Set oDefinitionManager = oProject.GetDefinitionManager</code>

## GetDependentFiles

Provides a list of the external files referenced in the project, including characteristic (for example, MDX) and coupled project files.

<b>UI Access</b>	N/A
------------------	-----

---

<b>Parameters</b>	None.
<b>Return Value</b>	List of referenced files.

<b>Python Syntax</b>	GetDependentFiles()
<b>Python Example</b>	<pre>files = oProject.GetDependentFiles()</pre>

<b>VB Syntax</b>	GetDependentFiles
<b>VB Example</b>	<pre>files = oProject.GetDependentFiles</pre>

## GetDesign

Returns the interface to a specific design in a given project.

<b>UI Access</b>	N/A								
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;designName&gt;</td> <td>String</td> <td>Name of the design.</td> </tr> </tbody> </table>			Name	Type	Description	<designName>	String	Name of the design.
Name	Type	Description							
<designName>	String	Name of the design.							
<b>Return Value</b>	Object of the specified design.								

<b>Python Syntax</b>	GetDesign (<designName>)
<b>Python Example</b>	<pre>oProject.GetDesign ("HFSSDesign1")</pre>

<b>VB Syntax</b>	GetDesign < <i>designName</i> >
<b>VB Example</b>	<code>oProject.GetDesign "HFSSDesign1"</code>

## GetDesigns

Obtains all designs in the current project.

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

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

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

## GetEDBHandle

Returns the EDB handle for the project.

**Important:**

This script is for internal Ansys use only.

<b>UI Access</b>	N/A
<b>Parameters</b>	None.
<b>Return Value</b>	String indicating the EDB handle for the project.

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

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

## GetLegacyName

Obtains the legacy name of a project.

**Note:**

This command is for internal Ansys use only.

<b>UI Access</b>	N/A
<b>Parameters</b>	None.
<b>Return Value</b>	String containing the legacy project name.

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

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

## GetName [Project]

Obtains the project name

<b>UI Access</b>	N/A
<b>Parameters</b>	None.
<b>Return Value</b>	String containing the project name, not including the path or extension.

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

--	--

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

## GetObjPath [Project]

Obtains the project name from the full path.

<b>UI Access</b>	N/A
<b>Parameters</b>	None.
<b>Return Value</b>	String containing only the project name.

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

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

## GetPath

Returns the location of the project on disk.

<b>UI Access</b>	N/A
<b>Parameters</b>	None.
<b>Return Value</b>	String containing the path to the project, not including the project name.

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

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

## 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.

<b>UI Access</b>	N/A		
<b>Parameters</b>	Name <code>&lt;includeReadOnly&gt;</code>	Type Boolean	Description (Optional) <ul style="list-style-type: none"><li>• True – Include read only props.</li><li>• False – Do not include read only props.</li></ul>
<b>Return Value</b>	Empty array.		

---

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

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

## GetPropValue [Project]

Returns the property value for the active project object, or specified property values.

<b>UI Access</b>	N/A		
<b>Parameters</b>	Name	Type	Description
	<code>&lt;propPath&gt;</code>	String	A child object's property path. See: <a href="#">Property Function Summary</a> .
<b>Return Value</b>	String of property value.		

<b>Python Syntax</b>	GetPropValue(<propPath>)
<b>Python Example</b>	<code>oProject.GetPropValue ("TeeModel/offset")</code>

<b>VB Syntax</b>	GetPropValue <propPath>
<b>VB Example</b>	<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 <i>&lt;PropTab&gt;</i>	Type String	Description One of the following, where tab titles are shown in parentheses: <ul style="list-style-type: none"><li>• PassedParameterTab ("Parameter Values")</li><li>• DefinitionParameterTab (Parameter Defaults")</li><li>• LocalVariableTab ("Variables" or "Local Variables")</li><li>• ProjectVariableTab ("Project variables")</li><li>• ConstantsTab ("Constants")</li><li>• BaseElementTab ("Symbol" or "Footprint")</li><li>• ComponentTab ("General")</li><li>• Component("Component")</li><li>• CustomTab ("Intrinsic Variables")</li><li>• Quantities ("Quantities")</li><li>• Signals ("Signals")</li></ul>
	<i>&lt;PropServer&gt;</i>	String	An object identifier, generally returned from another script method, such as ComplInst@R;2;3
Return Value	Array of strings containing the names of the appropriate properties.		

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

<b>VB Syntax</b>	GetProperties <PropTab>, <PropServer>
<b>VB Example</b>	<code>oEditor.GetProperties "PassedParameterTab", "k"</code>

## GetProperty Value

Returns the value of a single property belonging to a specific <PropServer> and <PropTab>. This function is available with the Project, Design or Editor objects, including definition editors.

**Tip:**

Use the script recording feature and edit a property, and then view the resulting script to see the format for that property.

<b>UI Access</b>	N/A		
<b>Parameters</b>	Name <PropTab>	Type String	Description One of the following, where tab titles are shown in parentheses: <ul style="list-style-type: none"><li>• PassedParameterTab ("Parameter Values")</li><li>• DefinitionParameterTab (Parameter Defaults")</li><li>• LocalVariableTab ("Variables" or "Local Variables")</li><li>• ProjectVariableTab ("Project variables")</li><li>• ConstantsTab ("Constants")</li><li>• BaseElementTab ("Symbol" or "Footprint")</li><li>• ComponentTab ("General")</li></ul>

		<ul style="list-style-type: none"> <li>• Component("Component")</li> <li>• CustomTab ("Intrinsic Variables")</li> <li>• Quantities ("Quantities")</li> <li>• Signals ("Signals")</li> </ul>
	<PropServer>	String An object identifier, generally returned from another script method, such as CompInst@R;2;3
	<PropName>	String Name of the property.
<b>Return Value</b>	String value of the property.	

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

<b>VB Syntax</b>	GetPropertyValues (<PropTab>, <PropServer>, <PropName>)
------------------	---------------------------------------------------------

<b>VB Example</b>	<pre>selectionArray = oEditor.GetSelections for k in selectionArray:     val = oEditor.GetProperty("PassedParameterTab", k, "R")     ...</pre>
-------------------	------------------------------------------------------------------------------------------------------------------------------------------------------------

## GetTopDesignList

Returns a list of top-level design names.

<b>UI Access</b>	N/A
<b>Parameters</b>	None.
<b>Return Value</b>	List of strings containing name of top-level designs.

<b>Python Syntax</b>	GetTopDesignList()
<b>Python Example</b>	oProject.GetTopDesignList()

<b>VB Syntax</b>	GetTopDesignList
<b>VB Example</b>	oProject.GetTopDesignList

## GetVariableValue

Gets the value of a single specified variable. To get the value of project variables, execute this command using `oProject`. To get the value of local variables, use `oDesign`.

<b>UI Access</b>	N/A		
<b>Parameters</b>	Name <code>&lt;VarName&gt;</code>	Type String	Description Name of the variable to access.
<b>Return Value</b>	String represents the value of the variable.		

<b>Python Syntax</b>	<code>GetVariableValue( &lt;VarName&gt; )</code>
<b>Python Example</b>	<code>oProject.GetVariableValue ("var_name")</code>

<b>VB Syntax</b>	<code>GetVariableValue &lt;VarName&gt;</code>
<b>VB Example</b>	<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`.

<b>UI Access</b>	N/A
------------------	-----

<b>Parameters</b>	None.
<b>Return Value</b>	Array of strings containing the variables.

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

<b>VB Syntax</b>	GetVariables
<b>VB Example</b>	<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.

<b>UI Access</b>	<b>Project &gt; Datasets &gt; Import.</b>		
<b>Parameters</b>	Name	Type	Description
	<datasetFilePath>	String	The full path to the file containing the dataset values. *.tab files recommended (see <a href="#">note</a> below).
<b>Return Value</b>	None.		

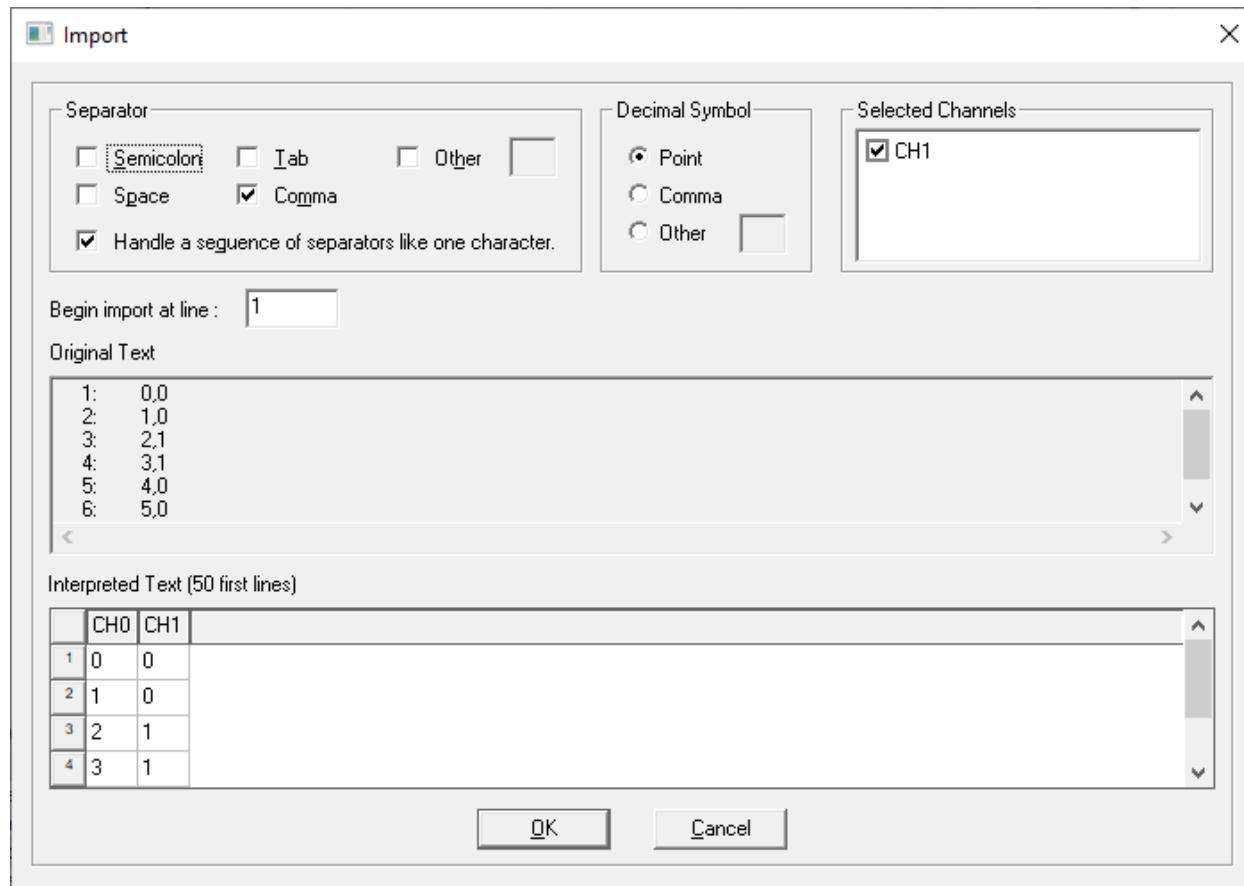
<b>Python Syntax</b>	ImportDataset (<datasetFileFullPath>,<optionalDatasetName>)
<b>Python Example</b>	<pre>oProject.ImportDataset('e:\tmp\dsdata.tab') oDesign.ImportDataset('e:\tmp\dsdata.tab') oProject.ImportDataset('e:\tmp\dsdata.tab', 'MyDatasetName') oDesign.ImportDataset('e:\tmp\dsdata.tab', 'MyDatasetName')</pre>

<b>VB Syntax</b>	ImportDataset <datasetFileFullPath>,<optionalDatasetName>
<b>VB Example</b>	<pre>oProject.ImportDataset "e:\tmp\dsdata.tab" oDesign.ImportDataset "e:\tmp\dsdata.tab" oProject.ImportDataset "e:\tmp\dsdata.tab", "MyDatasetName" oDesign.ImportDataset "e:\tmp\dsdata.tab", "MyDatasetName"</pre>

## Note About File Types:

Tab-delimited or space-delimited files with the extension \*.tab are the recommended file type. When using ImportDataset at the Design level, \*.tab is the only file type supported.

At the Project level, other file types are supported (for example, \*.csv). However, after calling the command, you must configure the file import format manually through the Electronics Desktop GUI by selecting **Project > Datasets** and clicking **Import**.



## 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].		
<b>Parameters</b>	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.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	InsertDesign (<DesignType>, <DesignName>, <TechnologyFile>:<SubCircuitID>, "")
<b>Python Example</b>	<pre>oProject.InsertDesign('Nexxim Circuit', 'MyDesigner', 'C:\\Program Files\\AnsysEM\\Designer\\', '')</pre>

<b>VB Syntax</b>	InsertDesign <DesignType>, <DesignName>, <TechnologyFile>:<SubCircuitID>, ""
<b>VB Example</b>	<pre>oProject.InsertDesign "Nexxim Circuit", "MyDesigner", "C:\\Program Files\\AnsysEM\\Designer\\", ""</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.

<b>UI Access</b>	N/A		
<b>Parameters</b>	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>&lt;stationaryPath&gt;</code>	String	Path.
<b>Return Value</b>	IDispatch string, such as 'IDispatch(IAltraSimScript)'		

<b>Python Syntax</b>	<code>InsertDesignWithWorkflow(&lt;type&gt;, &lt;workflowName&gt;, &lt;specName&gt;, &lt;fileName&gt;, &lt;libLoc&gt;, &lt;stationaryPath&gt;)</code>
<b>Python Example</b>	<pre>oProject.InsertDesignWithWorkflow ("Circuit Design", "Serial Design", "PCIe3 Stressed", "LongChannel", "SysLib", "C:\Program Files\AnsysEM\v232\Win64\syslib\MS - RT_duroid 6010 (Er=10.2) 0.010 inch, 0.5 oz copper.asty")</pre>

<b>VB Syntax</b>	<code>InsertDesignWithWorkflow &lt;type&gt;, &lt;workflowName&gt;, &lt;specName&gt;, &lt;fileName&gt;, &lt;libLoc&gt;, &lt;stationaryPath&gt;</code>
<b>VB Example</b>	<pre>oProject.InsertDesignWithWorkflow "Circuit Design", "Serial Design", "PCIe3 Stressed", "LongChannel", "SysLib", "C:\Program Files\AnsysEM\v232\Win64\syslib\MS - RT_duroid 6010" &amp; " (Er=10.2) 0.010 inch, 0.5 oz copper.asty"</pre>

## InsertToolObject

**Note:**

This command is for internal Ansys use only.

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

## Paste (Project Object)

Pastes a design in the active project.

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

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

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

## Redo [Project Level]

Reapplies the last project-level command.

<b>UI Access</b>	Edit > Redo.
------------------	--------------

<b>Parameters</b>	None.
<b>Return Value</b>	None.

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

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

## RemoveAllUnusedDefinitions

Removes all unused project definitions.

<b>UI Access</b>	N/A
<b>Parameters</b>	None.
<b>Return Value</b>	None.

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

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

## RemoveMaterial

Removes a material from a library.

UI Access	Remove Material(s) command in the material editor		
<b>Parameters</b>	Name	Type	Description
	<code>&lt;MaterialName&gt;</code>	String	Name of the material to be removed.
	<code>&lt;IsProjectMaterial&gt;</code>	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.
	<code>&lt;LibraryName&gt;</code>	String	Name of the user or personal library where the material resides.
	<code>&lt;LibraryLocation&gt;</code>	String	Location of library. Valid options:"UserLib" or "PersonalLib".
<b>Return Value</b>	None.		

<b>Python Syntax</b>	RemoveMaterial (<MaterialName>, <IsProjectMaterial>, <LibraryName>, <LibraryLocation>)
<b>Python Example</b>	<code>oDefinitionManager.RemoveMaterial ([ "Material1", false, "mo0907", "UserLib"])</code>

<b>VB Syntax</b>	RemoveMaterial <MaterialName>, <IsProjectMaterial>, <LibraryName>, <LibraryLocation>
<b>VB Example</b>	<pre>oDefinitionManager.RemoveMaterial "Material1", false, "mo0907", "UserLib"</pre>

## RemoveUnusedDefinitions

Removes any unused project definitions.

<b>UI Access</b>	<b>Tools &gt; Project Tools &gt; Remove Unused Definitions.</b>						
<b>Parameters</b>	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>&lt;Definitions&gt;</td><td>Array</td><td>Definitions to be removed, such as materials and surface materials.</td></tr></tbody></table>	Name	Type	Description	<Definitions>	Array	Definitions to be removed, such as materials and surface materials.
Name	Type	Description					
<Definitions>	Array	Definitions to be removed, such as materials and surface materials.					
<b>Return Value</b>	None.						

<b>Python Syntax</b>	RemoveUnusedDefinitions(<Definitions>)
<b>Python Example</b>	<pre>oProject.RemoveUnusedDefinitions (     [         [             "NAME:Materials",             "Al-Extruded"         ],</pre>

	<pre>[     "NAME:SurfaceMaterials",     "Steel-oxidised-surface" ] ])</pre>
--	---------------------------------------------------------------------------------------------

<b>VB Syntax</b>	RemoveUnusedDefinitions <Definitions>
<b>VB Example</b>	<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\(\)](#).

<b>UI Access</b>	Edit > Rename.									
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;NewName&gt;</td> <td>String</td> <td>Desired name of the project. The path is optional.</td> </tr> <tr> <td>&lt;OverWriteOk&gt;</td> <td>Boolean</td> <td> <ul style="list-style-type: none"> <li>True - overwrite the file on disk if it exists.</li> <li>False - prevent overwrite.</li> </ul> </td> </tr> </tbody> </table>	Name	Type	Description	<NewName>	String	Desired name of the project. The path is optional.	<OverWriteOk>	Boolean	<ul style="list-style-type: none"> <li>True - overwrite the file on disk if it exists.</li> <li>False - prevent overwrite.</li> </ul>
Name	Type	Description								
<NewName>	String	Desired name of the project. The path is optional.								
<OverWriteOk>	Boolean	<ul style="list-style-type: none"> <li>True - overwrite the file on disk if it exists.</li> <li>False - prevent overwrite.</li> </ul>								
<b>Return Value</b>	None.									

<b>Python Syntax</b>	Rename(<NewName>,<OverWriteOK>)
<b>Python Example</b>	<pre>oProject.Rename ("c:\projects\MyProject.aedt", True)</pre>

<b>VB Syntax</b>	Rename <NewName>,<OverWriteOK>
<b>VB Example</b>	<pre>oProject.Rename "c:\projects\MyProject.aedt", true</pre>

## Save

Saves the active project.

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

<b>Python Syntax</b>	Save()
<b>Python Example</b>	<pre>oProject.Save()</pre>

<b>VB Syntax</b>	Save
<b>VB Example</b>	<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.		
<b>Parameters</b>	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.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	For non-Schematic/Layout project: SaveAs ( <NewName> <OverWriteOK> )  For Schematic/Layout project: SaveAs ( <NewName> <OverWriteOK> <DefaultAction> <OverrideActions> )
<b>Python</b>	oProject.SaveAs ('D:/projects/project1.aedt', True)

<b>n Exam- ple</b>	<pre>---- oProject.SaveAs('D:/Projects/Project1.aedt', True, 'ef_overwrite', ['NAME:OverrideActions', ['NAME:ef_copy_no_overwrite', ['NAME:Files', '\$PROJECTDIR/circuit_models.inc']], ['NAME:ef_make_path_absolute', ['NAME:Files', '\$PROJECTDIR/SL_6s.sp']]])</pre>
----------------------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

<b>VB Syn- tax</b>	<p>For non-Schematic/Layout project: SaveAs &lt;<i>NewName</i>&gt; &lt;<i>OverWriteOK</i>&gt;</p> <p>For Schematic/Layout project: SaveAs &lt;<i>NewName</i>&gt; &lt;<i>OverWriteOK</i>&gt; &lt;<i>DefaultAction</i>&gt; &lt;<i>OverrideActions</i>&gt;</p>
<b>VB Exampl- es</b>	<pre>oProject.SaveAs "D:/projects/project1.aedt", true ---- oProject.SaveAs     "F:\Designer Projects\TA33097\HighSpeedChannel.aedt", true, "ef_overwrite", Array     ("NAME:OverrideActions",      Array("NAME:ef_copy_no_overwrite", Array("NAME:Files", "\$PROJECTDIR/circuit_mod- els.inc")),      Array("NAME:ef_make_path_absolute", Array("NAME:Files", "\$PROJECTDIR\SL_6s.sp")))</pre>

**Important:**

The DefaultAction and OverrideActions strings correspond to the following actions:

- **ef\_overwrite** – Copy file to new project directory and overwrite.
- **ef\_copy\_no\_overwrite** – Copy file to new project directory and don't overwrite.
- **ef\_make\_path\_absolute** – Change reference to point to file in old project directory.
- **Empty String** – Do nothing.

The DefaultAction is applied to all files that are NOT explicitly listed in the OverrideActions array. Those in the OverrideActions array are separate arrays for actions that are different from the default action; those actions are applied to the files listed in the same array:

- If OverrideActions are not specified, DefaultAction is applied to ALL files in project directory.

## SaveAsStandAloneProject

Saves the project as a standalone copy.

**Note:**

This script is not supported when the application is being controlled by Ansys Workbench.

<b>UI Access</b>	N/A		
<b>Parameters</b>	Name <code>&lt; projectName &gt;</code>	Type String	Description The desired name of the project, with directory and extension.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	SaveAsStandAloneProject( <projectName>)
<b>Python Example</b>	<code>oProject.SaveAsStandAloneProject('D:/projects/project1.aedt')</code>

<b>VB Syntax</b>	SaveAsStandAloneProject <projectName>
<b>VB Examples</b>	<code>oProject.SaveAsStandAloneProject "D:/projects/project1.aedt"</code>

## SaveProjectArchive

Saves the active project as an archive to the specified file path.

<b>UI Access</b>	<b>File &gt; Archive.</b>		
<b>Parameters</b>	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.
<b>Return Value</b>	<ArchiveNotes>	String	String describe the archive.
	None.		

<b>Python Syntax</b>	SaveProjectArchive(<archivefilepath>, <IncludeExternalFiles>, <IncludeResultsFiles>, <AdditionalFiles>, <ArchiveNotes>)
<b>Python</b>	<code>oProject.SaveProjectArchive("C:\\\\Users\\\\Documents\\\\Ansoft\\\\Project27.aedtz", True,</code>

<b>Example</b>	<code>False, [], "")</code>
----------------	-----------------------------

<b>VB Syntax</b>	<code>SaveProjectArchive &lt;archivefilepath&gt;, &lt;IncludeExternalFiles&gt;, &lt;IncludeResultsFiles&gt;, &lt;AdditionalFiles&gt;, &lt;ArchiveNotes&gt;</code>
<b>VB Example</b>	<code>oProject.SaveProjectArchive "C:\Documents\OptimTee.aedtz", true, false, Array(), "My notes"</code>

## SetActiveDefinitionEditor

Obtains a specified definition editor.

<b>UI Access</b>	N/A		
<b>Parameters</b>	Name	Type	Description
	<code>&lt;EditorName&gt;</code>	String	Name of the definition editor to set active, one of "SymbolEditor", "FootprintEditor".
<b>Return Value</b>	Object for the definition to be edited.		

<b>Python Syntax</b>	<code>SetActiveDefinitionEditor(&lt;EditorName&gt;, &lt;DefinitionName&gt;)</code>
<b>Python Example</b>	<code>oProject.SetActiveDefinitionEditor("SymbolEditor", "Simplorer Elements\Basic Elements\Circuit\Passive Elements:R")</code>

<b>VB Syntax</b>	SetActiveDefinitionEditor < <i>EditorName</i> >, < <i>DefinitionName</i> >
<b>VB Example</b>	<pre>oProject.SetActiveDefinitionEditor "SymbolEditor",_ "Simpler Elements\Basic Elements\Circuit\Passive Elements:R"</pre>

## SetActiveDesign

Sets a design to be the active design.

<b>UI Access</b>	N/A						
<b>Parameters</b>	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>&lt;<i>DesignName</i>&gt;</td><td>String</td><td>Name of the design to set as the active design.</td></tr></tbody></table>	Name	Type	Description	< <i>DesignName</i> >	String	Name of the design to set as the active design.
Name	Type	Description					
< <i>DesignName</i> >	String	Name of the design to set as the active design.					
<b>Return Value</b>	None.						

<b>Python Syntax</b>	SetActiveDesign (< <i>DesignName</i> >)
<b>Python Example</b>	<pre>oDesign = oProject.SetActiveDesign("SimplorerDesign2")</pre>

<b>VB Syntax</b>	SetActiveDesign < <i>DesignName</i> >
<b>VB Example</b>	<pre>Set oDesign = oProject.SetActiveDesign "SimplorerDesign2"</pre>

## SetPropValue [Project]

Sets a property value for an active project's child object.

<b>UI Access</b>	Edit Properties on ProjectTree objects.									
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><code>&lt;propPath&gt;</code></td> <td>String</td> <td>A child object's property path. See: <a href="#">Property Function</a>.</td> </tr> <tr> <td><code>&lt;newValue&gt;</code></td> <td>String</td> <td>New property value.</td> </tr> </tbody> </table>	Name	Type	Description	<code>&lt;propPath&gt;</code>	String	A child object's property path. See: <a href="#">Property Function</a> .	<code>&lt;newValue&gt;</code>	String	New property value.
Name	Type	Description								
<code>&lt;propPath&gt;</code>	String	A child object's property path. See: <a href="#">Property Function</a> .								
<code>&lt;newValue&gt;</code>	String	New property value.								
<b>Return Value</b>	<p>Boolean:</p> <ul style="list-style-type: none"> <li>• <b>True</b> – property found.</li> <li>• <b>False</b> – property not found.</li> </ul>									

<b>Python Syntax</b>	<code>SetPropValue(&lt;propPath&gt;, &lt;newValue&gt;)</code>
<b>Python Example</b>	<pre>oProject.SetPropValue("TeeModel/offset", "2mm") oProject.SetPropValue("TeeModel/Results/S Parameter Plot 1/Display Type", "Data Table")</pre>

<b>VB Syntax</b>	<code>SetPropValue &lt;propPath&gt;, &lt;newValue&gt;</code>
<b>VB Example</b>	<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
	<propTab>	String	<p>One of the following, where tab titles are shown in parentheses:</p> <ul style="list-style-type: none"> <li>• PassedParameterTab ("Parameter Values")</li> <li>• DefinitionParameterTab (Parameter Defaults")</li> <li>• LocalVariableTab ("Variables" or "Local Variables")</li> <li>• ProjectVariableTab ("Project variables")</li> <li>• ConstantsTab ("Constants")</li> <li>• BaseElementTab ("Symbol" or "Footprint")</li> <li>• ComponentTab ("General")</li> <li>• Component("Component")</li> <li>• CustomTab ("Intrinsic Variables")</li> <li>• Quantities ("Quantities")</li> <li>• Signals ("Signals")</li> </ul>
	<propServer>	String	An object identifier, generally returned from another script method, such as ComplInst@R;2;3
	<propName>	String	Name of the property.

	<code>&lt;propValue&gt;</code>	String	The value for the property
<b>Return Value</b>	None.		

<b>Python Syntax</b>	<code>SetPropertyValue(&lt;propTab&gt;, &lt;propServer&gt;, &lt;propName&gt;, &lt;propValue&gt;)</code>
<b>Python Example</b>	<code>oEditor SetPropertyValue ("PassedParameterTab", "k", "R", "2200")</code>

<b>VB Syntax</b>	<code>SetPropertyValue &lt;propTab&gt;, &lt;propServer&gt;, &lt;propName&gt;, &lt;propValue&gt;</code>
<b>VB Example</b>	<code>oEditor SetPropertyValue "PassedParameterTab", "k", "R", "2200"</code>

## SetVariableValue

Sets the value of a variable. To set the value of a project variable, execute this command using `oProject`. To set the value of a local variable, use `oDesign`.

<b>UI Access</b>	N/A											
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><code>&lt;VarName&gt;</code></td> <td>String</td> <td>Variable name.</td> </tr> <tr> <td><code>&lt;VarValue&gt;</code></td> <td>Value</td> <td>New value for the variable.</td> </tr> </tbody> </table>			Name	Type	Description	<code>&lt;VarName&gt;</code>	String	Variable name.	<code>&lt;VarValue&gt;</code>	Value	New value for the variable.
Name	Type	Description										
<code>&lt;VarName&gt;</code>	String	Variable name.										
<code>&lt;VarValue&gt;</code>	Value	New value for the variable.										
<b>Return Value</b>	None.											

<b>Python Syntax</b>	<code>SetVariableValue (&lt;VarName&gt;, &lt;VarValue&gt;)</code>
----------------------	-------------------------------------------------------------------

**Python Example**

```
oProject.SetVariableValue('$Var1', '3mm')
```

**VB Syntax**

```
SetVariableValue <VarName>, <VarValue>
```

**VB Example**

```
oProject.SetVariableValue "$Var1", "3mm"
```

## 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**

```
SimulateAll()
```

**Python Example**

```
oProject.SimulateAll()
```

**VB Syntax**

```
SimulateAll
```

**VB Example**

```
oProject.SimulateAll
```

## Undo [Project]

Cancels the last project-level command.

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

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

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

## UpdateDefinitions

Updates all definitions. The **Messages** window reports when definitions are updated, or warns when definitions cannot be found.

<b>UI Access</b>	Tools > Project Tools > Update Definitions. Click Select All, then Update.
<b>Parameters</b>	None.
<b>Return Value</b>	None.

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

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

# 6 - Object Oriented Property Scripting

Scripting in AEDT has been markedly enhanced via the convenient use of Object-Oriented access to retrieve or modify properties of objects in AEDT. This feature allows for much less code to be written to access object properties and enables much more readable code for our users, avoiding complex array input.

The primary gains of the use of Object-Oriented scripting are the ease with which properties of various existing objects in an Ansys Electronics DesktopProject/Design can be read, modified, and set. Along with this ease of implementation comes much more 'readable' code to aid in others' interpretation of custom scripts.

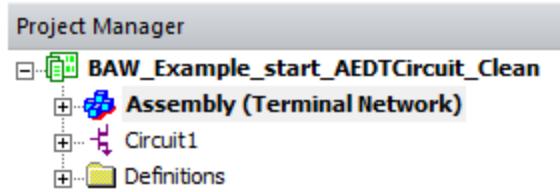
This App Note will discuss the following: The logic for syntax of access, Basic Attributes of Project and Design properties and examples of use.

## Object-Oriented Scripting

There are five basic functions that you use in Object Oriented scripting to retrieve and set properties:

1. GetChildNames()
2. GetChildObjects()
3. GetPropNames()
4. GetPropValue()
5. SetPropValue()

At a high level, use GetChildNames() to determine what object instances exist for a given object. An example is shown below to demonstrate for an AEDT Project shown that has two Designs.



If you open the Command Window that allows for executing python code, you first define the Project Object, oProject, and the Design Object, oDesign, as shown:

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

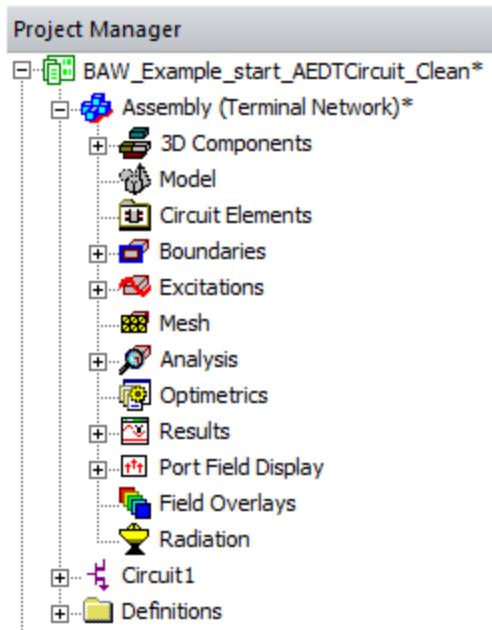
Once the objects have been defined, you can use GetChildNames() to learn what object instances exist for each. As an example, observe the Child Names of oProject, and you see a list of the Designs in the AEDT Project, per the GUI.

```
>>> oProject.GetChildNames()
['Assembly', 'Circuit1']
```

As another example, retrieve the names of the Object Instances available in oDesign, to see the various objects associated with a Design setup:

```
>>> oDesign.GetChildNames()
['Boundaries', 'Excitations', 'Circuit Elements', 'Model', 'Mesh', 'Analysis', 'Optimetrics', 'Port
Field Display', 'Field Overlays', 'Radiation', 'Results', '3D Modeler']
```

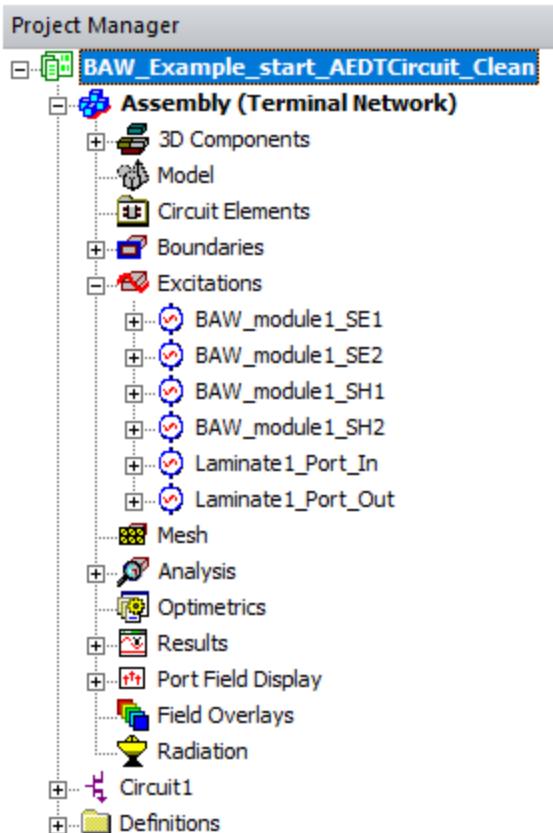
These names are what you would expect based on the Project Manager Layout:



In the **Project Manager** above, any object that has the '+' symbol is populated with children that you can query. Once you know the name of the object you want, you can instance it via the GetChildObject() command. This defines the instance to the desired object. As an example, set an instance for the Excitations:

```
>>> oExcitations = oDesign.GetChildObject('Excitations')
```

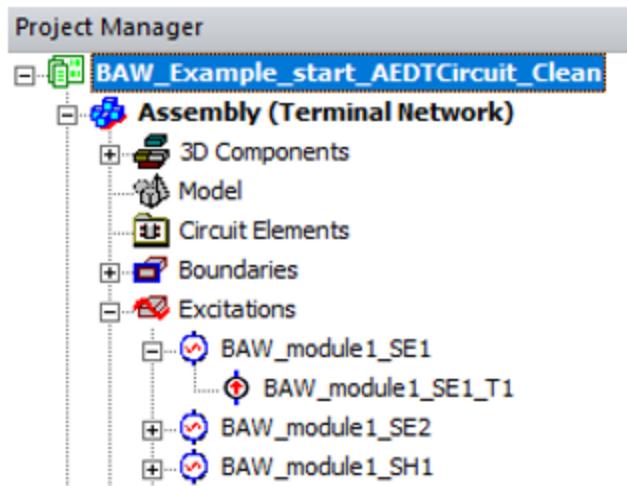
You now have an object, oExcitations defined to be the oDesign Child Object 'Excitations'. What does this mean? If you expand the Excitations dialogue in the Project Manager, you expect that the Child Names of this object would be the names of the Excitations as defined, in this case six ports:



```
>>> oExcitations.GetChildNames()
['Laminate1_Port_In', 'Laminate1_Port_Out', 'BAW_module1_SE1', 'BAW_module1_SE2', 'BAW_module1_SH1',
 'BAW module1 SH2']
```

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

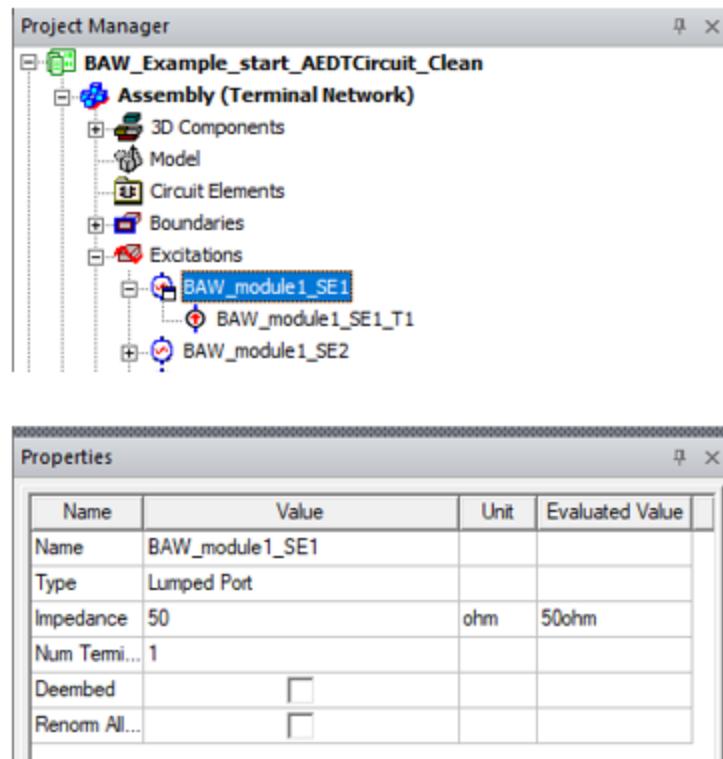
Expand the first port to see its expected Child Object, its Terminal, in the **Project Manager** Window:



Through scripting, first define the Port Object (in this example, oPort) using the 'GetChildObject()' command for the first port, 'BAW\_module1\_SE1.' Then determine its Object Child Name, the terminal definition:

```
>>> oPort = oExcitations.GetChildObject('BAW_module1_SE1')
>>> oPort.GetChildNames()
['BAW_module1_SE1_T1']
```

As the use and logic for GetChildNames() and GetChildObject() have been demonstrated, you can now explore the properties of each of these objects, if they exist. The function to determine what properties exist is GetPropNames(). Use this to determine what properties exist to be retrieved or modified for a given object. The properties available are readily identifiable in the **Property** window, by default located beneath the **Project Manager** window. For example, if you select a given port object, 'BAW\_module1\_SE1' the Property window populates as shown:



Name	Value	Unit	Evaluated Value
Name	BAW_module1_SE1		
Type	Lumped Port		
Impedance	50	ohm	50ohm
Num Termi... Deembed	1		
Renom All...	<input type="checkbox"/>		

If you execute the GetPropNames() function on the previously defined object, oPort, you see the same Property Names as available in the **Properties** window:

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

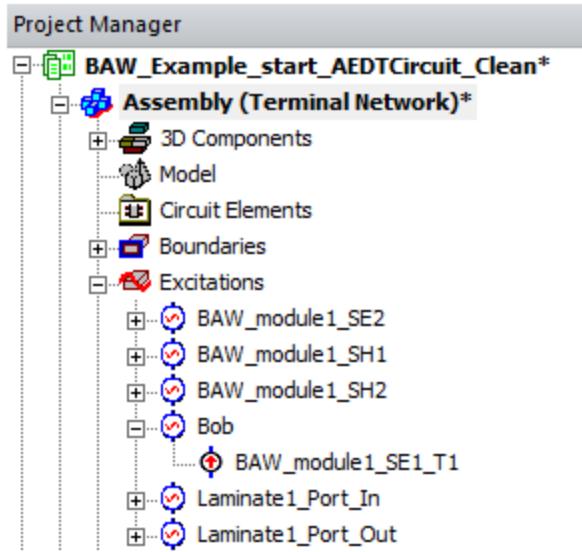
Once you identify the desired object and you know the desired property, you can access the value via GetPropValue(). For example, if you want to retrieve the name of the object oPort:

```
>>> oPort.GetPropValue('Name')  
'BAW_module1_SE1'
```

To change the value of the property, use the SetPropValue() function. The arguments for this function are ('Property Name', 'New Value'). For example, to change the name of the port to 'Bob':

```
>>> oPort.SetPropValue('Name', 'Bob')  
True
```

This function returns a Boolean 'True' if successful. The Project Manager window updates accordingly:



This approach for retrieving and setting properties is general and can be used for many aspects of an Ansys Electronics Desktop simulation. This Object-Oriented method of property identification and modification operates only on existing objects. Object-Oriented scripting cannot create new instances; you must revert to the functions in a given Module to do that. Not all Children of a given object may be accessible via the GetChildNames() command just yet. An example is given for Material property modification later in this App Note. However, if you need specific objects you can reference details in the Scripting Help or reach out to an Application Engineer.

## Material Properties and Examples

This section discusses the material properties and how to access and modify them. Because materials are globally defined, the objects are children of the Project, oProject, as shown below:

```
>>> oProject = oDesktop.GetActiveProject()
>>> oMaterials = oProject.GetChildObject('Materials')
>>> oMaterials.GetChildNames()
['vacuum', 'Cap_Mat', 'Outline_Mat', 'SolderMask_Mat', 'copper', 'pec']
```

All materials with a Project Definition, or assigned to an object, in the Project are accessible. For example, assume you want to see the conductivity of 'copper.' Follow the same flow as in the previous section:

```
>>> oCopper = oMaterials.GetChildObject('copper')
>>> oCopper.GetPropNames()
['Coordinate System Type', 'Coordinate System Type/Choices', 'Relative Permittivity Type', 'Relative Permittivity Type/Choices', 'Relative Permittivity', 'Relative Permeability Type', 'Relative Permeability Type/Choices', 'Relative Permeability', 'Bulk Conductivity Type', 'Bulk Conductivity Type/Choices', 'Bulk Conductivity', 'Dielectric Loss Tangent Type', 'Dielectric Loss Tangent Type/Choices', 'Dielectric Loss Tangent', 'Magnetic Loss Tangent Type', 'Magnetic Loss Tangent Type/Choices', 'Magnetic Loss Tangent', 'Electric Coercivity Type', 'Electric Coercivity Magnitude', 'Magnetic Coercivity Type', 'Magnetic Coercivity Magnitude', 'Thermal Conductivity Type', 'Thermal Conductivity Type/Choices', 'Thermal Conductivity', 'Magnetic Saturation Type', 'Magnetic Saturation', 'Lande G Factor Type', 'Lande G Factor', 'Delta H Type', 'Delta H', '- Measured Frequency Type', '- Measured Frequency', 'Core Loss Model', 'Core Loss Model/Choices', 'Mass Density Type', 'Mass Density', 'Composition', 'Composition/Choices', 'Specific Heat Type', 'Specific Heat', 'Young's Modulus Type', 'Young's Modulus Type/Choices', "Young's Modulus", "Poisson's Ratio Type", "Poisson's Ratio Type/Choices", "Poisson's Ratio", 'Thermal Expansion Coefficient Type', 'Thermal Expansion Coefficient Type/Choices', 'Thermal Expansion Coefficient', 'Magnetostriction Type', 'Inverse Magnetostriction Type', 'Thermal Material Type', 'Thermal Material Type/Choices', 'Solar Behavior Type', 'Solar Behavior Type/Choices', 'Solar Behavior']
```

The Material Property of interest is "Bulk Conductivity." So you create a "Cond" object to store the value and use the GetPropValue function to obtain it. Then name the Cond object to see the value:

```
>>> Cond = oCopper.GetPropValue('Bulk Conductivity')
>>> Cond
'58000000'
```

To change the conductivity, use the SetPropValue() function as shown below:

```
>>> oCopper.SetPropValue('Bulk Conductivity', '100')
True
>>> NewCond = oCopper.GetPropValue('Bulk Conductivity')
>>> NewCond
'100'
```

## Body Properties and Modification

The following example shows how to retrieve the properties of a Body in the model, in this case a Region object. Once you identify the desired property, you can modify it as needed.

```
>>> oModel = oDesign.GetChildObject('3D Modeler')
>>> oModel.GetChildNames()
['RadBox_Region_1']
>>> oRegion = oModel.GetChildObject('RadBox_Region_1')
>>> oRegion.GetPropNames()
['Name', 'Material', 'Solve Inside', 'Orientation', 'Orientation/Choices', 'Model', 'Group', 'Display
Wireframe', 'Material Appearance', 'Color', 'Color/Red', 'Color/Green', 'Color/Blue', 'Transparent']
```

## Retrieving Variables

Retrieving defined variables in a Design or Project is a common effort for automation. There are two types of variables, Design and Project. Project variables are preceded with a '\$' symbol and are retrieved in the Project object as it is globally defined to all Designs. Design variables do not have any preceding symbols and are retrieved in the Design object as their scope is limited to a given Design. The following example demonstrates the retrieval of Project Variable names and values, and then Design variable names and values.:

```
>>> oProjVar = oProject.GetChildObject("Variables")
>>> oProjVar.GetPropNames()
 ['$test']
>>> oProjVar.GetPropValue('$test')
 '0'
>>> oDesVar = oDesign.GetChildObject("Variables")
>>> oDesVar.GetPropNames()
 ['test']
>>> oDesVar.GetPropValue('test')
 '0'
```

## Retrieve Datasets and Values

The GetChildObject, GetChildTypes and GetChildNames functions operate on the oDesktop objects. This allows you to retrieve and view datasets and values. The dataset script wrapper store all values internally in SI units, and converts them back to user-supplied units when you request non-SI property values. For example, if you assigned a dataset to the example OptimTee project in HFSS, you could use these functions in the command window:

```
>>>oDesktop.GetChildTypes()
['Projects']

>>>oDesktop.GetChildNames()
['OptimTee']

>>>arrProjectNames = oDesktop.GetChildNames()

>>>tp = oDesktop.GetChildObject('OptimTee')

>>>tp.GetChildTypes()
```

```
['Design', 'Project Data']

>>>tp.GetChildNames('Project Data')

['Variables', 'Materials', 'Surface Materials', 'Datasets']

>>>ds = tp.GetChildObject('datasets')

>>>ds.GetChildNames()

 ['$ds1']

>>>ds1=ds.GetChildObject('$ds1')

>>>ds1.GetPropValue('[:, :, :]')

[[1.0, 4.0], [2.0, 5.0], [3.0, 6.0]]

>>>ds1.GetPropSIValue()

[[1.0, 4.0], [2.0, 5.0], [3.0, 6.0]]

>>>ds1.DimUnits

>>>ds1.DimUnits = ['mm', 'mm']

>>>ds1.DimUnits

['mm', 'mm']

>>>ds1.GetPropSIValue()

[[0.001, 0.004000000000000001], [0.002, 0.005000000000000001], [0.003000000000000001,
0.006000000000000001]]

>>>ds1.GetPropValue('[:, :, :]')

[[1.0, 4.0], [2.0, 5.0], [3.0, 6.0]]
```

## GetSolutionData API

Many users want to use scripts to extract solution data from Ansys Electronics Desktop for custom Post Processing. Scripting includes a new method to do this without having to export data to a file and then re-import it for use in a script. The new function is accessible via the “ReportSetup” Module. The function call is “GetSolutionDataPerVariation()”. A code snippet to extract Terminal S Parameter data is shown:

```
8  oModule = oDesign.GetModule("ReportSetup")
9  Results = oModule.GetSolutionDataPerVariation("Terminal.Solution.Data", "Setup1 :: Sweep1",
10  [
11      [
12          [
13              [
14                  [
15                      [
16                          [
17                              [
18                                  [
19                                      ##..Get Dependent and Independent Variable data for Nominal Variation
20                                      NominalData = Results[0]
21                                      ##..Get Independent Variable Data
22                                      ##..For second argument, if pass True then data is in SI Units
23                                      ##.....if pass False then data is in default scale units instead of SI
24                                      SweepValues = NominalData.GetSweepValues("Freq", True)
25                                      ##..Get Dependent Variable DataValues
26                                      ##..Note: ..Can pass any 'Y Component' Name
27                                      DataValues = NominalData.GetRealDataValues("dB(St(Terminal_1))")
```

The above code shows how you can extract the Dependent and Independent data to variables for easy manipulation. For more information on other functions available for this, see [GetSolutionDataPerVariation](#).

## Summary

Scripting has been advancing in Ansys Electronics Desktop to better allow you to customize and automate their repetitive or complex simulations. The ability to easily retrieve and set property values via the Object-Oriented scripting allows for ease or both writing and

reading. The ability to extract solution data within a script execution is a new functionality that markedly enables more advanced post processing.

Object oriented property scripting presents an easy to use, intuitive and object oriented representation of the data model. The framework supports query of objects and their properties including the edits of the data model in an object oriented fashion. With the new scripting framework, data exposure is intuitive and provides maximum coverage.

Each exposed script object supports the following COM functions:

- GetName
  - Return name of the object as text string
  - e.g. name of a design, solve setup, boundary, etc
- GetChildTypes
  - An object can have different types of children.
  - Return array of text string. Can be empty if the object's children are NOT categorized into different types.

For example, a design object has 3 children types. The following examples show how the commands run in the **Tools > Open Command Window** for IronPython.

```
>>> design.GetChildTypes()  
['Module', 'Editor', 'Design Data']
```

- GetChildNames
  - Input: [String – Type]. Default = “Module” and “Editor” for design script object. ‘All’ for other script objects.
  - Return an array of immediate children’s names, of a given type if specified

For example, a Mechanical design object has these children.

```
>>> design.GetChildNames()  
['Boundaries', 'Excitations', 'Optimetrics', 'Results', '3D Modeler']
```

Four of the children are of “Module” type

```
>>> design.GetChildNames("module")
['Boundaries', 'Excitations', 'Optimetrics', 'Results']
```

- **GetChildObject**

- Input: String -- Object path. The path may include multiple generations.
- Return the child object if found

For example,

```
>>> d = project.GetChildObject("hfss")
>>> d.GetChildObject("3d modeler").GetChildNames()
['Box1', 'Box1_1', 'Box1_1_1']
>>> project.GetChildObject("hfss/3d modeler").GetChildNames()
['Box1', 'Box1_1', 'Box1_1_1']
```

- **GetPropNames**

- Input: [BOOL - IncludeReadOnly] -- default to true
- Return an array of the object's properties

For example,

```
>>> geom = project.GetChildObject("hfss/3d modeler").GetChildObject("Box1")
>>> geom.GetPropNames()
['Name', 'Material', 'Material/SIValue', 'Material/EvaluatedValue', 'Solve Inside', 'Orientation',
 'Orientation/Choices', 'Model', 'Group', 'Display Wireframe', 'Material Appearance',
 'Color', 'Color/Red', 'Color/Green', 'Color/Blue', 'Transparent']
```

- GetPropValue
  - Input: String – Property Path. The path may include multiple generations.
  - Return the property value as VARIANT

For example,

```
>>> geom.GetPropertyValue("material")
'"vacuum"'
>>> geom.GetPropertyValue("xsize")
'3mm'
>>> op.GetPropertyValue("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.SetPropertyValue("model", False)
>>> boxcmd.SetPropertyValue("ysize", "4mm")
```

- GetPropEvaluatedValue (<PropName>)

For example,

```
oVar = oDesign.GetChildObject(" Variables/var")
oVar.GetPropEvaluatedValue()
```

- GetPropSIValue (<PropName>)

For example,

```
oCreateBox = oDesign.GetChildObject("3D Modeler/Box1/CreateBox:1")
oCreateBox.GetPropValue("xSize")
    return "length / 2"
oCreateBox.GetPropEvaluatedValue("xSize")
    return '0.4mm'
oCreateBox.GetPropSIValue("xSize")
    return 0.0004
```

## Additional Details Specific to AEDT Solvers

“3D Modeler” of 3D products and “Machine” of RMxprt are exposed as “Editor” type children of a design script object.

“Variables” and “Design Settings” are exposed as “Design Data” type children of a design script object.

The following “Module” types are exposed as “Module” type children of a design script object.

HFSS

- Boundaries, Excitations, Circuit Elements, Hybrid Regions, Analysis, Radiation, Field Overlays, Optimetrics, Results

HFSS 3D Layout

- Boundaries, Excitations, Circuit Elements, Analysis, Radiation, Field Overlays, Optimetrics, Results

Maxwell 3D/2D

- Boundaries, Excitations, Analysis, Field Overlays, Optimetrics, Results

**RMxprt**

- Analysis, Field Overlays, Optimetrics, Results

**Q3D**

- Boundaries, Nets, Analysis, Optimetrics,

**Q2D**

- Boundaries, Conductors, Analysis, Field Overlays, Optimetrics,

**Icepak**

- Thermal, Monitor, Mesh, Analysis, Field Overlays, Optimetrics, Results

**Mechanical**

- Boundaries, Excitations, Analysis, Field Overlays, Optimetrics, Results

**Circuit**

- Optimetrics, Results

**Circuit Netlist**

- Results

**EMIT**

- Coupling

**Simplorer/Twin Builder**

- Analysis, Optimetrics, Results

## Additional details on Boundaries/Excitations

Each design type presents its boundaries/excitations data in the project tree as different groups. For example, an HFSS design has Boundaries, Excitations, Circuit Elements and Hybrid Regions while a Icepak design has just a “Thermal” project tree folder.

These module script objects do not have properties

```
>>> project.GetChildObject("icepak/thermal").GetPropNames()  
[]
```

GetChildTypes of these module script objects returns the types of its immediate children

```
>>> d = p.GetChildObject("q2d")  
>>> d.GetChildObject("conductors").GetChildTypes()  
['NonIdealGround', 'SignalLine']
```

GetChildNames of these module object returns its immediate children

```
>>> p.GetChildObject("icepak/thermal").GetChildNames()  
['Source1', 'Resistance1', 'ConductingPlate1', 'Source2', 'Resistance2', 'ConductingPlate2',  
'Source3', 'Resistance3', 'ConductingPlate3']
```

GetChildNames can be invoked with a “type” and the returns will be filtered by that given type.

```
>>> p.GetChildObject("icepak/thermal").GetChildNames("resistance")  
['Resistance1', 'Resistance2', 'Resistance3']
```

Children of a module object are scriptable objects and have properties.

```
>>> port = p.GetChildObject("hfss/excitations/1")  
>>> port.GetPropNames(False)  
['Name', 'Deembed', 'Deembed Dist', 'Renorm All Terminals']
```

---

You can query/edit these properties

```
>>> port.GetPropValue("deembed")
False
>>> port.SetPropValue("deembed", True)
True
>>> port.GetPropValue("deembed")
True
```

A boundary/excitation script object can also have children. For example, HFSS terminal is a child of its port. Q3D source/sink can be children of a net.

```
>>> port.GetChildNames()
['Box1_T1']
>>> port.GetChildTypes()
['Terminal']
>>> p.GetChildObject("q3d/nets/s2").GetChildNames()
['Source2', 'Sink2']
>>> p.GetChildObject("q3d/nets/s2").GetChildTypes()
['Sink', 'Source']
```

## 3D component encapsulation

These script interfaces are compliant with encapsulation. For example,

- Design.GetChildObject("boundaries").GetChildNames() will not return component boundaries
- SetPropValues of component excitations can only be used to edit post processing settings such as 'Deembed', 'Deembed Dist' of a HFSS port.

## Additional details on Solve setup

All solve setups are children of the “Analysis” script object. This parent script object is also of the type “Module”.

```
>>> d = oDesktop.GetActiveProject().GetChildObject("hfss")
>>> d.GetChildNames()
['Boundaries', 'Excitations', 'Hybrid Regions', 'Circuit Elements', 'Analysis', 'Opti-
metrics', 'RadField', 'Results', '3D Modeler']
>>> setups = d.GetChildObject("analysis")
```

This module script object has no property

```
>>> setups.GetPropNames()
[]
```

The children of this module script object in a 3D design is not categorized into different types because the solve setup type is one-to-one to the solution type of a 3D design.

```
>>> setups.GetChildTypes()
[]
```

The children of this module script object in a 3DLayout and Simplorer/TwinBuilder design is categorized into different solve setup types, such as “Transient”, “AC” and “DC” in a Simplorer/TwinBuilder design and “HFSS”, “PlanarEM”, “SIwave” in a 3D layout design.

A solve setup script object can also have children. Children are typically frequency sweeps.

```
>>> setup.GetChildNames()
```

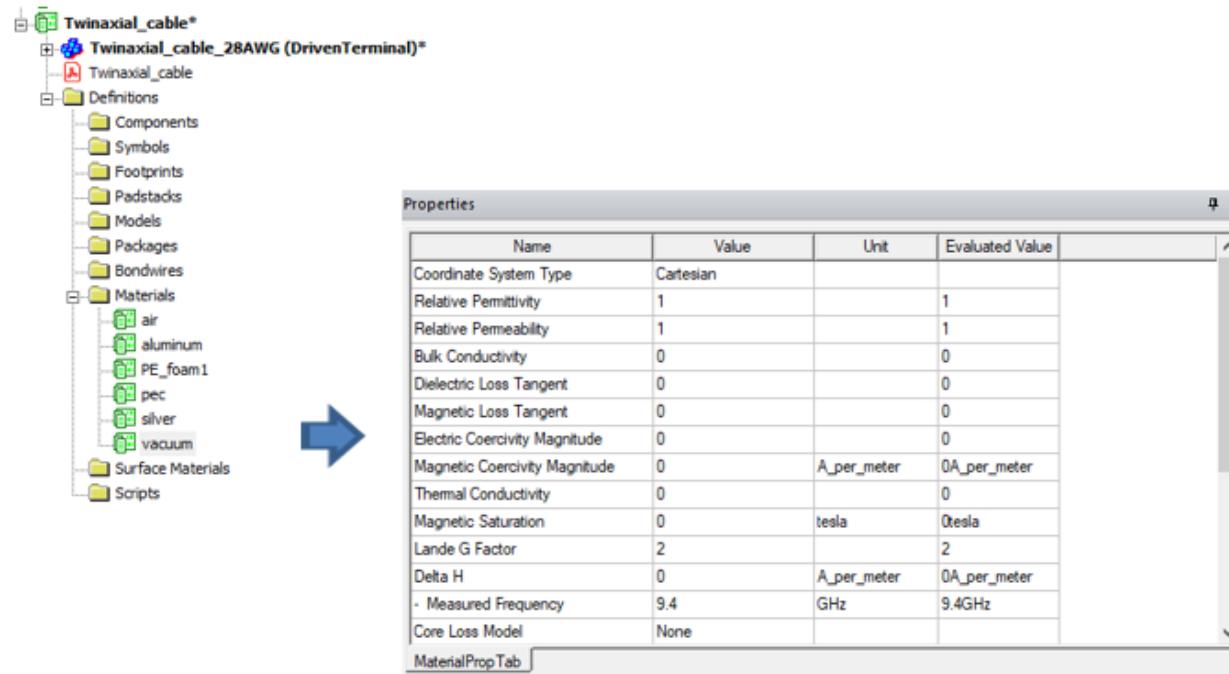
```
['Sweep', 'Sweep1', 'Sweep2']
>>> setup.GetChildTypes()
['Discrete', 'Interpolating']
>>> sweep1 = setup.GetChildObject("sweep")
```

### Related Topics

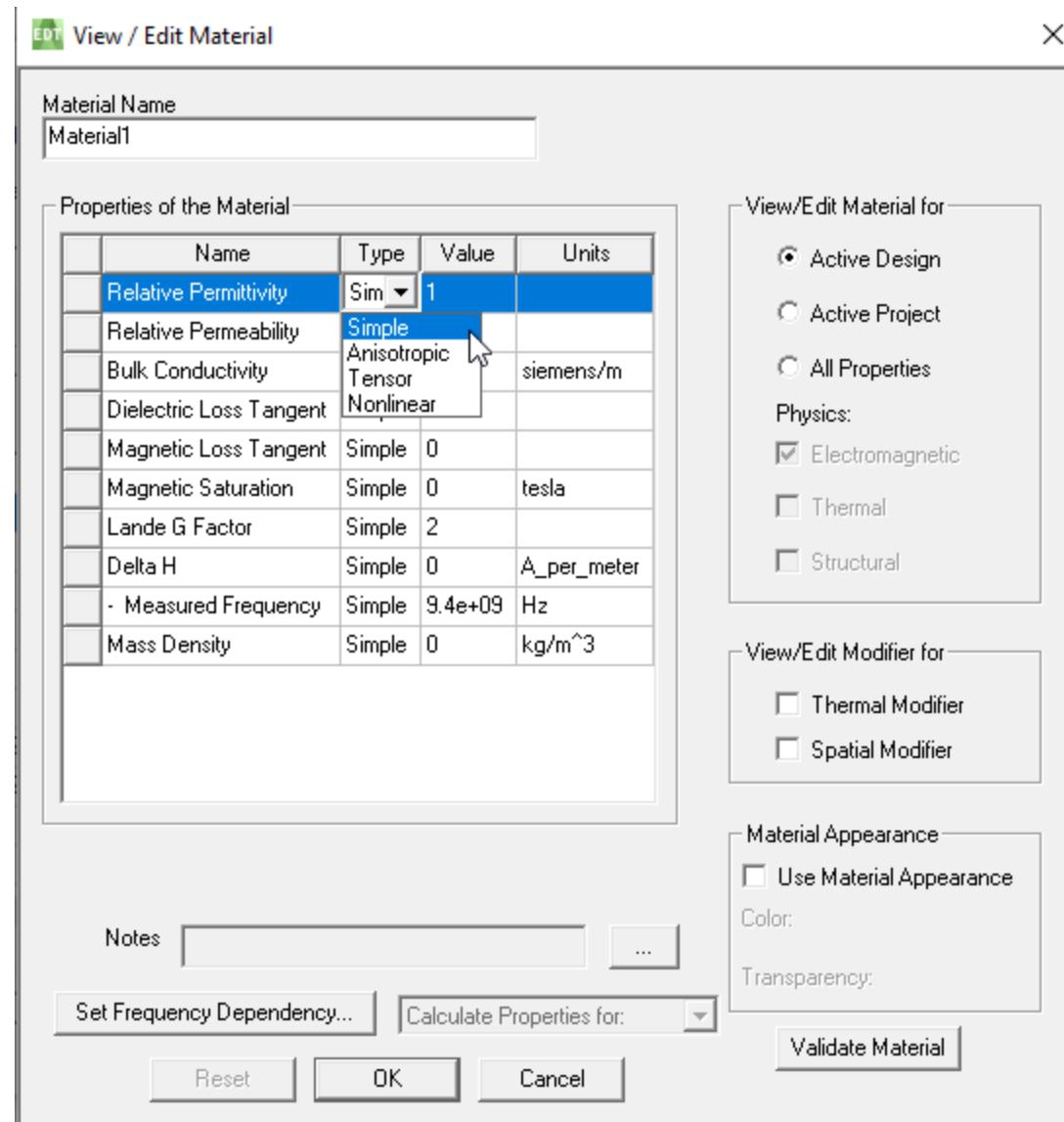
[ExampleGetLayeredImpedanceBoundaryPropertyNamesandValues.htm](#)

## Materials Scripting Support

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



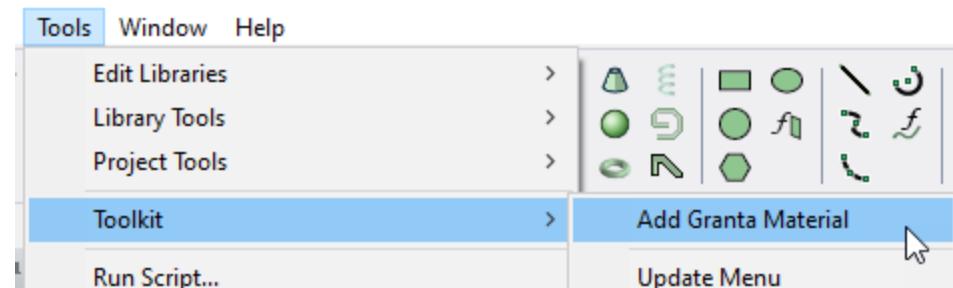
Some Material Properties like Relatively Permittivity may have values assigned as BH Curves or Tensors, as discussed in the Assigning Materials chapter of the online help.

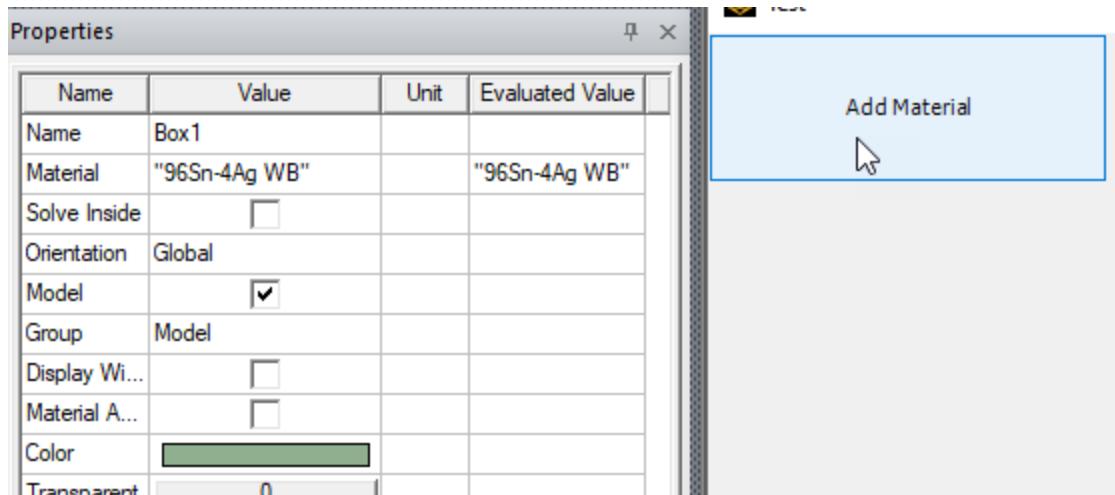


With this feature enabled you can then:

- Get/Set Simple material property
- Get/Set Anisotropic material property
- Get/Set Nonlinear material property
- Get/Set Vector material property
  - Components hide/shown as needed
- Get/Set Tensor material property
- Get/Set Choice material property
- [AddDefinitionFromBlock](#)
- [AddDefinitionFromLibFile](#)
- [GetExtendedDefinitionObject](#)

A new Toolkit allows you to select materials from the Granta materials gateway, such that project materials will automatically be added when you select a material from the gateway, and that the gateway itself is easily accessed from the materials.





What is not supported:

- Change of property type
- Custom material property, due to its complexity

## Object Oriented Scripting for Materials

Materials are Child objects of the Active Project. In the IronPython command window, you can execute GetPropNames() for a specified material as follows:

```
>>> omats = oDesktop.GetActiveProject().GetChildObject("Materials")
>>> omat = omats.GetChildObject("vacuum")
>>> omat.GetPropNames()
['Coordinate System Type', 'Coordinate System Type/Choices', 'Relative Permeability Type',
'Relative Permeability Type/Choices', 'Relative Permeability', 'Relative Permeability/SIValue',
```

```
'Relative Permeability/EvaluatedValue', 'Bulk Conductivity Type', 'Bulk Conductivity Type/Choices', 'Bulk Conductivity', 'Bulk Conductivity/SIValue', 'Bulk Conductivity/EvaluatedValue', 'Magnetic Coercivity Type', 'Magnetic Coercivity Magnitude', 'Magnetic Coercivity Magnitude/SIValue', 'Magnetic Coercivity Magnitude/EvaluatedValue', 'Composition', 'Composition/Choices', "Young's Modulus Type", "Young's Modulus Type/Choices", "Young's Modulus", "Young's Modulus/SIValue", "Young's Modulus/EvaluatedValue", "Poisson's Ratio Type", "Poisson's Ratio", "Poisson's Ratio Type/Choices", "Poisson's Ratio", "Poisson's Ratio/SIValue", "Poisson's Ratio/EvaluatedValue"]
```

### Examples showing change to material property type:

```
>>> omat.GetPropValue("Relative Permeability Type/Choices")
['Simple', 'Anisotropic', 'Tensor', 'Nonlinear']
>>> omat.GetPropValue("Relative Permeability Type")
'Nonlinear'
>>> omat.SetPropValue("Relative Permeability Type", "Simple")
True
>>> omat.GetPropValue("Relative Permeability Type")
'Simple'
>>> omat.SetPropValue("Relative Permeability", 10)
True
```

### Examples showing change to a vector component value

```
>>> omat.GetPropValue("Magnetic Coercivity Magnitude")
'0A_per_meter'
>>> omat.SetPropValue("Magnetic Coercivity Magnitude", "-1A_per_meter")
```

---

```
True
```

```
>>> omat.GetPropNames()
```

```
['Coordinate System Type', 'Coordinate System Type/Choices', 'Relative Permeability Type',  
'Relative Permeability Type/Choices', 'Relative Permeability', 'Relative Permeability/SIValue',  
'Relative Permeability/EvaluatedValue', 'Bulk Conductivity Type', 'Bulk Conductivity  
Type/Choices', 'Bulk Conductivity', 'Bulk Conductivity/SIValue', 'Bulk Con-  
ductivity/EvaluatedValue', 'Magnetic Coercivity Type', 'Magnetic Coercivity Magnitude', 'Mag-  
netic Coercivity Magnitude/SIValue', 'Magnetic Coercivity Magnitude/EvaluatedValue', 'Magnetic  
Coercivity Components', 'Magnetic Coercivity Components/Component1', 'Magnetic Coercivity Com-  
ponents/Component2', 'Magnetic Coercivity Components/Component3', 'Composition', 'Com-  
position/Choices', '- Stacking Factor Type', '- Stacking Factor', '- Stacking Factor/SIValue',  
'- Stacking Factor/EvaluatedValue', '- Stacking Direction', '- Stacking Direction/Choices',  
"Young's Modulus Type", "Young's Modulus Type/Choices", "Young's Modulus", "Young's Mod-  
ulus/SIValue", "Young's Modulus/EvaluatedValue", "Poisson's Ratio Type", "Poisson's Ratio  
Type/Choices", "Poisson's Ratio", "Poisson's Ratio/SIValue", "Poisson's Ratio/EvaluatedValue"]
```

```
>>> omat.SetPropValue("Magnetic Coercivity Components/Component2", 2)
```

```
True
```

```
>>> omat.GetPropValue("Magnetic Coercivity Components")
```

```
['Component1:=', '2', 'Component2:=', '2', 'Component3:=', '0']
```

## Change choice property value

```
>>> omat.GetPropValue("Composition/Choices")
```

```
['Solid', 'Lamination', 'Litz Wire']
```

```
>>> omat.SetPropValue("Composition", "Lamination")
```

```
True
```

```
>>> omat.GetPropNames()
```

```
['Coordinate System Type', 'Coordinate System Type/Choices', 'Relative Permeability Type',
'Relative Permeability Type/Choices', 'Relative Permeability', 'Relative Permeability/SIValue',
'Relative Permeability/EvaluatedValue', 'Bulk Conductivity Type', 'Bulk Conductivity
Type/Choices', 'Bulk Conductivity', 'Bulk Conductivity/SIValue', 'Bulk Con-
ductivity/EvaluatedValue', 'Magnetic Coercivity Type', 'Magnetic Coercivity Magnitude', 'Magnetic
Coercivity Magnitude/SIValue', 'Magnetic Coercivity Magnitude/EvaluatedValue', 'Magnetic Coer-
civity Components', 'Magnetic Coercivity Components/Component1', 'Magnetic Coercivity Com-
ponents/Component2', 'Magnetic Coercivity Components/Component3', 'Composition',
'Composition/Choices', '- Stacking Factor Type', '- Stacking Factor', '- Stacking Fact-
or/SIValue', '- Stacking Factor/EvaluatedValue', '- Stacking Direction', '- Stacking Dir-
ection/Choices', "Young's Modulus Type", "Young's Modulus Type/Choices", "Young's Modulus",
"Young's Modulus/SIValue", "Young's Modulus/EvaluatedValue", "Poisson's Ratio Type", "Poisson's
Ratio Type/Choices", "Poisson's Ratio", "Poisson's Ratio/SIValue", "Poisson's Ratio/E-
valuatedValue"]
```

```
>>> omat.SetPropValue("Magnetic Coercivity Components/Component2", 2)
```

```
True
```

```
>>> omat.GetPropValue("Magnetic Coercivity Components")
```

```
['Component1:=' , '2', 'Component2:=' , '2', 'Component3:=' , '0']
```

## Change choice property value

```
>>> omat.GetPropValue("Composition/Choices")
['Solid', 'Lamination', 'Litz Wire']
>>> omat.SetPropValue("Composition", "Lamination")
True
>>> omat.GetPropNames()
```

```
['Coordinate System Type', 'Coordinate System Type/Choices', 'Relative Permeability Type',  
'Relative Permeability Type/Choices', 'Relative Permeability', 'Relative Permeability/SIValue',  
'Relative Permeability/EvaluatedValue', 'Bulk Conductivity Type', 'Bulk Conductivity  
Type/Choices', 'Bulk Conductivity', 'Bulk Conductivity/SIValue', 'Bulk Con-  
ductivity/EvaluatedValue', 'Magnetic Coercivity Type', 'Magnetic Coercivity Magnitude', 'Mag-  
netic Coercivity Magnitude/SIValue', 'Magnetic Coercivity Magnitude/EvaluatedValue', 'Magnetic  
Coercivity Components', 'Magnetic Coercivity Components/Component1', 'Magnetic Coercivity Com-  
ponents/Component2', 'Magnetic Coercivity Components/Component3', 'Composition', 'Com-  
position/Choices', '- Stacking Factor Type', '- Stacking Factor', '- Stacking Factor/SIValue',  
'- Stacking Factor/EvaluatedValue', '- Stacking Direction', '- Stacking Direction/Choices',  
"Young's Modulus Type", "Young's Modulus Type/Choices", "Young's Modulus", "Young's Mod-  
ulus/SIValue", "Young's Modulus/EvaluatedValue", "Poisson's Ratio Type", "Poisson's Ratio  
Type/Choices", "Poisson's Ratio", "Poisson's Ratio/SIValue", "Poisson's Ratio/EvaluatedValue"]
```

>>>

This page intentionally  
left blank.

## 7 - Property Script Commands

Property script commands allow you to navigate through all objects and properties in a project. You can get and set all properties for all objects in the Project tree with simple data types.

Property Object is the base class defined for all script objects that support the properties Get and Set.

GetName ()

- Returns the name of the object.

GetChildTypes ()

- An object may have different types of children. For example, a design may have variables, modules, and editors.
- Returns an array of text strings; may be empty if the children are not divided into different types.

GetChildNames (<type>)

- <type> – Child type name. By default, returns all children names for all types.
- Returns an array of immediate children names, belonging to a type if specified.

GetChildObject (<objPath>)

- <objPath> – A child object path; can contain multiple generations (for example, designObject/moduleObject/SetupObject).
- Returns a child property object if the object is found.

GetPropNames (<bIncludeReadOnly>)

- <bIncludeReadOnly> – Optional; defaults to true. True includes read-only properties; False excludes read-only properties.
- Returns an array of the object's property names.

GetPropValue (<propertyPath>)

- <propertyPath> – The property's path; may be a child object's path appended with a property name (for example, TeeModel/Offset/SIValue).
- Returns the property value if found. Otherwise causes script error.

```
SetPropValue(<propertyPath>, <data>)
```

- <propertyPath> – The property's path; may be a child object's path appended with a property name (for example, TeeModel/Offset/SIValue).
- <data> – New data; type depends on property type.
- Returns True if updated successfully; False if new data is invalid.

For a detailed summary of how Property script commands are used in a range of contexts, including Variable objects, see: [Object Script Property Function Summary](#). Additional examples for these commands are listed under [Project Objects](#), [Design Objects](#), [3D Modeler](#), [Optimetrics](#), [Radiation Module](#) and [Reporter](#).

**Note:**

Older property commands should be executed by the oProject object.

```
Set oProject = oDesktop.SetActiveProject ("Project1")  
oProject.CommandName <args>
```

**Some of the topics covered in this chapter are as follows:**

[Conventions Used in this Chapter](#)

[ChangeProperty](#)

[GetArrayVariables](#)

[GetProperties](#)

[GetPropertyValue](#)[GetVariables](#)[GetVariableValue](#)[SetPropertyValue](#)[SetVariableValue](#)[Additional Property Scripting Example](#)[Example Use of Record Script and Edit Properties](#)

## Object Script Property Function Summary

### Object Path

The Object path can be used to navigate through objects and properties in an Ansys EM project.

- An Object path consisted of one or multiple Object-ID-Nodes separated by "/" . 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(<*bIncludeReadOnly*>);
  - <*bIncludeReadOnly*> – optional, default to true; True will include read-only properties, False will exclude read-only properties.
  - Returns an array of the object's property names.
- GetPropValue(<*propertyPath*>)
  - <*propertyPath*> – the property's full path. A property name or child object's path appended with a property name, like “TeeModel/Offset/SIValue”
  - Returns the property value if the property is found; otherwise causes script error.
- SetPropValue(<*propertyPath*>, <*data*>)
  - <*propertyPath*> – the property's full path. A property name or child object's path appended with a property name, like “TeeModel/Offset/Value”
  - <*data*> – new data, type is dependent on property type.
  - Returns True if property data is updated successfully; False if the new data is invalid.

## Project Object

Project Object inherited all functions defined in the Property Object. But it doesn't have property, GetPropValue() & SetPropValue() function can be used to set its child object's property.

- GetChildTypes() always return [“Design”, “Variable”].
- GetChildNames(type)  
GetChildNames() & GetChildName(“Design”) will return all Design names of the project.  
GetChildNames(“Variable”) return all project variable names.
- GetChildObject(objPath)

```
oDesign = oProject("TeeModel")
```

```
oVariable = oProject.GetChildObject("VariableName")
```

```
oReport = oProject.GetChildObject("TeeModel/Results/S Parameter Plot 1")
```

- GetPropNames(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("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 its properties by adding functions with property name appended to Get\_ & Set\_ prefix. To find what functions it provided enter dir(oVar) in the command window. It can be accessed by the project or design object's GetChildObject(VariableName) function.

```
oProjVar = oProject.GetChildObject("$VarName")
oVar = oProject.GetChildObject("DesignName/VarName")
oVar = oDesign.GetChildObject("variableName")
```

oProject..GetChildNames("Variable") will return all project variable names.  
oDesign.GetChildNames(Variable") will return all Design Variable names.

- GetChildTypes() always return empty array.
- GetChildNames() always return empty array , since variable has no child.
- GetChildObject(objPath) it has no child.
- GetPropNames(bIncludeReadOnly) ['EvaluatedValue', 'SIValue'] are read-only properties
  - Independent variable :['Value', 'EvaluatedValue', 'SIValue', 'Description', 'ReadOnly', 'Hidden', 'Sweep', 'Optimization/Included', 'Optimization/Min', 'Optimization/Max', 'Sensitivity/Included', 'Sensitivity/Min', 'Sensitivity/Max', 'Sensitivity/IDisp', 'Statistical', 'Statistical/Included', 'Tuning/Included', 'Tuning/Step', 'Tuning/Min', 'Tuning/Max'].
  - Dependent variable ['Value', 'EvaluatedValue', 'SIValue', 'Description', 'ReadOnly', 'Hidden', 'Sweep']
- GetPropValue(propName)
  - oVar.GetPropValue() return the variable value as text string.
  - oVar.GetPropValue("Value") return the variable value as text string.
  - oVar.GetPropValue("SIValue") return the SI-value of variable as number.
  - oVar.Get\_SIValue() also return the SI value.
- SetPropValue(propName, newValue)
  - oVar.SetPropValue("Value", 888)
  - oVar.SetPropValue("Sensitivity/Included", True)
  - oVar.SetPropValue("Sensitivity/Max", '1.8pF')
  - oVar.Set\_Sensitivity\_Max('1.8pF') also works as last call.
  - oVar.SetPropValue("Sensitivity", ['Min:=' , '0.8pF', 'Max:=' , '1.8pF'])
  - //set multiple attributes at one call:
    - oVar.SetPropValue("@", ["Value:=' , 288, 'Sensitivity', ['Included', True, 'Min', '0.0']] )
    - oVar.SetPropValue("", ["Value:=' , 288, 'Sensitivity', ['Included', True, 'Min', '0.0']] )

## Optimetrics Module Object:

Optimetrics Module Object inherited all functions defined in the Property Object. But it doesn't have property, GetPropValue() & SetPropValue() function can be used to set its child object's property..

- GetChildTypes() there are six type of children, they are ['OptiParametric', 'OptiOptimization', 'OptiSensitivity', 'OptiStatistical', 'OptiDesignExplorer', 'OptiDXDOE']. But the return array only included those that have setup defined, so it may be an empty array if no optimetrics setup is defined. The GetChildNames(type) function also recognized the type name without the prefix "Opti".
- GetChildNames(type)  
GetChildNames() will return all setup for all types.  
GetChildNames("OptiOptimization") & GetChildNames("Optimization") will return all Optimization setup.
- GetChildObject()  
oParamSetup = oOptModule.GetChildObject('ParametricSetup1') get the  
oOptSetup = oOptModule.GetChildObject('OptimizationSetup1')
- GetPropNames(bIncludeReadOnly) always return empty array since the it has no property.
- GetPropValue(propPath) may be used to get its child's property value  
oOptModule.GetPropValue("OptimizationSetup1\Optimizer") get the optimizer name for OptimizationSetup1
- SetPropValue(propPath, newValue) may be used to set its child's property value  
oOptModule.SetPropValue(ParametricSetup1\Enabled", False) //disable ParametricSetup1

## Optimetrics Setup Object

This is a new Object inherited all functions defined in the Property Object. But it doesn't have child. It is accessible through its parents.

```
oOptSetup = oOptModule.GetChildObject('OptimizationSetup1')
oOptSetup = oDesign.GetChildObject('Optimetrics\OptimizationSetup1')
oOptSetup = oProject.GetChildObject('TeeModel\Optimetrics\OptimizationSetup1')
```

- GetChildTypes() always return empty array.
- GetChildNames(type) always return empty array
- GetChildObject()
- GetPropNames(bIncludeReadOnly) will return the property names listed in the property window when the setup is selected.
- GetPropValue(propName)
  - oOptSetup.GetPropValue("Optimizer") return the selected optimizer name.
  - oOptSetup.GetPropValue("Optimizer/Choices") return all optimizer names.
- SetPropValue(propName, newValue)
  - oOptSetup.SetPropValue("Optimizer", "NotAnOptimzerName"); will return false.
  - oOptSetup.SetPropValue("Optimizer", "Quasi Newton"); return true, since "Quasi Newton" is one of the optimizer name returned as the Optimizer Choices.
- HasResult() return true if the setup is solved. Otherwise return false.
- Validate() return true if the setup is valid for analyze. Otherwise return false. Calling the SetPropValue() function to change the property may invalid the setup.

### **ReportSetup(Results) Module Object:**

ReportSetup module Object inherited all functions defined in the Property Object. But it doesn't have property, GetPropValue() & SetPropValue() function can be used to get/set its child object's property..

- GetChildTypes() always empty array.
- GetChildNames(type)
  - GetChildNames() return all report names
- •GetChildObject(objPath)
  - oRpt = oRptModule.GetChildObject("S Parameter Plot 1") return the report property object

```
oTrace = oRptModule.GetChildObject("S Parameter Plot 1/dB(S(Port1,Port1))") return the trace property object  
oAxisX = oRptModule.GetChildObject("S Parameter Plot 1/AxisX") return the axis X property object
```

- GetPropNames(bIncludeReadOnly) always return empty array since the it \has no property.
- GetPropValue()  
oRptModule.GetPropValue("S Parameter Plot 1/Display Type")
- SetPropValue()  
oRptModule.SetPropValue("S Parameter Plot 1/Display Type", "DataTable")

## ReportSetup(Results) Module Child Objects:

These are Property Objects. Its first level of child object is report. Report has trace, axis, header, Legend, and more children. Trace has curve as child etc.

Those child objects can be accessed by calling all levels of parent object's GetChildObject(path) function.

```
oRpt = oRptModule.GetChildObject(reportName)
```

```
oRpt = oDesign.GetChildObject("Results/reportName")
```

```
oTrace = oRpt.GetChildObject(traceName)
```

```
oTrace = oRptModule.GetChildObject(ReportName/TraceName)
```

- GetChildTypes() always return empty array.
- GetChildNames() get the object's child names. What will be returned will depended on the object instance.
- GetChildObject(objPath)
- GetPropNames(bIncludeReadOnly) will return the property names listed in the property window when the object is selected.
- GetPropValue(propName)  
oRpt.GetPropValue("Display Type") return the report's display type.  
oOptSetup.GetPropValue("Display Type/Choices") return all optimizer names.  
oTrace.GetPropValue("X Component")

- SetPropValue(propName, newValue)  
oTrace.SetPropValue("Primary sweep", "Freq")

### Radiation Module Object:

This inherited all functions defined in the Property Object. But it doesn't have property, GetPropValue() & SetPropValue() function can be used to set its child object's property.

- GetChildTypes() always return empty array, now its children
- GetChildNames(type)  
GetChildNames() return all setup names.
- GetChildObject(setupName) return the setup object as Property object.  
oOverlay= oRadModule.GetChildObject('Antenna Parameter Overlay1')  
oSphere = oRadModule.GetChildObject('Infinite Sphere1')
- GetPropNames() return empty array; it has no property.
- GetPropValue()  
oRadModule.GetPropValue('Line1/Num Points') //Get the the Line1 setups' "Num Points" property value.
- SetPropValue()  
oRadModule.SetPropValue('Line1/Num Points', 100); Set the Line1 setups' "Num Points" property to 100.

### Radiation Module Child Objects:

These are Property Objects. It also provides quick function call to get/set its properties by adding functions with property name appended to Get\_ & Set\_ prefix. To find what functions it provides enter dir(oVar) the command window.

Those child objects can be access by call all levels of parent object's GetChildObject(path) function.

```
oRadSetup = oRadModule.GetChildObject(setupName)  
oRadSetup = oDesign.GetChildObject(RadField/setupName)
```

- GetChildTypes() always return empty array.
- GetChildNames() always return empty array , since Radiation setup has no child.
- GetChildObject(objPath) it has no child.
- GetPropNames(bIncludeReadOnly) will return the property names listed in the property window when the setup is selected.
- GetPropValue(propName)
  - oRadSetup.GetPropValue("Num Points") return the line setup's "Num Points" property value.
  - oRadSetup.Get\_NumPoints() will also get the same value.
- SetPropValue(propName, newValue)
  - oRadSetup.SetPropValue('Num Points', 888)
  - oRadSetup.Set\_NumPoints(888)

## Conventions Used in this Chapter

General Definitions:

<b>Property</b>	A single item that can be modified in the <b>Properties</b> window or in the modal <b>Properties</b> pop-up window.
<b>&lt;PropServer&gt;</b>	The item whose properties are being modified. This is usually a compound name, giving all information needed by the editor, design, or project in order to locate the item.
<b>&lt;PropTab&gt;</b>	Corresponds to one tab in the <b>Properties</b> window, the one under which properties are being edited.
<b>&lt;PropName&gt;</b>	The name of a single property.

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

For Project Variables:

<b>&lt;PropServer&gt;</b>	"ProjectVariables"
<b>&lt;PropTab&gt;</b>	"ProjectVariableTab"

For Local Variables:

<b>&lt;PropServer&gt;</b>	"LocalVariables"
---------------------------	------------------

<b>&lt;PropTab&gt;</b>	"LocalVariableTab"
------------------------	--------------------

For Passed Parameters:

<b>&lt;PropServer&gt;</b>	"Instance:<Name of Circuit Instance>"
<b>&lt;PropTab&gt;</b>	"PassedParameter Tab"

For Definition Parameters:

<b>&lt;PropServer&gt;</b>	"DefinitionParameters"
<b>&lt;PropTab&gt;</b>	"DefinitionParameters"

For Modules and Editors:

<b>&lt;PropServer&gt;</b>	<ModuleName>:<ItemName> where <ItemName> is the boundary name, solution setup name, etc. For example, "BoundarySetup:PerfE1"
<b>&lt;PropTab&gt;</b>	Boundary Module: "HfssTab"  Mesh Operations Module: "MeshSetupTab"  Analysis Module: "HfssTab"  Optimetrics Module: "OptimetricsTab"  Solutions Module: <i>Does not support properties.</i>  Field Overlays Module: "FieldsPostProcessorTab"  Radiation Module: "RadFieldSetupTab"  Circuit Module: "CCircuitTab"  System Module: "SystemTab"  HFSS 3D Layout Module: "HFSS 3D LayoutTab"

	Nexxim Module: "NexximTab" Layout elements: "BaseElementTab" Schematic elements: "ComponentTab" Optimetrics Module: "OptimetricsTab"
--	-----------------------------------------------------------------------------------------------------------------------------------------------

For 3D Model Editor objects:

<b>&lt;PropServer&gt;</b>	Name of the object. For example, "Box1".
<b>&lt;PropTab&gt;</b>	"Geometry3DAttributeTab"

For 3D Model Editor operations:

<b>&lt;PropServer&gt;</b>	<ObjName>:<OperationName>:<int> where <int> is the operation's history index. For example, "Box2:CreateBox:2" refers to the second "CreateBox" operation in Box2's history.
<b>&lt;PropTab&gt;</b>	"Geometry3DCmdTab"

For Reporter operations on Report properties:

<b>&lt;PropServer&gt;</b>	<b>&lt;ReportSetup&gt;</b>
<b>&lt;ChangeProperty&gt;</b>	Array. For example, to set the company name in a plot header to "My Company":  Set oModule = oDesign.GetModule("ReportSetup")  oModule.ChangeProperty Array("NAME:AllTabs", _ Array("NAME:Header", _ Array("NAME:PropServers", _  "XY Plot1:Header"), Array("NAME:ChangedProps", _ Array("NAME:Company Name", "Value:=", "My Company"))))

**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
```

<b>&lt;propType&gt;</b>	<b>&lt;propTypeName&gt;</b>	<b>Property Description</b>
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

<b>&lt;tabType&gt;</b>	<b>Description</b>	<b>Objects with this tab type</b>
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>,
          <NewPropsArray>,
          <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 **RMxprt>Design Properties** panels. Specify the new zero-based row index of the variable or separator.

#### **ReadOnly**

Used to mark a property as "read only" so it can not be modified. In HFSS and Maxwell, this flag can only be set for variables and separators.

#### **Hidden**

Used to hide a property so it can not be viewed outside of the **Property** dialog box. In HFSS and Maxwell, this flag can only be set for variables and separators.

```
<PropTypeSpecificArgs>

    SeparatorProp: no arguments

    TextProp: "Value:=", <string>

    NumberProp: "Value:=", <double>

    ValueProp: "Value:=", <value>

    CheckboxProp: "Value:=", <bool>

    MenuProp: "Value:=", <string>

    PointProp "X:=", <double>, "Y:=", <double>

    VPointProp: "X:=", <value>, "Y:=", <value>

    V3DPointProp: "X:=", <value>, "Y:=", <value>,
                  "Z:=", <value>

    Material Button: "Material:=", <string>

    Color Button: "R:=", <int>, "G:=", <int>, "B:=", <int>

    Transparency Button: "Value:=", <double>
```

<PropTypeSpecificArgs> for VariableProps

Syntax:

```
"Value:=", <value>, <OptimizationFlagsArray>,
<TuningFlagsArray>, <SensitivityFlagsArray>,
<StatisticsFlagsArray>
```

Parameters:

```
<OptimizationFlagsArray>
```

```
Array("NAME:Optimization",
```

```
"Included:=", <bool>,
```

```
"Min:=", <value>,
```

```
"Max:=", <value>)
```

```
<Tuning_flagsArray>
```

```
Array("NAME:Tuning",
```

```
"Included:=", <bool>,
```

```
"Step:=", <value>,
```

```
"Min:=", <value>,
```

```
"Max:=", <value>)
```

```
<SensitivityFlagsArray>
```

```
Array("NAME:Sensitivity",
```

```
"Included:=", <bool>,
```

```
"Min:=", <value>,
```

```
"Max:=", <value>,
```

```
"IDisp:=", <value> )
```

```
<StatisticsFlagsArray>  
Array("NAME:Statistical",  
"Included:=", <bool>,  
"Dist:=", <Distribution>,  
"StdD:=", <value>,  
"Min:=", <value>,  
"Max:=", <value>,  
"Tol:=", <string>)
```

#### **<Distribution>**

Type: string

Value should be "Gaussian" or "Uniform"

#### **StdD**

Standard deviation.

#### **Min**

Low cut-off for the distribution.

#### **Max**

High cut-off for the distribution.

**Tol**

Tolerance for uniform distributions. Format is "<int>%".

Example: "20%".

*VB Example:* Adding a new project level variable "\$width":

```
oProject.ChangeProperty Array("NAME:AllTabs",_
    Array("NAME:ProjectVariableTab",_
        Array("NAME:PropServers", "ProjectVariables"),_
        Array("NAME:NewProps",_
            Array("NAME:$width",_
                "PropType:=", "VariableProp",_
                "Value:=", "3mm",_
                "Description:=", "my new variable"))))
```

*VB Example:* Deleting the design level variable "height":

```
oDesign.ChangeProperty Array("NAME:AllTabs",_
```

```
Array("NAME:LocalVariableTab",_
      Array("NAME:PropServers", "DefinitionParameters"),_
      Array("NAME:DeletedProps", "height"))
```

*VB Example:* Changing a property's value. If the following command were executed, then the value of the property "XSize" of the PropServer "Box1:CreateBox:1" on the "Geometry3DCmdTab" tab would be changed. (oEditor is the Geometry3D editor in HFSS.)

```
oEditor.ChangeProperty Array("NAME:AllTabs",_
                      Array("NAME:Geometry3DCmdTab",_
                      Array("NAME:PropServers","Box1:CreateBox:1"),_
                      Array("NAME:ChangedProps",_
                      Array("NAME:XSize", "Value:=", "1.4mil"))))
```

*VB Example:* Changing the Company Name, Design Name, the background color, and the Axis scaling in a Report.

```
Set oProject = oDesktop.SetActiveProject("wgcombiner")
Set oDesign = oProject.SetActiveDesign("HFSSDesign2")
Set oModule = oDesign.GetModule("ReportSetup")

oModule.ChangeProperty Array("NAME:AllTabs", Array("NAME:Header", _ Array("NAME:PropServers",
"XY Plot1:Header"), _
Array("NAME:ChangedProps", Array("NAME:Company Name", _
```

```
"Value:=", "My Company")))

oModule.ChangeProperty Array("NAME:AllTabs", Array("NAME:Header", _ Array("NAME:PropServers", "XY
Plot1:Header"), _
Array("NAME:ChangedProps", Array("NAME:Design Name", _
"Value:=", "WG Combiner")))

oModule.ChangeProperty Array("NAME:AllTabs", Array("NAME:General", _ Array("NAME:PropServers", "XY
Plot1:General"), _
Array("NAME:ChangedProps", Array("NAME:Back Color", _
"R:=", 128, "G:=", 255, "B:=", 255)))))

oModule.ChangeProperty Array("NAME:AllTabs", Array("NAME:Axis", _ Array("NAME:PropServers", "XY
Plot1:AxisX"), _
Array("NAME:ChangedProps", Array("NAME:Axis Scaling", _
"Value:=", "Log"))))
```

## PropHost Functions

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

### 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
-----------	----

	Name	Type	Description
<b>Parameters</b>	<tabType>	Integer	<p>One of the following, where tab titles are shown in parentheses:</p> <ul style="list-style-type: none"> <li>1 for PassedParameterTab ("Parameter Values")</li> <li>2 for DefinitionParameterTab ("Parameter Defaults")</li> <li>3 for LocalVariableTab ("Variables" or "Local Variables")</li> <li>4 for ProjectVariableTab ("Project variables")</li> <li>5 for ConstantsTab ("Constants")</li> <li>6 for BaseElementTab ("Symbol" or "Footprint")</li> <li>7 for ComponentTab ("General")</li> <li>28 for CustomTab ("Intrinsic Variables")</li> <li>39500 for Quantities ("Quantities")</li> <li>39506 for Signals ("Signals")</li> <li>39512 for Component ("Component")</li> </ul>
	<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.
<b>Return Value</b>	None		

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

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

## AddMenuProp2

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

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

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

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

## 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	<p>One of the following, where tab titles are shown in parentheses:</p> <ul style="list-style-type: none"> <li>1 for PassedParameterTab ("Parameter Values")</li> <li>2 for DefinitionParameterTab (Parameter Defaults")</li> <li>3 for LocalVariableTab ("Variables" or "Local Variables")</li> <li>4 for ProjectVariableTab ("Project variables")</li> <li>5 for ConstantsTab ("Constants")</li> <li>6 for BaseElementTab ("Symbol" or "Footprint")</li> <li>7 for ComponentTab ("General")</li> <li>28 for CustomTab ("Intrinsic Variables")</li> <li>39500 for Quantities ("Quantities")</li> <li>39506 for Signals ("Signals")</li> <li>39512 for Component ("Component")</li> </ul>
	<propType>	Integer	<p>One of the following to add new definition properties (e.g., to component, design or project):</p> <ul style="list-style-type: none"> <li>0 for TextProp (any string)</li> <li>1 for Menu (one,two,three)</li> <li>2 for CheckboxProp ("true" or "false" values)</li> <li>3 for VariableProp (useful for Project and Local Variables, expression)</li> <li>4 for VPointProp (x and y values, which can be expressions)</li> </ul>

		<p>5 for V3DPointProp (x, y and z values, which can be expressions)</p> <p>6 for NumberProp (simple number, no expression)</p> <p>7 for ColorProp (takes an RGB value)</p> <p>8 for PointProp (x and y values, no expressions)</p> <p>11 for SeparatorProp (used for property list organization, the value is the separator "title")</p> <p>13 for FileNameProp (special button property for filenames)</p> <p>39506 for GenericProp (for generics in Twin Builder, SML and VHDL)</p> <p>39500 for QuantityProp (for Twin Builder quantities, SML and VHDL)</p> <p>39504 for SignalProp (for VHDL signals in Twin Builder)</p> <p>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, QuantitiyProp in Quantities, and signalProp in Signals)</p>
	<propName>	String Name of the new menu property.
	<valueText>	String The property value
<b>Return Value</b>	None	

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

<b>VB Syntax</b>	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	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;tabType&gt;</td> <td>Type</td> <td>           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")         </td> </tr> <tr> <td>&lt;propType&gt;</td> <td>String</td> <td>One of the following to add new definition properties (e.g., to component, design or project):</td> </tr> </tbody> </table>			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):
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):										

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

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

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

## ExecuteScript

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

<b>UI Access</b>	NA						
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;scriptName&gt;</td> <td>String</td> <td>Name of a script stored in a script library</td> </tr> </tbody> </table>	Name	Type	Description	<scriptName>	String	Name of a script stored in a script library
Name	Type	Description					
<scriptName>	String	Name of a script stored in a script library					
<b>Return Value</b>	None						

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

<b>VB Syntax</b>	ExecuteScript(<scriptName>)
------------------	-----------------------------

**VB Example**

```
PropHost.ExecuteScript("PropChangeScript")
```

## GetApplication

Returns the application object currently running the script.

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

<b>UI Access</b>	NA						
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;propName&gt;</td> <td>String</td> <td>Name of the property</td> </tr> </tbody> </table>	Name	Type	Description	<propName>	String	Name of the property
Name	Type	Description					
<propName>	String	Name of the property					
<b>Return Value</b>	String						

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

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

## GetChangedProperty

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

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

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

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

## GetDescription

Finds named property and returns description string

<b>UI Access</b>	NA						
<b>Parameters</b>	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>&lt;propName&gt;</td><td>String</td><td>Name of the property</td></tr></tbody></table>	Name	Type	Description	<propName>	String	Name of the property
Name	Type	Description					
<propName>	String	Name of the property					
<b>Return Value</b>	String						

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

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

## 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.

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

## GetEvaluatedText

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

<b>UI Access</b>	NA								
<b>Parameters</b>	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>&lt;propname&gt;</td><td>String</td><td>The name of the property</td></tr></tbody></table>			Name	Type	Description	<propname>	String	The name of the property
Name	Type	Description							
<propname>	String	The name of the property							
<b>Return Value</b>	<p>String The Evaluated Value</p>								

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

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

## GetFileName

Finds the full path name to propName.

<b>UI Access</b>	NA					
<b>Parameters</b>	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead></table>			Name	Type	Description
Name	Type	Description				

	<propName>	String	Name of the property
<b>Return Value</b>	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.

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

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

**GetHidden**

Finds named property and returns its Hidden flag.

<b>UI Access</b>	NA						
<b>Parameters</b>	<table border="1"> <tr> <td>Name</td> <td>Type</td> <td>Description</td> </tr> <tr> <td>&lt;propName&gt;</td> <td>String</td> <td>Name of the property</td> </tr> </table>	Name	Type	Description	<propName>	String	Name of the property
Name	Type	Description					
<propName>	String	Name of the property					

<b>Return Value</b>	Returns 1 if property is hidden and 0 if it is not.
---------------------	-----------------------------------------------------

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

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

## 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 ComplInstance, 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](#).

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

## GetPropServers

Returns array of objects that have properties showing on tabTypeName.

<b>UI Access</b>	NA						
<b>Parameters</b>	<table border="1"> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> <tr> <td>None</td> <td></td> <td></td> </tr> </table>	Name	Type	Description	None		
Name	Type	Description					
None							
<b>Return Value</b>	Array of propserver names.						

<b>Python Syntax</b>	GetPropServers(<tabTypeName>)
<b>Python Example</b>	<pre>objects = PropHost.GetPropServers ("PassedParameterTab")</pre> <p>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 "ComInst@CAP_1".</p>

<b>VB Syntax</b>	GetPropServers(<tabTypeName>)
<b>VB Example</b>	<code>objects = PropHost.GetPropServers ("PassedParameterTab")</code>

	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 "ComplInst@CAP_1".
--	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

## GetPropTabType

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

<b>UI Access</b>	NA						
<b>Parameters</b>	<table border="1"><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;propName&gt;</td><td>String</td><td>Name of the property</td></tr></table>	Name	Type	Description	<propName>	String	Name of the property
Name	Type	Description					
<propName>	String	Name of the property					
<b>Return Value</b>	Returns string						

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

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

## GetReadOnly

Finds named property and returns its ReadOnly flag.

<b>UI Access</b>	NA						
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;propName&gt;</td> <td>String</td> <td>Name of the property</td> </tr> </tbody> </table>	Name	Type	Description	<propName>	String	Name of the property
Name	Type	Description					
<propName>	String	Name of the property					
<b>Return Value</b>	Returns 1 if property is read-only and 0 if it is not.						

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

<b>VB Syntax</b>	GetReadOnly(<propName>)
<b>VB Example</b>	<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.

<b>UI Access</b>	NA						
<b>Parameters</b>	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>&lt;propName&gt;</td><td>String</td><td>Name of the property</td></tr></tbody></table>	Name	Type	Description	<propName>	String	Name of the property
Name	Type	Description					
<propName>	String	Name of the property					
<b>Return Value</b>	String						

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

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

## GetText

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

<b>UI Access</b>	NA						
<b>Parameters</b>	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>&lt;propName&gt;</td><td>String</td><td>Name of the property</td></tr></tbody></table>	Name	Type	Description	<propName>	String	Name of the property
Name	Type	Description					
<propName>	String	Name of the property					

<b>Return Value</b>	String
---------------------	--------

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

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

## 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.

<b>UI Access</b>	NA						
<b>Parameters</b>	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>&lt;propname&gt;</td><td>String</td><td>The name of the property.</td></tr></tbody></table>	Name	Type	Description	<propname>	String	The name of the property.
Name	Type	Description					
<propname>	String	The name of the property.					
<b>Return Value</b>	Integer; non-zero for constant, zero for variable or expression.						

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

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

## PropertyExists

Finds named property and returns its property type.

<b>UI Access</b>	NA
------------------	----

<b>Parameters</b>	<table border="1"> <tr> <th>Name</th><th>Type</th><th>Description</th></tr> <tr> <td>&lt;propName&gt;</td><td>String</td><td>Name of the property</td></tr> </table>	Name	Type	Description	<propName>	String	Name of the property
Name	Type	Description					
<propName>	String	Name of the property					
<b>Return Value</b>	Returns 1 if property exists in any tab, 0 if it does not.						

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

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

## RemoveProp

Removes the named property from whichever tab it is found.

<b>UI Access</b>	NA						
<b>Parameters</b>	<table border="1"> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> <tr> <td>&lt;propName&gt;</td> <td>String</td> <td>Name of the property</td> </tr> </table>	Name	Type	Description	<propName>	String	Name of the property
Name	Type	Description					
<propName>	String	Name of the property					
<b>Return Value</b>	None						

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

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

## SetCallback

Finds named property and sets its Callback script.

<b>UI Access</b>	NA									
<b>Parameters</b>	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>&lt;propName&gt;</td><td>String</td><td>Name of the property</td></tr><tr><td>&lt;scriptName&gt;</td><td>String</td><td>Name of a script stored in a script library.</td></tr></tbody></table>	Name	Type	Description	<propName>	String	Name of the property	<scriptName>	String	Name of a script stored in a script library.
Name	Type	Description								
<propName>	String	Name of the property								
<scriptName>	String	Name of a script stored in a script library.								
<b>Return Value</b>	None									

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

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

## SetDescription

Finds named property and sets its description text.

<b>UI Access</b>	NA									
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;propName&gt;</td> <td>String</td> <td>Name of the property</td> </tr> <tr> <td>&lt;description&gt;</td> <td>String</td> <td>Descriptive text.</td> </tr> </tbody> </table>	Name	Type	Description	<propName>	String	Name of the property	<description>	String	Descriptive text.
Name	Type	Description								
<propName>	String	Name of the property								
<description>	String	Descriptive text.								
<b>Return Value</b>	None									

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

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

## SetHidden

Finds named property and sets its Hidden flag.

<b>UI Access</b>	NA									
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;propName&gt;</td> <td>String</td> <td>Name of the property</td> </tr> <tr> <td>&lt;hidden&gt;</td> <td>Boolean</td> <td>True or 1, false or 0.</td> </tr> </tbody> </table>	Name	Type	Description	<propName>	String	Name of the property	<hidden>	Boolean	True or 1, false or 0.
Name	Type	Description								
<propName>	String	Name of the property								
<hidden>	Boolean	True or 1, false or 0.								

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

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

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

## SetReadOnly

Finds named property and sets its ReadOnly flag.

<b>UI Access</b>	NA		
<b>Parameters</b>	Name	Type	Description
	<propName>	String	Name of the property
<b>Return Value</b>	None		

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

---

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

## SetText

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

<b>UI Access</b>	NA									
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;propName&gt;</td> <td>String</td> <td>Name of the property</td> </tr> <tr> <td>&lt;text&gt;</td> <td>String</td> <td>The value for the property.</td> </tr> </tbody> </table>	Name	Type	Description	<propName>	String	Name of the property	<text>	String	The value for the property.
Name	Type	Description								
<propName>	String	Name of the property								
<text>	String	The value for the property.								
<b>Return Value</b>	None									

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

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

## 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
Return Value	None		

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

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

### ChangeProperty (HFSS and Maxwell)

*Use:* Changes to properties are scripted using the ChangeProperty command. This command can be executed by the oEditor to change editor properties, by the oDesign to change design level properties, and by the oProject to change project level properties. The command can be used to create, edit, and/or remove properties. In HFSS and Maxwell, only Variable and Separator properties can be deleted.

Use the script recording feature and edit a property, and then view the resulting script entry or use GetPropertyValue for the desired property to see the expected format.

*Command:* None

*Syntax:* ChangeProperty Array("Name:AllTabs", <PropTabArray>, <PropTabArray>, ...)

ChangeProperty(<modulename>:<setup name>:<sweep name>)

*Return Value:* None

*Parameters:* <PropTabArray>

```
Array ("Name:<PropTab>",
<PropServersArray>,
<NewPropsArray>,
<ChangedPropsArray>,
<DeletedPropsArray>)
```

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

```
<NewPropsArray>
Array ("Name:NewProps", <PropdataArray>,
<PropdataArray>, ...)
```

```
<ChangedPropsArray>
Array ("Name:ChangedProps", <PropdataArray>,
<PropdataArray>, ...)
```

```
<DeletedPropsArray>  
  Array ("Name:DeletedProps", <PropName>,  
         <PropName>, ...)
```

```
<PropdataArray>  
  Array ("NAME:<PropName>",<br/>  
         "PropType:=", <PropType>,<br/>  
         "NewName:=", <string>,<br/>  
         "Description:=", <string>,<br/>  
         "NewRowPosition:=", <int>,<br/>  
         "ReadOnly:=", <bool>,<br/>  
         "Hidden:=", <bool>,<br/>  
         <PropTypeSpecificArgs>)
```

```
<PropType>
```

Type: string

Identifies the type of property when a new property is added. In HFSS and Maxwell, only separator properties and variable properties can be added.

```
"SeparatorProp"  
"VariableProp"  
"TextProp"
```

```
"NumberProp"  
"ValueProp"  
"CheckboxProp"  
"MenuProp"  
"PointProp"  
"VPointProp"  
"V3DPointProp"  
"ButtonProp"
```

**NewName**

Specify the new name of a property if the property's name is being edited. In HFSS and Maxwell, the name can only be changed for separators and variables.

**Description**

Specify a description of the property. In HFSS and Maxwell, the description can only be changed for separators and variables.

**NewRowPosition**

Used to reorder rows in the **Property** dialog box.

In HFSS, this only applies to the **Project>Project Variables** panel and the **Hfss>DesignProperties** panel.

In Maxwell, this only applies to the **Project>Project Variables** panel and the **Maxwell3D>DesignProperties**, **Maxwell2D>Design Properties**, or **RMxprt>Design Properties** panels.

Specify the new zero-based row index of the variable or separator.

---

### **ReadOnly**

Used to mark a property as "read only" so it can not be modified. In HFSS and Maxwell, this flag can only be set for variables and separators.

### **Hidden**

Used to hide a property so it can not be viewed outside of the **Property** dialog box. In HFSS and Maxwell, this flag can only be set for variables and separators.

```
<PropTypeSpecificArgs>
    SeparatorProp: no arguments
    TextProp: "Value:=", <string>
    NumberProp: "Value:=", <double>
    ValueProp: "Value:=", <value>
    CheckboxProp: "Value:=", <bool>
    MenuProp: "Value:=", <string>
    PointProp "X:=", <double>, "Y:=", <double>
    VPointProp: "X:=", <value>, "Y:=", <value>
    V3DPointProp: "X:=",<value>, "Y:=",<value>, "Z:=",<value>
    Material Button: "Material:=", <string>
    Color Button: "R:="",<int>, "G:="",<int>, "B:="",<int>
```

Transparency Button: "Value:=", <double>

<PropTypeSpecificArgs> for VariableProps

Syntax:

```
"Value:=", <value>, <OptimizationFlagsArray>,
<TuningFlagsArray>, <SensitivityFlagsArray>,
<StatisticsFlagsArray>
```

Parameters:

<OptimizationFlagsArray>

```
Array("NAME:Optimization",
"Included:=", <bool>,
"Min:=", <value>,
"Max:=", <value>)
```

<Tuning flagsArray>

```
Array("NAME:Tuning",
"Included:=", <bool>,
"Step:=", <value>,
"Min:=", <value>,
"Max:=", <value>)
```

```
<SensitivityFlagsArray>
Array("NAME:Sensitivity",
"Included:=", <bool>,
"Min:=", <value>,
"Max:=", <value>,
"IDisp:=", <value> )
```

```
<StatisticsFlagsArray>
Array("NAME:Statistical",
"Included:=", <bool>,
"Dist:=", <Distribution>,
"StdD:=", <value>,
"Min:=", <value>,
"Max:=", <value>,
"Tol:=", <string>)
```

```
<Distribution>
  Type: string
    Value should be "Gaussian" or "Uniform"
```

StdD

Standard deviation.

Min

Low cut-off for the distribution.

Max

High cut-off for the distribution.

Tol

Tolerance for uniform distributions. Format is "<int>%".

Example: "20%".

*VB Example:* Adding a new project level variable "\$width":

```
oProject.ChangeProperty Array("NAME:AllTabs",_
    Array("NAME:ProjectVariableTab",_
        Array("NAME:PropServers", "ProjectVariables"),_
        Array("NAME:NewProps",_
            Array("NAME:$width",_
```

```
"PropType:=", "VariableProp",  
"Value:=", "3mm",  
"Description:=", "my new variable")))
```

*VB Example:* Deleting the design level variable "height":

```
oDesign.ChangeProperty Array("NAME:AllTabs",  
    Array("NAME:LocalVariableTab",  
        Array("NAME:PropServers", "DefinitionParameters"),  
        Array("NAME:DeletedProps", "height")))
```

*VB Example:* Changing a property's value. If the following command were executed, then the value of the property "XSize" of the PropServer "Box1:CreateBox:1" on the "Geometry3DCmdTab" tab would be changed. (oEditor is the Geometry3D editor in HFSS.)

```
oEditor.ChangeProperty Array("NAME:AllTabs",  
    Array("NAME:Geometry3DCmdTab",  
        Array("NAME:PropServers", "Box1:CreateBox:1"),  
        Array("NAME:ChangedProps",  
            Array("NAME:XSize", "Value:=", "1.4mil"))))
```

*VB Example:* Changing the Company Name, Design Name, the background color, and the Axis scaling in a Report.

```
Set oProject = oDesktop.SetActiveProject("wgcombiner")
Set oDesign = oProject.SetActiveDesign("HFSSDesign2")
Set oModule = oDesign.GetModule("ReportSetup")
oModule.ChangeProperty Array("NAME:AllTabs", Array("NAME:Header", _
    Array("NAME:PropServers", "XY Plot1:Header"), _
    Array("NAME:ChangedProps", Array("NAME:Company Name", _
        "Value:=", "My Company"))))

oModule.ChangeProperty Array("NAME:AllTabs", Array("NAME:Header", _
    Array("NAME:PropServers", "XY Plot1:Header"), _
    Array("NAME:ChangedProps", Array("NAME:Design Name", _
        "Value:=", "WG Combiner"))))

oModule.ChangeProperty Array("NAME:AllTabs", Array("NAME:General", _
    Array("NAME:PropServers", "XY Plot1:General"), _
    Array("NAME:ChangedProps", Array("NAME:Back Color", _
        "R:=", 128, "G:=", 255, "B:=", 255)))))

oModule.ChangeProperty Array("NAME:AllTabs", Array("NAME:Axis", _
    Array("NAME:PropServers", "XY Plot1:AxisX"), _
    Array("NAME:ChangedProps", Array("NAME:Axis Scaling", _
```

```
"Value:=", "Log"))))
```

## GetArrayVariables

Returns a list of array variables. To get a list of indexed project variables, execute with oProject. To get a list of indexed local variables, use oDesign.

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

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

<b>VB Syntax</b>	GetArrayVariables
<b>VB Example</b>	<pre>oProject.GetArrayVariables oDesign.GetArrayVariables</pre>

## GetProperties

Gets a list of all the properties belonging to a specific <PropServer> and <PropTab>. This can be executed by the oProject, oDesign, or oEditor variables.

<b>UI Access</b>	N/A		
<b>Parameters</b>	Name	Type	Description
	<PropTab>	String	<p>One of the following, where tab titles are shown in parentheses:</p> <ul style="list-style-type: none"> <li>• PassedParameterTab ("Parameter Values")</li> <li>• DefinitionParameterTab (Parameter Defaults")</li> <li>• LocalVariableTab ("Variables" or "Local Variables")</li> <li>• ProjectVariableTab ("Project variables")</li> <li>• ConstantsTab ("Constants")</li> <li>• BaseElementTab ("Symbol" or "Footprint")</li> <li>• ComponentTab ("General")</li> <li>• Component("Component")</li> <li>• CustomTab ("Intrinsic Variables")</li> <li>• Quantities ("Quantities")</li> <li>• Signals ("Signals")</li> </ul>
	<PropServer>	String	An object identifier, generally returned from another script method, such as ComplInst@R;2;3
<b>Return Value</b>	Array of strings containing the names of the appropriate properties.		

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

<b>VB Syntax</b>	GetProperties <PropTab>, <PropServer>
<b>VB Example</b>	<code>oEditor.GetProperties "PassedParameterTab", "k"</code>

## GetProperty Value

Returns the value of a single property belonging to a specific <PropServer> and <PropTab>. This function is available with the Project, Design or Editor objects, including definition editors.

**Tip:**

Use the script recording feature and edit a property, and then view the resulting script to see the format for that property.

<b>UI Access</b>	N/A		
<b>Parameters</b>	Name <PropTab>	Type String	Description One of the following, where tab titles are shown in parentheses: <ul style="list-style-type: none"><li>• PassedParameterTab ("Parameter Values")</li><li>• DefinitionParameterTab (Parameter Defaults")</li><li>• LocalVariableTab ("Variables" or "Local Variables")</li><li>• ProjectVariableTab ("Project variables")</li><li>• ConstantsTab ("Constants")</li><li>• BaseElementTab ("Symbol" or "Footprint")</li></ul>

		<ul style="list-style-type: none"> <li>• ComponentTab ("General")</li> <li>• Component("Component")</li> <li>• CustomTab ("Intrinsic Variables")</li> <li>• Quantities ("Quantities")</li> <li>• Signals ("Signals")</li> </ul>
	<PropServer>	String An object identifier, generally returned from another script method, such as CompInst@R;2;3
	<PropName>	String Name of the property.
<b>Return Value</b>	String value of the property.	

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

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

## GetVariables

Returns a list of all defined variables. To get a list of project variables, execute this command using `oProject`. To get a list of local variables, use `oDesign`.

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

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

<b>VB Syntax</b>	<code>GetVariables</code>
<b>VB Example</b>	<code>oProject.GetVariables</code> <code>oDesign.GetVariables</code>

## GetVariableValue

Gets the value of a single specified variable. To get the value of project variables, execute this command using `oProject`. To get the value of local variables, use `oDesign`.

<b>UI Access</b>	N/A								
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;VarName&gt;</td> <td>String</td> <td>Name of the variable to access.</td> </tr> </tbody> </table>			Name	Type	Description	<VarName>	String	Name of the variable to access.
Name	Type	Description							
<VarName>	String	Name of the variable to access.							
<b>Return Value</b>	String represents the value of the variable.								

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

<b>VB Syntax</b>	GetVariableValue <VarName>
<b>VB Example</b>	<code>oProject.GetVariableValue "var_name"</code>

## SetPropertyValue

Sets the value of a single property belonging to a specific PropServer and PropTab. This function is available with the Project, Design or Editor objects, including definition editors. This is not supported for properties of the following types: ButtonProp, PointProp, V3DPointProp, and VPointProp. Only the ChangeProperty command can be used to modify these properties.

Use the script recording feature and edit a property, and then view the resulting script entry or use GetPropertyValue for the desired property to see the expected format.

<b>UI Access</b>	N/A								
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;propTab&gt;</td> <td>String</td> <td>           One of the following, where tab titles are shown in parentheses:  <ul style="list-style-type: none"> <li>• PassedParameterTab ("Parameter Values")</li> </ul> </td> </tr> </tbody> </table>			Name	Type	Description	<propTab>	String	One of the following, where tab titles are shown in parentheses: <ul style="list-style-type: none"> <li>• PassedParameterTab ("Parameter Values")</li> </ul>
Name	Type	Description							
<propTab>	String	One of the following, where tab titles are shown in parentheses: <ul style="list-style-type: none"> <li>• PassedParameterTab ("Parameter Values")</li> </ul>							

			<ul style="list-style-type: none"> <li>• DefinitionParameterTab ("Parameter Defaults")</li> <li>• LocalVariableTab ("Variables" or "Local Variables")</li> <li>• ProjectVariableTab ("Project variables")</li> <li>• ConstantsTab ("Constants")</li> <li>• BaseElementTab ("Symbol" or "Footprint")</li> <li>• ComponentTab ("General")</li> <li>• Component("Component")</li> <li>• CustomTab ("Intrinsic Variables")</li> <li>• Quantities ("Quantities")</li> <li>• Signals ("Signals")</li> </ul>
	<i>&lt;propServer&gt;</i>	String	An object identifier, generally returned from another script method, such as ComplInst@R;2;3
	<i>&lt;propName&gt;</i>	String	Name of the property.
	<i>&lt;propValue&gt;</i>	String	The value for the property
<b>Return Value</b>	None.		

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

---

<b>VB Syntax</b>	SetPropertyValue <propTab>, <propServer>, <propName>, <propValue>
<b>VB Example</b>	<code>oEditor SetPropertyValue "PassedParameterTab", "k", "R", "2200"</code>

## SetVariableValue

Sets the value of a variable. To set the value of a project variable, execute this command using `oProject`. To set the value of a local variable, use `oDesign`.

<b>UI Access</b>	N/A									
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;VarName&gt;</td> <td>String</td> <td>Variable name.</td> </tr> <tr> <td>&lt;VarValue&gt;</td> <td>Value</td> <td>New value for the variable.</td> </tr> </tbody> </table>	Name	Type	Description	<VarName>	String	Variable name.	<VarValue>	Value	New value for the variable.
Name	Type	Description								
<VarName>	String	Variable name.								
<VarValue>	Value	New value for the variable.								
<b>Return Value</b>	None.									

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

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

## 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("HFSSModell1")  
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.GetProperty("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 SetProperty "Geometry3DCmdTab",_
        "Box1:CreateBox:1", prop, new_val
        val = _
        oEditor SetProperty "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:\testinstall\Designer\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.GetProperty("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  
'
```

# 8 - Dataset Script Commands

Dataset commands should be executed by the oProject object:

```
Set oProject = oDesktop.SetActiveProject("Project1")
oProject.CommandName <args>
```

[AddDataSet](#)

[DeleteDataSet](#)

[EditDataSet](#)

[ExportDataSet](#)

[HasDataSet](#)

[ImportDataSet](#)

## AddDataset

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

<b>UI Access</b>	<b>Project &gt; Datasets &gt; Add.</b>		
<b>Parameters</b>	Name	Type	Description
	< DatasetdataArray>	Array	Array("NAME:<DatasetName>", Array("NAME:Coordinates", <CoordinateArray>, <CoordinateArray>, ...))
	<DatasetName>	String	Name of the dataset.
<b>Return Value</b>	None.		

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

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

**VB Syntax****AddDataset <DatasetdataArray>**

**VB Example**

```
oProject.AddDatasetArray("NAME:ds1",
    Array("NAME:Coordinates",
        Array("NAME:Coordinate", "X:=", 1, "Y:=", 2,
            Array("NAME:Coordinate", "X:=", 3, "Y:=", 4),
            Array("NAME:Coordinate", "X:=", 5, "Y:=", 7),
            Array("NAME:Coordinate", "X:=", 6, "Y:=", 20)))
oDesign.AddDatasetArray("NAME:ds1",
    Array("NAME:Coordinates",
        Array("NAME:Coordinate", "X:=", 1, "Y:=", 2,
            Array("NAME:Coordinate", "X:=", 3, "Y:=", 4),
            Array("NAME:Coordinate", "X:=", 5, "Y:=", 7),
            Array("NAME:Coordinate", "X:=", 6, "Y:=", 20))))
```

## DeleteDataset

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

<b>UI Access</b>	<b>Project &gt; Datasets &gt; Remove.</b>		
<b>Parameters</b>	Name <i>&lt;DatasetName&gt;</i>	Type String	Description Name of the dataset found in the project.
<b>Return Value</b>	None.		

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

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

## EditDataset

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

<b>UI Access</b>	Project > Datasets > Edit.									
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;OriginalName&gt;</td> <td>String</td> <td>Name of the original dataset.</td> </tr> <tr> <td>&lt;DatasetdataArray&gt;</td> <td>Array</td> <td>Data for the modified dataset.</td> </tr> </tbody> </table>	Name	Type	Description	<OriginalName>	String	Name of the original dataset.	<DatasetdataArray>	Array	Data for the modified dataset.
Name	Type	Description								
<OriginalName>	String	Name of the original dataset.								
<DatasetdataArray>	Array	Data for the modified dataset.								
<b>Return Value</b>	None.									

<b>Python Syntax</b>	EditDataset (<OriginalName> <DatasetdataArray>)
<b>Python Example</b>	<pre>oProject.EditDataset ("ds1" ["NAME:ds2",  ["NAME:Coordinates",</pre>

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

```

        "X:=", 3, "Y:=", 4
    ]
]
)

```

<b>VB Syntax</b>	EditDataset < <i>OriginalName</i> > < <i>DatasetdataArray</i> >
<b>VB Example</b>	<pre> oProject.EditDataset "ds1" Array("NAME:ds2",     Array("NAME:Coordinates",Array("NAME:Coordinate",         "X:=", 1, "Y:=", 2), Array("NAME:Coordinate",         "X:=", 3, "Y:=", 4))) oDesign.EditDataset "ds1" Array("NAME:ds2",     Array("NAME:Coordinates",Array("NAME:Coordinate",         "X:=", 1, "Y:=", 2), Array("NAME:Coordinate",         "X:=", 3, "Y:=", 4))) </pre>

## ExportDataset

Exports a dataset to a named file. This can be executed by the oProject, or oDesign variables.

<b>UI Access</b>	<b>Project &gt; Datasets &gt; Export.</b>								
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;<i>datasetFilePath</i>&gt;</td> <td>String</td> <td>The full path to the file.</td> </tr> </tbody> </table>			Name	Type	Description	< <i>datasetFilePath</i> >	String	The full path to the file.
Name	Type	Description							
< <i>datasetFilePath</i> >	String	The full path to the file.							

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

<b>Python Syntax</b>	ExportDataset (<datasetFileFullPath>)
<b>Python Example</b>	<pre>oProject.ExportDataset ('e:/tmp/dsdata.txt') oDesign.ExportDataset ('e:/tmp/dsdata.txt')</pre>

<b>VB Syntax</b>	ExportDataset <datasetFileFullPath>
<b>VB Example</b>	<pre>oProject.ExportDataset "e:/tmp/dsdata.txt" oDesign.ExportDataset "e:/tmp/dsdata.txt"</pre>

## HasDataset

Determines whether a specified dataset exists.

<b>UI Access</b>	N/A						
<b>Parameters</b>	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>&lt;datasetName&gt;</td><td>String</td><td>Name of the specified dataset.</td></tr></tbody></table>	Name	Type	Description	<datasetName>	String	Name of the specified dataset.
Name	Type	Description					
<datasetName>	String	Name of the specified dataset.					
<b>Return Value</b>	<p>Integer:</p> <ul style="list-style-type: none"><li>• <b>1</b> - dataset exists.</li><li>• <b>0</b> - dataset does not exist.</li></ul>						

<b>Python Syntax</b>	HasDataset (<datasetName>)
<b>Python Example</b>	<code>oDesign.HasDataset ('\$ds1')</code>

<b>VB Syntax</b>	HasDataset <datasetName>
<b>VB Example</b>	<code>oDesign.HasDataset "\$ds1"</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.

<b>UI Access</b>	<b>Project &gt; Datasets &gt; Import.</b>		
<b>Parameters</b>	Name	Type	Description
	<datasetFilePath>	String	The full path to the file containing the dataset values. *.tab files recommended (see <a href="#">note</a> below).
<b>Return Value</b>	None.		

<b>Python Syntax</b>	ImportDataset (<datasetFilePath>,<optionalDatasetName>)
<b>Python Example</b>	<pre>oProject.ImportDataset ('e:\tmp\dsdata.tab') oDesign.ImportDataset ('e:\tmp\dsdata.tab') oProject.ImportDataset ('e:\tmp\dsdata.tab', 'MyDatasetName')</pre>

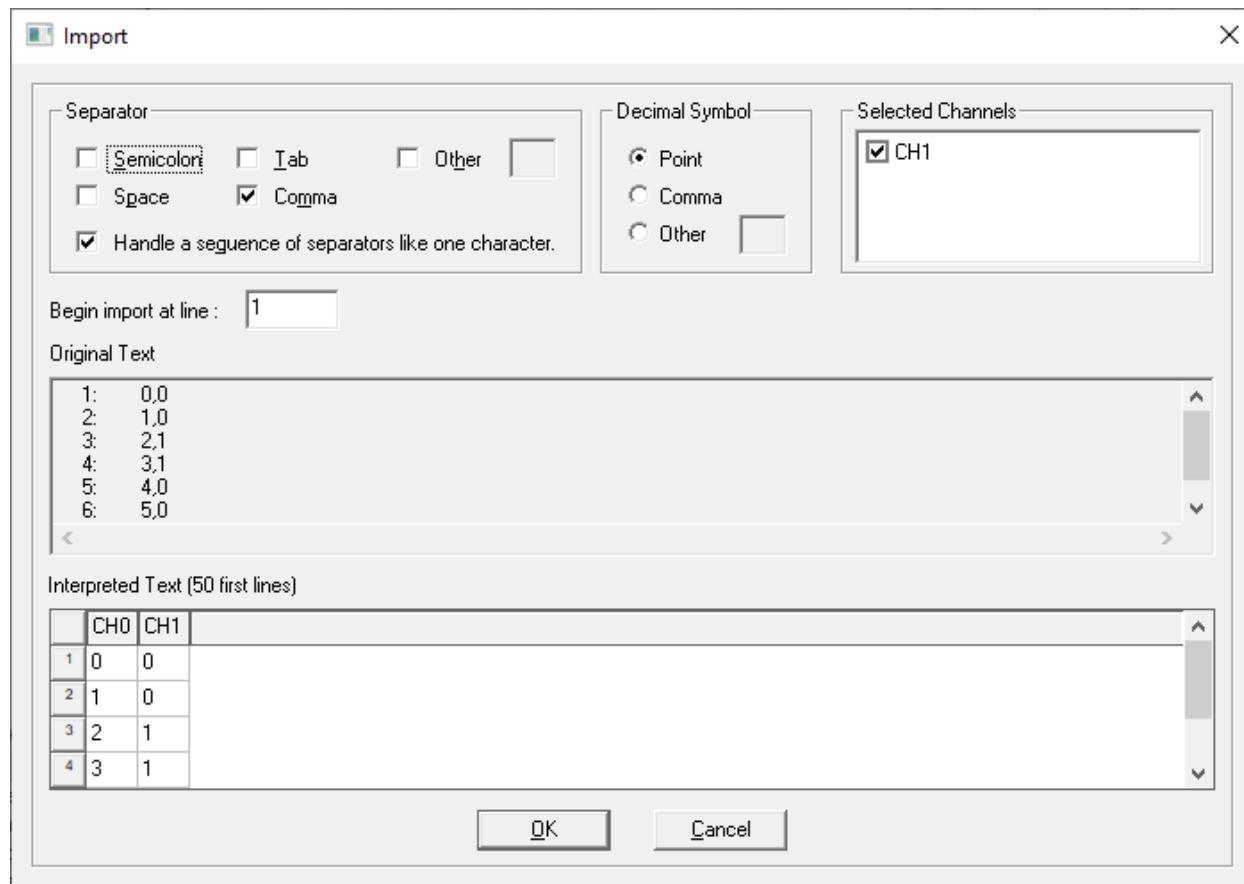
```
oDesign.ImportDataset('e:\tmp\dsdata.tab', 'MyDatasetName')
```

<b>VB Syntax</b>	ImportDataset <datasetFilePath>, <optionalDatasetName>
<b>VB Example</b>	<pre>oProject.ImportDataset "e:\tmp\dsdata.tab" oDesign.ImportDataset "e:\tmp\dsdata.tab" oProject.ImportDataset "e:\tmp\dsdata.tab", "MyDatasetName" oDesign.ImportDataset "e:\tmp\dsdata.tab", "MyDatasetName"</pre>

## Note About File Types:

Tab-delimited or space-delimited files with the extension \*.tab are the recommended file type. When using ImportDataset at the Design level, \*.tab is the only file type supported.

At the Project level, other file types are supported (for example, \*.csv). However, after calling the command, you must configure the file import format manually through the Electronics Desktop GUI by selecting **Project > Datasets** and clicking **Import**.



This page intentionally  
left blank.

# 9 - 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:

- [Analysis Module](#) – "AnalysisSetup"
- [Boundary Module](#) – "BoundarySetup"
- [Field Overlays Module](#) – "FieldsReporter"
- Mesh Module – "MeshSetup"
- [Optimetrics Module](#) – "Optimetrics"
- [Radiation Module](#) – "RadField"
- [Radiation Setup Manager Module](#) – "RadiationSetupMgr"
- Reduce Matrix Module – "ReduceMatrix"
- [Reporter Module](#) – "ReportSetup"
- [Simulation Setup Module](#) – "SimSetup"
- [Solutions Module](#) – "Solutions"

[AddDataset](#)

[AddDesignVariablesForDynamicLink](#)

[AddDynamicLink](#)

[AddModelingProperties](#)

[Analyze](#)

[AnalyzeAll](#)

[AnalyzeAllNominal](#)

[AnalyzeDistributed](#)

[ApplyMeshOps](#)

[AssignDCThickness](#)

ChangePortProperty [Nexxim]

[ChangeProperty](#)

ClearLinkedData

[ConstructVariationString](#)

[CopyItemCommand](#)

[CreateReport](#)

[DeleteDataset](#)

[DeleteDesignInstance](#)

[DeleteFieldVariation](#)

[DeleteFullVariation](#)

[DeleteLinkedDataVariation](#)

[DeleteOutputVariable](#)

DeletePort [Nexxim]

[DeleteSource \[Nexxim\]](#)

[EditCoSimulationOptions](#)

[EditDataset](#)

**EditDesignSettings**

[EditImportData](#)

[EditInfiniteArray](#)

[EditNotes](#)

[EditOptions](#)

[ExportConvergence](#)

[ExportDataset](#)

[ExportForHSpice](#)

[ExportNMFDATA](#)

[ExportMatrixData](#)

[ExportMeshStats](#)

[ExportNetworkData](#)

[ExportProfile](#)

[ExportReport](#)

[GenerateMesh](#)

[GetActiveEditor](#)

[GetAllPorts](#)

[GetChildNames \[Design\]](#)

[GetChildObject \[Design\]](#)  
[GetChildTypes \[Design\]](#)  
[GetConfigurableData](#)  
[GetData](#)  
[GetDesignID](#)  
[GetDesignName](#)  
[GetDesignType](#)  
[GetDesignValidationInfo](#)  
[GetEditor](#)  
[GetGeometryIdsForAllNetLayerCombinations](#)  
[GetGeometryIdsForNetLayerCombinations](#)  
[GetManagedFilesPath](#)  
[GetModule](#)  
[GetName](#)  
[GetNominalVariation](#)  
[GetNoteText](#)  
[GetObjPath](#)  
[GetOutputVariableValue](#)  
[GetOutputVariables](#)  
[GetPostProcessingVariables](#)

[GetPropHost](#)  
[GetPropNames \[Design\]](#)  
[GetPropValue \[Design\]](#)  
[GetProperties](#)  
[GetPropertyValue](#)  
[GetSelections](#)  
[GetSolutionType](#)  
[GetSolveInsideThreshold](#)  
[GetVariableValue](#)  
[GetVariables](#)  
ImportDataFilePath [Nexxim]  
[ImportDataset](#)  
[InsertDesign](#)  
[OverlayCurrents](#)  
[OverlayFarField](#)  
[OverlayMesh](#)  
[OverlayNearField](#)  
[PasteDesign](#)  
PasteItemCommand  
[PushExcitations](#)  
[Redo](#)

---

[RemoveImportData \(Layout Editor\)](#)

[RemoveModelingProperties](#)

[RenameDesignInstance](#)

[RenameImportData \[Nexxim\]](#)

[RenameSource](#)

[ReportTemplates](#)

[RunToolkit](#)

[SetActiveEditor](#)

[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
	< 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",             [                 "NAME:Coordinate", </pre>

```
        "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",         Array("NAME:Coordinate", "X:=", 1, "Y:=", 2,         Array("NAME:Coordinate", "X:=", 3, "Y:=", 4),         </pre>

	Array ("NAME:Coordinate", "X:=", 5, "Y:=", 7), Array ("NAME:Coordinate", "X:=", 6, "Y:=", 20)))
--	----------------------------------------------------------------------------------------------------

## 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	<ul style="list-style-type: none"><li>In the <b>Schematic Editor</b> or <b>Layout Editor</b>, right click the new instance of the linked design and click <b>Add Design Variables for Dynamic Link</b>.</li><li>In the <b>Project Manager</b>, right click on <b>[Project Name] &gt; [Design Name] &gt; [Dynamic Link Design]</b> and click <b>Add Design Variables for Dynamic Link</b>.</li></ul>			
Parameters	<table border="1"><tr><td>Name <i>&lt;InstanceID&gt;</i></td><td>Type String</td><td>Description Instance ID for the dynamic link component</td></tr></table>	Name <i>&lt;InstanceID&gt;</i>	Type String	Description Instance ID for the dynamic link component
Name <i>&lt;InstanceID&gt;</i>	Type String	Description 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")</pre>

```
oDesign.AddDesignVariablesForDynamicLink "6"
```

## 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 Slwave.

**Note:**

HFSS 3D Layout designs can only accept dynamic links from HFSS.

UI Access	<ul style="list-style-type: none"> <li>Open the Circuit or HFSS 3D Layout design to which you want to add the link. On the <b>Component Libraries</b> menu, open the <b>Symbols</b> tab, and then click <b>Import Models</b>. Select the appropriate icon from the panel of icons. The <b>Import Components</b> dialog opens.</li> <li>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.</li> </ul>																					
Parameters	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><code>&lt;designName&gt;</code></td> <td>String</td> <td>Design source for the dynamic link. Optional, defaults to the first design in the project pointed to by <code>projectPath</code>.</td> </tr> <tr> <td><code>&lt;projectPath&gt;</code></td> <td>String</td> <td>Project source for the dynamic link. Optional, defaults to the current active project.</td> </tr> <tr> <td><code>&lt;linkName&gt;</code></td> <td>String</td> <td>Name applied to the dynamic link. Optional, defaults to the <code>designName</code>.</td> </tr> <tr> <td><code>&lt;solutionName&gt;</code></td> <td>String</td> <td>Solution in source design which will be referenced by the dynamic link. Optional, defaults to the first <code>setup:sweep</code>.</td> </tr> <tr> <td><code>&lt;transLinePort&gt;</code></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 <code>""</code>. This value is ignored for non-HFSS design types.</td> </tr> <tr> <td><code>&lt;ReduceMatrix&gt;</code></td> <td>String</td> <td>For Q3D and 2D extractor designs, reduce matrix which will be</td> </tr> </tbody> </table>	Name	Type	Description	<code>&lt;designName&gt;</code>	String	Design source for the dynamic link. Optional, defaults to the first design in the project pointed to by <code>projectPath</code> .	<code>&lt;projectPath&gt;</code>	String	Project source for the dynamic link. Optional, defaults to the current active project.	<code>&lt;linkName&gt;</code>	String	Name applied to the dynamic link. Optional, defaults to the <code>designName</code> .	<code>&lt;solutionName&gt;</code>	String	Solution in source design which will be referenced by the dynamic link. Optional, defaults to the first <code>setup:sweep</code> .	<code>&lt;transLinePort&gt;</code>	String	For HFSS designs only, port name for transmission lines. It is created only if it is <i>not</i> empty. Optional, defaults to <code>""</code> . This value is ignored for non-HFSS design types.	<code>&lt;ReduceMatrix&gt;</code>	String	For Q3D and 2D extractor designs, reduce matrix which will be
Name	Type	Description																				
<code>&lt;designName&gt;</code>	String	Design source for the dynamic link. Optional, defaults to the first design in the project pointed to by <code>projectPath</code> .																				
<code>&lt;projectPath&gt;</code>	String	Project source for the dynamic link. Optional, defaults to the current active project.																				
<code>&lt;linkName&gt;</code>	String	Name applied to the dynamic link. Optional, defaults to the <code>designName</code> .																				
<code>&lt;solutionName&gt;</code>	String	Solution in source design which will be referenced by the dynamic link. Optional, defaults to the first <code>setup:sweep</code> .																				
<code>&lt;transLinePort&gt;</code>	String	For HFSS designs only, port name for transmission lines. It is created only if it is <i>not</i> empty. Optional, defaults to <code>""</code> . This value is ignored for non-HFSS design types.																				
<code>&lt;ReduceMatrix&gt;</code>	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.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	AddDynamicLink (<designName>, <projectPath>, <linkName>, <solutionName>, <transLinePort>, <reducedMatrix>, <enableCableModeling>, <description>)
<b>Python Example</b>	<pre> oDesign = oProject.SetActiveDesign("Circuit1") oDesign.AddDynamicLink("TeeModel", "\$PersonalLib\\TeeWrongFile.aedt", "TeeLink", "Setup1 : LastAdaptive", "Port1", "", False, "desc") </pre>

<b>VB Syntax</b>	AddDynamicLink <designName>, <projectPath>, <linkName>, <solutionName>, <transLinePort>, <reducedMatrix>, <enableCableModeling>, <description>
<b>VB Example</b>	<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.

<b>UI Access</b>	Right-click a solution setup in the project tree, and select <b>Analyze</b> .						
<b>Parameters</b>	<table border="1"> <tr> <td>Name</td> <td>Type</td> <td>Description</td> </tr> <tr> <td>&lt;setup&gt;</td> <td>String</td> <td>Setup name.</td> </tr> </table>	Name	Type	Description	<setup>	String	Setup name.
Name	Type	Description					
<setup>	String	Setup name.					
<b>Return Value</b>	<p>Integer value:</p> <ul style="list-style-type: none"> <li>• <b>0</b> – Success</li> <li>• <b>Else</b> – Error</li> </ul>						

<b>Python Syntax</b>	Analyze (<setup>)
<b>Python Example</b>	oDesign.Analyze("Setup1")

<b>VB Syntax</b>	Analyze <setup>
------------------	-----------------

**VB Example**

```
oDesign.Analyze "Setup1"
```

## AnalyzeAll [design]

Runs all solution setups and Optimetrics setups for the current design instance.

<b>UI Access</b>	N/A
<b>Parameters</b>	None.
<b>Return Value</b>	None.

**Python Syntax**

```
AnalyzeAll()
```

**Python Example**

```
oDesign.AnalyzeAll()
```

**VB Syntax**

```
AnalyzeAll
```

**VB Example**

```
oDesign.AnalyzeAll
```

## AnalyzeAllNominal

Runs all defined setups.

<b>UI Access</b>	Right-click <b>Analysis</b> in the project tree, and select <b>Analyze All</b> .
<b>Parameters</b>	None.

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

<b>Python Syntax</b>	AnalyzeAllNominal()
<b>Python Example</b>	<code>oDesign.AnalyzeAllNominal()</code>

<b>VB Syntax</b>	AnalyzeAllNominal
<b>VB Example</b>	<code>oDesign.AnalyzeAllNominal</code>

## AnalyzeDistributed

Performs a distributed analysis.

<b>UI Access</b>	N/A						
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><code>&lt;SetupName&gt;</code></td> <td>String</td> <td>String name of the setup to be analyzed.</td> </tr> </tbody> </table>	Name	Type	Description	<code>&lt;SetupName&gt;</code>	String	String name of the setup to be analyzed.
Name	Type	Description					
<code>&lt;SetupName&gt;</code>	String	String name of the setup to be analyzed.					
<b>Return Value</b>	<p>Integer value:</p> <ul style="list-style-type: none"> <li>• <b>0</b> – Success</li> <li>• <b>Else</b> – Error</li> </ul>						

<b>Python Syntax</b>	AnalyzeDistributed( <code>&lt;SetupName&gt;</code> )
<b>Python Example</b>	<code>oDesign.AnalyzeDistributed('Setup1')</code>

<b>VB Syntax</b>	AnalyzeDistributed <SetupName>
<b>VB Example</b>	<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.

<b>UI Access</b>	HFSS 3D Layout > Analysis Setup > Generate Mesh.						
<b>Parameters</b>	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>&lt;SetupNameArray&gt;</td><td>Array</td><td>Array containing the string names of setups.</td></tr></tbody></table>	Name	Type	Description	<SetupNameArray>	Array	Array containing the string names of setups.
Name	Type	Description					
<SetupNameArray>	Array	Array containing the string names of setups.					
<b>Return Value</b>	Integer value: <ul style="list-style-type: none"><li>• 0 – Success</li><li>• Else – Error</li></ul>						

<b>Python Syntax</b>	<code>ApplyMeshOps(&lt;SetupNameArray&gt;)</code>
<b>Python Example</b>	<code>oDesign.ApplyMeshOps(['Setup1', 'Setup2'])</code>

<b>VB Syntax</b>	ApplyMeshOps(<SetupNameArray>)
<b>VB Example</b>	status = oDesign.ApplyMeshOps Array("Setup1", "Setup2")

## AssignDCThickness

Assigns DC thickness to more accurately compute DC resistance of a thin conducting object for which **Solve Inside** is *not* selected.

UI Access	HFSS > Boundaries > Assign DC Thickness.		
<b>Parameters</b>	Name	Type	Description
	<objectNameArray>	Array	Array containing string object name(s). These objects will have their thickness changed.
	<thicknessValueArray>	Array	Array containing string thickness values, including units. The values are applied to the objects in the objectNameArray, in order (the first object receives the first value, the second object the second value, and so on). An empty string leaves the value as-is.  Alternately, can be string "Infinite" for infinite thickness or "Effective" to have it calculated automatically.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	AssignDCThickness (<objectNameArray>, <thicknessValueArray>, <enableAuto>)
<b>Python Example</b>	<pre> oModule.AssignDCThickness (     [         "pad1",         "pin1",     ] ) </pre>

```
    "pad2",
    "pad3",
    "pad4",
    "pin2",
    "pin3",
    "pin4",
    "gnd"
],
[
    "1mm",
    "\"><Infinite>\",
    "\"><Effective>\",
    "2mm",
    "\"><Infinite>\",
    "\"><Effective>\",
    "3mm",
    "\"><Infinite>\",
    "\"><Effective>\"
],
"EnableAuto"
```

	)
--	---

<b>VB Syntax</b>	AssignDCThickness <objectNameArray>,<thicknessValueArray>, <enableAuto>
<b>VB Example</b>	<pre> oModule.AssignDCThickness Array("pad1", "pin1", "pad2", "pad3", "pad4", "pin2", "pin3", "pin4", "gnd"), Array("2mm",       "" &amp; Chr(34) &amp; "&lt;Effective&gt;" &amp; Chr(34) &amp; """",       "" &amp; Chr(34) &amp; "&lt;Infinite&gt;" &amp; Chr(34) &amp; """",       "1mm", "" &amp; Chr(34) &amp; "&lt;Infinite&gt;" &amp; Chr(34) &amp; """",       "" &amp; Chr(34) &amp; "&lt;Effective&gt;" &amp; Chr(34) &amp; """",       "3mm", "" &amp; Chr(34) &amp; "&lt;Infinite&gt;" &amp; Chr(34) &amp; """",       "" &amp; Chr(34) &amp; "&lt;Effective&gt;" &amp; Chr(34) &amp; ""), "EnableAuto" </pre>

## ChangeProperty[ReportSetup]

Change the properties for a Report.

<b>UI Access</b>	Double-click on a report to change its properties.		
<b>Parameters</b>	Name <PropertyArray>	Type Array	Description Varies, depending on the properties associated with the select object command.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	ChangeProperty(<PropertyArray>)
<b>Python Example</b>	<pre>oModule = oDesign.GetModule("ReportSetup") oModule.ChangeProperty(     [         "NAME:AllTabs",         [             "NAME:General",             [                 "NAME:PropServers",                 "XY Plot 1:General"             ],             [                 "NAME:ChangedProps",                 [                     "NAME:Use Scientific Notation",                     "Value:=", True                 ]             ]         ]     ] )</pre>

```
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  
            ]  
        ]  
    ]  
)
```

**VB  
Syntax**

ChangeProperty <PropertyArray>

<b>VB Example</b>	<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:PropServers", _ "XY Plot 1:General"), Array("NAME:ChangedProps", Array("NAME:Use Scientific Notation", "Value:=", _ true))) </pre>
-------------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

## ConstructVariationString

Lists and orders the variables and values associated with a design variation.

<b>UI Access</b>	N/A		
<b>Parameters</b>	Name <i>&lt;ArrayOfVariableNames&gt;</i>	Type Array of Strings	Description List of variable names.
	<ArrayOfVariableValuesIncludingUnits>	Array of Strings	List of variable values, including units, in the same order as the list of names.
<b>Return Value</b>	Returns variation string with the variables ordered to correspond to the order of variables in design variations. The values for the variables are inserted into the variation string. For an example of how ConstructionVariationString can be used, see the Python example.		

<b>Python Syntax</b>	ConstructVariationString( <i>&lt;ArrayOfVariableNames&gt;</i> , <i>&lt;ArrayOfVariableValuesIncludingUnits&gt;</i> )
<b>Python Example</b>	varStr = oDesign.ConstructVariationString(["xx", "yy"], ["2mm", "1mm"])

	<code>oDesign.ExportProfile("Setup1", varStr, "C:\profile.prof")</code>
--	-------------------------------------------------------------------------

<b>VB Syntax</b>	ConstructVariationString <ArrayOfVariableNames>, <ArrayOfVariableValuesIncludingUnits>
<b>VB Example</b>	<code>oDesign.ConstructVariationString Array("x_size", "y_size"), Array("2mm", "1mm")</code>

## 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")`  
y, l

## CreateReport

Creates a new report with a single trace and adds it to the **Results** branch in the project tree.

<b>UI Access</b>	Right-click on <b>Results</b> > <b>Create [Type] Report</b>		
<b>Parameters</b>	Name	Type	Description
	<code>&lt;ReportName&gt;</code>	String	Name of report.
	<code>&lt;ReportType&gt;</code>	String	Type of report.

		<p>Possible values are:</p> <p>"Modal S Parameters" - Only for Driven Modal solution-type problems with ports.</p> <p>"Terminal S Parameters" - Only for Driven Terminal solution-type problems with ports.</p> <p>"Eigenmode Parameters" - Only for Eigenmode solution-type problems.</p> <p>"Fields"</p> <p>"Far Fields" - Only for problems with radiation or PML boundaries.</p> <p>"Near Fields" - Only for problems with radiation or PML boundaries.</p> <p>"Emission Test"</p>
<DisplayType>	String	<p>Type of display.</p> <p>If ReportType is "Modal S Parameters", "Terminal S Parameters", or "Eigenmode Parameters", then set to one of the following:</p> <p>"Rectangular Plot", "Polar Plot", "Radiation Pattern", "Smith Chart", "Data Table", "3D Rectangular Plot", "3D Rectangular Bar Plot" or "3D Polar Plot".</p> <p>If &lt;ReportType&gt; is "Fields", then set to one of the following:</p> <p>"Rectangular Plot", "Polar Plot", "Radiation Pattern", "Rectangular Contour Plot", "Data Table", "3D Rectangular Plot", or "3D Rectangular Bar Plot".</p> <p>If &lt;ReportType&gt; is "Far Fields" or "Near Fields", then set to one of the following:</p> <p>"Rectangular Plot", "Radiation Pattern", "Rectangular Contour Plot", "Data Table",</p>

		<p>"3D Rectangular Plot", "3D Rectangular Bar Plot" or "3D Polar Plot"</p> <p>If &lt;ReportType&gt; is "Emission Test", then set to one of the following:</p> <p>"Rectangular Plot" or "Data Table"</p>
<SolutionName>	String	Name of the solution as listed in the <b>Modify Report</b> dialog box.
<ContextArray>	Array	<p>Context for which the expression is being evaluated. This can be an empty string if there is no context.</p> <p>Array("Domain:=", &lt;DomainType&gt;)  &lt;DomainType&gt; ex. "Sweep" or "Time"</p> <p>Array("Context:=", &lt;GeometryType&gt;)  &lt;GeometryType&gt;  ex. "Infinite Spheren", "Spheren", "Polylinen"</p>
<FamiliesArray>	Array	<p>Contains sweep definitions for the report.</p> <p>Array("&lt;VariableName&gt;:= ", &lt;ValueArray&gt;)  &lt;ValueArray&gt;  Array("All") or Array("Value1", "Value2", ... "ValueN")  examples of &lt;VariableName&gt; "Freq", "Theta", "Distance"</p>
<ReportdataArray>	Array	<p>This array contains the report quantity and X, Y, and (Z) axis definitions.</p> <p>Array("X Component:=", &lt;VariableName&gt;, "Y Component:=", &lt;VariableName&gt;   &lt;ReportQuantityArray&gt;)  &lt;ReportQuantityArray&gt;  ex. Array("dB(S(Port1, Port1))")</p>

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

<b>Python Syntax</b>	CreateReport(<ReportName>, <ReportType>, <DisplayType>, <SolutionName>, <ContextArray>, <FamiliesArray>, <ReportdataArray>, <ExtendedInfo>)
<b>Python Example</b>	<p>Three examples are given below: (nominal, Optimetrics, spectral)</p> <p><b>Nominal Example:</b></p> <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> <p><b>Optimetrics Example:</b></p> <pre>oModule.CreateReport ( "XY Plot 3", "Standard",_                       "Rectangular Plot", "TR", ["NAME:Context",_                       "OptiSetup:=", "ParametricSetup1", "SimValueCor_                       [1, 0, 2, 0, false, false, 0, 1,_                       0, 1, 1, "", 0, 0]], ["Time:=", ["All"],_</pre>

```
"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", "K  
false, "0", "MH", false, "100", "TE", false, "4  
"TH", false, "40", "TS", false, "0ns", "UF", fa  
"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"  
        ]  
    ]  
]  
)  
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  
    ]  
]  
]  
])  
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>Show Outline",
            "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: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))"
        ],
        [
            "NAME:ChangedProps",
            [
                "NAME:Outline Width",

```

```
        "Value:=" , "2"
    ]
]
])
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))"
        ],
        [
            "NAME:ChangedProps",
            [
                "NAME:Bar Width",
                "Value:=" , "Wide"
            ]
        ]
    ]
])
```

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

<b>VB Syntax</b>	<pre>CreateReport &lt;ReportName&gt;, &lt;ReportType&gt;, &lt;DisplayType&gt;, &lt;SolutionName&gt;, &lt;ContextArray&gt;, &lt;FamiliesArray&gt;, &lt;ReportdataArray&gt;, &lt;ExtendedInfo&gt;</pre>
<b>VB Example</b>	<pre>oModule.CreateReport "3D Cartesian Plot1", "Far Fields", _     "3D Cartesian Plot", "Setup1 : LastAdaptive", _     Array("Context:=", "Infinite Sphere1", "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", _</pre>

```

    "Rectangular Contour Plot", "Setup1 : LastAdaptive", Array("Context:=", _
    "Infinite Sphere1"), Array("_u:=", Array("All"), "_v:=", Array("All"), "Freq:=",
    Array( _
        "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.

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

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

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

## DeleteFieldVariation

Deletes field variations, fields and meshes, or fields and meshes for specified variations.

<b>UI Access</b>	[Solver] > Results > Clean Up Solutions > [Fields Only / Fields and Meshes].												
<b>Parameters</b>	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>&lt;variationKeys&gt;</td><td>Array</td><td>Array containing either "All" string to select all variations, or strings of variations to include.</td></tr><tr><td>&lt;deleteMesh&gt;</td><td>Boolean</td><td>When true, deletes mesh data.</td></tr><tr><td>&lt;deleteLinkedData&gt;</td><td>Boolean</td><td>When true, deletes linked data.</td></tr></tbody></table>	Name	Type	Description	<variationKeys>	Array	Array containing either "All" string to select all variations, or strings of variations to include.	<deleteMesh>	Boolean	When true, deletes mesh data.	<deleteLinkedData>	Boolean	When true, deletes linked data.
Name	Type	Description											
<variationKeys>	Array	Array containing either "All" string to select all variations, or strings of variations to include.											
<deleteMesh>	Boolean	When true, deletes mesh data.											
<deleteLinkedData>	Boolean	When true, deletes linked data.											
<b>Return Value</b>	None.												

<b>Python Syntax</b>	DeleteFieldVariation(<variationKeys>, <deleteMesh>, <deleteLinkedData>)
<b>Python Example</b>	<pre>oProject = oDesktop.GetActiveProject()</pre>

```

oDesign = oProject.GetActiveDesign()
Design.DeleteFieldVariation(['All'], True, False)

```

<b>VB Syntax</b>	DeleteFieldVariation <variationKeys>, <deleteMesh>, <deleteLinkedData>
<b>VB Example</b>	<pre> Set oProject = oDesktop.SetActiveProject "OptimTee" Set oDesign = oProject.SetActiveDesign "TeeModel" oDesign.DeleteFieldVariation Array("offset=" &amp; Chr(39) &amp; "0.0947046688081817in" &amp; Chr(39) &amp; ""), true, false </pre>

## DeleteFullVariation

Deletes either all solution data, or selected variation data.

<b>UI Access</b>	<b>[HFSS 3D Layout] &gt; Results &gt; Clean Up Solutions.</b>		
<b>Parameters</b>	Name	Type	Description
	<variationKeys>	Array	Array containing either "All" string to select all variations, or strings of variations to include.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	DeleteFullVariation(<variationKeys>, <deleteLinkedData>)
<b>Python Example</b>	<code>oDesign.DeleteFullVariation(['All']), false)</code>

<b>VB Syntax</b>	DeleteFullVariation <variationKeys>, <deleteLinkedData>
------------------	---------------------------------------------------------

**VB Example**

```
oDesign.DeleteFullVariation Array('All'), false
```

## DeleteLinkedDataVariation

Deletes the linked data of specified variations.

<b>UI Access</b>	<b>[HFSS 3D Layout] &gt; Results &gt; Clean Up Solutions.</b>		
<b>Parameters</b>	Name <i>&lt;DesignVariationKeys&gt;</i>	Type Array	Description Array containing strings of the variations keys whose linked data are going to be deleted.
<b>Return Value</b>	None.		

**Python Syntax**

```
DeleteLinkedDataVariation(<DesignVariationKeys>)
```

**Python Example**

```
oDesign.DeleteLinkedDataVariation(["current='0.9mA'", "current='1.0mA'"])
```

**VB Syntax**

```
DeleteLinkedDataVariation <DesignVariationKeys>
```

**VB Example**

```
oDesign.DeleteLinkedDataVariation Array("current='0.9mA'", "current='1.0mA'")
```

## 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**.

<b>Parameters</b>	<table border="1"> <tr> <th>Name</th><th>Type</th><th>Description</th></tr> <tr> <td>&lt;OutputVarName&gt;</td><td>String</td><td>Name of the output variable.</td></tr> </table>	Name	Type	Description	<OutputVarName>	String	Name of the output variable.
Name	Type	Description					
<OutputVarName>	String	Name of the output variable.					
<b>Return Value</b>	None.						

<b>Python Syntax</b>	DeleteOutputVariable (<OutputVarName>)
<b>Python Example</b>	<pre>oModule = oDesign.GetModule("OutputVariable") oModule.DeleteOutputVariable ("testNew")</pre>

<b>VB Syntax</b>	DeleteOutputVariable <OutputVarName>
<b>VB Example</b>	<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.

<b>UI Access</b>	Right-click on <b>Results</b> , select <b>Browse Solutions...</b> , click <b>Delete</b> button in the dialog.									
<b>Parameters</b>	<table border="1"> <tr> <td>Name</td> <td>Type</td> <td>Description</td> </tr> <tr> <td>&lt;SoluParams&gt;</td> <td>Array</td> <td>Structured array. Array (&lt;DataSpecifierArray&gt;, ...)</td> </tr> <tr> <td>&lt;DataSpecifierArray&gt;</td> <td>Array</td> <td>Structured array. Array (&lt;DesignVariationKey&gt;, &lt;SetupName&gt;, &lt;Sol-</td> </tr> </table>	Name	Type	Description	<SoluParams>	Array	Structured array. Array (<DataSpecifierArray>, ...)	<DataSpecifierArray>	Array	Structured array. Array (<DesignVariationKey>, <SetupName>, <Sol-
Name	Type	Description								
<SoluParams>	Array	Structured array. Array (<DataSpecifierArray>, ...)								
<DataSpecifierArray>	Array	Structured array. Array (<DesignVariationKey>, <SetupName>, <Sol-								

			nName>)
<b>Return Value</b>	None.		

<b>Python Syntax</b>	DeleteSolutionVariation(<SoluParams>)
<b>Python Example</b>	<pre>oModule.DeleteSolutionVariation([     ["width='2in'", "Setup1", "Adaptive_1"],     ["width='2in'", "Setup1", "Sweep1"] ])</pre>

<b>VB Syntax</b>	DeleteSolutionVariation <SoluParams>
<b>VB Example</b>	<pre>oModule.DeleteSolutionVariation Array( _     Array("width='2in'", "Setup1", "Adaptive_1"), _     Array("width='2in'", "Setup1", "Sweep1"))</pre>

## DeleteOutputVariable

Deletes an existing output variable. The variable can only be deleted if it is not in use by any traces.

<b>UI Access</b>	<b>HFSS 3D Layout &gt; Results &gt; Output Variables.</b> In the <b>Output Variables</b> window, click <b>Delete</b> .								
<b>Parameters</b>	<table border="1"> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> <tr> <td>&lt;OutputVarName&gt;</td> <td>String</td> <td>Name of the output variable.</td> </tr> </table>			Name	Type	Description	<OutputVarName>	String	Name of the output variable.
Name	Type	Description							
<OutputVarName>	String	Name of the output variable.							

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

<b>Python Syntax</b>	DeleteOutputVariable (<OutputVarName>)
<b>Python Example</b>	<pre>oModule = oDesign.GetModule("OutputVariable") oModule.DeleteOutputVariable ("testNew")</pre>

<b>VB Syntax</b>	DeleteOutputVariable <OutputVarName>
<b>VB Example</b>	<pre>Set oModule = oDesign.GetModule("OutputVariable") oModule.DeleteOutputVariable "testNew"</pre>

## DistributeAnalysis

Performs a distributed analysis.

<b>UI Access</b>	N/A						
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;SetupName&gt;</td> <td>String</td> <td>String name of the setup to be analyzed.</td> </tr> </tbody> </table>	Name	Type	Description	<SetupName>	String	String name of the setup to be analyzed.
Name	Type	Description					
<SetupName>	String	String name of the setup to be analyzed.					
<b>Return Value</b>	<p>Integer value:</p> <ul style="list-style-type: none"> <li>• <b>0</b> – Success</li> <li>• <b>Else</b> – Error</li> </ul>						

<b>Python Syntax</b>	DistributeAnalysis(<SetupName>)
<b>Python Example</b>	oDesign.DistributeAnalysis('Setup1')

<b>VB Syntax</b>	DistributeAnalysis <SetupName>
<b>VB Example</b>	oDesign.DistributeAnalysis "Setup1"

## 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")
```

## 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")))
```

ua

---

## EditLayoutForLayoutComponent

Opens the definition component for editing in HFSS 3D Layout for a Layout Component in an HFSS 3D design.

<b>UI Access</b>	<b>Edit Layout...</b>		
<b>Parameters</b>	Name <i>&lt;ComponentDefinitionName&gt;</i>	Type String	Description Name of the Layout Component Definition.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	EditLayoutForLayoutComponent ([Name:LayoutComponent EDB", Definitions:=", [ <i>&lt;ComponentDefinitionName&gt;</i> ])
<b>Python Example</b>	<pre>oDesign.EditLayoutForLayoutComponent (     [         "NAME:Layout Component EDB",         "Definitions:="           , ["Diff_Via_diffViaNominal"]     ] )</pre>

## EditNotes

Updates the notes for the design.

<b>UI Access</b>	<b>HFSS 3D Layout &gt; Edit Notes.</b>
------------------	----------------------------------------

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	oDesign.EditNotes("My design notes.")

VB Syntax	EditNotes <DesignNotes>
VB Example	oDesign.EditNotes "My design notes."

## 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>

## 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)
```

## ExportConvergence

For a given variation, exports convergence data (max mag delta S, E, freq) to \*.conv file.

UI Access	Results > Solution Data. Select Convergence tab and click Export.		
Parameters	Name	Type	Description
	<Setup>	String	Setup name.
	<VariationKeys>	String	The variation. Pass empty string for the current nominal variation.
	<FilePath>	String	Full path to desired *.conv file location.
	<OverwritelfExists>	Boolean	<i>Optional.</i> If True, overwrites any existing file at the same path.
Return Value	None.		

Python Syntax	ExportConvergence(<Setup>, <VariationKeys>, <FilePath> <OverwritelfExists>)
---------------	-----------------------------------------------------------------------------

<b>Python Example</b>	<code>oDesign.ExportConvergence('Setup1', 'x_size = 2mm', 'c:\convergence.conv')</code>
-----------------------	-----------------------------------------------------------------------------------------

<b>VB Syntax</b>	<code>ExportConvergence &lt;Setup&gt;, &lt;VariationKeys&gt;, &lt;FilePath&gt; &lt;OverwriteIfExists&gt;</code>
<b>VB Example</b>	<code>oDesign.ExportConvergence "Setup1", "x_size = 2mm", "c:\convergence.conv"</code>

## ExportDataset

Exports a dataset to a named file. This can be executed by the oProject, or oDesign variables.

<b>UI Access</b>	<b>Project &gt; Datasets &gt; Export.</b>		
<b>Parameters</b>	Name <code>&lt;datasetFileFullPath&gt;</code>	Type String	Description The full path to the file.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	<code>ExportDataset(&lt;datasetFileFullPath&gt;)</code>
<b>Python Example</b>	<code>oProject.ExportDataset('e:/tmp/dsdata.txt')</code> <code>oDesign.ExportDataset('e:/tmp/dsdata.txt')</code>

<b>VB Syntax</b>	<code>ExportDataset &lt;datasetFileFullPath&gt;</code>
<b>VB Example</b>	<code>oProject.ExportDataset "e:/tmp/dsdata.txt"</code> <code>oDesign.ExportDataset "e:/tmp/dsdata.txt"</code>

## 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.

<b>UI Access</b>	N/A		
<b>Parameters</b>	Name	Type	Description
	<DesignVariation>	String	Desgin 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.
<b>Return Value</b>	None.		
<b>Python Syntax</b>	ExportForHSpice(<DesignVariation>, <SoluSelections>, <SpiceType>, <BandWidth>, <FWSFile>, <LumpedElementFile>, <PoleZeroSpiceFile>, <PoleZeroMatlabFile>, <PartialFractionFile>, <FittingError>, <MinimumOrder>, <MaximumOrder>, [Optional <UseCommonGround>], [Optional <EnforcePassivity>])		
<b>Python Example</b>	oModule.ExportForHSpice("width='2in'",		

	<pre>["Setup1:Sweep1"], 1, 0, "c:\mydir\Sweep1.fws", "", "", "", "", .005, 20, 200)</pre>
<b>VB Syntax</b>	ExportForHSpice <DesignVariation>, <SoluSelections>, <SpiceType>, <BandWidth>, <FWSFile>, <LumpedElementFile>, <PoleZeroSpiceFile>, <PoleZeroMatlabFile>, <PartialFractionFile>, <FittingError>, <MinimumOrder>, <MaximumOrder>, [Optional <UseCommonGround>], [Optional <EnforcePassivity>]
<b>VB Example</b>	<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> <LumpedElementFilenme> <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  
<LumpedElementFilenme>

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> <LumpedElementFilenme> <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
```

## ExportMatrixData

*Use:* Exports matrix in Matlab or spreadsheet format.

*Command:* In the **Matrix** tab of the **Solution** dialog box, click **Export>RLGC**.

*Syntax:* ExportMatrixData <FileName>, <SolnType>, <DesignVariationKey>, <Solution>, <Matrix>, <ResUnit>, <IndUnit>, <CapUnit>, <CondUnit>, <Frequency>, <MatrixType>, <PassNumber>, <ACPlusDCResistance>

*Return Value:* none

*Parameters:* <FileName>

Type: <string>

The path and name of the file where the data will be exported. The file extension determines the file format.

<SolnType>

Type: <string>

One of the following "C", "AC RL" and "DC RL"

<DesignVariationKey>

Type: <string>

Design variation string

Example: "radius = 3mm"

The empty variation string ("") is interpreted to mean the current nominal variation.

<Solution>

<SolveSetup>:<Soln>

**Parameters:**

<SolveSetup>

Type: <string>

Name of the solve setup

<Soln>

Type: <string>

Name of the solution at a certain adaptive pass.

<Matrix>

Type: <string>

Either "Original" or one of the reduce matrix setup names

<ResUnit>

Type: <string>

Unit used for the resistance value

<IndUnit>

Type: <string>

Unit used for the inductance value

<CapUnit>

Type: <string>

Unit used for the capacitance value

<CondUnit>

```
Type: <string>
Unit used for the conductance value

<Frequency>
Type: <double>
Frequency in hertz.

<MatrixType>
Type: <String>
Value: "Maxwell", "Spice", "Couple" , (one or all of these).
Matrix type to export.

<PassNumber>
Type: <integer>
Pass number.

<ACPlusDCResistance>
Type: <bool>
Default Value: false
Add DC and AC resistance and export the matrix.

VB Example:oDesign.ExportMatrixData "C:/temp/3DScripts/rlgc_lvl.lvl", "C, DC RL, AC RL", "", _
"Setup1:LastAdaptive", "Original", "ohm", "nH", "pF", "mSie", 100000000, _ "Max-
well,Spice,Couple", 0
```

**Python Syn-
tax**

```
ExportMatrixData ([FileName, SolnType, DesignVariationKey, Solution, Matrix, ResUnit, IndUnit, CapUnit,
CondUnit, Frequency, MatrixType, PassNumber, ACPlusDCResistance])
```

**Python Example**

```
oDesign.ExportMatrixData
(["C:/temp/3DScripts/rlgc_lvl lvl", "C, DC RL, AC RL", "", _ "Setup1:LastAdaptive",
"Original", "ohm", "nH", "pF", "mSie", 100000000, _ "Maxwell,Spice,Couple",
0])
```

## ExportMeshStats

Exports mesh statistics to a file.

<b>UI Access</b>	N/A		
<b>Parameters</b>	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.
	<overwritelfExists>	Boolean	(Optional). Defaults to True. If True, script overwrites any existing file of the same name. If False, no overwrite.
<b>Return Value</b>	None.		

**Python Syntax**

```
ExportMeshStats(<setupName>, <variationString>, <filePath>, <overwritelfExists>)
```

**Python Example**

```
oDesign.ExportMeshStats ("Setup1", "radius=3mm", "C:\mydir\meshstats.ms",
True)
```

**VB Syntax**

```
ExportMeshStats <setupName>, <variationString>, <filePath>, <overwritelfExists>
```

**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
	<i>&lt;DesignVariationKey&gt;</i>	String	Design variation key. Pass empty string for the current nominal variation.
	<i>&lt;SolnSelectionArray&gt;</i>	Array	Array of selected solutions.  Array (<SolnSelector>, <SolnSelector>, ...)  If more than one array entry, this indicates a combined Interpolating sweep.
	<i>&lt;SolnSelector&gt;</i>	String	Solution setup name and solution name, separated by a colon.
	<i>&lt;FileFormat&gt;</i>	Integer	File format value.  2 : Tab delimited spreadsheet format (.tab)  3 : Touchstone (.sNp)  4 : CitiFile (.cit)  7 : Matlab (.m)  8 : Terminal Z0 spreadsheet
	<i>&lt;OutFile&gt;</i>	String	Full path to the file to write out.
	<i>&lt;FreqsArray&gt;</i>	Array	The frequencies to export. The <i>&lt;FreqsArray&gt;</i> 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.
	<i>&lt;DoRenorm&gt;</i>	Boolean	Specifies whether to renormalize the data before export.
	<i>&lt;RenormImped&gt;</i>	Double	Real impedance value in ohms, for renormalization. Required in syntax, but ignored if DoRenorm is false.

	<b>&lt;DataType&gt;</b>	String	Optional. Type: "S", "Y", or "Z". The matrix to export.
	<b>&lt;pass&gt;</b>	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.
	<b>&lt;ComplexFormat&gt;</b>	Integer	Optional. Type: "0", "1", or "2" The format to use for the exported data. 0 = Magnitude/Phase. 1= Real/Imaginary. 2= db/Phase.
	<b>&lt;Precision&gt;</b>	Integer	Optional. Touchstone number of digits precision. Default if not specified is 15.
	<b>&lt;UseExportFreqs&gt;</b>	Boolean	Specifies whether to use export frequencies.
	<b>&lt;IncludeGammaComments&gt;</b>	Boolean	Specifies whether to include Gamma and Impedance comments.
	<b>&lt;SupportNonStdExport&gt;</b>	Boolean	Specifies whether to support non-standard Touchstone extensions for mixed reference impedance.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	ExportNetworkData(<DesignVariationKey>, <SolnSelectionArray>, <SolnSelector>, <FileFormat>, <OutFile>, <FrequenciesArray>, <DoRenorm>, <RenormImped>, [Optional <DataType>], [Optional <pass>], [Optional <ComplexFormat>], [Optional <Precision>], [Optional <UseExportFreqs>], [Optional <IncludeGammaComments>], [Optional <SupportNonStdExport>])
<b>Python Example</b>	<pre>oModule.ExportNetworkData("", ["Setup1:Sweep1"], 3, "C://Documents/package_HFSSDesign1.s4p", ['all'], True, 50, "S", -1, 0, 15, True, True, True</pre>

<b>VB Syntax</b>	ExportNetworkData < <i>DesignVariationKey</i> >, < <i>SolnSelectionArray</i> >, < <i>SolnSelector</i> >, < <i>FileFormat</i> >, < <i>OutFile</i> >, < <i>FreqsArray</i> >, < <i>DoRenorm</i> >, < <i>RenormImped</i> >, [Optional < <i>DataType</i> >], [Optional < <i>pass</i> >], [Optional < <i>ComplexFormat</i> >], [Optional < <i>Precision</i> >], [Optional < <i>UseExportFreqs</i> >], [Optional < <i>IncludeGammaComments</i> >], [Optional < <i>SupportNonStdExport</i> >]
<b>VB Example</b>	<pre> oModule.ExportNetworkData "width='2in'", _     Array("Setup1:Sweep1"), 2, "c:\mydir\out.tab", _     Array("all"), false, 0  oModule.ExportNetworkData "width='2in'", _     Array("Setup1:Sweep1", "Setup1:Sweep2"), 3, _     "c:\mydir\out.s2p", Array(1.0e9, 1.5e9, 2.0e9), _     true, 50.0 </pre>

## ExportNetworkData

Exports matrix solution data to a file.

<b>UI Access</b>	N/A														
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;<i>DesignVariationKey</i>&gt;</td> <td>String</td> <td>Design variation key. Pass empty string for the current nominal variation.</td> </tr> <tr> <td>&lt;<i>SolnSelectionArray</i>&gt;</td> <td>Array</td> <td>           Array of selected solutions.            Array(&lt;<i>SolnSelector</i>&gt;, &lt;<i>SolnSelector</i>&gt;, ...)            If more than one array entry, this indicates a combined Interpolating sweep.         </td> </tr> <tr> <td>&lt;<i>SolnSelector</i>&gt;</td> <td>String</td> <td>Solution setup name and solution name, separated by a colon.</td> </tr> </tbody> </table>	Name	Type	Description	< <i>DesignVariationKey</i> >	String	Design variation key. Pass empty string for the current nominal variation.	< <i>SolnSelectionArray</i> >	Array	Array of selected solutions. Array(< <i>SolnSelector</i> >, < <i>SolnSelector</i> >, ...) If more than one array entry, this indicates a combined Interpolating sweep.	< <i>SolnSelector</i> >	String	Solution setup name and solution name, separated by a colon.		
Name	Type	Description													
< <i>DesignVariationKey</i> >	String	Design variation key. Pass empty string for the current nominal variation.													
< <i>SolnSelectionArray</i> >	Array	Array of selected solutions. Array(< <i>SolnSelector</i> >, < <i>SolnSelector</i> >, ...) If more than one array entry, this indicates a combined Interpolating sweep.													
< <i>SolnSelector</i> >	String	Solution setup name and solution name, separated by a colon.													

	<code>&lt;FileFormat&gt;</code>	Integer	File format value. 2 : Tab delimited spreadsheet format (.tab) 3 : Touchstone (.sNp) 4 : CitiFile (.cit) 7 : Matlab (.m) 8 : Terminal Z0 spreadsheet
	<code>&lt;OutFile&gt;</code>	String	Full path to the file to write out.
	<code>&lt;FreqsArray&gt;</code>	Array	The frequencies to export. The <code>&lt;FreqsArray&gt;</code> argument contains a vector (e.g. "1GHz", "2GHz", ...) to use, or "all". To export all frequencies, use <code>Array("all")</code> . If no frequencies are specified, all frequencies are used.
	<code>&lt;DoRenorm&gt;</code>	Boolean	Specifies whether to renormalize the data before export.
	<code>&lt;RenormImped&gt;</code>	Double	Real impedance value in ohms, for renormalization. Required in syntax, but ignored if <code>DoRenorm</code> is false.
	<code>&lt;DataType&gt;</code>	String	Optional. Type: "S", "Y", or "Z". The matrix to export.
	<code>&lt;pass&gt;</code>	Integer	Optional. The pass to export. This is ignored if the <code>sourceName</code> is a frequency sweep. Leaving out this value or specifying -1 gets all passes.
	<code>&lt;ComplexFormat&gt;</code>	Integer	Optional. Type: "0", "1", or "2"  The format to use for the exported data.  0 = Magnitude/Phase.  1= Real/Imaginary.  2= db/Phase.
	<code>&lt;Precision&gt;</code>	Integer	Optional. Touchstone number of digits precision. Default if not specified is 15.
	<code>&lt;UseExportFreqs&gt;</code>	Boolean	Specifies whether to use export frequencies.

	<code>&lt;IncludeGammaComments&gt;</code>	Boolean	Specifies whether to include Gamma and Impedance comments.
	<code>&lt;SupportNonStdExport&gt;</code>	Boolean	Specifies whether to support non-standard Touchstone extensions for mixed reference impedance.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	<code>ExportNetworkData(&lt;DesignVariationKey&gt;, &lt;SolnSelectionArray&gt;, &lt;SolnSelector&gt;, &lt;FileFormat&gt;, &lt;OutFile&gt;, &lt;FreqsArray&gt;, &lt;DoRenorm&gt;, &lt;RenormImped&gt;, [Optional &lt;DataType&gt;], [Optional &lt;pass&gt;], [Optional &lt;ComplexFormat&gt;], [Optional &lt;Precision&gt;], [Optional &lt;UseExportFreqs&gt;], [Optional &lt;IncludeGammaComments&gt;], [Optional &lt;SupportNonStdExport&gt;])</code>
<b>Python Example</b>	<pre> oModule.ExportNetworkData("", ["Setup1:Sweep1"], 3, "C://Documents/package_HFSSDesign1.s4p", ['all'], True, 50, "S", -1, 0, 15, True, True, True </pre>

<b>VB Syntax</b>	<code>ExportNetworkData &lt;DesignVariationKey&gt;, &lt;SolnSelectionArray&gt;, &lt;SolnSelector&gt;, &lt;FileFormat&gt;, &lt;OutFile&gt;, &lt;FreqsArray&gt;, &lt;DoRenorm&gt;, &lt;RenormImped&gt;, [Optional &lt;DataType&gt;], [Optional &lt;pass&gt;], [Optional &lt;ComplexFormat&gt;], [Optional &lt;Precision&gt;], [Optional &lt;UseExportFreqs&gt;], [Optional &lt;IncludeGammaComments&gt;], [Optional &lt;SupportNonStdExport&gt;]</code>
<b>VB Example</b>	<pre> oModule.ExportNetworkData "width='2in'", _     Array("Setup1:Sweep1"), 2, "c:\mydir\out.tab", _     Array("all"), false, 0 oModule.ExportNetworkData "width='2in'", _     Array("Setup1:Sweep1", "Setup1:Sweep2"), 3, _     "c:\mydir\out.s2p", Array(1.0e9, 1.5e9, 2.0e9), _ </pre>

	true, 50.0
--	------------

## ExportNMFData

*Use:* Exports s-parameters in neutral file format.

*Command:* In the **matrix** tab of the **Solution** dialog box, click **Export->S-Parameter**. Then select the **Neutral Model** file format.

*Syntax:* ExportNetworkData <SolutionName> <FileName> <ReduceMatrix><Reference Impedance> <FrequencyArray> <DesignVariation> <Format> <Length> <PassNumber>

*Return Value:* None

*Parameters:* <SolutionName>

Type: <String>

Format: <SetupName>:<SolutionName>

Solution that is exported.

<FileName>

Type:<String>

The name of the file. The file extension will determine the file format.

<ReduceMatrix>

Type: <string>

Either "Original" or one of the reduce matrix setup name

<Reference Impedance>

Type: <Double>

Reference impedance

<FrequencyArray>

Type: <Array of doubles>

Value: Array(<double>, <double>....)

Frequency points in the sweep that is exported.

<DesignVariation>

Type: <String>

Design variation at which the solution is exported.

<Format>

Type: <String>

Values: "MagPhase", "RealImag", "DbPhase".

The format in which the sparameters will be exported.

<Length>

Type:<String>

Length for exporting sparameters.

<PassNumber>

Type:<integer>

Pass number.

*VB Example:*

```
oDesign.ExportNMFDATA "Setup7 : LastAdaptive", "C:/temp/neu.nmf", "Original", _ 50, Array  
(10000000, 20000000, 30000000, 40000000, 50000000, 60000000, 70000000, _ 80000000, 90000000,  
100000000), "", "MagPhase", "7meter"
```

**Python Syntax**

```
ExportNMFDData(  
    "",  
    # Empty string.  
    1,  
    #The number 1.  
    "Name",  
    # This is the full path of the file from which the solution is loaded  
    "ExportFile",  
    # full path of file to export to  
    [ "NAME:Frequencies" ],  
    #optional, if none defined all frequencies are used  
    [ "NAME:NMFOptions",  
    #Export NMF options object  
    "DataTypes:=", [ "S" ],  
    #DataTypes can be "S", "Y", "Z", "G", and "Z0", for S , Y, Z matrix, Gamma and Z0 (zero)  
    "DisplayFormat:=", "MA",  
    #DisplayFormat "MA", "RI", "DB"  
    "FileType:=", "",  
    # Export File Type
```

```
2 - Spreadsheet(*.tab)
3 - Touchstone(*.sNp)
4 - Citifile(*.cit)
6 - Neutral format(*.nmf)
7 - Matlab format(*.m)

    "Renormalize:=", false,
#Renormalize true/false

    "RefImpedance:=", 50,
# Reference Impedance

    "Precision:=", 8,
# Number of digits Precision

    "Variables:=", [ "FF", "cap", "Rs"]
# Array of variables

    "Variations:=", [ "", "", ""]
# Array of variations to export solutions for

[ "NAME:ConstantVars"]

#Array of variables that are constant, can be empty

[ "NAME: DependentVars"]

#Array of variables that are dependent, can be empty

    "MatrixSize:=", 2,
#Matrix size, optional (used in nmf file header)
```

	<pre>"CreateNPortModel:=", true #Create a model based on the exported file true/false ])</pre>
<b>Python Example</b>	<pre>oTool = oDesktop.GetTool("NdExplorer") oTool.ExportNetworkData("", True, "C:/.../1000HM.S2P", "\$SYSLIB/Test.s2p", "", [     "NAME:Frequencies",     500000000,     1000000000,     10000000000,     25000000000,     50000000000,     75000000000,      100000000000 ], [     "NAME:TSOptions",     "DataTypes:=" , ["S"],</pre>

```

        "DisplayFormat:="           , "DB",
        "FileType:="               , 3,
        "Renormalize:="            , True,
        "RefImpedance:="           , 50,
        "Precision:="              , 6,
        "UseMultipleCores:="       , False,
        "NumberOfCores:="          , 1,
        "Comments:="                , True,
        "Noise:="                  , False
    ] )

```

## ExportProfile

Exports a solution profile to file.

<b>UI Access</b>	N/A		
<b>Parameters</b>	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.
	<overwriteIfExists>	Boolean	Optional. <ul style="list-style-type: none"> <li>• <b>True</b> - overwrites any existing file of the same name.</li> <li>• <b>False</b> - no overwrite.</li> </ul>
<b>Return Value</b>	None.		

<b>Python Syntax</b>	ExportProfile(<SetupName>, <VariationString>, <filePath>, <overwritelfExists>)
<b>Python Example</b>	<code>oDesign.ExportProfile('Setup1', 'radius=3mm', 'C:\Files\Profile.prof', True)</code>

<b>VB Syntax</b>	ExportProfile <SetupName>, <VariationString>, <filePath>, <overwritelfExists>
<b>VB Example</b>	<code>oDesign.ExportProfile "Setup1", "radius=3mm", "C:\Files\Profile.prof", True</code>

## 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.

<b>UI Access</b>	[HFSS 3D Layout] > Analysis Setup > Generate Mesh.						
<b>Parameters</b>	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>&lt;SimulationNames&gt;</td><td>Array</td><td>List of simulation names.</td></tr></tbody></table>	Name	Type	Description	<SimulationNames>	Array	List of simulation names.
Name	Type	Description					
<SimulationNames>	Array	List of simulation names.					
<b>Return Value</b>	<p>Integer:</p> <ul style="list-style-type: none"><li>• <b>0</b> – Success</li><li>• <b>Else</b> – Failure</li></ul>						
<b>Python Syntax</b>	GenerateMesh(<SimulationNames>)						
<b>Python Example</b>	<code>oDesign.GenerateMesh(['Setup1','Setup2','Setup3'])</code>						
<b>VB Syntax</b>	GenerateMesh <SimulationNames>						

**VB Example**

```
oDesign.GenerateMesh Array('Setup1','Setup2','Setup3')
```

## 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.

<b>UI Access</b>	N/A		
<b>Parameters</b>	Name <type>	Type String	Description Optional. Valid options are "Module", "Editor", and "Variable". Default returns Module and Editor names.
<b>Return Value</b>	Names of children for the queried object.		
<b>Python Syntax</b>	GetChildNames (<type>)		
<b>Python Example</b>	oDesign.GetChildNames ("Variable")		
<b>VB Syntax</b>	GetChildNames <type>		

**VB Example**

```
oDesign.GetChildNames "Variable"
```

## GetChildObject [Design]

Returns the design's child objects.

<b>UI Access</b>	N/A		
<b>Parameters</b>	Name <i>&lt;path&gt;</i>	Type String	Description The path may include multiple generations (for example, "designObject/moduleObj/SetupObject"). See: <a href="#">Object Path</a> .
<b>Return Value</b>	A child object if one is found. Otherwise script error.		

**Python Syntax**

```
GetChildObject(<path>)
```

**Python Example**

```
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")
```

**VB Syntax**

```
GetChildObject <path>
```

<b>VB Example</b>	<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.

<b>UI Access</b>	N/A
<b>Parameters</b>	None.
<b>Return Value</b>	String: "Module", "Editor", or "Variable"

<b>Python Syntax</b>	GetChildTypes()
<b>Python Example</b>	<code>oDesign.GetChildTypes ()</code>

<b>VB Syntax</b>	GetChildTypes
<b>VB Example</b>	<code>oDesign.GetChildTypes</code>

## GetConfigurableData

Obtains configurable data of a specified type

**Note:**

This command is for internal Ansys use only.

<b>UI Access</b>	N/A						
<b>Parameters</b>	<table border="1"><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;type&gt;</td><td>String</td><td>Specified type.</td></tr></table>	Name	Type	Description	<type>	String	Specified type.
Name	Type	Description					
<type>	String	Specified type.					
<b>Return Value</b>	None.						

<b>Python Syntax</b>	GetConfigurableData(<type>)
<b>Python Example</b>	<code>oDesign.GetConfigurableData ("model")</code>

<b>VB Syntax</b>	GetConfigurableData <type>
<b>VB Example</b>	<code>oDesign.GetConfigurableData "model"</code>

## GetData

**Note:**

This command is for internal Ansys use only.

---

<b>Python Syntax</b>	GetData()
<b>Python Example</b>	oDesign.GetData()

## GetDesignID

Returns the unique identification number of the active design.

<b>UI Access</b>	N/A
<b>Parameters</b>	None.
<b>Return Value</b>	String indicating the unique identification number of the active design.

<b>Python Syntax</b>	GetDesignID()
<b>Python Example</b>	<pre>oDesign = oProject.GetActiveDesign() oDesign.GetDesignID()</pre>

<b>VB Syntax</b>	GetDesignType
<b>VB Example</b>	<pre>Set oDesign = oProject.GetActiveDesign oDesign.GetDesignID</pre>

## GetDesignName

Returns the name of the active design.

<b>UI Access</b>	N/A
<b>Parameters</b>	None.
<b>Return Value</b>	String indicating the name of the active design.

<b>Python Syntax</b>	<code>GetDesignName()</code>
<b>Python Example</b>	<pre>oDesign = oProject.GetActiveDesign() oDesign.GetDesignName()</pre>

<b>VB Syntax</b>	<code>GetDesignName</code>
<b>VB Example</b>	<pre>Set oDesign = oProject.GetActiveDesign oDesign.GetDesignName</pre>

## GetDesignType

Returns the design type of the active design.

<b>UI Access</b>	N/A
<b>Parameters</b>	None.
<b>Return Value</b>	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", "RMxprt", or "Twin Builder").

---

<b>Python Syntax</b>	GetDesignType()
<b>Python Example</b>	<pre>oDesign = oProject.GetActiveDesign() oDesign.GetDesignType()</pre>

<b>VB Syntax</b>	GetDesignType
<b>VB Example</b>	<pre>Set oDesign = oProject.GetActiveDesign oDesign.GetDesignType</pre>

ain

## GetDesignValidationInfo

Returns the design's validation information.

<b>UI Access</b>	N/A		
<b>Parameters</b>	Name <i>&lt;validationResultFilePath&gt;</i>	Type String	Description Full path for validation file, including *.xml extension.
<b>Return Value</b>	Boolean: <ul style="list-style-type: none"> <li>• 0 – Failure (validation not done)</li> <li>• 1 – Success (results written to specified location; see file format below)</li> </ul>		

<b>Python Syntax</b>	GetDesignValidationInfo( <i>&lt;validationResultFilePath&gt;</i> )
<b>Python Example</b>	<pre>oDesign.GetDesignValidationInfo('E:/MyProjects/Validation.xml')</pre>

<b>VB Syntax</b>	GetDesignValidationInfo <validationResultFilePath>
<b>VB Example</b>	<code>oDesign.GetDesignValidationInfo "E:/MyProjects/Validation.xml"</code>

## GetEdgePositionAtNormalizedParameter

Gets the position on an edge with normalized parameter.

<b>UI Access</b>	N/A		
<b>Parameters</b>	Name	Type	Description
	<EdgeID>	Integer	Edge ID.
	<NormParam>	Double	Normalized parameter.(Proportional to the length of the specified edge)  For example, 0 leads to the start vertex position of the edge, 1 leads to the end vertex position of the edge, 0.5 leads to the mid position on the edge.
<b>Return Value</b>	Array of string containing the x, y and z coordinate values.		

<b>Python Syntax</b>	GetEdgePositionAtNormalizedParameter(<EdgeID>, <NormParam>)
<b>Python Example</b>	<code>oEditor.GetEdgePositionAtNormalizedParameter(5, 0)</code> <code>oEditor.GetEdgePositionAtNormalizedParameter(5, 1)</code> <code>oEditor.GetEdgePositionAtNormalizedParameter(5, 0.5)</code>

<b>VB Syntax</b>	GetEdgePositionAtNormalizedParameter <EdgeID>, <NormParam>
<b>VB Example</b>	<code>oEditor.GetEdgePositionAtNormalizedParameter 5, 0</code>

	<code>oEditor.GetEdgePositionAtNormalizedParameter 5, 1 oEditor.GetEdgePositionAtNormalizedParameter 5, 0.5</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.

<b>Python Syntax</b>	GetEditor(<editorName>)
<b>Python Example</b>	<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 ("&lt;no-net&gt;", "Trace", "HFSS Setup : Last Adaptive")</pre>
Example with Variables	<pre>oProject = oDesktop.SetActiveProject("Spiral_Inductor_Microstrip") oDesign = oProject.SetActiveDesign("Spiral")</pre>

```

res = oDesign.GetGeometryIdsForNetLayerCombination("<no-net>", "Trace", "HFSS Setup : Last Adaptive")
res = ['Surface', 'FacesList', '21', '22']

```

<b>VB Syntax</b>	GetGeometryIdsForNetLayerCombinations
<b>VB Example</b>	<pre> oDesign = oProject.GetActiveDesign() oDesign.GetGeometryIdsForAllNetLayerCombinations "&lt;no-net&gt;", "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](#).

<b>UI Access</b>	N/A						
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;setupName&gt;</td> <td>String</td> <td>Name of the setup.</td> </tr> </tbody> </table>	Name	Type	Description	<setupName>	String	Name of the setup.
Name	Type	Description					
<setupName>	String	Name of the setup.					
<b>Return Value</b>	Array of strings indicating face and edge ID numbers.						

<b>Python Syntax</b>	GetGeometryIdsForAllNetLayerCombinations ()
<b>Python Example</b>	<pre> oDesign = oProject.GetActiveDesign() </pre>

	<pre>oDesign.GetGeometryIdsForAllNetLayerCombinations("HFSS Setup : Last Adaptive")</pre>
<b>Example with Variables</b>	<pre>oProject = oDesktop.SetActiveProject("Spiral_Inductor_Microstrip") oDesign = oProject.SetActiveDesign("Spiral") res = oDesign.GetGeometryIdsForAllNetLayerCombinations("HFSS Setup : Last Adaptive") res = ['PlotGeomInfo for &lt;no-net&gt;/&lt;no-layer&gt; (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 &lt;no-net&gt;/AirbridgeMetal (net/layer combination):', 'Surface', 'FacesList', '14', 'PlotGeomInfo for &lt;no-net&gt;/BottomGround (net/layer combination):', 'Surface', 'FacesList', '29', 'PlotGeomInfo for &lt;no-net&gt;/Trace (net/layer combination):', 'Surface', 'FacesList', '21', '22']</pre>

<b>VB Syntax</b>	GetGeometryIdsForAllNetLayerCombinations
<b>VB Example</b>	<pre>oDesign = oProject.GetActiveDesign() oDesign.GetGeometryIdsForAllNetLayerCombinations "HFSS Setup : Last Adaptive"</pre>

## GetManagedFilePath

Get the path to the project's results folder.

<b>UI Access</b>	N/A
<b>Parameters</b>	None.

<b>Return Value</b>	String containing path where project results are located.
---------------------	-----------------------------------------------------------

<b>Python Syntax</b>	<code>oDesign.GetManagedFilesPath()</code>
<b>Python Example</b>	<code>oDesign.GetManagedFilesPath()</code>

<b>VB Syntax</b>	<code>oDesign.GetManagedFilesPath</code>
<b>VB Example</b>	<code>oDesign.GetManagedFilesPath</code>

## GetModule

Returns the IDispatch for the specified module.

<b>UI Access</b>	N/A		
<b>Parameters</b>	Name <code>&lt;modulename&gt;</code>	Type String	Description One of the following: <ul style="list-style-type: none"><li>• <a href="#">Analysis Module</a> – "AnalysisSetup"</li><li>• <a href="#">Boundary Module</a> – "BoundarySetup"</li><li>• <a href="#">Field Overlays Module</a> – "FieldsReporter"</li><li>• Mesh Module – "MeshSetup"</li><li>• <a href="#">Optimetrics Module</a> – "Optimetrics"</li><li>• <a href="#">Radiation Module</a> – "RadField"</li><li>• <a href="#">Radiation Setup Manager Module</a> – "RadiationSetupMgr"</li><li>• Reduce Matrix Module – "ReduceMatrix"</li></ul>

			<ul style="list-style-type: none"><li>• <a href="#">Reporter Module</a> – "ReportSetup"</li><li>• <a href="#">Simulation Setup Module</a> – "SimSetup"</li><li>• <a href="#">Solutions Module</a> – "Solutions"</li></ul>
<b>Return Value</b>	Module IDispatch		

<b>Python Syntax</b>	GetModule (<modulename>)
<b>Python Example</b>	oModule = oDesign.GetModule("SimSetup")

<b>VB Syntax</b>	GetModule <modulename>
<b>VB Example</b>	set oModule = oDesign.GetModule "SimSetup"

## GetModule (Layout Editor)

Get the IDispatch for the specified module.

*Command:* None

*Syntax:* GetModule <"modulename"> <object\_variable>

*Return Value:* Module IDispatch

*Parameters:* <"modulename">

Type: string

<object\_variable>

Type: string

*VB Example:* oDesign.GetModule "modulename" object\_variable

## GetModule (RadiationSetupMgr)

Returns the IDispatch for the specified module.

UI Access	N/A		
Parameters	Name <modulename>	Type String	Description One of the following: <ul style="list-style-type: none"><li>• <a href="#">Analysis Module</a> – "AnalysisSetup"</li><li>• <a href="#">Boundary Module</a> – "BoundarySetup"</li><li>• <a href="#">Field Overlays Module</a> – "FieldsReporter"</li><li>• Mesh Module – "MeshSetup"</li><li>• <a href="#">Optimetrics Module</a> – "Optimetrics"</li><li>• <a href="#">Radiation Module</a> – "RadField"</li><li>• <a href="#">Radiation Setup Manager Module</a> – "RadiationSetupMgr"</li><li>• Reduce Matrix Module – "ReduceMatrix"</li><li>• <a href="#">Reporter Module</a> – "ReportSetup"</li><li>• <a href="#">Simulation Setup Module</a> – "SimSetup"</li><li>• <a href="#">Solutions Module</a> – "Solutions"</li></ul>
Return Value	Module IDispatch		

<b>Python Syntax</b>	GetModule (<modulename>)
<b>Python Example</b>	<pre>oModule = oDesign.GetModule("SimSetup")</pre>

<b>VB Syntax</b>	GetModule <modulename>
<b>VB Example</b>	<pre>set oModule = oDesign.GetModule("SimSetup")</pre>

## GetName

Returns the name and ID of the active design. If the design is a sub-circuit, returns the parent design also.

<b>UI Access</b>	N/A
<b>Parameters</b>	None.
<b>Return Value</b>	String indicating the name and ID of the active design and parent design (if the active design is a sub-circuit).

<b>Python Syntax</b>	GetName()
<b>Python Example</b>	<pre>design_name = oDesign.GetName()</pre>

<b>VB Syntax</b>	GetName
<b>VB Example</b>	<pre>design_name = oDesign.GetName</pre>

## 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

## GetNominalVariation

Returns the current nominal variation.

<b>UI Access</b>	N/A
<b>Parameters</b>	None.
<b>Return Value</b>	String containing current nominal variation.

<b>Python Syntax</b>	GetNominalVariation()
<b>Python Example</b>	oDesign.GetNominalVariation()

<b>VB Syntax</b>	GetNominalVariation
------------------	---------------------

<b>VB Example</b>	<code>oDesign.GetNominalVariation</code>
-------------------	------------------------------------------

## GetNoteText

Returns the text of the note attached to a design.

<b>UI Access</b>	N/A
<b>Parameters</b>	None.
<b>Return Value</b>	String: text of the design note.

<b>Python Syntax</b>	<code>GetNoteText()</code>
<b>Python Example</b>	<code>oDesign.GetNoteText ()</code>

<b>VB Syntax</b>	<code>GetNoteText</code>
<b>VB Example</b>	<code>oDesign.GetNoteText</code>

## GetObjPath [Design]

Obtains the path to the design.

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

<b>Python Syntax</b>	GetObjPath()
<b>Python Example</b>	<code>oDesign.GetObjPath()</code>

<b>VB Syntax</b>	GetObjPath
<b>VB Example</b>	<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.

<b>UI Access</b>	N/A																	
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><code>&lt;OutputVarName&gt;</code></td> <td>String</td> <td>Name of the output variable.</td> </tr> <tr> <td><code>&lt;IntrinsicVariation&gt;</code></td> <td>String</td> <td>A set of intrinsic variable value pairs to use when evaluating the output expression.  HFSS example:  <code>"Freq='20GHz' Theta='20deg' Phi='30deg'"</code></td> </tr> <tr> <td><code>&lt;SolutionName&gt;</code></td> <td>String</td> <td>Name of the solution as listed in the output variable UI. For example, <code>"Setup1 : Last Adaptive"</code>.</td> </tr> <tr> <td><code>&lt;ReportType&gt;</code></td> <td>String</td> <td>The name of the report type as seen in the output variable UI.  Possible values in Layout/Schematic Editor are:           <ul style="list-style-type: none"> <li>• "Standard"</li> </ul> </td> </tr> </tbody> </table>			Name	Type	Description	<code>&lt;OutputVarName&gt;</code>	String	Name of the output variable.	<code>&lt;IntrinsicVariation&gt;</code>	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>	<code>&lt;SolutionName&gt;</code>	String	Name of the solution as listed in the output variable UI. For example, <code>"Setup1 : Last Adaptive"</code> .	<code>&lt;ReportType&gt;</code>	String	The name of the report type as seen in the output variable UI.  Possible values in Layout/Schematic Editor are: <ul style="list-style-type: none"> <li>• "Standard"</li> </ul>
Name	Type	Description																
<code>&lt;OutputVarName&gt;</code>	String	Name of the output variable.																
<code>&lt;IntrinsicVariation&gt;</code>	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>																
<code>&lt;SolutionName&gt;</code>	String	Name of the solution as listed in the output variable UI. For example, <code>"Setup1 : Last Adaptive"</code> .																
<code>&lt;ReportType&gt;</code>	String	The name of the report type as seen in the output variable UI.  Possible values in Layout/Schematic Editor are: <ul style="list-style-type: none"> <li>• "Standard"</li> </ul>																

			<ul style="list-style-type: none"> <li>• "Load Pull"</li> <li>• "Constellation"</li> <li>• "Data Table"</li> <li>• "Eye Diagram"</li> <li>• "Statistical"</li> </ul>
	<ContextArray>	Array	<p>Structured array containing context for which the output variable expression is being evaluated. Can be empty.</p> <pre>Array("Context:=", &lt;string&gt;)</pre>
<b>Return Value</b>	Double value of the output variable.		

<b>Python Syntax</b>	GetOutputVariableValue(<OutputVarName>, <IntrinsicVariation>, <SolutionName>, <ReportTypeName>, <ContextArray>)
<b>Python Example</b>	<pre>Val = oDesign.GetOutputVariableValue("test", "Freq = '20Ghz' Theta='20deg' Phi='30deg'", "TR", "Standard", [])</pre>

<b>VB Syntax</b>	GetOutputVariableValue <OutputVarName>, <IntrinsicVariation>, <SolutionName>, <ReportTypeName>, <ContextArray>
<b>VB Example</b>	<pre>Val = oDesign.GetOutputVariableValue "test", "Freq = '20Ghz' Theta='20deg' Phi='30deg'", "TR", "Standard", Array()</pre>

## GetOutputVariables

Returns the list of output variables.

<b>UI Access</b>	N/A
<b>Parameters</b>	None.
<b>Return Value</b>	Array containing all output variables.

<b>Python Syntax</b>	GetOutputVariables()
<b>Python Example</b>	<code>oDesign.GetOutputVariables ()</code>

<b>VB Syntax</b>	GetOutputVariables
<b>VB Example</b>	<code>oDesign.GetOutputVariables</code>

## GetPostProcessingVariables

Returns the list of post-processing variables.

<b>UI Access</b>	N/A
<b>Parameters</b>	None.
<b>Return Value</b>	Returns array containing variables.

<b>Python Syntax</b>	GetPostProcessingVariables()
<b>Python Example</b>	<code>oDesign.GetPostProcessingVariables()</code>

<b>VB Syntax</b>	GetPostProcessingVariables
<b>VB Example</b>	<code>oDesign.GetPostProcessingVariables</code>

## GetPropHost

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

*Command:* None

*Syntax:* GetPropHost()

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

Returns the interface to the properties of the complInstance.

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

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

## GetProperties

Gets a list of all the properties belonging to a specific <PropServer> and <PropTab>. This can be executed by the oProject, oDesign, or oEditor variables.

<b>UI Access</b>	N/A		
<b>Parameters</b>	Name	Type	Description
	<PropTab>	String	<p>One of the following, where tab titles are shown in parentheses:</p> <ul style="list-style-type: none"> <li>• PassedParameterTab ("Parameter Values")</li> <li>• DefinitionParameterTab (Parameter Defaults")</li> <li>• LocalVariableTab ("Variables" or "Local Variables")</li> <li>• ProjectVariableTab ("Project variables")</li> <li>• ConstantsTab ("Constants")</li> <li>• BaseElementTab ("Symbol" or "Footprint")</li> <li>• ComponentTab ("General")</li> <li>• Component("Component")</li> <li>• CustomTab ("Intrinsic Variables")</li> <li>• Quantities ("Quantities")</li> <li>• Signals ("Signals")</li> </ul>
<b>Return Value</b>	Array of strings containing the names of the appropriate properties.		

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

<b>VB Syntax</b>	GetProperties <PropTab>, <PropServer>
------------------	---------------------------------------

**VB Example**

```
oEditor.GetProperties "PassedParameterTab", "k"
```

## GetPropertyValue

Returns the value of a single property belonging to a specific <PropServer> and <PropTab>. This function is available with the Project, Design or Editor objects, including definition editors.

**Tip:**

Use the script recording feature and edit a property, and then view the resulting script to see the format for that property.

UI Access	N/A		
	Name	Type	Description
Parameters	<PropTab>	String	<p>One of the following, where tab titles are shown in parentheses:</p> <ul style="list-style-type: none"><li>• PassedParameterTab ("Parameter Values")</li><li>• DefinitionParameterTab (Parameter Defaults")</li><li>• LocalVariableTab ("Variables" or "Local Variables")</li><li>• ProjectVariableTab ("Project variables")</li><li>• ConstantsTab ("Constants")</li><li>• BaseElementTab ("Symbol" or "Footprint")</li><li>• ComponentTab ("General")</li><li>• Component("Component")</li><li>• CustomTab ("Intrinsic Variables")</li><li>• Quantities ("Quantities")</li></ul>

		<ul style="list-style-type: none"> <li>• Signals ("Signals")</li> </ul>
<PropServer>	String	An object identifier, generally returned from another script method, such as CompInst@R;2;3
<PropName>	String	Name of the property.
<b>Return Value</b>	String value of the property.	

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

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

## GetPropNames [Design]

Returns array containing names of property. For designs, always returns an empty array because the design has no property.

<b>UI Access</b>	N/A		
<b>Parameters</b>	Name <i>&lt;includeReadOnly&gt;</i>	Type Boolean	Description (Optional). <ul style="list-style-type: none"><li>• True - include read only property.</li><li>• False - do not include read only property.</li></ul>
<b>Return Value</b>	Empty array.		
<b>Python Syntax</b>	GetPropNames( <i>&lt;includeReadOnly&gt;</i> )		
<b>Python Example</b>	<pre>oDesign.GetPropNames()</pre>		
<b>VB Syntax</b>	GetPropNames <i>&lt;includeReadOnly&gt;</i>		
<b>VB Example</b>	<pre>oDesign.GetPropNames</pre>		

## GetPropertyValue [Design]

Returns the property value for the active design object, or specified property values.

<b>UI Access</b>	N/A		
<b>Parameters</b>	Name <i>&lt;propPath&gt;</i>	Type String	Description A child object's property path. See: <a href="#">Object Property Script Function Summary</a> .
<b>Return Value</b>	The property value (integer, string, or array of strings) of the specified child object.		
<b>Python Syntax</b>	GetPropertyValue( <i>&lt;propPath&gt;</i> )		

<b>Python Example</b>	<pre>oDesign.GetPropValue('offset/SIValue') oDesign.GetPropValue('Results/S Parameter Plot 1/Display Type') oDesign.GetPropValue('Results/S Parameter Plot 1/Display Type/Choices')</pre>
-----------------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

<b>VB Syntax</b>	GetPropValue(<propPath>)
<b>VB Example</b>	<pre>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")</pre>

## GetSelections [Design]

This script serves no function at the Design level. See: [GetSelections \(Layout Editor\)](#), [GetSelections \(Model Editor\)](#), or Get Selections (Schematic Editor).

## GetSolutionType

Returns solution type of the design.

<b>UI Access</b>	N/A
<b>Parameters</b>	None.
<b>Return Value</b>	<p>String containing the solution type.</p> <p>Possible values are: "SBR+", "HFSS [Modal   Terminal] [ Network   Composite]", "Transient [Network   Composite]", "Eigenmode", or "Characteristic".</p>

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

<b>Python Example</b>	<code>oDesign.GetSolutionType()</code>
-----------------------	----------------------------------------

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

<b>VB Example</b>	<code>oDesign.GetSolutionType</code>
-------------------	--------------------------------------

## GetSolveInsideThreshold

Returns the solve inside threshold. This command does not apply to HFSS-IE.

<b>UI Access</b>	N/A
<b>Parameters</b>	None.
<b>Return Value</b>	Double representing the solve inside threshold.

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

<b>Python Example</b>	<code>oDesign.GetSolveInsideThreshold()</code>
-----------------------	------------------------------------------------

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

<b>VB Example</b>	<code>oDesign.GetSolveInsideThreshold</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`.

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

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

<b>VB Syntax</b>	<code>GetVariables</code>
<b>VB Example</b>	<code>oProject.GetVariables</code> <code>oDesign.GetVariables</code>

## GetVariableValue

Gets the value of a single specified variable. To get the value of project variables, execute this command using `oProject`. To get the value of local variables, use `oDesign`.

<b>UI Access</b>	N/A
<b>Parameters</b>	<code>Name</code>   <code>Type</code>   <code>Description</code>

	<VarName>	String	Name of the variable to access.
<b>Return Value</b>	String represents the value of the variable.		

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

<b>VB Syntax</b>	GetVariableValue <VarName>
<b>VB Example</b>	<code>oProject.GetVariableValue "var_name"</code>

## GetVariationVariableValue

Returns the value for a specified variation's variable.

<b>UI Access</b>	N/A		
<b>Parameters</b>	Name	Type	Description
	<VariationString>	String	The name of the design variation.
	<VariableName>	String	The name of the variable.
<b>Return Value</b>	Returns a double precision value in SI units, interpreted to mean the value of the variable contained in the variation string.		

<b>Python Syntax</b>	GetVariationVariableValue(<VariationString>, <VariableName>)
----------------------	--------------------------------------------------------------

<b>Python Example</b>	<code>oDesign.GetVariationVariableValue('x_size = 2mm y_size = 1mm', 'y_size')</code>
-----------------------	-------------------------------------------------------------------------------------------

<b>VB Syntax</b>	<code>GetVariationVariableValue &lt;VariationString&gt;, &lt;VariableName&gt;</code>
<b>VB Example</b>	<code>varval = oDesign.GetVariationVariableValue "x_size = 2mm y_size = 1mm", "y_size"</code>

## 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.

<b>UI Access</b>	<b>Project &gt; Datasets &gt; Import.</b>		
<b>Parameters</b>	Name	Type	Description
	<code>&lt;datasetFullPath&gt;</code>	String	The full path to the file containing the dataset values. *.tab files recommended (see <a href="#">note</a> below).
<b>Return Value</b>	None.		

<b>Python Syntax</b>	<code>ImportDataset (&lt;datasetFullPath&gt;,&lt;optionalDatasetName&gt;)</code>
<b>Python Example</b>	<code>oProject.ImportDataset ('e:\tmp\dsdata.tab') oDesign.ImportDataset ('e:\tmp\dsdata.tab')</code>

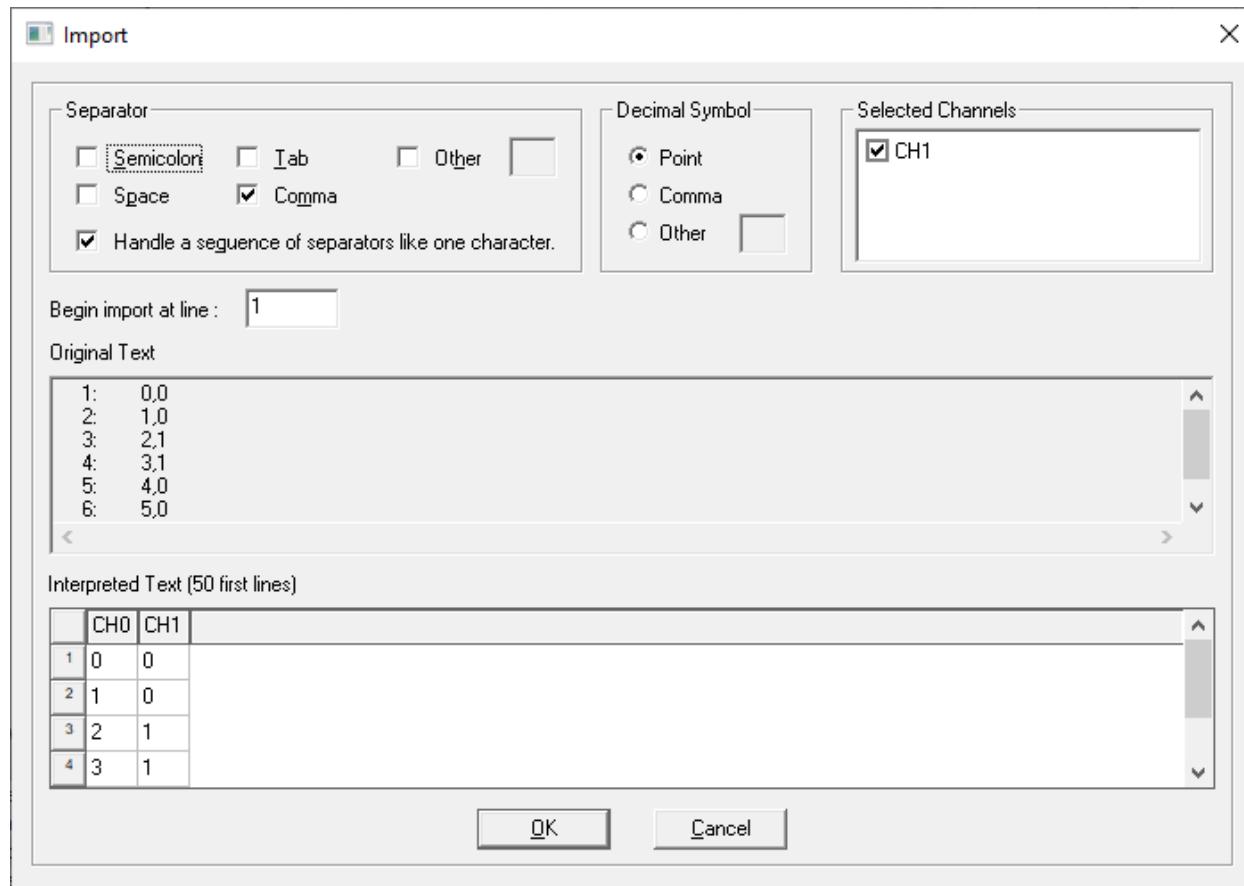
```
oProject.ImportDataset('e:\tmp\dsdata.tab', 'MyDatasetName')
oDesign.ImportDataset('e:\tmp\dsdata.tab', 'MyDatasetName')
```

<b>VB Syntax</b>	ImportDataset <datasetFileFullPath>, <optionalDatasetName>
<b>VB Example</b>	<pre>oProject.ImportDataset "e:\tmp\dsdata.tab" oDesign.ImportDataset "e:\tmp\dsdata.tab" oProject.ImportDataset "e:\tmp\dsdata.tab", "MyDatasetName" oDesign.ImportDataset "e:\tmp\dsdata.tab", "MyDatasetName"</pre>

## Note About File Types:

Tab-delimited or space-delimited files with the extension \*.tab are the recommended file type. When using ImportDataset at the Design level, \*.tab is the only file type supported.

At the Project level, other file types are supported (for example, \*.csv). However, after calling the command, you must configure the file import format manually through the Electronics Desktop GUI by selecting **Project > Datasets** and clicking **Import**.



## 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].		
<b>Parameters</b>	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.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	InsertDesign (<DesignType>, <DesignName>, <TechnologyFile>:<SubCircuitID>, "")
<b>Python Example</b>	<pre>oProject.InsertDesign('Nexxim Circuit', 'MyDesigner', 'C:\\Program Files\\AnsysEM\\Designer\\', '')</pre>

<b>VB Syntax</b>	InsertDesign <DesignType>, <DesignName>, <TechnologyFile>:<SubCircuitID>, ""
<b>VB Example</b>	<pre>oProject.InsertDesign "Nexxim Circuit", "MyDesigner", "C:\\Program Files\\AnsysEM\\Designer\\", ""</pre>

## 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	<b>Edit &gt; Paste.</b>		
<b>Parameters</b>	Name <code>&lt;pasteOption&gt;</code>	Type Integer	Description One of the following: <ul style="list-style-type: none"><li>• <b>0</b> – Link to the existing design that was copied</li><li>• <b>1</b> – Create a new copy of the design and keep the original layers of the design being copied</li><li>• <b>2</b> – Create a new copy of the design and merge the layers of the design being copied.</li></ul>
<b>Return Value</b>	None.		

<b>Python Syntax</b>	<code>PasteDesign(&lt;pasteOption&gt;)</code>
<b>Python Example</b>	<pre> oProject.CopyDesign('DesignB') oDesign = oProject.SetActiveDesign('DesignA') oDesign.PasteDesign(1) </pre>

<b>VB Syntax</b>	<code>PasteDesign &lt;pasteOption&gt;</code>
<b>VB Example</b>	<pre> oProject.CopyDesign "B" Set oDesign = oProject.SetActiveDesign ("A") oDesign.PasteDesign 1 </pre>

## PasteItemCommand (Layout Editor)

Paste tree items, such as Alstrasim 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	<ul style="list-style-type: none"><li>• Layout Editor</li><li>• Schematic Editor</li><li>• Select a Nexxim solution in a 3D Layout design, right click, and choose <b>Push Excitations</b>.</li></ul>		
Parameters	Name <Reference Designation>	Type String	Description "<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:

```
Array ("NAME:options",
       "CalcThevenin =", <true|false>,
       "Sol:=", "<solution name>")
```

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:

```
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>")
```

winType can have the following values:

- Rectangular

			<ul style="list-style-type: none"> <li>• Bartlett</li> <li>• Blackman</li> <li>• Hamming</li> <li>• Hanning</li> <li>• Kaiser</li> <li>• Welch</li> <li>• Weber</li> <li>• Lanzcos</li> </ul>
<b>Return Value</b>	None		

**Note:**

For more information about the component name, schematic IDs, and formats, see the section [Editor Scripting IDs](#).

<b>VB Syntax</b>	PushExcitations "<refdes>", Array("NAME:options", "transient:=", ["CalcThevenin =", <true>], Array("start:=", <start time>, "stop:=", <stop time>, "maxHarmonics:=", <max harmonics>, "winType:=", <>window>, ["widthPct:=", <width percentage>], ["kaiser:=", <Kaiser value>], ["correctCoherentGain:=", true]), "Sol:=", "<solution name>")
<b>VB Example</b>	<p>For a transient solution:</p> <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")

```

## Redo [Design]

Reapplies the last design-level command.

<b>UI Access</b>	<b>Edit &gt; Redo.</b>
<b>Parameters</b>	None.

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

<b>Python Syntax</b>	Redo()
<b>Python Example</b>	<code>oDesign.Redo ()</code>

<b>VB Syntax</b>	Redo
<b>VB Example</b>	<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.

<b>UI Access</b>	Right-click a design instance in the project tree, and then click <b>Rename</b> on the shortcut menu.		
<b>Parameters</b>	Name	Type	Description
	<OldName>	String	The current name of the design, which must be the design on which this command is invoked.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	RenameDesignInstance (<OldName>, <NewName>)
<b>Python Example</b>	<code>oDesign.RenameDesignInstance ("Design1", "Design2")</code>

<b>VB Syntax</b>	RenameDesignInstance <OldName>, <NewName>
<b>VB Example</b>	<code>oDesign.RenameDesignInstance "Design1", "Design2"</code>

## **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.

<b>UI Access</b>	[HFSS 3D Layout] > Toolkit > [Script Name].												
<b>Parameters</b>	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>&lt;LibraryType&gt;</td><td>String</td><td>Name of the library in which the script is located.</td></tr><tr><td>&lt;ToolkitName&gt;</td><td>String</td><td>Name of the Python script.</td></tr><tr><td>&lt;ToolkitArg&gt;</td><td>Array</td><td>Structured array containing arguments required by the toolkit script.</td></tr></tbody></table>	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.
Name	Type	Description											
<LibraryType>	String	Name of the library in which the script is located.											
<ToolkitName>	String	Name of the Python script.											
<ToolkitArg>	Array	Structured array containing arguments required by the toolkit script.											
<b>Return Value</b>	User-defined solution(s) and/or output(s).												
<hr/>													

<b>Python Syntax</b>	RunToolkit(<LibraryType>, <ToolkitName>, <ToolkitArg>)
<b>Python Example</b>	oDesign.RunToolkit("SysLib", "Hearing Aid Compliance", [])

<b>VB Syntax</b>	RunToolkit <LibraryType>, <ToolkitName>, <ToolkitArg>
<b>VB Example</b>	oDesign.RunToolkit "SysLib", "Wavelength Calculator", Array() oDesign.RunToolkit "SysLib", "Hearing Aid Compliance", Array()

## SARSetup

Sets up for the specific absorption rate (SAR) computation. This command does not apply to HFSS-IE.

UI Access	HFSS > Fields > SAR Setting.		
Parameters	Name	Type	Description
	<TissueMass>	Double	Value represents mass of tissue between 1 and 10 in grams.
	<MaterialDensity>	Double	Density of material in gram/cm <sup>3</sup> .
	<VoxelSize>	Double	Size of a voxel in millimeters.
<AverageSARMethod>	Integer	0 - IEEE std 1528.	
		1 - Gridless, i.e. classical Ansoft method.	
Return Value	None.		

Python Syntax	SARSetup(<TissueMass>, <MaterialDensity>, <TissueObjectListID>, <VoxelSize>, <AverageSARMethod>)
Python Example	oDesign.SARSetup(1.0, 1.0, 678, 1.0, 0)

VB Syntax	SARSetup <TissueMass>, <MaterialDensity>, <TissueObjectListID>, <VoxelSize>, <AverageSARMethod>
VB Example	oDesign.SARSetup 1.0, 1.0, 678, 1.0, 0

## SetActiveEditor

Sets the active editor.

UI Access	N/A.		
Parameters	Name	Type	Description

	<EditorName>	String	Text of the design's notes.
<b>Return Value</b>	Editor object.		

<b>Python Syntax</b>	SetActiveEditor(<EditorName>)
<b>Python Example</b>	<code>oDesign.SetActiveEditor('3D Modeler')</code>

<b>VB Syntax</b>	SetActiveEditor <EditorName>
<b>VB Example</b>	<code>oDesign.SetActiveEditor "3D Modeler"</code>

## 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>`

## SetAllowMaterialOverride

Sets the option to allow material override.

<b>UI Access</b>	N/A.		
<b>Parameters</b>	Name <code>&lt;allowMaterialOverride&gt;</code>	Type Integer	Description 1 - allow material override. 0 - does not allow material override.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	<code>SetAllowMaterialOverride(&lt;allowMaterialOverride&gt;)</code>
<b>Python Example</b>	<code>oDesign.SetAllowMaterialOverride(1)</code>

<b>VB Syntax</b>	<code>SetAllowMaterialOverride &lt;allowMaterialOverride&gt;</code>
<b>VB Example</b>	<code>oDesign.SetAllowMaterialOverride 1</code>

## SetBackgroundMaterial

Sets the design's background material.

**Important:**

You can only use this script in 2D Extractor if there is no surface ground in the design *and* the problem type is "open".

<b>UI Access</b>	From the <b>Project Manager</b> , right-click the design and select <b>Set Background Material</b> .
------------------	------------------------------------------------------------------------------------------------------

<b>Parameters</b>	Name <i>&lt;matName&gt;</i>	Type String	Description Material name.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	SetBackgroundMaterial( <i>&lt;matName&gt;</i> )
<b>Python Example</b>	<code>oDesign.SetBackgroundMaterial ('vacuum')</code>

<b>VB Syntax</b>	SetBackgroundMaterial <i>matName</i>
<b>VB Example</b>	<code>oDesign.SetBackgroundMaterial "vacuum"</code>

## SetDesignMode

Switches between HFSS 3D Layout operating modes.

<b>UI Access</b>	Edit Layout...		
<b>Parameters</b>	Name <i>&lt;modeName&gt;</i>	Type String	Description Enter "IC" or "General" to change operating modes.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	SetDesignMode( <i>&lt;modeName&gt;</i> )
<b>Python Example</b>	<code>oDesign.SetDesignMode ("IC")</code>

<b>VB Syntax</b>	SetDesignMode <modeName>
<b>VB Example</b>	<code>oDesign.SetDesignMode "IC"</code>

## SetDesignSettings

Sets the design settings.

UI Access	HFSS > Design Settings		
<b>Parameters</b>	Name <i>&lt;DesignSettings&gt;</i>	Type Array	Description ["NAME:<DesignSettingsName>"," "Use Advanced DC Extrapolation:=" <boolean>, "Port Validation Settings:=" <String of ["Standard"   "Extended"]> "Calculate Lossy Dielectrics:=" <boolean> "Use Power S:=" <boolean>, "Export After Simulation:=" <boolean>, "Export FRTM After Simulation:=" <boolean>, "Export Dir:=" <string of path>, "Allow Material Override:=" <boolean>, "Calculate Lossy Dielectrics:=" <boolean>, "EnabledObjects:=" <array of objects>, "Maximum Number of Frequencies for Broadband Adapt:=" <integer>]  <i>&lt;ModelValidationSettings&gt;</i>
		Array	["NAME:Validation Options",

		"EntityCheckLevel:=" <String of ["strict"   "none"   "warningOnly"   "basic"]>, "IgnoreUnclassifiedObjects:=" <boolean>, "SkipIntersectionChecks:=" <boolean>), "Design Validation Settings:=" "Perform full validations" )
<b>Return Value</b>	None.	

<b>Python Syntax</b>	SetDesignSettings(<propName>, <description>)
<b>Python Example</b>	Design.SetDesignSettings( [ "NAME:Design Settings Data", "Use Power S:=" , False, "Export FRTM After Simulation:=", True, "Allow Material Override:=", False, "Calculate Lossy Dielectrics:=", False, "Perform Minimal validation:=", False, "EnabledObjects:=" , [], "Port Validation Settings:=", "Standard", "Save Adaptive support files:=", True ],

```
[  
    "NAME:Model Validation Settings",  
    "EntityCheckLevel:=", "Strict",  
    "IgnoreUnclassifiedObjects:=", False,  
    "SkipIntersectionChecks:=", False  
])
```

<b>Python Syntax</b>	<code>SetDesignSettings (&lt;DesignSettings&gt;, &lt;ModelValidationSettings&gt;)</code>
<b>Python Example</b>	<pre>oDesign.SetDesignSettings(     [         "NAME:Design Settings Data",         "Use Advanced DC Extrapolation:=", True,         "Use Power S:=", False,         "Export After Simulation:=", False,         "Allow Material Override:=", True,         "Calculate Lossy Dielectrics:=", True,         "Perform Minimal validation:=", False,         "EnabledObjects:=", [],         "Port Validation Settings:=", "Standard"         "Maximum Number of Frequencies for Broadband Adapt:=", 5</pre>

```
],  
[  
    "NAME:Model Validation Settings",  
    "EntityCheckLevel:=", "Strict",  
    "IgnoreUnclassifiedObjects:=", False,  
    "SkipIntersectionChecks:=", False  
])
```

<b>VB Syntax</b>	SetDesignSettings <DesignSettings>, <ModelValidationSettings>
<b>VB Example</b>	<pre>oDesign.SetDesignSettings     Array(         "NAME:Design Settings Data",         "Use Advanced DC Extrapolation:=", True,         "Use Power S:=", False,         "Export After Simulation:=", False,         "Allow Material Override:=", True,         "Calculate Lossy Dielectrics:=", True,         "Perform Minimal validation:=", False,         "EnabledObjects:=", [])</pre>

```
"Port Validation Settings:=", "Standard"
"Maximum Number of Frequencies for Broadband Adapt:=", 5
),
Array(
    "NAME:Model Validation Settings",
    "EntityCheckLevel:=", "Strict",
    "IgnoreUnclassifiedObjects:=", False,
    "SkipIntersectionChecks:=", False
)
```

## 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*

```
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 HFSS 3D show outlines of objects across layers during view changes. This improves the visualization performance.

UI Access	Fast Transformation		
Parameters	Name <i>&lt;Layout ComponentName&gt;</i>	Type String	Description Name of the Layout Component.
	Boolean	Type String	Description True or False
Return Value	None.		

Python Syntax	SetFastTransformationForLayoutComponent ("<LayoutComponentName>", <boolean> )
Python Example	oDesign.SetFastTransformationForLayoutComponent ("LC1_1", False)

## SetLengthSettings

Sets the distributed and lumped lengths of the design, as well as the rise time.

UI Access	N/A		
Parameters	Name <i>&lt;DistributedUnits&gt;</i>	Type String	Description Length units used for post-processing.
	<i>&lt;LumpedLength&gt;</i>	Type String	Description Length of design in post-processing.

---

	<i>&lt;RiseTime&gt;</i>	String	Time used in post-processing.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	SetLengthSettings(<DistributedUnits>, <LumpedLength>, <RiseTime>)
<b>Python Example</b>	<code>oDesign.SetLengthSettings ('mm', '7meter', '1s')</code>

<b>VB Syntax</b>	SetLengthSettings<DistributedUnits>, <LumpedLength>, <RiseTime>)
<b>VB Example</b>	<code>oDesign.SetLengthSettings "mm", "7meter", "1s"</code>

## SetObjectAttributesForLayoutComponent

Sets visualization attributes for layout components in HFSS 3D for a named component for layers, net and objects.

UI Access	Object Attributes		
Parameters	Name <i>&lt;Layout ComponentName&gt;</i>	Type String	Description Name of the Layout Component.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	SetObjectAttributesForLayoutComponent (["<LayoutComponentName>',[ <ObjectAttributesinNetMode>][<ObjectAttributesinLayerMode>],[ <ObjectAttributesinObjectMode>])
<b>Python Example</b>	<code>oDesign.SetObjectAttributesForLayoutComponent (</code> [ <code>"Name :="</code> <code>,</code> <code>"LC1_1",</code>

```
"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]

] ,
[

    "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],
            "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]
        ]
    )
}
```

## SetPropValue [Design]

Sets the property value for the active property object.

<b>UI Access</b>	Edit properties on Project Tree objects.									
<b>Parameters</b>	<table border="1"><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;propPath&gt;</td><td>String</td><td>A child object's property path. See: <a href="#">Object Property Script Function Summary</a>.</td></tr><tr><td>&lt;Value&gt;</td><td>String</td><td>New property value.</td></tr></table>	Name	Type	Description	<propPath>	String	A child object's property path. See: <a href="#">Object Property Script Function Summary</a> .	<Value>	String	New property value.
Name	Type	Description								
<propPath>	String	A child object's property path. See: <a href="#">Object Property Script Function Summary</a> .								
<Value>	String	New property value.								
<b>Return Value</b>	Boolean: <ul style="list-style-type: none"><li>• <b>True</b> - property found and the new value is valid.</li><li>• <b>False</b> - property not found.</li></ul>									

<b>Python Syntax</b>	SetPropValue(<propPath>, <Value>)
<b>Python Example</b>	<pre>oDesign.SetPropValue("offset", "12mm") oDesign.SetPropValue("Results/S Parameter Plot 1/Display Type", "Rectangular Plot")</pre>

<b>VB Syntax</b>	SetPropValue <propPath>, <Value>
<b>VB Example</b>	<pre>oDesign.SetPropValue "offset", "12mm" oDesign.SetPropValue "Results/S Parameter Plot 1/Display Type", "Rectangular Plot"</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	<p>One of the following, where tab titles are shown in parentheses:</p> <ul style="list-style-type: none"> <li>• PassedParameterTab ("Parameter Values")</li> <li>• DefinitionParameterTab (Parameter Defaults")</li> <li>• LocalVariableTab ("Variables" or "Local Variables")</li> <li>• ProjectVariableTab ("Project variables")</li> <li>• ConstantsTab ("Constants")</li> <li>• BaseElementTab ("Symbol" or "Footprint")</li> <li>• ComponentTab ("General")</li> <li>• Component("Component")</li> <li>• CustomTab ("Intrinsic Variables")</li> <li>• Quantities ("Quantities")</li> <li>• Signals ("Signals")</li> </ul>

	<code>&lt;propServer&gt;</code>	String	An object identifier, generally returned from another script method, such as ComplInst@R;2;3
	<code>&lt;propName&gt;</code>	String	Name of the property.
	<code>&lt;propValue&gt;</code>	String	The value for the property
<b>Return Value</b>	None.		

<b>Python Syntax</b>	<code>SetPropertyValue(&lt;propTab&gt;, &lt;propServer&gt;, &lt;propName&gt;, &lt;propValue&gt;)</code>
<b>Python Example</b>	<code>oEditor SetPropertyValue ("PassedParameterTab", "k", "R", "2200")</code>

<b>VB Syntax</b>	<code>SetPropertyValue &lt;propTab&gt;, &lt;propServer&gt;, &lt;propName&gt;, &lt;propValue&gt;</code>
<b>VB Example</b>	<code>oEditor SetPropertyValue "PassedParameterTab", "k", "R", "2200"</code>

## SetShowLayoutForLayoutComponent

Layout visualization, rather than bounding box, for a Layout Component in an HFSS 3D design.

<b>UI Access</b>	<b>Edit Layout...</b>		
	Name	Type	Description
<b>Parameters</b>	<code>&lt;Layout ComponentName&gt;</code>	String	Name of the Layout Component.
	Boolean	String	True or False
<b>Return Value</b>	None.		

<b>Python Syntax</b>	SetShowLayoutForLayoutComponent ("<LayoutComponentName>", <boolean> )
<b>Python Example</b>	oDesign.SetShowLayoutForLayoutComponent ("LC1_1", True)

## SetSolutionType

Sets the solution type for the design.

UI Access	HFSS > Solution Type.		
<b>Parameters</b>	Name	Type	Description
	<SolutionType>	String	Possible values are: "SBR+", "HFSS [Modal   Terminal] [ Network   Composite]", "Transient [Network   Composite]", "Eigenmode", or "Characteristic".
	EnableAutoOpen:=,	Boolean	Only . <ul style="list-style-type: none"> <li>• <b>True</b> - turn on auto open mode.</li> <li>• <b>False</b> - turn off auto open mode.</li> </ul>
	<ModelExteriorAsIE>	String	Only Applies for Driven Solution Type with Auto Open Mode as True. Possible values are: Radiation, FEBI, PML
<b>Return Value</b>	None.		

<b>Python Syntax</b>	SetSolutionType(<SolutionType>, <AutoOpenMode>, <ModelExteriorAsIE>)
<b>Python Example</b>	oDesign.SetSolutionType ("HFSS Modal Network", [

```
        "NAME:Options",
        "EnableAutoOpen:="      , False
    ] )

oDesign.SetSolutionType("Transient Network",
[
    "NAME:Options",
    "EnableAutoOpen:="      , False
])

oDesign.SetSolutionType("HFSS Hybrid Modal Network",
[
    "NAME:Options",
    "EnableAutoOpen:="      , False
])

oDesign.SetSolutionType("SBR+",
[
    "NAME:Options",
    "EnableAutoOpen:="      , False
])
```

	])
--	----

<b>VB Syntax</b>	SetSolutionType <SolutionType>, <AutoOpenMode>, <ModelExteriorAsIE>
<b>VB Example</b>	<pre> oDesign.SetSolutionType "Transient Network", Array("NAME:Options", "EnableAutoOpen:=", true, "BoundaryType:=", "Radiation") oDesign.SetSolutionType "SBR+", Array("NAME:Options", "EnableAutoOpen:=", false) oDesign.SetSolutionType "HFSS Terminal Composite", Array("NAME:Options", "EnableAutoOpen:=", false) </pre>

## SetSolveInsideThreshold

Sets the solve inside threshold to the specified double.

<b>UI Access</b>	N/A		
<b>Parameters</b>	Name <threshold>	Type Double	Description Siemens/m.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	SetSolveInsideThreshold(<threshold>)
<b>Python Example</b>	oDesign.SetSolveInsideThreshold(100000)

<b>VB Syntax</b>	SetSolveInsideThreshold <threshold>
------------------	-------------------------------------

**VB Example**

```
oDesign.SetSolveInsideThreshold 100000
```

## SetSourceContexts

For Near or Far Field projects for Driven Modal or Driven Terminal Network Analysis Solutions, specifies the port name and all modes/terminals of that port to be enabled as Source Context.

<b>UI Access</b>	<b>HFSS &gt; Fields &gt; Edit Sources.</b>						
<b>Parameters</b>	<table border="1"><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;SourceId&gt;</td><td>Array</td><td>Name of modes/terminals to be set as source context.</td></tr></table>	Name	Type	Description	<SourceId>	Array	Name of modes/terminals to be set as source context.
Name	Type	Description					
<SourceId>	Array	Name of modes/terminals to be set as source context.					
<b>Return Value</b>	None.						

**Python Syntax**

```
SetSourceContexts(<SourceId>)
```

**Python Example**

```
oModule.SetSourceContexts( [ "Box1_T1", "Box1_T2", "Box1_T3", "Current1", "IncPWave1" ] )
```

**VB Syntax**

```
SetSourceContexts <SourceId>
```

**VB Example**

```
oModule.SetSourceContexts _  
    Array("Box1_T1", "Box1_T2", "Box1_T3", "Current1", "IncPWave1")
```

## SetVariableValue

Sets the value of a variable. To set the value of a project variable, execute this command using `oProject`. To set the value of a local variable, use `oDesign`.

<b>UI Access</b>	N/A		
<b>Parameters</b>	Name	Type	Description
	<code>&lt;VarName&gt;</code>	String	Variable name.
<b>Return Value</b>	None.		

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

<b>VB Syntax</b>	<code>SetVariableValue &lt;VarName&gt;, &lt;VarValue&gt;</code>
<b>VB Example</b>	<code>oProject.SetVariableValue "\$Var1", "3mm"</code>

## Solve

Performs one or more simulation. The next script command will not be executed until the simulation(s) are complete.

<b>UI Access</b>	Select solution setup(s). Right-click and select <b>Analyze</b> .		
<b>Parameters</b>	Name	Type	Description
	<code>&lt;SimulationNames&gt;</code>	Array	Array containing string simulation names.

<b>Return Value</b>	<p>Integer:</p> <ul style="list-style-type: none"><li>• <b>0</b> – Simulation(s) completed.</li><li>• <b>1</b> – Simulation error.</li><li>• <b>-1</b> – Command execution error.</li></ul>
---------------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

<b>Python Syntax</b>	Solve < <i>SimulationNames</i> >
<b>Python Example</b>	<code>oDesign.Solve(['Setup1', 'Setup2', 'Setup3'])</code>

<b>VB Syntax</b>	Solve < <i>SimulationNames</i> >
<b>VB Example</b>	<code>oDesign.Solve Array("Setup1", "Setup2", "Setup3")</code>

## StartAnalysis (Layout Editor)

Simulates all setups

*Command:* None

*Syntax:* StartAnalysis

*Return Value:* None

*Parameters:* None

*VB Example:* `oDesign.StartAnalysis`

## StopSimLink

Stops linked simulation.

<b>UI Access</b>	N/A		
<b>Parameters</b>	Name	Type	Description
	<simId>	Integer	ID of specified simulation.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	StopSimLink(<simId>, <abort>)
<b>Python Example</b>	oDesign.StopSimLink(18, True)

<b>VB Syntax</b>	StopSimLink <simId>, <abort>
<b>VB Example</b>	oDesign.StopSimLink 18, true

## Undo [Design]

Cancels the last design-level command.

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

<b>Python Syntax</b>	Undo()
<b>Python Example</b>	<code>oDesign.Undo ()</code>

<b>VB Syntax</b>	Undo
<b>VB Example</b>	<code>oDesign.Undo</code>

## 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.

<b>Python Syntax</b>	ValidateLink()
<b>Python Example</b>	oDesign.ValidateLink()

## ValidateLink

**Note:**

This command is for internal Ansys use only.

<b>Python Syntax</b>	ValidateLink()
<b>Python Example</b>	oDesign.ValidateLink()

This page intentionally  
left blank.

# 10 - 3D Modeler Editor Script Commands

3D Modeler commands should be executed by the "3D Modeler" editor:

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

## Conventions Used in this Chapter:

### <AttributesArray>

<AttributesArray> takes the following structure:

```
Array("NAME:Attributes",
      "Name:=", <string>,
      "Flags:=", <string>,
      "Color:=", <string>,
      "Transparency:=", <value>,
      "PartCoordinateSystem:=", <string>,
      "UDMId:=", <string>,
      "MaterialValue:=", <string>,
      "SurfaceMaterialValue:=", <string>,
      "Solveinside:=", <boolean>,
      "ShellElement:=", <boolean>,
      "ShellElementThickness:=", <string>,
      "IsMaterialEditable:=", <boolean>,
```

```
"UseMaterialAppearance:=", <boolean>,  
"IsLightweight:=", <boolean>)
```

Where:

- **Flags** – Takes a string containing "NonModel" and/or "Wireframe", separated by the # character. For example, "NonModel#Wireframe".
- **Color** – Takes a string containing an RGB triple, formatted as "<RGB>". For example, "(255 255 255)".
- **Transparency** – Takes a value between 0 and 1.
- **PartCoordinateSystem** – Orientation of the primitive. The name of one of the defined coordinate systems should be specified.
- **UDMId** – Takes a string containing an ID.
- **MaterialValue** – Takes a string of the material name.
- **SurfaceMaterialValue** – Takes a string of the surface material name.
- **Solveinside** – Takes a boolean value.
- **ShellElement** – Takes a boolean value specifying whether or not a shell element is present.
- **ShellElementThickness** – Takes a string containing the shell element thickness. If element is not present, pass empty string.
- **IsMaterialEditable** – Takes a boolean value.
- **IsLightweight** – Takes a boolean value.

## <SelectionsArray>

<SelectionsArray> typically takes the following structure:

```
Array ("NAME:Selections",  
"Selections:=", <string>)
```

Where:

---

- **Selections** – Takes a comma-separated list of parts on which to perform the action. For example, "Rect1, Rect2, Rect3".

In some cases, <SelectionsArray> takes additional parameters:

```
Array ("NAME:Selections",
      "AllowRegionDependentPartSelectionForPMLCreation:=", <boolean>,
      "AllowRegionSelectionForPMLCreation:=", <boolean>,
      "Selections:=", <string>,
      "NewPartsModelFlag:=", <string>,
      "UseCurrentCS:=", <boolean>)
```

Where:

- **AllowRegionDependentPartSelectionForPMLCreation** – Takes a boolean value. See individual script for whether this parameter is required.
- **AllowRegionSelectionForPMLCreation** – Takes a boolean value. See individual script for whether this parameter is required.
- **Selections** – Takes a comma-separated list of parts on which to perform the action. For example, "Rect1, Rect2, Rect3".
- **NewPartsModelFlag** – Takes either string "Model" or string "Nonmodel". See individual script for whether this parameter is required.
- **UseCurrentCS** – Takes a boolean value. See individual script for whether this parameter is required. Use [GetActiveCoordinateSystem](#) to determine the current CS.

#### Note:

*Selections* is the only parameter required in *all* 3D Modeler Editor scripts. See individual scripts for additional required parameters.

## Organization

3D Modeler editor scripts are organized into the following categories:

[Draw Menu Commands](#)

[Edit Menu Commands](#)

[Modeler Menu Commands](#)

[Cable Modeling Commands](#)

[Other oEditor Commands](#)

## **Draw Menu Commands**

[Create3D Component](#)

[CreateBondwire](#)

[CreateBox](#)

[CreateCircle](#)

[CreateCone](#)

[CreateCutplane](#)

[CreateCylinder](#)

[CreateEllipse](#)

[CreateEquationCurve](#)

[CreateEquationSurface](#)

[CreateHelix](#)

[CreatePoint](#)

[CreatePolyline](#)

[CreateRectangle](#)  
[CreateRegion](#)  
[CreateRegularPolygon](#)  
[CreateRegularPolyhedron](#)  
[CreateSphere](#)  
[CreateSpiral](#)  
[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 <code>&lt;GeometryData&gt;</code>	Type Array	Description Structured array containing geometry data:  <code>Array ("NAME:GeometryData",         "ComponentName:=", "&lt;string&gt;",         "Owner:=", "&lt;string&gt;",         "Email:=", "&lt;string&gt;",         "Company:=", "&lt;string&gt;",         "Version:=", "&lt;string&gt;",         "Date:=", "&lt;string&gt;",         "Notes:=", "&lt;string&gt;",         "HasLabel:=", &lt;boolean&gt;,         "IncludedParts:=", &lt;array&gt;,         "IncludedCS:=", &lt;array&gt;,         "ReferenceCS:=", &lt;string&gt;,         "IncludedParameters:=", &lt;array&gt;);</code>

			"ParameterDescription:=", <array>)
	<DesignData>	Array	Structured array containing design data:  Array ("NAME:DesignData",  "Boundaries:=", <array>,  "Excitations:=", <array>,  "MeshOperations:=", <array>)
	<FileName>	String	Full file path to 3D component.
	<ImageFile>	Array	Optional. Structured array containing file path of 3D component image:  Array ("NAME:ImageFile",  "ImageFile:=", <string>)
<b>Return Value</b>	None.		

<b>Python Syntax</b>	Create3DComponent(<GeometryData>, <DesignData>, <FileName>, <ImageFile>)
<b>Python Example</b>	<pre>oEditor.Create3DComponent ([ "NAME:GeometryData",   "ComponentName:=", "Connector",   "Owner:=", "",   "Email:=", "",   "Company:=", "",   "Version:=", "1.0",   "Date:=", "11:41:01 AM Aug 28, 2014",</pre>

	<pre> "Notes:=", "",  "HasLabel:=", false,  "IncludedParts:=", ["Box1", "Cylinder1", "Cone1"],  "IncludedCS:=", ["RelativeCS1"],  "ReferenceCS:=", "Global",  "IncludedParameters:=", ["htcone", "lr", "htcyl", "zs", "radcyl", "xs", "\$rp", "\$con"],  "ParameterDescription:=", []],  ["NAME:DesignData",  "Boundaries:=", ["PerfE1", "FiniteCond1"],  "Excitations:=", ["1"],  "MeshOperations:=", []],  "C:/tmp/Connector.a3dcomp",  ["NAME:ImageFile", "ImageFile:=", ""]] </pre>
--	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

<b>VB Syntax</b>	Create3DComponent <GeometryData>, <DesignData>, <FileName>, <ImageFile>
<b>VB Example</b>	<pre> oEditor.Create3DComponent Array ("NAME:GeometryData", "ComponentName:=", "Connector", </pre>

```
"Owner:=", "",  
"Email:=", "",  
"Company:=", "",  
"Version:=", "1.0",  
"Date:=", "11:41:01 AM Aug 28, 2014",  
"Notes:=", "",  
"HasLabel:=", false,  
"IncludedParts:=", Array( "Box1", "Cylinder1", "Cone1"),  
"IncludedCS:=", Array("RelativeCS1"),  
"ReferenceCS:=", "Global",  
"IncludedParameters:=", Array("htcone", "lr", "htcyl", "zs", "radcyl", "xs", "$rp",  
"$con"),  
"ParameterDescription:=", Array(),  
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	<b>Draw &gt; Bondwire.</b>		
	Name	Type	Description
<b>Parameters</b>	<Parameters>	Array	<p>Structured array.</p> <pre> Array ("NAME:BondwireParameters",       "WireType:=", &lt;string ("JEDEC_4Points", "JEDEC_5Points", or "LOW")&gt;,       "WireDiameter:=", &lt;string&gt;,       "NumSides:=", &lt;value&gt;,       "XPadPos:=", &lt;value&gt;,       "YPadPos:=", &lt;value&gt;,       "ZPadPos:=", &lt;value&gt;,       "XDir:=", &lt;value&gt;,       "YDir:=", &lt;value&gt;,       "ZDir:=", &lt;value&gt;,       "Distance:=", &lt;value&gt;,       "h1:=", &lt;value&gt;,       "h2:=", &lt;value&gt;,       "alpha:=", &lt;value&gt;,       "beta:=", &lt;value&gt;,       "WhichAxis:=", &lt;string ("X", "Y", or "Z")&gt;,       "ReverseDirection:=", &lt;boolean&gt;)     </pre>

	<code>&lt;AttributesArray&gt;</code>	Array	Structured array. See: <a href="#">AttributesArray</a> .
<b>Return Value</b>	None.		

<b>Python Syntax</b>	<code>CreateBondwire(&lt;Parameters&gt;, &lt;Attributes&gt;)</code>
<b>Python Example</b>	<pre> oEditor.CreateBondwire(     [         "NAME:BondwireParameters",         "WireType:=" , "JEDEC_4Points",         "WireDiameter:=" , "0.025mm",         "NumSides:=" , "6",         "XPadPos:=" , "1.6mm",         "YPadPos:=" , "-0.2mm",         "ZPadPos:=" , "0mm",         "XDir:=" , "-2.2mm",         "YDir:=" , "-1.4mm",         "ZDir:=" , "0mm",         "Distance:=" , "2.60768096208106mm",         "h1:=" , "0.2mm",         "h2:=" , "0mm",         "alpha:=" , "80deg",         "beta:=" , "0",         "WhichAxis:=" , "Z",     ] ) </pre>

```
"ReverseDirection:="      , True
],
[ "NAME:Attributes",
  "Name:="                  , "Bondwire1",
  "Flags:="                 , "",
  "Color:="                 , "(143 175 143)",
  "Transparency:="          , 0,
  "PartCoordinateSystem:=", "Global",
  "UDMId:="                 , "",
  "MaterialValue:="          , "\"vacuum\"",
  "SurfaceMaterialValue:=", "\\"",
  "SolveInside:="            , True,
  "ShellElement:="           , False,
  "ShellElementThickness:=", "0mm",
  "IsMaterialEditable:="    , True,
  "UseMaterialAppearance:=", False,
  "IsLightweight:="          , False
])
```

---

**VB****CreateBondwire <Parameters>, <Attributes>**

Syntax	
<b>VB Example</b>	<pre> oEditor.CreateBondwire Array("NAME:BondwireParameters", "WireType:=", "LOW", "WireDiameter:=",                            _"0.025mm", "NumSides:=", "6", "XPadPos:=", "6mm", "YPadPos:=", "-2.5mm", "ZPadPos:=",                            _"0mm", "XDir:=", "-10mm", "YDir:=", "4mm", "ZDir:=", "0mm", "Distance:=",                            _"10.770329614269mm", "h1:=", "0.2mm", "h2:=", "0mm", "alpha:=", "80deg", "beta:=",                            _"80deg", "WhichAxis:=", "Z", "ReverseDirection:=", false), Array("NAME:Attributes",                            "Name:=",                            _"Bondwire2", "Flags:="", "", "Color:=", "(143 175 143)", "Transparency:=", 0,                            "PartCoordinateSystem:=",                            _"Global", "UDMID:=", "", "MaterialValue:=", "" &amp; Chr(34) &amp; "copper" &amp; Chr(34) &amp; "", "Sur-                            faceMaterialValue:=",                            _"" &amp; Chr(34) &amp; "" &amp; Chr(34) &amp; "", "SolveInside:=", false, "ShellElement:=",                            _false, "ShellElementThickness:=", "0mm", "IsMaterialEditable:=", true, "UseMa-                            terialAppearance:=",                            _false, "IsLightweight:=", false) </pre>

## CreateBox

Creates a box.

UI Access	Draw > Box.								
Parameters	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;Parameters&gt;</td> <td>Array</td> <td> <p>Structured array.</p> <p>Array ("NAME:BoxParameters",  "XPosition:=", &lt;string&gt;,</p> </td></tr> </tbody> </table>	Name	Type	Description	<Parameters>	Array	<p>Structured array.</p> <p>Array ("NAME:BoxParameters",  "XPosition:=", &lt;string&gt;,</p>		
Name	Type	Description							
<Parameters>	Array	<p>Structured array.</p> <p>Array ("NAME:BoxParameters",  "XPosition:=", &lt;string&gt;,</p>							

			<pre>"YPosition:=", &lt;string&gt;, "ZPosition:=", &lt;string&gt;, "XSize:=", &lt;string&gt;, "YSize:=", &lt;string&gt;, "ZSize:=", &lt;string&gt;)</pre>
	<code>&lt;AttributesArray&gt;</code>	Array	Structured array. See: <a href="#">AttributesArray</a> .
<b>Return Value</b>	None.		

<b>Python Syntax</b>	<code>CreateBox(&lt;Parameters&gt;, &lt;Attributes&gt;)</code>
<b>Python Example</b>	<pre>oEditor.CreateBox(     [         ["NAME:BoxParameters",             "XPosition:=" , "0.5mm",             "YPosition:=" , "-6.5mm",             "ZPosition:=" , "0mm",             "XSize:=" , "2mm",             "YSize:=" , "1.5mm",             "ZSize:=" , "1.5mm"         ],         ["NAME:Attributes",             ...         ]     ] )</pre>

```

    "Name:="                  , "Box3",
    "Flags:="                 , "",
    "Color:="                 , "(143 175 143)",
    "Transparency:="          , 0,
    "PartCoordinateSystem:="   , "Global",
    "UDMID:="                 , "",
    "MaterialValue:="          , "\copper\",
    "SurfaceMaterialValue:="   , "\",
    "SolveInside:="            , False,
    "ShellElement:="           , False,
    "ShellElementThickness:="  , "0mm",
    "IsMaterialEditable:="     , True,
    "UseMaterialAppearance:="  , False,
    "IsLightweight:="          , False
  ]
)

```

<b>VB Syntax</b>	CreateBox <Parameters>, <Attributes>
<b>VB Example</b>	<pre> oEditor.CreateBox Array("NAME:BoxParameters", "XPosition:=", "0mm", "YPosition:=", _ "4mm", "ZPosition:=", "0mm", "XSize:=", "-4mm", "YSize:=", "1mm", "ZSize:=", _ "2mm"), Array("NAME:Attributes", "Name:=", "Box4", "Flags:=", "", "Color:=", _ </pre>

```

    "(143 175 143)", "Transparency:=", 0, "PartCoordinateSystem:=", "Global", "UDMID:=",
    -
    "", "MaterialValue:=", "" & Chr(34) & "copper" & Chr(34) & "", "Sur-
    faceMaterialValue:=", _
    "" & Chr(34) & "" & Chr(34) & "", "SolveInside:=", false, "ShellElement:=", _
    false, "ShellElementThickness:=", "0mm", "IsMaterialEditable:=", true, _
    "UseMaterialAppearance:=", false, "IsLightweight:=", false)

```

## CreateCircle

Creates a circle.

UI Access	Draw > Circle.		
Parameters	Name <i>&lt;Parameters&gt;</i>	Type Array	Description Structured array. Array ("NAME:CircleParameters",           "IsCovered:=", <boolean>,           "XCenter:=", <value>,           "YCenter:=", <value>,           "ZCenter:=", <value>,           "Radius:=", <value>,           "WhichAxis:=", <string>           "NumSegments:=", <string containing integer>)
	<i>&lt;AttributesArray&gt;</i>	Array	Structured array. See: <a href="#">AttributesArray</a> .

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

<b>Python Syntax</b>	CreateCircle(<Parameters>, <Attributes>)
<b>Python Example</b>	<pre> oEditor.CreateCircle(     [         ["NAME:CircleParameters",             "IsCovered:=" , True,             "XCenter:=" , "5.5mm",             "YCenter:=" , "-3mm",             "ZCenter:=" , "0mm",             "Radius:=" , "0.707106781186548mm",             "WhichAxis:=" , "Z",             "NumSegments:=" , "0"         ],         ["NAME:Attributes",             "Name:=" , "Circle1",             "Flags:=" , "",             "Color:=" , "(143 175 143)",             "Transparency:=" , 0,             "PartCoordinateSystem:=" , "Global",             "UDMID:=" , "",             "MaterialValue:=" , "\"copper\""         ]     ] ) </pre>

```
"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:=", "" &amp; Chr(34) &amp; "copper" &amp; Chr(34) &amp; "", "Sur- faceMaterialValue:=", _ "" &amp; Chr(34) &amp; "" &amp; Chr(34) &amp; "", "SolveInside:=", false, "ShellElement:=", _ false, "ShellElementThickness:=", "0mm", "IsMaterialEditable:=", true, "UseMa- terialAppearance:=", _ false, "IsLightweight:=", false)</pre>

## CreateCone

Creates a cone.

UI Access	Draw > Cone.		
<b>Parameters</b>	Name <i>&lt;Parameters&gt;</i>	Type Array	Description Structured array.  Array ("NAME:ConeParameters", "XCenter:=", <string>, "YCenter:=", <string>, "ZCenter:=", <string>, "WhichAxis:=", <string>, "Height:=", <string>, "BottomRadius:=", <string>, "TopRadius:=", <string>)
	<i>&lt;AttributesArray&gt;</i>	Array	Structured array. See: <a href="#">AttributesArray</a> .
<b>Return Value</b>	None.		

<b>Python Syntax</b>	CreateCone(<Parameters>, <Attributes>)
<b>Python Example</b>	<pre>oEditor.CreateCone(     [ "NAME:ConeParameters",         "XCenter:=" , "3mm",         "YCenter:=" , "-4.5mm",         "ZCenter:=" , "0mm",         "WhichAxis:=" , "X",         "Height:=" , "10mm",         "BottomRadius:=" , "1mm",         "TopRadius:=" , "0.5mm" ] )</pre>

```
"ZCenter:=" , "0mm",
"WhichAxis:=" , "Z",
"Height:=" , "2.5mm",
"BottomRadius:=" , "2.82842712474619mm",
"TopRadius:=" , "2.23606797749979mm"
],
["NAME:Attributes",
 "Name:=" , "Cone1",
 "Flags:=" , "",
 "Color:=" , "(143 175 143)",
 "Transparency:=" , 0,
 "PartCoordinateSystem:=", "Global",
 "UDMId:=" , "",
 "MaterialValue:=" , "\"copper\"",
 "SurfaceMaterialValue:=" , "\"\"",
 "SolveInside:=" , False,
 "ShellElement:=" , False,
 "ShellElementThickness:=" , "0mm",
 "IsMaterialEditable:=" , True,
 "UseMaterialAppearance:=" , False,
```

	"IsLightweight:=" , False ])
--	---------------------------------

VB Syntax	CreateCone <Parameters>, <Attributes>
VB Example	<pre> oEditor.CreateCone Array("NAME:ConeParameters", "XCenter:=", "3mm", "YCenter:=", _  "ZCenter:=", "0mm", "WhichAxis:=", "Z", "Height:=", "4mm", "BottomRadius:=", _  "1.11803398874989mm", "TopRadius:=", "2.06155281280883mm"), Array("NAME:Attributes",  "Name:=", _  "Cone1", "Flags:=", "", "Color:=", "(143 175 143)", "Transparency:=", 0, "PartCoordinateSystem:=", _  "Global", "UDMID:=", "", "MaterialValue:=", "" &amp; Chr(34) &amp; "copper" &amp; Chr(34) &amp; "",  "SurfaceMaterialValue:=", _  "" &amp; Chr(34) &amp; "" &amp; Chr(34) &amp; "", "SolveInside:=", false, "ShellElement:=", _  false, "ShellElementThickness:=", "0mm", "IsMaterialEditable:=", true, "UseMaterialAppearance:=", _  false, "IsLightweight:=", false) </pre>

## CreateCutplane

Creates a cutplane.

UI Access	Draw > Plane.		
Parameters	Name <Parameters>	Type Array	Description Structured array. Array ("NAME:PlaneParameters",

			<pre>"PlaneBaseX:=", &lt;string&gt;, "PlaneBaseY:=", &lt;string&gt;, "PlaneBaseZ:=", &lt;string&gt;, "PlaneNormalX:=", &lt;string&gt;, "PlaneNormalY:=", &lt;string&gt;), "PlaneNormalZ:=", &lt;string&gt;)</pre>
	<AttributesArray>	Array	See: <a href="#">AttributesArray</a> . CreateCutplane only takes the Name and Color attributes.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	CreateCutplane(<Parameters>, <Attributes>)
<b>Python Example</b>	<pre>oEditor.CreateCutplane( ["NAME:PlaneParameters",  "PlaneBaseX:="          , "-0.6mm",  "PlaneBaseY:="          , "-0.8mm",  "PlaneBaseZ:="          , "0mm",  "PlaneNormalX:="        , "1.2mm",  "PlaneNormalY:="        , "0.2mm",  "PlaneNormalZ:="        , "0mm" ],</pre>

```
[ "NAME:Attributes",
  "Name:=" , "Plane1",
  "Color:=" , "(143 175 143)"
])
```

<b>VB Syntax</b>	CreateCutplane <Parameters>, <Attributes>
<b>VB Example</b>	<pre>oEditor.CreateCutplane Array("NAME:PlaneParameters", "PlaneBaseX:=", "0.6mm", "PlaneBaseY:=", _ "1.2mm", "PlaneBaseZ:=", "0mm", "PlaneNormalX:=", "0.4mm", "PlaneNormalY:=", _ "0.6mm", "PlaneNormalZ:=", "0mm"), Array("NAME:Attributes", "Name:=", "Plane2", "Col- or:=", _ "(143 175 143)")</pre>

## CreateCylinder

Creates a cylinder.

UI Access	Draw > Cylinder.								
Parameters	<table border="1"> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> <tr> <td>&lt;Parameters&gt;</td> <td>Array</td> <td> <p>Structured array.</p> <p>Array ("NAME:CylinderParameters",  "XCenter:=", &lt;string&gt;,  "YCenter:=", &lt;string&gt;,  "ZCenter:=", &lt;string&gt;,</p> </td></tr> </table>	Name	Type	Description	<Parameters>	Array	<p>Structured array.</p> <p>Array ("NAME:CylinderParameters",  "XCenter:=", &lt;string&gt;,  "YCenter:=", &lt;string&gt;,  "ZCenter:=", &lt;string&gt;,</p>		
Name	Type	Description							
<Parameters>	Array	<p>Structured array.</p> <p>Array ("NAME:CylinderParameters",  "XCenter:=", &lt;string&gt;,  "YCenter:=", &lt;string&gt;,  "ZCenter:=", &lt;string&gt;,</p>							

			"Radius:=", <string>, "Height:=", <string>, "WhichAxis:=", <string>, "NumSides:=", <string containing integer>)
	<AttributesArray>	Array	Structured array. See: <a href="#">AttributesArray</a> .
<b>Return Value</b>	None.		

<b>Python Syntax</b>	CreateCylinder(<Parameters>, <Attributes>)
<b>Python Example</b>	<pre> oEditor.CreateCylinder(     [         ["NAME:CylinderParameters",             "XCenter:="           , "6mm",             "YCenter:="           , "-4.5mm",             "ZCenter:="           , "0mm",             "Radius:="            , "0.5mm",             "Height:="             , "4.5mm",             "WhichAxis:="          , "Z",             "NumSides:="           , "0"         ],         ["NAME:Attributes",             "Name:="              , "Cylinder1",         ]     ] ) </pre>

```

    "Flags:="           , """",
    "Color:="          , "(143 175 143)",
    "Transparency:="   , 0,
    "PartCoordinateSystem:=", "Global",
    "UDMID:="          , """",
    "MaterialValue:="   , "\"copper\"",
    "SurfaceMaterialValue:=", "\"\"",
    "SolveInside:="     , False,
    "ShellElement:="    , False,
    "ShellElementThickness:=", "0mm",
    "IsMaterialEditable:=" , True,
    "UseMaterialAppearance:=", False,
    "IsLightweight:="    , False
)

```

<b>VB Syntax</b>	CreateCylinder <Parameters>, <Attributes>
<b>VB Example</b>	<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:=", _ </pre>

```

"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)

```

## CreateEllipse

Creates an ellipse.

UI Access	Draw > Ellipse.		
<b>Parameters</b>	Name <i>&lt;Parameters&gt;</i>	Type Array	Description Structured array.  Array ("NAME:EllipseParameters", "IsCovered:=", <string>, "XCenter:=", <string>, "YCenter:=", <string>, "ZCenter:=", <string>, "MajRadius:=", <string>, "Ratio:=", <string>, "WhichAxis:=", <string>, "NumSegments:=", <string>)
	<i>&lt;AttributesArray&gt;</i>	Array	Structured array. See: <a href="#">AttributesArray</a> .

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

<b>Python Syntax</b>	CreateEllipse(<Parameters>, <Attributes>)
<b>Python Example</b>	<pre> oEditor.CreateEllipse(     [         ["NAME:EllipseParameters",             "IsCovered:=" , True,             "XCenter:=" , "0.6mm",             "YCenter:=" , "-0.6mm",             "ZCenter:=" , "0mm",             "MajRadius:=" , "0.2mm",             "Ratio:=" , "7",             "WhichAxis:=" , "Z",             "NumSegments:=" , "0"         ],         ["NAME:Attributes",             "Name:=" , "Ellipse1",             "Flags:=" , "",             "Color:=" , "(143 175 143)",             "Transparency:=" , 0,             "PartCoordinateSystem:=" , "Global",             "UDMId:=" , ""         ]     ] ) </pre>

```

    "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, "PartCoordinateSystem:=", - "Global", "UDMID:=", "", "MaterialValue:=", "" &amp; Chr(34) &amp; "copper" &amp; Chr(34) &amp; "", "Sur- faceMaterialValue:=", - "" &amp; Chr(34) &amp; "" &amp; Chr(34) &amp; "", "SolveInside:=", false, "ShellElement:=", _</pre>

```
false, "ShellElementThickness:=", "0mm", "IsMaterialEditable:=", true, "UseMaterialAppearance:=",
false, "IsLightweight:=", false)
```

## CreateEquationCurve

Creates an equation-based curve.

UI Access	Draw > Equation-Based Curve.		
<b>Parameters</b>	Name <i>&lt;Parameters&gt;</i>	Type Array	Description Structured array.  Array ("NAME:EquationBasedCurveParameters", "XtFunction:=", <string>, "YtFunction:=", <string>, "ZtFunction:=", <string>, "tStart:=", <string>, "tEnd:=", <string>, "NumOfPointsOnCurve:=", <string>, "Version:=", <integer>), <polylineArray(optional)>)
	<i>&lt;AttributesArray&gt;</i>	Array	Structured array. See: <a href="#">AttributesArray</a> .
<b>Return Value</b>	None.		

<b>Python Syntax</b>	CreateEquationCurve(<Parameters>, <Attributes>)
----------------------	-------------------------------------------------

**Python Example**

```
oEditor.CreateEquationCurve(  
    [ "NAME:EquationBasedCurveParameters",  
        "XtFunction:=" , "1",  
        "YtFunction:=" , "3",  
        "ZtFunction:=" , "32",  
        "tStart:=" , "1",  
        "tEnd:=" , "3",  
        "NumOfPointsOnCurve:=" , "0",  
        "Version:=" , 1,  
    [ "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,
    "ShellElement:="          , False,
    "ShellElementThickness:=", "0mm",
    "IsMaterialEditable:="   , True,
    "UseMaterialAppearance:=", False,
    "IsLightweight:="         , False
  ]
)

```

<b>VB Syntax</b>	CreateEquationCurve <Parameters>, <Attributes>
<b>VB Example</b>	<pre> oEditor.CreateEquationCurve Array("NAME:EquationBasedCurveParameters", "XtFunction:=", _ "1", "YtFunction:=", "3", "ZtFunction:=", "32", "tStart:=", "1", "tEnd:=", "3", "NumOfPointsOnCurve:=", _ </pre>

```

"0", "Version:=", 1, Array("NAME:PolylineXSection", "XSectionType:=", "None", "XSectionOrient:=", _
"Auto", "XSectionWidth:=", "0", "XSectionTopWidth:=", "0", "XSectionHeight:=", _
"0", "XSectionNumSegments:=", "0", "XSectionBendType:=", "Corner")), Array("NAME:Attributes", "Name:=", _
"EquationCurve2", "Flags:=", "", "Color:=", "(143 175 143)", "Transparency:=", _
0, "PartCoordinateSystem:=", "Global", "UDMID:=", "", "MaterialValue:=", _
"" & Chr(34) & "copper" & Chr(34) & "", "SurfaceMaterialValue:=", "" & Chr(34) & "" &
Chr(34) & "", "SolveInside:=", _
false, "ShellElement:=", false, "ShellElementThickness:=", "0mm", "IsMaterialEditable:=", _
true, "UseMaterialAppearance:=", false, "IsLightweight:=", false)

```

## CreateEquationSurface

Creates an equation-based surface.

UI Access	Draw > Equation-Based Surface.		
	Name	Type	Description
Parameters	<Parameters>	Array	<p>Structured array.</p> <p>Array ("NAME:EquationBasedSurfaceParameters",  "XuvFunction:=" , &lt;string equation containing  Function, Operators and/or quantities _u, _v, or PI&gt;,  "YuvFunction:=" , &lt;string equation containing  Function, Operators and/or quantities _u, _v, or PI&gt;,</p>

			<pre>"ZuvFunction:=" , &lt;string equation containing Function, Operators and/or quantities _u, _v, or PI&gt;, "uStart:=" , &lt;string&gt;, "uEnd:=" , &lt;string&gt;, "vStart:=" , &lt;string&gt;, "vEnd:=" , &lt;string&gt;, "Version:=" , &lt;integer&gt;)</pre>
	<code>&lt;AttributesArray&gt;</code>	Array	Structured array. See: <a href="#">AttributesArray</a> .
<b>Return Value</b>	None.		

<b>Python Syntax</b>	CreateEquationSurface(<Parameters>, <Attributes>)
<b>Python Example</b>	<pre>oEditor.CreateEquationSurface (     [ "NAME:EquationBasedSurfaceParameters",         "XuvFunction:=" , "_u",         "YuvFunction:=" , "_v",         "ZuvFunction:=" , "sin(_u)+cos(_v)",         "uStart:=" , "1",         "uEnd:=" , "10",         "vStart:=" , "1",         "vEnd:=" , "10",         "Version:=" , 1     ], </pre>

```
[ "NAME:Attributes",
  "Name:=" , "EquationSurface1",
  "Flags:=" , "",
  "Color:=" , "(143 175 143)",
  "Transparency:=" , 0,
  "PartCoordinateSystem:=", "Global",
  "UDMId:=" , "",
  "MaterialValue:=" , "\"copper\"",
  "SurfaceMaterialValue:=" , "\"\"",
  "SolveInside:=" , False,
  "ShellElement:=" , False,
  "ShellElementThickness:=" , "0mm",
  "IsMaterialEditable:=" , True,
  "UseMaterialAppearance:=" , False,
  "IsLightweight:=" , False
])
```

<b>VB Syntax</b>	CreateEquationSurface <Parameters>, <Attributes>
<b>VB Example</b>	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", "IsMa-
    terialEditable:=", _
    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: <a href="#">SelectionsArray</a> .
	<Parameters>	Array	<p>Structured array.</p> <p>Array ("NAME:HelixParameters",</p> <p style="padding-left: 20px;">"XCenter:=" , &lt;string&gt;,</p> <p style="padding-left: 20px;">"YCenter:=" , &lt;string&gt;,</p> <p style="padding-left: 20px;">"ZCenter:=" , &lt;string&gt;,</p> <p style="padding-left: 20px;">"XStartDir:=" , &lt;string&gt;,</p> <p style="padding-left: 20px;">"YStartDir:=" , &lt;string&gt;,</p>

			<pre>"ZStartDir:=" , &lt;string&gt;, "NumThread:=" , &lt;string&gt;, "RightHand:=" , &lt;boolean&gt;, "RadiusIncrement:=" , &lt;string&gt;, "Thread:=" , &lt;string&gt;)</pre>
<b>Return Value</b>	None.		

<b>Python Syntax</b>	CreateHelix(<SelectionsArray>, <Parameters>)
<b>Python Example</b>	<pre>oEditor.CreateHelix(     ["NAME:Selections",         "Selections:=" , "EquationSurface2",         "NewPartsModelFlag:=" , "Model"     ],     ["NAME:HelixParameters",         "XCenter:=" , "10000mm",         "YCenter:=" , "40000mm",         "ZCenter:=" , "0mm",         "XStartDir:=" , "0mm",         "YStartDir:=" , "10000mm",         "ZStartDir:=" , "0mm",</pre>

```

    "NumThread:="           , "1",
    "RightHand:="          , True,
    "RadiusIncrement:="    , "0mm",
    "Thread:="              , "1mm"
  ] )

```

<b>VB Syntax</b>	CreateHelix <SelectionsArray>, <Parameters>
<b>VB Example</b>	<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	<p>Structured array.</p> <p>Array ("NAME:PointParameters",</p>

			"PointX:=", <value>, "PointY:=", <value>, "PointZ:=", <value>)  <AttributesArray>
	<AttributesArray>	Array	Structured array. See: <a href="#">AttributesArray</a> . CreatePoint takes only the Name and Color attributes.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	CreatePoint(<Parameters>, <Attributes>)
<b>Python Example</b>	<pre>oEditor.CreatePoint(      ["NAME:PointParameters",         "PointX:=" , "0.2mm",          "PointY:=" , "-0.2mm",          "PointZ:=" , "0mm"     ],      ["NAME:Attributes",          "Name:=" , "Point1",          "Color:=" , "(143 175 143)"</pre>

	])
--	----

<b>VB Syntax</b>	CreatePoint <Parameters>, <Attributes>
<b>VB Example</b>	<pre> oEditor.CreatePoint  Array ("NAME:PointParameters",       "PointX:=" , "0.2mm",       "PointY:=" , "-0.2mm",       "PointZ:=" , "0mm") ,       Array ("NAME:Attributes",       "Name:=" , "Point1",       "Color:=" , "(143 175 143)")</pre>

## CreatePolyline

Creates a polyline.

<b>UI Access</b>	Draw > Line.								
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;Parameters&gt;</td> <td>Array</td> <td> <p>Structured array.</p> <pre> Array ("NAME:PolylineParameters",       "IsPolylineCovered:=" , &lt;bool&gt;,       "IsPolylineClosed:=" , &lt;bool&gt;,       &lt;PolylinePointsArray&gt;,       &lt;PolylineSegmentsArray&gt;)</pre> </td> </tr> </tbody> </table>	Name	Type	Description	<Parameters>	Array	<p>Structured array.</p> <pre> Array ("NAME:PolylineParameters",       "IsPolylineCovered:=" , &lt;bool&gt;,       "IsPolylineClosed:=" , &lt;bool&gt;,       &lt;PolylinePointsArray&gt;,       &lt;PolylineSegmentsArray&gt;)</pre>		
Name	Type	Description							
<Parameters>	Array	<p>Structured array.</p> <pre> Array ("NAME:PolylineParameters",       "IsPolylineCovered:=" , &lt;bool&gt;,       "IsPolylineClosed:=" , &lt;bool&gt;,       &lt;PolylinePointsArray&gt;,       &lt;PolylineSegmentsArray&gt;)</pre>							

	<code>&lt;PolylinePointsArray&gt;</code>	Array	Array ("NAME:PolylinePoints", <OnePointArray>, <OnePointArray>, ...)
	<code>&lt;OnePointArray&gt;</code>	Array	Array ("NAME:PLPoint", "X:=", <value>, "Y:=", <value>, "Z:=", <value>))
	<code>&lt;PolylineSegmentsArray&gt;</code>	Array	<PolylineSegmentsArray> Array ("NAME:PolylineSegments", <OneSegmentArray>, <OneSegmentArray>, ...)
	<code>&lt;OneSegmentArray&gt;</code>	Array	Array ("NAME:PLSegment", "SegmentType:=", <"Line", "Arc", "Spline", or "AngularArc">, "StartIndex:=", <value>, "NoOfPoints:=", <value>)
	<code>&lt;AttributesArray&gt;</code>	Array	Structured array. See: <a href="#">AttributesArray</a> .
<b>Return Value</b>	None.		

<b>Python Syntax</b>	CreatePolyline(<Parameters>, <Attributes>)
<b>Python Example</b>	<pre>oEditor.CreatePolyline(     ["NAME:PolylineParameters",         "IsPolylineCovered:="      , True,</pre>

```
"IsPolylineClosed:="      , False,  
[ "NAME:PolylinePoints",  
  [ "NAME:PLPoint",  
    "X:="           , "20000mm",  
    "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"
],
[ "NAME:Attributes",
"Name:=" , "Polyline1",
```

```

    "Flags:="           , """",
    "Color:="          , "(143 175 143)",
    "Transparency:="   , 0,
    "PartCoordinateSystem:=", "Global",
    "UDMID:="          , """",
    "MaterialValue:="   , "\"copper\"",
    "SurfaceMaterialValue:=", "\"\"",
    "SolveInside:="     , False,
    "ShellElement:="    , False,
    "ShellElementThickness:=", "0mm",
    "IsMaterialEditable:=" , True,
    "UseMaterialAppearance:=", False,
    "IsLightweight:="    , False
  )
)

```

<b>VB Syntax</b>	CreatePolyline <Parameters>, <Attributes>
<b>VB Example</b>	<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:=", _ </pre>

```

    "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) & "",
    "SurfaceMaterialValue:=", _
    "" & Chr(34) & "" & Chr(34) & "", "SolveInside:=", false, "ShellElement:=", _
    false, "ShellElementThickness:=", "0mm", "IsMaterialEditable:=", true, "UseMa-
    terialAppearance:=", _
    false, "IsLightweight:=", false)

```

## CreateRectangle

Creates a rectangle.

UI Access	Draw > Rectangle.		
Parameters	Name	Type	Description
	<Parameters>	Array	<p>Structured array.</p> <p>Array ("NAME:RectangleParameters",      "IsCovered:=" , &lt;boolean&gt;,</p>

			<pre>"XStart:=" , &lt;string&gt;, "YStart:=" , &lt;string&gt;, "ZStart:=" , &lt;string&gt;, "Width:=" , &lt;string&gt;, "Height:=" , &lt;string&gt;, "WhichAxis:=" , &lt;string "X", "Y", or "Z"&gt;)</pre>
	<b>&lt;AttributesArray&gt;</b>	Array	Structured array. See: <a href="#">AttributesArray</a> .
<b>Return Value</b>	None.		

<b>Python Syntax</b>	CreateRectangle(<Parameters>, <Attributes>)
<b>Python Example</b>	<pre>oEditor.CreateRectangle( [ "NAME:RectangleParameters",   "IsCovered:=" , True,   "XStart:=" , "-80000mm",   "YStart:=" , "-90000mm",   "ZStart:=" , "0mm",   "Width:=" , "20000mm",   "Height:=" , "30000mm",   "WhichAxis:=" , "Z" ], [ "NAME:Attributes", ]</pre>

```

    "Name:="                  , "Rectangle1",
    "Flags:="                 , "", ,
    "Color:="                 , "(143 175 143)",
    "Transparency:="          , 0,
    "PartCoordinateSystem:=", "Global",
    "UDMID:="                 , "", ,
    "MaterialValue:="          , "\"copper\"",
    "SurfaceMaterialValue:=", "\\"",
    "SolveInside:="            , False,
    "ShellElement:="           , False,
    "ShellElementThickness:=", "0mm",
    "IsMaterialEditable:="     , True,
    "UseMaterialAppearance:=", False,
    "IsLightweight:="          , False
  ]
)

```

<b>VB Syntax</b>	CreateRectangle <Parameters>, <Attributes>
<b>VB Example</b>	<pre> oEditor.CreateRectangle Array("NAME:RectangleParameters", "IsCovered:=", true, "XStart:=", _ "10000mm", "YStart:=", "-40000mm", "ZStart:=", "0mm", "Width:=", "40000mm", "Height:=", </pre>

```

    -
    "10000mm", "WhichAxis:=", "Z"), Array("NAME:Attributes", "Name:=", "Rectangle2", "Flag-
s:=", _
    "", "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)

```

## CreateRegion

Creates a region containing the design.

UI Access	Draw > Region.		
Parameters	Name	Type	Description
	<Parameters>	Array	<p>Structured array.</p> <pre>         Array("NAME:RegionParameters",             "+XPaddingType:=", &lt;string "Percentage Offset",             "Absolute Offset", or "Absolute Position",             "+XPadding:=", &lt;string X value&gt;,             "-XPaddingType:=", &lt;string "Percentage Offset",             "Absolute Offset", or "Absolute Position",             "-XPadding:=", &lt;string -X value&gt;,             "+YPaddingType:=", &lt;string "Percentage Offset", </pre>

		<pre>"Absolute Offset", or "Absolute Position", "+YPadding:=", &lt;string Y value&gt;, "-YPaddingType:=", &lt;string "Percentage Offset", "Absolute Offset", or "Absolute Position", "-YPadding:=", &lt;string -Y value&gt;, "+ZPaddingType:=", &lt;string "Percentage Offset", "Absolute Offset", or "Absolute Position", "+ZPadding:=", &lt;string Z value&gt;, "-ZPaddingType:=", &lt;string "Percentage Offset", "Absolute Offset", or "Absolute Position", "-ZPadding:=", &lt;string -Z value&gt;)</pre>
	<p><b>&lt;AttributesArray&gt;</b></p>	<p>Array</p> <p>Structured array. See: <a href="#">AttributesArray</a>.</p>
<b>Return Value</b>	None.	

<b>Python Syntax</b>	CreateRegion(<Parameters>, <Attributes>)
<b>Python Example</b>	<pre>oEditor.CreateRegion ( [ "NAME:RegionParameters",     "+XPaddingType:=" , "Percentage Offset",     "+XPadding:=" , "0",     "-XPaddingType:=" , "Percentage Offset",     "-XPadding:=" , "0",</pre>

```
"+YPaddingType:=", "Percentage Offset",
"+YPadding:=", "0",
"-YPaddingType:=", "Percentage Offset",
"-YPadding:=", "0",
"+ZPaddingType:=", "Percentage Offset",
"+ZPadding:=", "0",
"-ZPaddingType:=", "Percentage Offset",
"-ZPadding:=", "0"
],
["NAME:Attributes",
"Name:=", "Region",
"Flags:=", "Wireframe#",
"Color:=", "(143 175 143)",
"Transparency:=", 0,
"PartCoordinateSystem:=", "Global",
"UDMID:=", "",
"MaterialValue:=", "\\"vacuum\\",
"SurfaceMaterialValue:=", "\\"\\",
"SolveInside:=", False,
"ShellElement:=", False,
"ShellElementThickness:=", "nan ",
```

```

    "IsMaterialEditable:=" , True,
    "UseMaterialAppearance:=", False,
    "IsLightweight:=" , False
]
)

```

<b>VB Syntax</b> <pre>CreateRegion &lt;Parameters&gt;, &lt;Attributes&gt;</pre>	<pre> oEditor.CreateRegion Array("NAME:RegionParameters", "+XPaddingType:=", _ "Percentage Offset", "+XPadding:=", "0", "-XPaddingType:=", "Percentage Offset", "-XPad- ding:=", _ "0", "+YPaddingType:=", "Percentage Offset", "+YPadding:=", "0", "-YPaddingType:=", _ "Percentage Offset", "-YPadding:=", "0", "+ZPaddingType:=", "Percentage Offset", "+ZPad- ding:=", _ "0", "-ZPaddingType:=", "Percentage Offset", "-ZPadding:=", "0"), Array("NAME:At- tributes", "Name:=", _ "Region", "Flags:=", "Wireframe#", "Color:=", "(143 175 143)", "Transparency:=", _ 0, "PartCoordinateSystem:=", "Global", "UDMID:=", "", "MaterialValue:=", _ "" &amp; Chr(34) &amp; "vacuum" &amp; Chr(34) &amp; "", "SurfaceMaterialValue:=", "" &amp; Chr(34) &amp; "" &amp; Chr(34) &amp; "", "SolveInside:=", _ false, "ShellElement:=", false, "ShellElementThickness:=", "nan ", "IsMa- terialEditable:=", _ true, "UseMaterialAppearance:=", false, "IsLightweight:=", false) </pre>
------------------------------------------------------------------------------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

## CreateRegularPolygon

Creates a regular polygon.

UI Access	Draw > Regular Polygon.		
<b>Parameters</b>	Name <i>&lt;Parameters&gt;</i>	Type Array	Description 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")
	<i>&lt;AttributesArray&gt;</i>	Array	Structured array. See: <a href="#">AttributesArray</a> .
<b>Return Value</b>	None.		

<b>Python Syntax</b>	CreateRegularPolygon(<Parameters>, <Attributes>)
<b>Python Example</b>	<code>oEditor.CreateRegularPolygon (</code>

```
[ "NAME:RegularPolygonParameters",
    "IsCovered:="           , True,
    "XCenter:="             , "-70000mm",
    "YCenter:="             , "-100000mm",
    "ZCenter:="             , "0mm",
    "XStart:="              , "-50000mm",
    "YStart:="              , "-80000mm",
    "ZStart:="              , "0mm",
    "NumSides:="            , "12",
    "WhichAxis:="           , "Z"
],
[ "NAME:Attributes",
    "Name:="                , "Polygon1",
    "Flags:="               , "",
    "Color:="               , "(143 175 143)",
    "Transparency:="         , 0,
    "PartCoordinateSystem:=", "Global",
    "UDMId:="               , "",
    "MaterialValue:="        , "\"copper\"",
    "SurfaceMaterialValue:=", "\"\""
]
```

	<pre>     "SolveInside:="           , False,     "ShellElement:="         , False,     "ShellElementThickness:=", "0mm",     "IsMaterialEditable:="   , True,     "UseMaterialAppearance:=", False,     "IsLightweight:="        , False   ] ) </pre>
--	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

<b>VB Syntax</b>	CreateRegularPolygon <Parameters>, <Attributes>
<b>VB Example</b>	<pre> oEditor.CreateRegularPolygon Array("NAME:RegularPolygonParameters", "IsCovered:=",   true, "XCenter:=", "-60000mm", "YCenter:=", "40000mm", "ZCenter:=", "0mm", "XStart:=",   -   "-50000mm", "YStart:=", "50000mm", "ZStart:=", "0mm", "NumSides:=", "8", "WhichAxis:=",   -   "Z"), Array("NAME:Attributes", "Name:=", "Polygon2", "Flags:=", "", "Color:=",   -(143 175 143), "Transparency:=", 0, "PartCoordinateSystem:=", "Global", "UDMId:=",   "", "MaterialValue:=", "" &amp; Chr(34) &amp; "copper" &amp; Chr(34) &amp; "", "Sur-   faceMaterialValue:=",   -   "" &amp; Chr(34) &amp; "" &amp; Chr(34) &amp; "", "SolveInside:=", false, "ShellElement:=",   -   false, "ShellElementThickness:=", "0mm", "IsMaterialEditable:=", true, "UseMa-   terialAppearance:=",   -   false, "IsLightweight:=", false) </pre>

## CreateRegularPolyhedron

Creates a regular polyhedron.

UI Access	Draw > Regular Polyhedron.		
<b>Parameters</b>	Name <i>&lt;Parameters&gt;</i>	Type Array	Description Structured array.  Array ("NAME:PolyhedronParameters", "XCenter:=" , <string>, "YCenter:=" , <string>, "ZCenter:=" , <string>, "XStart:=" , <string>, "YStart:=" , <string>, "ZStart:=" , <string>, "Height:=" , <string>, "NumSides:=" , <string containing number greater than 2>, "WhichAxis:=" , <string "X", "Y", or "Z">)
	<i>&lt;AttributesArray&gt;</i>	Array	Structured array. See: <a href="#">AttributesArray</a> .
<b>Return Value</b>	None.		

Python Syntax	CreateRegularPolyhedron(<Parameters>, <Attributes>)
---------------	-----------------------------------------------------

**Python Example**

```
oEditor.CreateRegularPolyhedron(  
    ["NAME:PolyhedronParameters",  
        "XCenter:=", "40000mm",  
        "YCenter:=", "-80000mm",  
        "ZCenter:=", "0mm",  
        "XStart:=", "50000mm",  
        "YStart:=", "-70000mm",  
        "ZStart:=", "0mm",  
        "Height:=", "50000mm",  
        "NumSides:=", "8",  
        "WhichAxis:=", "Z"  
    ["NAME:Attributes",  
        "Name:=", "RegularPolyhedron1",  
        "Flags:=", "",  
        "Color:=", "(143 175 143)",  
        "Transparency:=", 0,  
        "PartCoordinateSystem:=", "Global",  
        "UDMId:=", "",  
        "MaterialValue:=", "\copper\",  
        "SurfaceMaterialValue:=", "\"\\\"",  
    ])
```

```

    "SolveInside:="           , False,
    "ShellElement:="         , False,
    "ShellElementThickness:=", "0mm",
    "IsMaterialEditable:="   , True,
    "UseMaterialAppearance:=", False,
    "IsLightweight:="        , False
)

```

<b>VB Syntax</b>	CreateRegularPolyhedron <Parameters>, <Attributes>
<b>VB Example</b>	<pre> oEditor.CreateRegularPolyhedron Array("NAME:PolyhedronParameters", "XCenter:=", "-10000mm", "YCenter:=", "-50000mm", "ZCenter:=", "0mm", "XStart:=", "0mm", "YStart:=", - "-50000mm", "ZStart:=", "0mm", "Height:=", "90000mm", "NumSides:=", "8", "WhichAxis:=", - "Z"), Array("NAME:Attributes", "Name:=", "RegularPolyhedron2", "Flags:=", "", "Col- or:=", _ "(143 175 143)", "Transparency:=", 0, "PartCoordinateSystem:=", "Global", "UDMID:=", "", "MaterialValue:=", "" &amp; Chr(34) &amp; "copper" &amp; Chr(34) &amp; "", "Sur- faceMaterialValue:=", "" &amp; Chr(34) &amp; "" &amp; Chr(34) &amp; "", "SolveInside:=", false, "ShellElement:=", false, "ShellElementThickness:=", "0mm", "IsMaterialEditable:=", true, "UseMa- </pre>

```
terialAppearance:=", _  
false, "IsLightweight:=", false)
```

## CreateSphere

Creates a sphere.

UI Access	Draw > Sphere.		
<b>Parameters</b>	Name <i>&lt;Parameters&gt;</i>	Type Array	Description Structured array.  Array ("NAME:SphereParameters", "XCenter:=", <string>, "YCenter:=", <string>, "ZCenter:=", <string>, "Radius:=", <string>)
	<i>&lt;AttributesArray&gt;</i>	Array	Structured array. See: <a href="#">AttributesArray</a> .
<b>Return Value</b>	None.		

<b>Python Syntax</b>	CreateSphere(<Parameters>, <Attributes>)
<b>Python Example</b>	<pre>oEditor.CreateSphere(     ["NAME:SphereParameters",         "XCenter:=" , "-40000mm",         "YCenter:=" , "-130000mm",         "ZCenter:=" , "0mm",         "Radius:=" , "10000mm"])</pre>

```
    "Radius:=" , "22360.6797749979mm"  
],  
[ "NAME:Attributes",  
    "Name:=" , "Sphere1",  
    "Flags:=" , "",  
    "Color:=" , "(143 175 143)",  
    "Transparency:=" , 0,  
    "PartCoordinateSystem:=", "Global",  
    "UDMId:=" , "",  
    "MaterialValue:=" , "\"copper\"",  
    "SurfaceMaterialValue:=", "\",\"",  
    "SolveInside:=" , False,  
    "ShellElement:=" , False,  
    "ShellElementThickness:=" , "0mm",  
    "IsMaterialEditable:=" , True,  
    "UseMaterialAppearance:=" , False,  
    "IsLightweight:=" , False  
])
```

---

**VB****CreateSphere <Parameters>, <Attributes>**

Syntax	
VB Example	<pre> oEditor.CreateSphere Array("NAME:SphereParameters", "XCenter:=", "-12000mm", "YCen- ter:=", _ "9000mm", "ZCenter:=", "0mm", "Radius:=", "1000mm"), Array("NAME:Attributes", "Name:=", _ "Sphere2", "Flags:=", "", "Color:=", "(143 175 143)", "Transparency:=", 0, "PartCoordi- nateSystem:=", _ "Global", "UDMID:=", "", "MaterialValue:=", "" &amp; Chr(34) &amp; "copper" &amp; Chr(34) &amp; "", "Sur- faceMaterialValue:=", _ "" &amp; Chr(34) &amp; "" &amp; Chr(34) &amp; "", "SolveInside:=", false, "ShellElement:=", _ false, "ShellElementThickness:=", "0mm", "IsMaterialEditable:=", true, "UseMa- terialAppearance:=", _ false, "IsLightweight:=", false) </pre>

## CreateSpiral

Creates a spiral by sweeping the specified object(s).

UI Access	Draw > Spiral.											
	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;SelectionsArray&gt;</td> <td>Array</td> <td>Structured array. See: <a href="#">SelectionsArray</a>.</td> </tr> <tr> <td>&lt;Parameters&gt;</td> <td>Array</td> <td>Structured array.             Array ("NAME:SpiralParameters",           "XCenter:=" , &lt;string&gt;,           "YCenter:=" , &lt;string&gt;,           "ZCenter:=" , &lt;string&gt;, </td> </tr> </tbody> </table>	Name	Type	Description	<SelectionsArray>	Array	Structured array. See: <a href="#">SelectionsArray</a> .	<Parameters>	Array	Structured array.  Array ("NAME:SpiralParameters",           "XCenter:=" , <string>,           "YCenter:=" , <string>,           "ZCenter:=" , <string>,		
Name	Type	Description										
<SelectionsArray>	Array	Structured array. See: <a href="#">SelectionsArray</a> .										
<Parameters>	Array	Structured array.  Array ("NAME:SpiralParameters",           "XCenter:=" , <string>,           "YCenter:=" , <string>,           "ZCenter:=" , <string>,										
Parameters												

			<pre>"YStartDir:=" , &lt;string&gt;, "ZStartDir:=" , &lt;string&gt;, "NumThread:=" , &lt;string&gt;, "RightHand:=" , &lt;boolean&gt;, "RadiusIncrement:=" , &lt;string&gt;)</pre>
<b>Return Value</b>	None.		

<b>Python Syntax</b>	CreateSpiral(<Parameters>, <Attributes>)
<b>Python Example</b>	<pre>oEditor.CreateSpiral( ["NAME:Selections",  "Selections:=" , "Polygon2",  "NewPartsModelFlag:=" , "Model" ], ["NAME:SpiralParameters",  "XCenter:=" , "-70000mm",  "YCenter:=" , "50000mm",  "ZCenter:=" , "0mm",  "XStartDir:=" , "-60000mm",  "YStartDir:=" , "-10000mm",  "ZStartDir:=" , "0mm",</pre>

```

    "NumThread:="           , "1",
    "RightHand:="          , True,
    "RadiusIncrement:="    , "1mm"
]
)

```

<b>VB Syntax</b>	CreateSpiral <Parameters>, <Attributes>
<b>VB Example</b>	<pre> oEditor.CreateSpiral Array("NAME:Selections", "Selections:=", "Rectangle2", "NewPartsModelFlag:=", _ "Model"), Array("NAME:SpiralParameters", "XCenter:=", "30000mm", "YCenter:=", _  "-30000mm", "ZCenter:=", "0mm", "XStartDir:=", "-90000mm", "YStartDir:=", _  "-30000mm", "ZStartDir:=", "0mm", "NumThread:=", "1", "RightHand:=", false, "Radi- usIncrement:=", _  "1mm") </pre>

## CreateTorus

Creates a torus.

UI Access	Draw > Torus.								
Parameters	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;Parameters&gt;</td> <td>Array</td> <td> <p>Structured array.</p> <pre> Array("NAME:TorusParameters",       "XCenter:=" , &lt;string&gt;,       "YCenter:=" , &lt;string&gt;,       </pre> </td></tr> </tbody> </table>	Name	Type	Description	<Parameters>	Array	<p>Structured array.</p> <pre> Array("NAME:TorusParameters",       "XCenter:=" , &lt;string&gt;,       "YCenter:=" , &lt;string&gt;,       </pre>		
Name	Type	Description							
<Parameters>	Array	<p>Structured array.</p> <pre> Array("NAME:TorusParameters",       "XCenter:=" , &lt;string&gt;,       "YCenter:=" , &lt;string&gt;,       </pre>							

			<pre>"ZCenter:=" , &lt;string&gt;, "MajorRadius:=" , &lt;string&gt;, "MinorRadius:=" , &lt;string&gt;, "WhichAxis:=" , &lt;string "X", "Y", or "Z"&gt;)</pre>
	<b>&lt;AttributesArray&gt;</b>	Array	Structured array. See: <a href="#">AttributesArray</a> .
<b>Return Value</b>	None.		

<b>Python Syntax</b>	<code>CreateTorus(&lt;Parameters&gt;, &lt;Attributes&gt;)</code>
<b>Python Example</b>	<pre>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:=" , ""         ]     ] )</pre>

```

    "Color:="           , "(143 175 143)",
    "Transparency:="   , 0,
    "PartCoordinateSystem:=", "Global",
    "UDMID:="          , "",
    "MaterialValue:="   , "\"copper\"",
    "SurfaceMaterialValue:=", "\\"",
    "SolveInside:="     , False,
    "ShellElement:="    , False,
    "ShellElementThickness:=", "0mm",
    "IsMaterialEditable:=" , True,
    "UseMaterialAppearance:=", False,
    "IsLightweight:="    , False
  ]
)

```

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, "PartCoordinateSystem:=", _  "Global", "UDMID:=", "", "MaterialValue:=", "" &amp; Chr(34) &amp; "copper" &amp; Chr(34) &amp; "", _  </pre>

```

"SurfaceMaterialValue:=", _
"" & Chr(34) & "" & Chr(34) & "", "SolveInside:=", false, "ShellElement:=", _
false, "ShellElementThickness:=", "0mm", "IsMaterialEditable:=", true, "UseMa-
terialAppearance:=", _
false, "IsLightweight:=", false)

```

## CreateUserDefinedModel

Creates a user-defined model.

UI Access	Draw > User-Defined Model > [Model].		
Parameters	Name <i>&lt;Parameters&gt;</i>	Type Array	Description Structured array.  Array ("NAME:UserDefinedModelParameters", <definitionArray>, <optionsArray>, <geometryParamsArray>, "DllName:=", <string filepath>, "Library:=", <string>, "Version:=", <string> "ConnectionID:=", <string>)
	<definitionArray>	Array	Structured array containing string "NAME:Definition".
	<optionsArray>	Array	Structured array containing string "NAME:Options".
	<geometryParamsArray>	Array	Structured array containing arrays for individual parameters:

```
Array("NAME:GeometryParameters",
      Array("NAME:UDMParam",
            "Name:=" , <string>,
            "Value:=" , <string>,
            "PropType2:=" , <integer>,
            "PropFlag2:=" , <integer>))
```

Required UDM parameters depend on the UDM being created. To see which properties apply to a UDM, right-click the UDM in the Project Tree and select **Properties**. Then select the **Parameters** tab.

PropType2 can be any of the following:

- **0** – Property takes a string value.
- **1** – Property is a menu option.
- **2** – Property takes a number (integer or double).
- **3** – Property takes a value (numbers, variables, or expressions).
- **4** – Property is a file name.
- **5** – Property corresponds to a check box.
- **6** – Property specifies a 3D position.

PropFlag2 can be any of the following:

- **0** – No flags
- **1** – Read-only
- **2** – Must be integer
- **4** – Must be real

			<ul style="list-style-type: none"> <li>• <b>8 – Hidden</b></li> </ul> <p>PropFlag2 values can be combined. For example, a read-only property that must be an integer would take the value 3. A hidden property that must be real would take the value 12.</p> <p>These values are further described in the <code>User-DefinedPrimitiveStructures.h</code> file included with the installation under <code>AnsysEM[Version]\Win64\UserDefinedPrimitives\Examples\Headers</code></p>
<b>Return Value</b>	None.		

<b>Python Syntax</b>	<code>CreateUserDefinedModel(&lt;Parameters&gt;)</code>
<b>Python Example</b>	

<b>VB Syntax</b>	<code>CreateUserDefinedModel &lt;Parameters&gt;</code>
<b>VB Example</b>	

## CreateUserDefinedPart

Creates a user-defined part.

<b>UI Access</b>	<b>Draw &gt; User-Defined Primitive &gt; [Part].</b>		
<b>Parameters</b>	Name <code>&lt;Parameters&gt;</code>	Type Array	Description Structured array.  <code>Array ("NAME:UserDefinedPrimitiveParameters",</code>

			<pre>"DllName:=" , &lt;string&gt;, "Version:=" , &lt;string&gt;, "NoOfParameters:=" , &lt;integer&gt;, "Library:=" , &lt;string&gt;, &lt;paramVectorArray&gt;)</pre>
	<i>&lt;paramVectorArray&gt;</i>	Array	<p>Structured array containing arrays for each pair:</p> <pre>Array("NAME:ParamVector",       &lt;pair&gt;, &lt;pair&gt;, &lt;pair&gt;,...)</pre>
	<i>&lt;pair&gt;</i>	Array	<p>Structured array:</p> <pre>Array("NAME:Pair",       "Name:=" , &lt;string&gt;,       "Value:=" , &lt;string&gt;)</pre>
	<i>&lt;Attributes&gt;</i>	Array	Structured array. See: <a href="#">AttributesArray</a> .
<b>Return Value</b>	None.		

<b>Python Syntax</b>	CreateUserDefinedPart(<Parameters>, <Attributes>)
<b>Python Example</b>	<pre>oEditor.CreateUserDefinedPart(     ["NAME:UserDefinedPrimitiveParameters",      "DllName:=" , "RMxprt/LapCoil.dll",      "Version:=" , "16.0",      "NoOfParameters:=" , 22,</pre>

```
"Library:=" , "syslib",
  [ "NAME:ParamVector",
    [ "NAME:Pair",
      "Name:=" , "DiaGap",
      "Value:=" , "100mm"
    ],
    [ "NAME:Pair",
      "Name:=" , "DiaYoke",
      "Value:=" , "20mm"
    ],
    [ "NAME:Pair",
      "Name:=" , "Length",
      "Value:=" , "100mm"
    ],
    [ "NAME:Pair",
      "Name:=" , "Skew",
      "Value:=" , "0deg"
    ],
    [ "NAME:Pair",
      "Name:=" , "Slots",

```

```
"Value:=" , "18"
],
[ "NAME:Pair",
  "Name:=" , "SlotType",
  "Value:=" , "1"
],
[ "NAME:Pair",
  "Name:=" , "Hs0",
  "Value:=" , "1mm"
],
[ "NAME:Pair",
  "Name:=" , "Hs1",
  "Value:=" , "1mm"
],
[ "NAME:Pair",
  "Name:=" , "Hs2",
  "Value:=" , "10mm"
],
[ "NAME:Pair",
  "Name:=" , "Bs0",
  "Value:=" , "2.5mm"
```

```
        ] ,  
        [ "NAME:Pair",  
          "Name:=" , "Bs1",  
          "Value:=" , "8mm"  
        ] ,  
        [ "NAME:Pair",  
          "Name:=" , "Bs2",  
          "Value:=" , "5mm"  
        ] ,  
        [ "NAME:Pair",  
          "Name:=" , "Rs",  
          "Value:=" , "0mm"  
        ] ,  
        [ "NAME:Pair",  
          "Name:=" , "FilletType",  
          "Value:=" , "0"  
        ] ,  
        [ "NAME:Pair",  
          "Name:=" , "Layers",  
          "Value:=" , "2"
```

```
],  
["NAME:Pair",  
 "Name:=" , "CoilPitch",  
 "Value:=" , "4"  
],  
["NAME:Pair",  
 "Name:=" , "EndExt",  
 "Value:=" , "5mm"  
],  
["NAME:Pair",  
 "Name:=" , "SpanExt",  
 "Value:=" , "25mm"  
],  
["NAME:Pair",  
 "Name:=" , "BendAngle",  
 "Value:=" , "0deg"  
],  
["NAME:Pair",  
 "Name:=" , "SegAngle",  
 "Value:=" , "10deg"  
],
```

```
[ "NAME:Pair",
  "Name:=" , "LenRegion",
  "Value:=" , "200mm"
] ,
[ "NAME:Pair",
  "Name:=" , "InfoCoil",
  "Value:=" , "0"
]
],
[ "NAME:Attributes",
  "Name:=" , "LapCoill",
  "Flags:=" , "",
  "Color:=" , "(143 175 143)",
  "Transparency:=" , 0,
  "PartCoordinateSystem:=", "Global",
  "UDMId:=" , "",
  "MaterialValue:=" , "\"copper\"",
  "SurfaceMaterialValue:=" , "\"\"",
  "SolveInside:=" , False,
```

```

    "ShellElement:="           , False,
    "ShellElementThickness:=", "0mm",
    "IsMaterialEditable:="   , True,
    "UseMaterialAppearance:=", False,
    "IsLightweight:="         , False
  ])

```

VB Syntax	CreateUserDefinedPart <Parameters>, <Attributes>
VB Example	<pre> oEditor.CreateUserDefinedPart Array("NAME:UserDefinedPrimitiveParameters", "DllName:", _ "SegmentedHelix/RectHelix.dll", "Version:=", "1.0", "NoOfParameters:=", 8, "Library:", _ "syslib", Array("NAME:ParamVector", Array("NAME:Pair", "Name:=", "RectHeight", "Value:=", _ "1mm"), Array("NAME:Pair", "Name:=", "RectWidth", "Value:=", "2mm"), Array("NAME:Pair", "Name:=", _ "StartHelixRadius", "Value:=", "10mm"), Array("NAME:Pair", "Name:=", "RadiusChange", "Value:=", _ "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:=", _ </pre>

```

"RectHelix1", "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)

```

## Edit3DComponent

Edits a specified 3D component.

<b>UI Access</b>	N/A		
	<b>Name</b>	Type	Description
<b>Parameters</b>	<compName>	String	Component name.
	<Parameters>	Array	<p>Structured array.</p> <p>Array ("NAME:EditComponentParametersData",</p> <p>"NewComponentName:=", &lt;string&gt;,</p> <p>"GeometryParameters:=", &lt;string&gt;,</p> <p>"MaterialParameters:=", &lt;string&gt;,</p> <p>"DesignParameters:=", &lt;string&gt;,</p> <p>&lt;ComponentMeshing&gt;,</p> <p>&lt;Excitations&gt;)</p>

	<code>&lt;ComponentMeshing&gt;</code>	Array	Structured array.  Array ("NAME:Component Meshing", "MeshAssembly:=", <boolean>)
	<code>&lt;Excitations&gt;</code>	Array	Structured array containing array of suppressed excitations.  Array ("NAME:Excitations", "Suppressed:=", <array>)
<b>Return Value</b>	None.		

<b>Python Syntax</b>	<code>Edit3DComponent (&lt;compName&gt;, &lt;Parameters&gt;)</code>
<b>Python Example</b>	<pre> oEditor.Edit3DComponent (     "Connector1",     [         "NAME:EditComponentParametersData",         "NewComponentName:=", "Connector2",         "GeometryParameters:=", "",         "MaterialParameters:=", "",         "DesignParameters:=", "",         [             "NAME:Component Meshing",             "MeshAssembly:=", False],         [             "NAME:Excitations",             "Suppressed:=", []         ]     ] ) </pre>

	)
--	---

<b>VB Syntax</b>	Edit3DComponent <compName>, <Parameters>
<b>VB Example</b>	<pre>oEditor.Edit3DComponent     "Connector1",     Array ("NAME:EditComponentParametersData",         "NewComponentName:=", "Connector2",         "GeometryParameters:=", "",         "MaterialParameters:=", "",         "DesignParameters:=", "",         Array ("NAME:Component Meshing",             "MeshAssembly:=", false),         Array ("NAME:Excitations",             "Suppressed:=", Array ())     )</pre>

## Edit3DComponentDefinition

Edits definitions of a specified 3D component.

<b>UI Access</b>	N/A		
<b>Parameters</b>	Name <Parameters>	Type Array	Description Structured array.

			<pre>Array ("NAME&gt;EditComponentParametersData",       "OriginalComponentName:=", &lt;string&gt;,       "NewComponentName:=", &lt;string&gt;,       "GeometryParameters:=", &lt;string&gt;,       "MaterialParameters:=", &lt;string&gt;,       "DesignParameters:=", &lt;string&gt;,       &lt;ComponentMeshing&gt;,       &lt;Excitations&gt;)</pre>
	<i>&lt;ComponentMeshing&gt;</i>	Array	<p>Structured array.</p> <pre>Array ("NAME"&gt;Component Meshing",       "MeshAssembly:=", &lt;boolean&gt;)</pre>
	<i>&lt;Excitations&gt;</i>	Array	<p>Structured array containing array of suppressed excitations.</p> <pre>Array ("NAME"&gt;Excitations",       "Suppressed:=", &lt;array&gt;)</pre>
<b>Return Value</b>	None.		
<b>Python Syntax</b>	<pre>Edit3DComponent (&lt;Parameters&gt;)</pre>		
<b>Python Example</b>	<pre>oEditor.Edit3DComponent (     [ "NAME&gt;EditComponentParametersData",       "OriginalComponentName:=", "Connector1",       "NewComponentName:=", "Connector2",       "GeometryParameters:=", "" ,</pre>		

```
"MaterialParameters:="", "",  
"DesignParameters:="", "",  
[ "NAME:Component Meshing",  
    "MeshAssembly:=", False],  
[ "NAME:Excitations",  
    "Suppressed:=", []]  
]  
)
```

VB Syntax	Edit3DComponent <Parameters>
<b>VB Example</b>	<pre>oEditor.Edit3DComponent Array("NAME&gt;EditComponentParametersData",     "OriginalComponentName:=", "Connector1",     "NewComponentName:=", "Connector2",     "GeometryParameters:="", "",     "MaterialParameters:="", "",     "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: <a href="#">SelectionsArray</a> .
	<Parameters>	Array	Structured array.  Array ("NAME:PolylineParameters", "IsPolylineCovered:=", <bool>, "IsPolylineClosed:=", <bool>, <PolylinePointsArray>, <PolylineSegmentsArray>)
	<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>)
<b>Return Value</b>	None.		

<b>Python Syntax</b>	EditPolyline(<SelectionsArray>, <Parameters>)
<b>Python Example</b>	<pre>oEditor.EditPolyline(     ["NAME:Selections",      "Selections:=", "Polyline1"]      ["NAME:PolylineParameters",      "IsPolylineCovered:=", True,      "IsPolylineClosed:=", False,      ["NAME:PolylinePoints",       ["NAME:PLPoint",        "X:=" , "2000mm",        "Y:=" , "-2000mm",        "Z:=" , "0mm"       ],       ["NAME:PLPoint",        ["NAME:PLPoint",         "X:=" , "2000mm",         "Y:=" , "-2000mm",         "Z:=" , "0mm"        ]      ]    ]</pre>

```
"X:=" , "-9000mm",
"Y:=" , "2000mm",
"Z:=" , "0mm"
] ,
[ "NAME:PLPoint",
  "X:=" , "10000mm",
  "Y:=" , "-14000mm",
  "Z:=" , "0mm"
]
],
[ "NAME:PolylineSegments",
  [ "NAME:PLSegment",
    "SegmentType:=" , "Line",
    "StartIndex:=" , 0,
    "NoOfPoints:=" , 2
  ],
  [ "NAME:PLSegment",
    "SegmentType:=" , "Line",
    "StartIndex:=" , 1,
    "NoOfPoints:=" , 2
  ]
]
```

```

        ],
        [
            "NAME:PolylineXSection",
            "XSectionType:=" , "None",
            "XSectionOrient:=" , "Auto",
            "XSectionWidth:=" , "0mm",
            "XSectionTopWidth:=" , "0mm",
            "XSectionHeight:=" , "0mm",
            "XSectionNumSegments:=" , "0",
            "XSectionBendType:=" , "Corner"
        ]
    )
)

```

<b>VB Syn- tax</b>	EditPolyline <SelectionsArray>, <Parameters>
<b>VB Ex- ample</b>	<pre> oEditor.EditPolyline Array("NAME:Selections", "Selections:=", "Polyline1") Array ("NAME:PolylineParameters", "IsPolylineCovered:=", true, "IsPolylineClosed:=",  false, Array("NAME:PolylinePoints", Array("NAME:PLPoint", "X:=", "40000mm", "Y:=",  "50000mm", "Z:=", "0mm"), Array("NAME:PLPoint", "X:=", "-15000mm", "Y:=",  "-50000mm", "Z:=", _ "0mm)), Array("NAME:PolylineSegments", Array("NAME:PLSegment",  "SegmentType:=", "Line", "StartIndex:=", _</pre>

```

0, "NoOfPoints:=", 2)), Array("NAME:PolylineXSection", "XSectionType:=", "None", "XSectionOrient:=", _
"Auto", "XSectionWidth:=", "0mm", "XSectionTopWidth:=", "0mm", "XSectionHeight:=", _
"0mm", "XSectionNumSegments:=", "0", "XSectionBendType:=", "Corner")), Array("NAME:Attributes", "Name:=", _
"Polyline2", "Flags:=", "", "Color:=", "(143 175 143)", "Transparency:=", 0, "PartCoordinateSystem:=", _
"Global", "UDMID:=", "", "MaterialValue:=", "" & Chr(34) & "copper" & Chr(34) & "", "SurfaceMaterialValue:=", _
"" & Chr(34) & "" & Chr(34) & "", "SolveInside:=", false, "ShellElement:=", _
false, "ShellElementThickness:=", "0mm", "IsMaterialEditable:=", true, "UseMaterialAppearance:=", _
false, "IsLightweight:=", false)

```

## Get3DComponentDefinitionNames

Gets names of 3D component definitions.

<b>UI Access</b>	N/A
<b>Parameters</b>	None.
<b>Return Value</b>	Array of strings containing component definition names.

<b>Python Syntax</b>	Get3DComponentDefinitionNames()
<b>Python Example</b>	<code>oEditor.Get3DComponentDefinitionNames ()</code>

<b>VB Syntax</b>	Get3DComponentDefinitionNames()
<b>VB Example</b>	<pre>Dim defNames defNames = oEditor.Get3DComponentDefinitionNames()</pre>

### Get3DComponentInstanceNames

Returns instance names of 3D component definitions.

<b>UI Access</b>	N/A						
<b>Parameters</b>	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>&lt;DefinitionName&gt;</td><td>String</td><td>Definition name.</td></tr></tbody></table>	Name	Type	Description	<DefinitionName>	String	Definition name.
Name	Type	Description					
<DefinitionName>	String	Definition name.					
<b>Return Value</b>	Array containing instance names.						

<b>Python Syntax</b>	Get3DComponentInstanceNames(<DefinitionName>)
<b>Python Example</b>	<pre>oEditor.Get3DComponentInstanceNames ("Connector")</pre>

<b>VB Syntax</b>	Get3DComponentInstanceNames <DefinitionName>
<b>VB Example</b>	<pre>oEditor.Get3DComponentInstanceNames "Connector"</pre>

### Get3DComponentMaterialNames

Returns material names for a specified \*.a3dcomp format 3D component.

<b>UI Access</b>	N/A
------------------	-----

<b>Parameters</b>	Name <i>&lt;InstanceName&gt;</i>	Type String	Description Component instance name.
<b>Return Value</b>	Array containing material names.		

<b>Python Syntax</b>	Get3DComponentMaterialNames( <i>&lt;InstanceName&gt;</i> )
<b>Python Example</b>	<code>oEditor.Get3DComponentMaterialNames ("Connector1.a3dcomp")</code>

<b>VB Syntax</b>	Get3DComponentMaterialNames < <i>InstanceName</i> >
<b>VB Example</b>	<code>oEditor.Get3DComponentMaterialNames "Connector1.a3dcomp"</code>

## Get3DComponentMaterialProperties

Returns material properties for a specified 3D component.

<b>UI Access</b>	N/A		
<b>Parameters</b>	Name <i>&lt;MaterialName&gt;</i>	Type String	Description Material name.
<b>Return Value</b>	Array containing material properties.		

<b>Python Syntax</b>	Get3DComponentMaterialProperties( <i>&lt;MaterialName&gt;</i> )
<b>Python Example</b>	<code>oEditor.Get3DComponentMaterialProperties ('Connector1:Material01')</code>

<b>VB Syntax</b>	Get3DComponentMaterialProperties < <i>MaterialName</i> >
------------------	----------------------------------------------------------

**VB Example**

```
oEditor.Get3DComponentMaterialProperties "Connector1:Material01"
```

## Get3DComponentParameters

Returns parameters for a specified 3D component.

<b>UI Access</b>	N/A								
<b>Parameters</b>	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>&lt;compName&gt;</td><td>String</td><td>3D component name.</td></tr></tbody></table>			Name	Type	Description	<compName>	String	3D component name.
Name	Type	Description							
<compName>	String	3D component name.							
<b>Return Value</b>	Array containing component parameters.								

**Python Syntax**

```
Get3DComponentParameters(<compName>)
```

**Python Example**

```
oEditor.Get3DComponentParameters ('Connector')
```

<b>VB Syntax</b>	Get3DComponentParameters <compName>		
<b>VB Example</b>	<pre>oEditor.Get3DComponentParameters "Connector"</pre>		

## Get3DComponentPartNames

Returns part names for a specified 3D component.

<b>UI Access</b>	N/A								
<b>Parameters</b>	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>&lt;InstanceName&gt;</td><td>String</td><td>3D component instance.</td></tr></tbody></table>			Name	Type	Description	<InstanceName>	String	3D component instance.
Name	Type	Description							
<InstanceName>	String	3D component instance.							

<b>Return Value</b>	Array containing part names.
---------------------	------------------------------

<b>Python Syntax</b>	Get3DComponentParameters(<InstanceName>)
<b>Python Example</b>	<code>oEditor.Get3DComponentPartNames ('Connector')</code>

<b>VB Syntax</b>	Get3DComponentParameters <InstanceName>
<b>VB Example</b>	<code>oEditor.Get3DComponentPartNames "Connector"</code>

## Insert3DComponent

Inserts a 3D component.

<b>UI Access</b>	Draw > 3D Component Library > Browse > [Component].						
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;ComponentData&gt;</td> <td>Array</td> <td>           Structured array.            Array ("NAME:InsertComponentData",            "Parameters:=", &lt;string&gt;,            "TargetCS:=", &lt;string&gt;,            "ComponentFile:=", &lt;string filepath&gt;)         </td> </tr> </tbody> </table>	Name	Type	Description	<ComponentData>	Array	Structured array. Array ("NAME:InsertComponentData", "Parameters:=", <string>, "TargetCS:=", <string>, "ComponentFile:=", <string filepath>)
Name	Type	Description					
<ComponentData>	Array	Structured array. Array ("NAME:InsertComponentData", "Parameters:=", <string>, "TargetCS:=", <string>, "ComponentFile:=", <string filepath>)					
<b>Return Value</b>	None.						

<b>Python Syntax</b>	Insert3DComponent(<ComponentData>)
<b>Python Example</b>	<code>oEditor.Insert3DComponent (</code>

```
[ "NAME:InsertComponentData",
  "Parameters:=", "",
  "TargetCS:=", "Global",
  "ComponentFile:=", "C:\tmp\Connector.a3dcomp" ] )
```

<b>VB Syntax</b>	Insert3DComponent < <i>ComponentData</i> >
<b>VB Example</b>	<pre>oEditor.Insert3DComponent Array("NAME:InsertComponentData",       "Parameters:=", "",       "TargetCS:=", "Global",       "ComponentFile:=", "C:\tmp\Connector.a3dcomp")</pre>

## InsertComponent

Inserts a component.

<b>UI Access</b>	N/A		
<b>Parameters</b>	Name	Type	Description

	<code>&lt;ComponentData&gt;</code>	Array	Structured array.  Array("NAME:InsertComponentData", "Parameters:=", <string>, "TargetCS:=", <string>, "ComponentFile:=", <string filepath>)
<b>Return Value</b>	None.		

<b>Python Syntax</b>	<code>InsertComponent(&lt;ComponentData&gt;)</code>
<b>Python Example</b>	<code>oEditor.InsertComponent(     [ "NAME:InsertComponentData",         "Parameters:=", "",         "TargetCS:=", "Global",         "ComponentFile:=", "C:\\tmp\\Connector.a3dcomp" ] )</code>

<b>VB Syntax</b>	<code>InsertComponent &lt;ComponentData&gt;</code>
<b>VB Example</b>	<code>oEditor.InsertComponent     Array("NAME:InsertComponentData",           "Parameters:=", "",           "TargetCS:=", "Global",           "ComponentFile:=", "C:\\tmp\\Connector.a3dcomp")</code>

## InsertPolylineSegment

Inserts a polyline segment before or after a specified existing segment.

UI Access	Draw > Line Segment > [Selection].		
<b>Parameters</b>	Name <i>&lt;Parameters&gt;</i>	Type Array	Description Structured array.  Array ("NAME:Insert Polyline Segment", "Selections:=" , <string>, "Segment Indices:=" , <array containing integers>, "At Start:=" , <boolean>, "SegmentType:=" , <string "Line", "Arc", "Spline", or "AngularArc">, <PolylinePointsArray>)
	<PolylinePointsArray>	Array	Structured array. See: <a href="#">CreatePolyline</a> .
<b>Return Value</b>	None.		

<b>Python Syntax</b>	InsertPolylineSegment(<Parameters>)
<b>Python Example</b>	<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"
      ]
    ]
  ]
)

```

<b>VB Syntax</b>	InsertPolylineSegment <Parameters>
<b>VB Example</b>	<pre> oEditor.InsertPolylineSegment Array("NAME:Insert Polyline Segment", "Selections:=",   "Polyline1&gt;CreatePolyline:1", "Segment Indices:=", Array(1), "At Start:=",   false, "SegmentType:=", "Spline", Array("NAME:PolylinePoints", Array("NAME:PLPoint",   "X:=",   </pre>

```

"-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.		
	Name <i>&lt;SelectionsArray&gt;</i>	Type Array	Description Structured array. See: <a href="#">SelectionsArray</a> .
Parameters	<i>&lt;PathSweepParametersArray&gt;</i>	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>)
---------------	---------------------------------------------------------------

<b>Python Example</b>	<pre> oEditor.SweepAlongPath(     [         "NAME:Selections",         "Selections:=" , "Rectangle1,Polyline1",         "NewPartsModelFlag:=" , "Model"     ],     [         "NAME:PathSweepParameters",         "DraftAngle:=" , "0deg",         "DraftType:=" , "Round",         "CheckFaceFaceIntersection:=", False,         "TwistAngle:=" , "0deg"     ] ) </pre>
-----------------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

<b>VB Syntax</b>	SweepAlongPath < <i>SelectionsArray</i> >, < <i>PathSweepParametersArray</i> >
<b>VB Example</b>	<pre> oEditor.SweepAlongPath      Array("NAME:Selections", "Selections:=",           "Polygon1,Polyline1"),_     Array("NAME:PathSweepParameters", _           "DraftAngle:=" , "0deg",_           "DraftType:=" , "Round",_ </pre>

	<pre>"CheckFaceFaceIntersection:=", False,_ "TwistAngle:=", "30deg")</pre>
--	----------------------------------------------------------------------------

## SweepAlongVector

Sweeps the specified 1D or 2D parts along a vector.

UI Access	Draw > Sweep > Along Vector.		
	Name	Type	Description
Parameters	<SelectionsArray> <VecSweepParametersArray>	Array Array	Structured array. See: <a href="#">SelectionsArray</a> . 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	<code>SweepAlongVector(&lt;SelectionsArray&gt;, &lt;VecSweepParametersArray&gt;)</code>
Python Example	<code>oEditor.SweepAlongVector(</code>

```
[

    "NAME:Selections",
    "Selections:=", "Rectangle1",
    "NewPartsModelFlag:=", "Model"

],


[

    "NAME:VectorSweepParameters",
    "DraftAngle:=", "0deg",
    "DraftType:=", "Round",
    "CheckFaceFaceIntersection:=", False,
    "SweepVectorX:=", "0mm"
    "SweepVectorY:=", "0mm"
    "SweepVectorZ:=", "12mm"
])
)
```

<b>VB Syntax</b>	SweepAlongVector <SelectionsArray>, <VecSweepParametersArray>
<b>VB Example</b>	<pre> oEditor.SweepAlongPath      Array("NAME:Selections",_         "Selections:=", "Rectangle1",_         "NewPartsModelFlag:=", "Model")      Array("NAME:VectorSweepParameters", _</pre>

```
"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.		
	Name <i>&lt;SelectionsArray&gt;</i>	Type Array	Description Structured array. See: <a href="#">SelectionsArray</a> .
Parameters	<i>&lt;AxisSweepParametersArray&gt;</i>	Type Array	<p>Array ("NAME:AxisSweepParameters",</p> <pre>"DraftAngle:=", &lt;value&gt;, "DraftType:=", &lt;string&gt;, "CheckFaceFaceIntersection:=", &lt;bool&gt;, "SweepAxis:=", &lt;value&gt; "SweepAngle:=", &lt;value&gt; "NumOfSegments:=", &lt;value&gt;)</pre> <p>Possible values for DraftType are "Extended", "Round", and "Natural".</p>

		Possible values for SweepAxis are "X", "Y", and "Z".
<b>Return Value</b>	None.	

<b>Python Syntax</b>	SweepAroundAxis(<SelectionsArray>, <AxisSweepParametersArray>)
<b>Python Example</b>	<pre> oEditor.SweepAroundAxis(     [         "NAME:Selections",         "Selections:=" , "Rectangle1",         "NewPartsModelFlag:=" , "Model"     ],     [         "NAME:AxisSweepParameters",         "DraftAngle:=" , "0deg",         "DraftType:=" , "Round",         "CheckFaceFaceIntersection:=", False,         "SweepAxis:=" , "X"         "SweepAngle:=" , "360deg"         "NumOfSegments:=" , "12"     ] ) </pre>

<b>VB Syntax</b>	SweepAroundAxis <SelectionsArray>, <AxisSweepParametersArray>
<b>VB Example</b>	<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 <SelectionsArray> <parameters>	Type Array Array	Description Structured array. See: <a href="#">SelectionsArray</a> . Structured array. Array ("NAME: Parameters", "NAME:SweepFaceAlongNormalToParameters",

			"FacesToDetach:=", <faceIDarray>, "LengthOfSweep:=", "<value><units>")
<b>Return Value</b>	None		

<b>Python Syntax</b>	SweepFacesAlongNormal(<SelectionsArray> <parameters>)
<b>Python Example</b>	<pre> oEditor.SweepFacesAlongNormal (     ["NAME:Selections",         "Selections:=", "Rectangle1",         "NewPartsModelFlag:=", "Model"],     ["NAME:Parameters",         "NAME:SweepFaceAlongNormalToParameters",         "FacesToDetach:=", [183],         "LengthOfSweep:=", "0.1mm"]) </pre>

<b>VB Syntax</b>	SweepFacesAlongNormal <SelectionsArray> <parameters>
<b>VB Example</b>	<pre> oEditor.SweepFacesAlongNormal     Array ("NAME:Selections",         "Selections:=", "Rectangle1",         "NewPartsModelFlag:=", "Model"), </pre>

```
        Array ("NAME:Parameters",
                "NAME:SweepFaceAlongNormalToParameters",
                "FacesToDetach:=", Array(57),
                "LengthOfSweep:=", "0.5mm")
    )
```

## SweepFacesAlongNormalWithAttributes

Sweep a face along normal, and specify attributes of the new object.

UI Access	Modeler > Surface > Sweep Faces Along Normal		
<b>Parameters</b>	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: <a href="#">SelectionsArray</a> .
	<parameters>	Array	Array ("NAME:Parameters",                 "NAME:SweepFaceAlongNormalToParameters",                 "FacesToDetach:=", <faceIDarray>,                 "LengthOfSweep:=", "<value><units>")
<AttributesArray>	Array	Structured array. See: <a href="#">AttributesArray</a> .	
<b>Return Value</b>	None		

<b>Python Syntax</b>	SweepFacesAlongNormalWithAttributes(<SelectionsArray>, <parameters>, <AttributesArray>)
<b>Python Example</b>	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,
    "ShellElementThickness:=", "0mm",
    "IsMaterialEditable:=", True,
    "UseMaterialAppearance:=", False,
```

	"IsLightweight:=", False])
--	----------------------------

<b>VB Syntax</b>	SweepFacesAlongNormalWithAttributes <SelectionsArray>, <parameters>, <AttributesArray>
<b>VB Example</b>	<pre> oEditor.SweepFacesAlongNormalWithAttributes      Array("NAME:Selections",           "Selections:=", "Rectangle1",           "NewPartsModelFlag:=", "Model"),     Array("NAME:Parameters",           "NAME:SweepFaceAlongNormalToParameters",           "FacesToDetach:=", Array(57),           "LengthOfSweep:=", "0.5mm"),     Array("NAME:Attributes",           "Name:=", "Box3",           "Flags:=", "",           "Color:=", "(143 175 143)",           "Transparency:=", 0,           "PartCoordinateSystem:=", "Global",           "UDMId:=", "",           "MaterialValue:=", "\"copper\""), </pre>

```
"SurfaceMaterialValue:=", "\\"\\\"",  
"SolveInside:=", false,  
"ShellElement:=", false,  
"ShellElementThickness:=", "0mm",  
"IsMaterialEditable:=", true,  
"UseMaterialAppearance:=", false,  
"IsLightweight:=", false)  
)
```

## **UpdateComponentDefinition**

Updates a 3D component's definition.

UI Access	Draw > 3D Component Library > Definitions.		
Parameters	Name	Type	Description
	<data>	Array	<p>Structured array.</p> <pre data-bbox="840 899 1744 910">Array("NAME:UpdateDefinitionData",</pre> <pre data-bbox="840 910 1744 922">"DefinitionNames:=", &lt;string&gt;,</pre> <pre data-bbox="840 922 1744 935">"Passwords:=", &lt;array of strings&gt;)</pre>
Return Value	None.		

<b>Python Syntax</b>	UpdateComponentDefinition(<data>)
<b>Python Example</b>	<pre>oEditor.UpdateComponentDefinition(     [ 'NAME:UpdateDefinitionData',       ... ] )</pre>

```
'DefinitionNames:=' , 'Connector, Magic_Tee',
'Passwords:=' , [ '' , '' ]
]
)
```

<b>VB Syntax</b>	UpdateComponentDefinition <data>
<b>VB Example</b>	<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.

<b>UI Access</b>	N/A		
<b>Parameters</b>	Name <i>&lt;SelectionsArray&gt;</i>	Type Array	Description Structured array. See: <a href="#">SelectionsArray</a> .
<b>Return Value</b>	None.		

<b>Python Syntax</b>	Copy( <i>&lt;SelectionsArray&gt;</i> )
<b>Python Example</b>	<pre>oEditor.Copy([     "NAME:Selections",     "Selections:=", "Box1"])</pre>

<b>VB Syntax</b>	Copy < <i>SelectionsArray</i> >
<b>VB Example</b>	<pre>oEditor.Copy Array(     "NAME:Selections",     "Selections:=", "Box1")</pre>

## DeletePolylinePoint

Deletes either a start point or an end point from an existing polyline segment.

UI Access	Edit > Delete [Start/End] Point		
<b>Parameters</b>	Name <i>&lt;DeletePointArray&gt;</i>	Type Array	Description 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>)
<b>Return Value</b>	None.		

<b>Python Syntax</b>	DeletePolylinePoint (<DeletePointArray>)
<b>Python Example</b>	<pre>oEditor.DeletePolylinePoint(["NAME:Delete Point",                              "Selections:=", "Polyline1:CreatePolyline:1",                              "Segment Index:=", 1,                              "At Start:=", True])</pre>

<b>VB Syn- tax</b>	DeletePolylinePoint <DeletePointArray>
------------------------	----------------------------------------

<b>VB Example</b>	<pre><code>oEditor.DeletePolylinePoint Array("NAME:Delete Point", "Selections:=", "Polyline1&gt;CreatePolyline:1", "Segment Index:=", 1, "At Start:=", True)</code></pre>
-------------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------

## DuplicateAlongLine

Duplicates specified parts along a line.

UI Access	Edit > Duplicate > Along Line.		
Parameters	Name	Type	Description
	<code>&lt;SelectionsArray&gt;</code>	Array	<p>Structured array.</p> <pre><code>Array("NAME:Selections",       "Selections:=" , &lt;string&gt;,       "NewPartsModelFlag:=" , &lt;string&gt;)</code></pre>
	<code>&lt;ParametersArray&gt;</code>	Array	<p>Structured array.</p> <pre><code>Array("NAME:DuplicateToAlongLineParameters",       "CreateNewObjects:=" , &lt;boolean&gt;,       "XComponent:=" , &lt;string&gt;,       "YComponent:=" , &lt;string&gt;,       "ZComponent:=" , &lt;string&gt;,       "NumClones:=" , &lt;string containing number greater       than 1&gt;)</code></pre>
	<code>&lt;OptionsArray&gt;</code>	Array	<p>Structured array.</p> <pre><code>Array("NAME:Options",       "DuplicateAssignments:=" , &lt;boolean&gt;)</code></pre>
	<code>&lt;CreateGroup&gt;</code>	Array	Optional. Structured array.

		Array ("CreateGroupsForNewObjects:=", <boolean>)
<b>Return Value</b>	None.	

<b>Python Syntax</b>	DuplicateAlongLine (<SelectionsArray>, <ParametersArray>, <OptionsArray>, <CreateGroup>)
<b>Python Example</b>	<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>

```
[ "CreateGroupsForNewObjects:=", False
])
```

<b>VB Syntax</b>	DuplicateAlongLine <SelectionsArray>, <ParametersArray>, <OptionsArray>, <CreateGroup>
<b>VB Example</b>	<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.**

	Name	Type	Description
<b>Parameters</b>	<code>&lt;SelectionsArray&gt;</code>	Array	Structured array.  <code>Array ("NAME:Selections",         "Selections:=" , &lt;string&gt;,         "NewPartsModelFlag:=" , &lt;string&gt;)</code>
	<code>&lt;ParametersArray&gt;</code>	Array	Structured array.  <code>Array ("NAME:DuplicateAroundAxisParameters",         "CreateNewObjects:=" , &lt;boolean&gt;,         "WhichAxis:=" , &lt;string&gt;,         "AngleStr:=" , &lt;string&gt;,         "NumClones:=" , &lt;string containing number greater         than 1&gt;)</code>
	<code>&lt;OptionsArray&gt;</code>	Array	Structured array.  <code>Array ("NAME:Options",         "DuplicateAssignments:=" , &lt;boolean&gt;)</code>
	<code>&lt;CreateGroup&gt;</code>	Array	Optional. Structured array.  <code>Array ("CreateGroupsForNewObjects:=" , &lt;boolean&gt;)</code>
<b>Return Value</b>	None.		

<b>Python Syntax</b>	<code>DuplicateAroundAxis (&lt;SelectionsArray&gt;, &lt;ParametersArray&gt;, &lt;OptionsArray&gt;, &lt;CreateGroup&gt;)</code>
<b>Python Example</b>	<code>oEditor.DuplicateAroundAxis (</code>

```
[ "NAME:Selections",
    "Selections:=", "Box1",
    "NewPartsModelFlag:=", "Model"
],
[ "NAME:DuplicateAroundAxisParameters",
    "CreateNewObjects:=", True,
    "WhichAxis:=", "Z",
    "AngleStr:=", "90deg",
    "NumClones:=", "2"
],
[ "NAME:Options",
    "DuplicateAssignments:=", False
],
[ "CreateGroupsForNewObjects:=", False
])
```

<b>VB Syntax</b>	DuplicateAroundAxis <SelectionsArray>, <ParametersArray>, <OptionsArray>, <CreateGroup>
<b>VB Example</b>	<pre>oEditor.DuplicateAroundAxis Array("NAME:Selections",     "Selections:=", "Box1",     "NewPartsModelFlag:=", "Model")</pre>

```

Array("NAME:DuplicateAroundAxisParameters",
      "CreateNewObjects:=", true,
      "WhichAxis:=", "Z",
      "AngleStr:=", "90deg",
      "NumClones:=", "2")

Array("NAME:Options",
      "DuplicateAssignments:=", false)

Array("CreateGroupsForNewObjects:=", false)

```

## DuplicateMirror

Duplicates specified parts according to a mirror plane.

UI Access	Edit > Duplicate > Mirror.		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	<p>Structured array.</p> <pre> Array("NAME:Selections",       "Selections:=" , &lt;string&gt;,       "NewPartsModelFlag:=" , &lt;string&gt;) </pre>
	<ParametersArray>	Array	<p>Structured array.</p> <pre> Array("NAME:DuplicateToMirrorParameters",       "DuplicateMirrorBaseX:=" , &lt;string&gt;,       "DuplicateMirrorBaseY:=" , &lt;string&gt;,       "DuplicateMirrorBaseZ:=" , &lt;string&gt;) </pre>

		<pre>"DuplicateMirrorNormalX:=", &lt;string&gt;, "DuplicateMirrorNormalY:=", &lt;string&gt;, "DuplicateMirrorNormalZ:=", &lt;string&gt;)</pre>
	<b>&lt;OptionsArray&gt;</b>	<b>Array</b> Structured array. <pre>Array("NAME:Options",       "DuplicateAssignments:=", &lt;boolean&gt;)</pre>
	<b>&lt;CreateGroup&gt;</b>	<b>Array</b> Optional. Structured array. <pre>Array("CreateGroupsForNewObjects:=", &lt;boolean&gt;)</pre>
<b>Return Value</b>	None.	

<b>Python Syntax</b>	<pre>DuplicateMirror (&lt;SelectionsArray&gt;, &lt;ParametersArray&gt;, &lt;OptionsArray&gt;, &lt;CreateGroup&gt;)</pre>
<b>Python Example</b>	<pre>oEditor.DuplicateMirror(   ["NAME:Selections",     "Selections:=", "Box1",     "NewPartsModelFlag:=", "Model"   ],   ["NAME:DuplicateToMirrorParameters",     "DuplicateMirrorBaseX:=", "-0.4mm",     "DuplicateMirrorBaseY:=", "-1.2mm",     "DuplicateMirrorBaseZ:=", "0mm",     "DuplicateMirrorNormalX:=", "0.124034734589208mm",</pre>

```
    "DuplicateMirrorNormalY:=", "0.992277876713668mm",
    "DuplicateMirrorNormalZ:=", "0mm"
],
[ "NAME:Options",
    "DuplicateAssignments:=", False
],
[ "CreateGroupsForNewObjects:=", False
])
```

<b>VB Syntax</b>	DuplicateMirror <SelectionsArray>, <ParametersArray>, <OptionsArray>, <CreateGroup>
<b>VB Example</b>	<pre>oEditor.DuplicateMirror     Array("NAME:Selections",         "Selections:=", "Box1",         "NewPartsModelFlag:=", "Model")     Array("NAME:DuplicateToMirrorParameters",         "DuplicateMirrorBaseX:=", "-0.4mm",         "DuplicateMirrorBaseY:=", "-1.2mm",         "DuplicateMirrorBaseZ:=", "0mm",         "DuplicateMirrorNormalX:=", "0.124034734589208mm",</pre>

```

    "DuplicateMirrorNormalY:=", "0.992277876713668mm",
    "DuplicateMirrorNormalZ:=", "0mm")

Array("NAME:Options",
      "DuplicateAssignments:=", false)

Array("CreateGroupsForNewObjects:=", false)

```

## Mirror

Mirrors specified part(s).

UI Access	Edit > Arrange > Mirror.		
	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: <a href="#">SelectionsArray</a> .
Parameters	<MirrorParameters>	Array	<p>Structured array.</p> <pre> Array("NAME:MirrorParameters",       "MirrorBaseX:=" , &lt;string&gt;,       "MirrorBaseY:=" , &lt;string&gt;,       "MirrorBaseZ:=" , &lt;string&gt;,       "MirrorNormalX:=" , &lt;string&gt;,       "MirrorNormalY:=" , &lt;string&gt;,       "MirrorNormalZ:=" , &lt;string&gt;) </pre>
Return Value	None.		

### Python Syntax

```
Mirror(<SelectionsArray>, <MirrorParameters>)
```

	<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>
--	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

<b>VB Syntax</b>	Mirror < <i>SelectionsArray</i> > < <i>MirrorParameters</i> >
<b>VB Example</b>	<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	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;SelectionsArray&gt;</td> <td>Array</td> <td>Structured array. See: <a href="#">SelectionsArray</a>.</td> </tr> <tr> <td>&lt;TranslateParameters&gt;</td> <td>Array</td> <td>Structured array.  Array ([ "NAME:TranslateParameters",           "TranslateVectorX:=" , &lt;string&gt;,           "TranslateVectorY:=" , &lt;string&gt;,           "TranslateVectorZ:=" , &lt;string&gt;)</td> </tr> </tbody> </table>	Name	Type	Description	<SelectionsArray>	Array	Structured array. See: <a href="#">SelectionsArray</a> .	<TranslateParameters>	Array	Structured array.  Array ([ "NAME:TranslateParameters", "TranslateVectorX:=" , <string>, "TranslateVectorY:=" , <string>, "TranslateVectorZ:=" , <string>)		
Name	Type	Description										
<SelectionsArray>	Array	Structured array. See: <a href="#">SelectionsArray</a> .										
<TranslateParameters>	Array	Structured array.  Array ([ "NAME:TranslateParameters", "TranslateVectorX:=" , <string>, "TranslateVectorY:=" , <string>, "TranslateVectorZ:=" , <string>)										
Return Value	None.											

Python Syntax	Move(<SelectionsArray>, <TranslateParameters>)
Python Example	oEditor.Move (

```
[ "NAME:Selections",
    "Selections:=", "Box1_1",
    "NewPartsModelFlag:=", "Model"
],
[ "NAME:TranslateParameters",
    "TranslateVectorX:=", "-0.5mm",
    "TranslateVectorY:=", "0.1mm",
    "TranslateVectorZ:=", "0mm"
])
```

VB Syntax	Move <SelectionsArray> <TranslateParameters>
<b>VB Example</b>	<pre>oEditor.Move Array("NAME:Selections",     "Selections:=", "Box1_1",     "NewPartsModelFlag:=", "Model") Array("NAME:TranslateParameters",     "TranslateVectorX:=", "-0.5mm",     "TranslateVectorY:=", "0.1mm",     "TranslateVectorZ:=", "0mm")</pre>

## OffsetFaces

Offsets the faces of selected part(s).

<b>UI Access</b>	<b>Edit &gt; Arrange &gt; Offset.</b>		
<b>Parameters</b>	Name <i>&lt;SelectionsArray&gt;</i>	Type Array	Description Structured array. See: <a href="#">SelectionsArray</a> .
	<i>&lt;OffsetParameters&gt;</i>	Array	Structured array.  Array ("NAME:OffsetParameters", "OffsetDistance:=" , <string>)
<b>Return Value</b>	None.		

<b>Python Syntax</b>	OffsetFaces (<SelectionsArray>, <OffsetParameters>)
<b>Python Example</b>	<pre> oEditor.OffsetFaces (     ["NAME:Selections",         "Selections:=", "Box1_1",         "NewPartsModelFlag:=", "Model"     ],     ["NAME:OffsetParameters",         "OffsetDistance:=", "16mm"     ] ) </pre>

<b>VB Syntax</b>	OffsetFaces <SelectionsArray> <OffsetParameters>
------------------	--------------------------------------------------

**VB Example**

```
oEditor.OffsetFaces  
Array("NAME:Selections",  
      "Selections:=", "Box1_1",  
      "NewPartsModelFlag:=", "Model")  
  
Array("NAME:OffsetParameters",  
      "OffsetDistance:=", "16mm")
```

## Paste (Model Editor)

Pastes previously copied object(s). See: [Copy](#).

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

<b>Python Syntax</b>	Paste()
<b>Python Example</b>	<pre>oEditor.Copy( [ "NAME:Selections",   "Selections:=" , "Box1_2" ]) oEditor.Paste()</pre>

<b>VB Syntax</b>	Paste
<b>VB Example</b>	<pre>oEditor.Copy Array("NAME:Selections", "Selections:=" , "Box1_2") oEditor.Paste</pre>

## Rotate

Rotates specified object(s).

<b>UI Access</b>	<b>Edit &gt; Arrange &gt; Rotate.</b>		
<b>Parameters</b>	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: <a href="#">SelectionsArray</a> .
	<RotateParameters>	Array	Structured array.  Array ("NAME:RotateParameters", "RotateAxis:=" , <string "X", "Y", or "Z">, "RotateAngle:=" , <string>)
<b>Return Value</b>	None.		

<b>Python Syntax</b>	Rotate(<SelectionsArray>, <RotateParameters>)
<b>Python Example</b>	<pre>oEditor.Rotate(     ["NAME:Selections",         "Selections:=", "Box1_1",         "NewPartsModelFlag:=", "Model"     ],     ["NAME:RotateParameters", </pre>

```

    "RotateAxis:=", "Z",
    "RotateAngle:=", "90deg"
]
)

```

<b>VB Syntax</b>	Rotate <SelectionsArray> <RotateParameters>
<b>VB Example</b>	<pre> oEditor.Rotate  Array("NAME:Selections",       "Selections:=", "Box1_1",       "NewPartsModelFlag:=", "Model")  Array("NAME:RotateParameters",       "RotateAxis:=", "Z",       "RotateAngle:=", "90deg") </pre>

## Scale

Scales specified object(s).

<b>UI Access</b>	Edit > Scale.									
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;SelectionsArray&gt;</td> <td>Array</td> <td>Structured array. See: <a href="#">SelectionsArray</a>.</td> </tr> <tr> <td>&lt;ScaleParameters&gt;</td> <td>Array</td> <td>Structured array.    Array ("NAME:ScaleParameters",       "ScaleX:=" , &lt;string containing scale factor&gt;, </td> </tr> </tbody> </table>	Name	Type	Description	<SelectionsArray>	Array	Structured array. See: <a href="#">SelectionsArray</a> .	<ScaleParameters>	Array	Structured array.  Array ("NAME:ScaleParameters",       "ScaleX:=" , <string containing scale factor>,
Name	Type	Description								
<SelectionsArray>	Array	Structured array. See: <a href="#">SelectionsArray</a> .								
<ScaleParameters>	Array	Structured array.  Array ("NAME:ScaleParameters",       "ScaleX:=" , <string containing scale factor>,								

			"ScaleY:=" , <string containing scale factor>, "ScaleZ:=" , <string containing scale factor>)
<b>Return Value</b>	None.		

<b>Python Syntax</b>	Scale (<SelectionsArray>, <ScaleParameters>)
<b>Python Example</b>	<pre> oEditor.Scale(     [ "NAME:Selections",         "Selections:=", "Box1",         "NewPartsModelFlag:=", "Model"     ],     [ "NAME:ScaleParameters",         "ScaleX:=", "2",         "ScaleY:=", "2",         "ScaleZ:=", "2"     ] ) </pre>

<b>VB Syntax</b>	Scale <SelectionsArray> <ScaleParameters>
<b>VB Example</b>	<pre> oEditor.Scale     Array("NAME:Selections",         "Selections:=", "Box1",         "NewPartsModelFlag:=", "Model") ) </pre>

	Array("NAME:ScaleParameters", "ScaleX:=", "2", "ScaleY:=", "2", "ScaleZ:=", "2")
--	-------------------------------------------------------------------------------------------

## Modeler Menu Commands

[AssignMaterial](#)

[Chamfer](#)

[Connect](#)

[CoverLines](#)

[CoverSurfaces](#)

[CreateEntityList](#)

[CreateFaceCS](#)

[CreateGroup](#)

[CreateObjectCS](#)

[CreateObjectFromEdges](#)

[CreateObjectFromFaces](#)

[CreateRelativeCS](#)

[DeleteEmptyGroups](#)

[DeleteLastOperation](#)

[DetachFaces](#)

[EditEntityList](#)  
[EditFaceCS](#)  
[EditObjectCS](#)  
[EditRelativeCS](#)  
[Export](#)  
[ExportModelImageToFile](#)  
[ExportModelMeshToFile](#)  
[Fillet](#)  
[FlattenGroup](#)  
[Generate History](#)  
[GetActiveCoordinateSystem](#)  
[GetCoordinateSystems](#)  
[HealObject](#)  
[Import](#)  
[ImportDXF](#)  
[ImportGDSII \[Modeler\]](#)  
[Intersect](#)  
[MoveCSToEnd](#)  
[MoveEntityToGroup](#)  
[MoveFaces](#)  
[ProjectSheet](#)

---

[PurgeHistory](#)[ReplaceWith3DComponent](#)[Section](#)[SeparateBody](#)[SetModelUnits](#)[SetWCS](#)[ShowWindow](#)[Split](#)[Subtract](#)[SweepFacesAlongNormal](#)[ThickenSheet](#)[UncoverFaces](#)[Unite](#)[Ungroup](#)[WrapSheet](#)

## **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
	< <i>argFaceList</i> >	Array	[ "NAME:<SpecifiedName>",

---

		"BaseFaces:=" , <FaceNumberArray>, "SnapFaces:=" , <FaceNumberArray>]
	<FaceNumberArray>	Array Array of number represents specified faces.
<b>Return Value</b>	None.	

<b>Python Syntax</b>	AlignFaces(<argFaceList>)
<b>Python Example</b>	<pre> oEditor.AlignFaces (     "NAME:Entity List",     "BaseFaces:=" , [ 9 ],     "SnapFaces:=" , [ 18 ] ) </pre>

<b>VB Syntax</b>	AlignFaces <argFaceList>
<b>VB Example</b>	<pre> oEditor.AlignFaces     Array("NAME:Entity List",         "BaseFaces:=" , [ 9 ],         "SnapFaces:=" , [ 18 ]) ) </pre>

## AssignMaterial

Assigns a material to specified object(s).

<b>UI Access</b>	<b>Modeler &gt; Assign Material.</b>
------------------	--------------------------------------

	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: <a href="#">SelectionsArray</a> .
	<AttributesArray>	Array	Structured array. See: <a href="#">AttributesArray</a> .
<b>Parameters</b>			This script supports the following attributes: <ul style="list-style-type: none"><li>• MaterialValue</li><li>• SolveInside</li><li>• ShellElement</li><li>• ShellElementThickness</li><li>• IsMaterialEditable</li><li>• UseMaterialAppearance</li><li>• IsLightweight</li></ul>
<b>Return Value</b>	None.		

**Python Syntax**

```
AssignMaterial(<SelectionsArray>, <AttributesArray>)
```

```
oEditor.AssignMaterial(  
    ["NAME:Selections",  
        "AllowRegionDependentPartSelectionForPMLCreation:=", True,  
        "AllowRegionSelectionForPMLCreation:=", True,  
        "Selections:=", "Box1"  
    ],  
    ["NAME:Attributes",  
        "MaterialValue:=", "diamond",  
        "SolveInside:=", False,  
        "ShellElement:=", False,  
        "ShellElementThickness:=", "nan",  
        "IsMaterialEditable:=", True,  
        "UseMaterialAppearance:=", False,  
        "IsLightweight:=", False  
    ])
```

**VB Syntax**

AssignMaterial &lt;SelectionsArray&gt; &lt;AttributesArray&gt;

**VB Example**

```

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)

```

**Chamfer**

Creates a chamfer.

UI Access	Modeler > Chamfer.		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: <a href="#">SelectionsArray</a> .
	<Parameters>	Array	Structured array.

		<pre>Array("NAME:Parameters",       Array("NAME:ChamferParameters",             "Edges:=" , &lt;array containing integer&gt;,             "Vertices:=" , &lt;array&gt;,             "LeftDistance:=" , &lt;string&gt;,             "RightDistance:=" , &lt;string&gt;,             "ChamferType:=" , &lt;string "Symmetric", "Left Distance-Angle", "Right Distance-Angle", or "Left Distance-Right Distance")       )</pre>
<b>Return Value</b>	None.	

<b>Python Syntax</b>	Chamfer(<SelectionsArray>, <Parameters>)
<b>Python Example</b>	<pre>oEditor.Chamfer(   ["NAME:Selections",    "Selections:="           , "Box2",    "NewPartsModelFlag:="     , "Model"   ],   ["NAME:Parameters",    ["NAME:ChamferParameters",     "Edges:="                 , [42],     "Vertices:="               , []]</pre>

```
"LeftDistance:="           , "0.1mm",
"RightDistance:="          , "0.1mm",
"ChamferType:="            , "Symmetric"
]
])
```

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.

---

<b>UI Access</b>	<b>Modeler &gt; Cleanup Model History.</b>
<b>Parameters</b>	None.
<b>Return Value</b>	None.

<b>Python Syntax</b>	CleanUpModel()
<b>Python Example</b>	<code>oEditor.CleanUpModel ()</code>

<b>VB Syntax</b>	CleanUpModel
<b>VB Example</b>	<code>oEditor.CleanUpModel</code>

## Connect

Connects two or more 1D polyline objects or 2D sheet objects.

<b>UI Access</b>	<b>Modeler &gt; Surface &gt; Connect.</b>		
<b>Parameters</b>	Name <code>&lt;SelectionsArray&gt;</code>	Type Array	Description Structured array. See: <a href="#">SelectionsArray</a> .
<b>Return Value</b>	None.		

<b>Python Syntax</b>	Connect( <code>&lt;SelectionsArray&gt;</code> )
<b>Python Example</b>	<code>oEditor.Connect(</code> <code>["NAME:Selections",</code> <code>"Selections:=", "Polyline2,Polyline1"])</code>

<b>VB Syntax</b>	Connect <SelectionsArray>
<b>VB Example</b>	<pre>oEditor.Connect Array(     "NAME:Selections",     "Selections:=", "Polyline2,Polyline1")</pre>

## CoverLines

Covers two or more 1D objects to form a sheet.

<b>UI Access</b>	<b>Modeler &gt; Surface &gt; Cover Lines.</b>		
<b>Parameters</b>	Name <SelectionsArray>	Type Array	Description Structured array. See: <a href="#">SelectionsArray</a> .
<b>Return Value</b>	None.		

<b>Python Syntax</b>	CoverLines(<SelectionsArray>)
<b>Python Example</b>	<pre>oEditor.CoverLines( [     "NAME:Selections",     "Selections:=", "Polyline3,Polyline4",     "NewPartsModelFlag:=", "Model"] )</pre>

<b>VB Syntax</b>	CoverLines <SelectionsArray>
------------------	------------------------------

<b>VB Example</b>	<pre> oEditor.CoverLines Array(     "NAME:Selections",     "Selections:=", "Polyline3,Polyline4",     "NewPartsModelFlag:=", "Model") </pre>
-------------------	----------------------------------------------------------------------------------------------------------------------------------------------

## CoverSurfaces

Covers two or more faces to form a solid object.

<b>UI Access</b>	<b>Modeler &gt; Surface &gt; Cover Faces.</b>						
<b>Parameters</b>	<table border="1" data-bbox="443 621 1896 719"> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> <tr> <td>&lt;SelectionsArray&gt;</td> <td>Array</td> <td>Structured array. See: <a href="#">SelectionsArray</a>.</td> </tr> </table>	Name	Type	Description	<SelectionsArray>	Array	Structured array. See: <a href="#">SelectionsArray</a> .
Name	Type	Description					
<SelectionsArray>	Array	Structured array. See: <a href="#">SelectionsArray</a> .					
<b>Return Value</b>	None.						

<b>Python Syntax</b>	CoverSurfaces (<SelectionsArray>)
<b>Python Example</b>	<pre> oEditor.CoverSurfaces (     ["NAME:Selections",         "Selections:=", "Obj1_Face1,Obj2_Face2",         "NewPartsModelFlag:=", "Model"     ] ) </pre>

<b>VB Syntax</b>	CoverSurfaces <SelectionsArray>
<b>VB Example</b>	<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.		
<b>Parameters</b>	Name <i>&lt;Parameters&gt;</i>	Type Array	Description Structured array.  Array ("NAME:GeometryEntityListParameters", "EntityType:=", <string "Object" or "Face">, "EntityList:=", <string of object names or IDs> See <a href="#">GetObjectIDByName</a> for returning object IDs.
	<i>&lt;AttributesArray&gt;</i>	Array	Structured array. See: <a href="#">AttributesArray</a> . CreateEntityList takes only the "Name" parameter.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	CreateEntityList(<Parameters>,<AttributesArray>)
<b>Python Example</b>	<pre>oEditor.CreateEntityList( ["NAME:GeometryEntityListParameters", "EntityType:=", "Object",</pre>

```

    "EntityList:=", "Bondwire1,Bondwire2"
],
[ "NAME:Attributes",
  "Name:=", "Objectlist1"
]
)

```

<b>VB Syntax</b>	CreateEntityList<Parameters> <AttributesArray>
<b>VB Example</b>	<pre> oEditor.CreateEntityList Array(   "NAME:GeometryEntityListParameters",   "EntityType:=", "Object",   "EntityList:=", "Bondwire1,Bondwire2")   Array("NAME:Attributes",   "Name:=", "Objectlist1") </pre>

## CreateFaceCS

Creates a Face Coordinate System from a selected face.

<b>UI Access</b>	<b>Modeler &gt; Coordinate System &gt; Create &gt; FaceCS.</b>								
<b>Parameters</b>	<table border="1"> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> <tr> <td>&lt;Parameters&gt;</td> <td>Array</td> <td>           Structured array.            Array ("NAME:FaceCSParameters",           &lt;OriginArray&gt;,           "MoveToEnd:=", &lt;boolean&gt;,         </td> </tr> </table>	Name	Type	Description	<Parameters>	Array	Structured array. Array ("NAME:FaceCSParameters",           <OriginArray>,           "MoveToEnd:=", <boolean>,		
Name	Type	Description							
<Parameters>	Array	Structured array. Array ("NAME:FaceCSParameters",           <OriginArray>,           "MoveToEnd:=", <boolean>,							

			<pre> "FaceID:=" , &lt;integer&gt;, &lt;AxisPosnArray&gt;, "WhichAxis:=" , &lt;string "X", "Y", or "Z"&gt;, "ZRotationAngle:=" , &lt;string&gt;, "XOffset:=" , &lt;string&gt;, "YOffset:=" , &lt;string&gt;, "AutoAxis:=" , &lt;boolean&gt; </pre>
<OriginArray>	Array	Structured array.	<p>Array ("NAME:Origin",</p> <pre> "IsAttachedToEntity:=" , &lt;boolean&gt;, "EntityID:=" , &lt;integer&gt;, "FacetedBodyTriangleIndex:=" , &lt;integer&gt;, "TriangleVertexIndex:=" , &lt;integer&gt;, "PositionType:=" , &lt;string "FaceCenter", "EdgeCenter", "OnVertex", "OnEdge", or "OnFace"&gt;, "UParam:=" , &lt;float between 0 and 1 representing the relative position of the point on the edge or face&gt;, "VParam:=" , &lt;float between 0 and 1 representing the relative position of the point on the edge or face&gt;, "XPosition:=" , &lt;string&gt;, </pre>

		<pre>"YPosition:=" , &lt;string&gt;, "ZPosition:=" , &lt;string&gt;)  IsAttachedToEntity specifies whether the point is anchored to a vertex, edge, or face. If True, provide UParam and VParam. If False, provide XPosition, YPosition, and ZPosition to provide fixed position. Pass "0" for unused parameters.</pre>
	<AxisPosnArray>	<p>Structured array.</p> <pre>Array("NAME:AxisPosn",       "IsAttachedToEntity:=" , &lt;boolean&gt;,       "EntityID:=" , &lt;integer&gt;,       "FacetedBodyTriangleIndex:=" , &lt;integer&gt;,       "TriangleVertexIndex:=" , &lt;integer&gt;,       "PositionType:=" , &lt;string "FaceCenter", "EdgeCenter", "OnVertex", "OnEdge", or "OnFace"&gt;,       "UParam:=" , &lt;float&gt;,       "VParam:=" , &lt;float&gt;,       "XPosition:=" , &lt;string&gt;,       "YPosition:=" , &lt;string&gt;,       "ZPosition:=" , &lt;string&gt;)</pre>
	<AttributesArray>	Structured array. See: <a href="#">AttributesArray</a> .
<b>Return Value</b>	None.	

<b>Python Syntax</b>	CreateFaceCS(<Parameters>,<AttributesArray>)
----------------------	----------------------------------------------

**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"
])

```

<b>VB Syntax</b>	CreateFaceCS <Parameters> <AttributesArray>
<b>VB</b>	oEditor.CreateFaceCS Array("NAME:FaceCSParameters", Array("NAME:Origin", "IsAt-

<b>Example</b> <pre>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 <Parameters>	Type Array	Description Structured array.  Array ("NAME:GroupParameter", "ParentGroupID:=" , <string>, "Parts:=" , <string>,

			"SubmodelInstances:=" , <string>, "Groups:=" , <string>)
<b>Return Value</b>	None.		

<b>Python Syntax</b>	CreateGroup(<Parameters>)
<b>Python Example</b>	<pre>oEditor.CreateGroup (     [ "NAME:GroupParameter",         "ParentGroupID:=" , "Model",         "Parts:=" , "Box1,Box2,Box3",         "SubmodelInstances:=" , "", ,         "Groups:=" , ""     ] )</pre>

<b>VB Syntax</b>	CreateGroup <Parameters>
<b>VB Example</b>	<pre>oEditor.CreateGroup Array("NAME:GroupParameter", "ParentGroupID:=", "Model", _     "Parts:=", "Box1,Box2,Box3", "SubmodelInstances:=", "", "Groups:=", "")</pre>

## CreateObjectCS

Creates an object coordinate system from a selected object.

<b>UI Access</b>	Modeler > Coordinate System > Create > Object > [Offset / Rotated / Both].
------------------	----------------------------------------------------------------------------

	Name	Type	Description
<b>Parameters</b>	<code>&lt;Parameters&gt;</code>	Array	<p>Structured array.</p> <pre>Array ("NAME:ObjectCSParameters",       &lt;OriginArray&gt;       "MoveToEnd:=" , &lt;boolean&gt;,       "ReverseXAxis:=" , &lt;boolean&gt;,       "ReverseYAxis:=" , &lt;boolean&gt;,       &lt;xAxisArray / xAxisPosArray&gt;       &lt;yAxisArray / yAxisPosArray&gt;)</pre> <p><b>Note:</b> <code>xAxisArray</code> and <code>xAxisPosArray</code> differ. Use <code>xAxisArray</code> for absolute position and <code>xAxisPosArray</code> for relative position. Do the same for <code>yAxisArray</code> and <code>yAxisPosArray</code>.</p>
	<code>&lt;OriginArray&gt;</code>	Array	<p>Structured array.</p> <pre>Array ("NAME:Origin",       "IsAttachedToEntity:=" , &lt;boolean&gt;,       "EntityID:=" , &lt;integer&gt;,       "FacetedBodyTriangleIndex:=" , &lt;integer&gt;,       "TriangleVertexIndex:=" , &lt;integer&gt;,       "PositionType:=" , &lt;string "OnVertex",       "EdgeCenter", "FaceCenter", "OnEdge", or "AbsolutePosition"&gt;,       "UParam:=" , &lt;float between 0 and 1 representing       the relative position of the point on the edge or</pre>

		<pre>face&gt;,     "VParam:=" , &lt;float between 0 and 1 representing     the relative position of the point on the edge or     face&gt;,     "XPosition:=" , &lt;string&gt;,     "YPosition:=" , &lt;string&gt;,     "ZPosition:=" , &lt;string&gt;)</pre> <p>IsAttachedToEntity specifies whether the point is anchored. If True, provide UParam and VParam. If False, provide XPosition, YPosition, and ZPosition to provide fixed position. Pass "0" for unused parameters.</p>
<xAxisArray>	Array	<p>Structured array for absolute position:</p> <pre>Array ("NAME:xAxis",     "DirectionType:=" , "AbsoluteDirection",     "EdgeID:=" , &lt;integer&gt;,     "FaceID:=" , &lt;integer&gt;,     "xDirection:=" , &lt;string&gt;,     "yDirection:=" , &lt;string&gt;,     "zDirection:=" , &lt;string&gt;,     "UParam:=" , &lt;float&gt;,     "VParam:=" , &lt;float&gt;)</pre>
<xAxisPosArray>	Array	<p>Structured array for relative position:</p> <pre>Array ("NAME:xAxisPos",     "IsAttachedToEntity:=" , &lt;boolean&gt;,</pre>

			<pre> "EntityID:=" , &lt;integer&gt;, "FacetedBodyTriangleIndex:=" , &lt;integer&gt;, "TriangleVertexIndex:=" , &lt;integer&gt;, "PositionType:=" , &lt;string "OnVertex", "EdgeCenter", "FaceCenter", or "OnEdge"&gt;, "UParam:=" , &lt;float&gt;, "VParam:=" , &lt;float&gt;, "XPosition:=" , &lt;string&gt;, "YPosition:=" , &lt;string&gt;, "ZPosition:=" , &lt;string&gt; </pre>
<yAxisArray>	Array	Structured array for absolute position:	<pre> Array ("NAME:yAxis", "DirectionType:=" , "AbsoluteDirection", "EdgeID:=" , &lt;integer&gt;, "FaceID:=" , &lt;integer&gt;, "xDirection:=" , &lt;string&gt;, &gt;yDirection:=" , &lt;string&gt;, "zDirection:=" , &lt;string&gt;, "UParam:=" , &lt;float&gt;, "VParam:=" , &lt;float&gt; </pre>
<yAxisPosArray>	Array	Structured array for relative position:	

		<pre>Array("NAME:yAxisPos",       "IsAttachedToEntity:=" , &lt;boolean&gt;,       "EntityID:=" , &lt;integer&gt;,       "FacetedBodyTriangleIndex:=" , &lt;integer&gt;,       "TriangleVertexIndex:=" , &lt;integer&gt;,       "PositionType:=" , &lt;string "OnVertex", "EdgeCenter", "FaceCenter", or "OnEdge"&gt;,       "UParam:=" , &lt;float&gt;,       "VParam:=" , &lt;float&gt;,       "XPosition:=" , &lt;string&gt;,       "YPosition:=" , &lt;string&gt;,       "ZPosition:=" , &lt;string&gt;)</pre>
	<AttributesArray>	Array Structured array. See: <a href="#">AttributesArray</a> .
<b>Return Value</b>	None.	

<b>Python Syntax</b>	CreateObjectCS(<Parameters>,<AttributesArray>)
<b>Python Example</b>	<pre>oEditor.CreateObjectCS(   [ "NAME:ObjectCSPARAMETERS",     [ "NAME:Origin",       "IsAttachedToEntity:=" , True,       "EntityID:=" , 59,       "FacetedBodyTriangleIndex:=" , -1,</pre>

```
    "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"
            ]
        )
    )
]

```

<b>VB Syntax</b>	CreateObjectCS <Parameters> <AttributesArray>
<b>VB Example</b>	<pre> oEditor.CreateObjectCS Array("NAME:ObjectCSParameters", Array("NAME:Origin", "IsAttachedToEntity:=", _  false, "EntityID:=", -1, "FacetedBodyTriangleIndex:=", -1, "TriangleVertexIndex:=", _</pre>

```

-1, "PositionType:=", "AbsolutePosition", "UParam:=", 0, "VParam:=", 0, "XPosition:=",
-
"0mm", "YPosition:=", "0mm", "ZPosition:=", "0mm"), "MoveToEnd:=", false, "ReverseXAxis:=",
-
false, "ReverseYAxis:=", false, Array("NAME:xAxisPos", "IsAttachedToEntity:=", true,
"EntityID:=", -
80, "FacetedBodyTriangleIndex:=", -1, "TriangleVertexIndex:=", -1, "PositionType:=",
-
"EdgeCenter", "UParam:=", 0, "VParam:=", 0, "XPosition:=", "0", "YPosition:=",
-
"0", "ZPosition:=", "0"), Array("NAME:yAxisPos", "IsAttachedToEntity:=", true,
"EntityID:=", -
69, "FacetedBodyTriangleIndex:=", -1, "TriangleVertexIndex:=", -1, "PositionType:=",
-
"EdgeCenter", "UParam:=", 0, "VParam:=", 0, "XPosition:=", "0", "YPosition:=",
-
"0", "ZPosition:=", "0)), Array("NAME:Attributes", "Name:=", "ObjectCS2",
"PartName:=", -
"Box3")

```

## CreateObjectFromEdges

Creates an object from the specified object edge.

UI Access	Modeler > Edge > Create Object From Edge		
Parameters	Name <SelectionsArray>	Type Array	Description Structured array. See: <a href="#">SelectionsArray</a> .
	<ParametersArray>	Array	Structured array. Array ("NAME:Parameters",

			<pre>        Array ("NAME:BodyFromEdgeToParameters",                 "Edges:=", &lt;array containing                 integer edges&gt;             )         )     ) &lt;CreateGroupsForNewObjects&gt;</pre>
	<CreateGroupsForNewObjects>	Array	<p>Structured array.</p> <pre>Array ("CreateGroupsForNewObjects:=", &lt;boolean True to create groups for new objects; else False&gt;)</pre>
<b>Return Value</b>	None.		

<b>Python Syntax</b>	CreateObjectFromEdges(<SelectionsArray>, <ParametersArray>, <CreateGroupsForNewObjects>)
<b>Python Example</b>	<pre>oEditor.CreateObjectFromEdges (     [         ["NAME:Selections",          "Selections:=", "Box2",          "NewPartsModelFlag:=", "Model"         ],         ["NAME:Parameters",          [              ["NAME:BodyFromEdgeToParameters",               "Edges:=", [41]             ],             ...         ],         ...     ],     ... )</pre>

```
[ "CreateGroupsForNewObjects:=", False
])
```

<b>VB Syntax</b>	CreateObjectFromEdges <SelectionsArray> <ParametersArray> <CreateGroupsForNewObjects>
<b>VB Example</b>	<pre>oEditor.CreateObjectFromEdges Array("NAME:Selections", "Selections:=", "Box1", "NewPartsModelFlag:=", "Model"), Array("NAME:Parameters", Array("NAME:BodyFromEdgeToParameters", "Edges:=", Array( 13))), Array("CreateGroupsForNewObjects:=", false)</pre>

## CreateObjectFromFace

Creates 2D objects from specified face(s).

<b>UI Access</b>	N/A		
<b>Parameters</b>	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: <a href="#">SelectionsArray</a> .
	<Parameters>	Array	Structured array. Array ("NAME:Parameters", <BodyFromFaceToParameters>)
	<CreateGroupsForNewObjects>	Array	Optional. Structured array. Array ("CreateGroupsForNewObjects:=", <boolean>)
<b>Return Value</b>	None.		

<b>Python Syntax</b>	CreateObjectFromFace (<SelectionsArray>,<Parameters>, <CreateGroupsForNewObjects>)
<b>Python Example</b>	<pre> oEditor.CreateObjectFromFace(     [         "NAME:Selections",         "Selections:=" , "Box3",         "NewPartsModelFlag:=" , "Model"     ],     [         "NAME:Parameters",         [             "NAME:BodyFromFaceToParameters",             "FacesToDetach:=" , [68]         ]     ],     [         "CreateGroupsForNewObjects:=", False     ] ) </pre>

<b>VB Syntax</b>	CreateObjectFromFace <SelectionsArray> <Parameters> <CreateGroupsForNewObjects>
<b>VB Example</b>	<pre> oEditor.CreateObjectFromFace Array("NAME:Selections", "Selections:=", "Box3",     "NewPartsModelFlag:=" ,     "Model") Array("NAME:Parameters", Array("NAME:BodyFromFaceToParameters",     "FacesToDetach:=", Array(68))) Array("CreateGroupsForNewObjects:=", false) </pre>

## CreateObjectFromFaces

Creates 2D objects from specified face(s).

UI Access	Modeler > Surface > Create Object from Face		
<b>Parameters</b>	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: <a href="#">SelectionsArray</a> .
	<Parameters>	Array	Structured array. Array ("NAME:Parameters", <BodyFromFaceToParameters>)
	<CreateGroupsForNewObjects>	Array	Structured array. Array ("CreateGroupsForNewObjects:=", <boolean>)
Return Value	None.		

Python Syntax	CreateObjectFromFaces (<SelectionsArray>,<Parameters>,<CreateGroupsForNewObjects>)
<b>Python Example</b>	<pre>oEditor.CreateObjectFromFaces (     [         ["NAME:Selections",          "Selections:="           , "Box3",          "NewPartsModelFlag:="     , "Model"         ],         ["NAME:Parameters",          ["NAME:BodyFromFaceToParameters", </pre>

```

        "FacesToDetach:="           , [68]
    ]
],
["CreateGroupsForNewObjects:=", False
])

```

<b>VB Syntax</b>	CreateObjectFromFaces <SelectionsArray> <Parameters> <CreateGroupsForNewObjects>
<b>VB Example</b>	<pre> oEditor.CreateObjectFromFaces Array("NAME:Selections", "Selections:=", "Box3", "NewPartsModelFlag:=" , "Model") Array("NAME:Parameters", Array("NAME:BodyFromFaceToParameters", "FacesToDetach:=", Array(68)))  Array("CreateGroupsForNewObjects:=", False) </pre>

## CreateRelativeCS

Creates a Relative Coordinate System.

UI Access	Modeler > Coordinate System > Create > Relative CS > [Offset / Rotated / Both].								
Parameters	<table border="1"> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> <tr> <td>&lt;Parameters&gt;</td> <td>Array</td> <td> <p>Structured array.</p> <p>Array ("NAME:RelativeCSPParameters",  "Mode:=" , "Axis/Position",  "OriginX:=" , &lt;string&gt;,  "OriginY:=" , &lt;string&gt;,</p> </td> </tr> </table>	Name	Type	Description	<Parameters>	Array	<p>Structured array.</p> <p>Array ("NAME:RelativeCSPParameters",  "Mode:=" , "Axis/Position",  "OriginX:=" , &lt;string&gt;,  "OriginY:=" , &lt;string&gt;,</p>		
Name	Type	Description							
<Parameters>	Array	<p>Structured array.</p> <p>Array ("NAME:RelativeCSPParameters",  "Mode:=" , "Axis/Position",  "OriginX:=" , &lt;string&gt;,  "OriginY:=" , &lt;string&gt;,</p>							

			<pre>"OriginZ:=" , &lt;string&gt;, "XAxisXvec:=" , &lt;string&gt;, "XAxisYvec:=" , &lt;string&gt;, "XAxisZvec:=" , &lt;string&gt;, "YAxisXvec:=" , &lt;string&gt;, "YAxisYvec:=" , &lt;string&gt;, "YAxisZvec:=" , &lt;string&gt;)</pre>
	<i>&lt;AttributesArray&gt;</i>	Array	Structured array. See: <a href="#">AttributesArray</a> . CreateRelativeCS supports only the "Name" parameter.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	CreateRelativeCS(<Parameters>,<AttributesArray>)
<b>Python Example</b>	<pre>oEditor.CreateRelativeCS(     [         "NAME:RelativeCSParameters",         "Mode:=" , "Axis/Position",         "OriginX:=" , "0.62mm",         "OriginY:=" , "-0.7mm",         "OriginZ:=" , "0mm",         "XAxisXvec:=" , "1mm",         "XAxisYvec:=" , "0mm",</pre>

```

    "XAxisZvec:="           , "0mm",
    "YAxisXvec:="           , "0mm",
    "YAxisYvec:="           , "1mm",
    "YAxisZvec:="           , "0mm"
],
[ "NAME:Attributes",
  "Name:="                 , "RelativeCS1"
]
)

```

<b>VB Syntax</b>	CreateRelativeCS <Parameters> <AttributesArray>
<b>VB Example</b>	<pre> oEditor.CreateRelativeCS Array("NAME:RelativeCSParameters", "Mode:=", "Axis/Position", "OriginX:=", _ "0mm", "OriginY:=", "0mm", "OriginZ:=", "0mm", "XAxisXvec:=", "0.66mm", "XAxisYvec:=", - "0.28mm", "XAxisZvec:=", "0mm", "YAxisXvec:=", "0.06mm", "YAxisYvec:=", "0.14mm", "YAxisZvec:=", _ "0mm"), Array("NAME:Attributes", "Name:=", "RelativeCS1") </pre>

## DeleteEmptyGroups

Deletes group(s) from the history tree.

<b>UI Access</b>	<b>Modeler &gt; Group &gt; Delete Empty.</b>		
<b>Parameters</b>	Name	Type	Description

	<i>&lt;Parameters&gt;</i>	Array	Structured array. Array("Groups:=", <array of string group IDs>)
<b>Return Value</b>	None.		

<b>Python Syntax</b>	DeleteEmptyGroups( <i>&lt;Parameters&gt;</i> )
<b>Python Example</b>	<pre>oEditor.DeleteEmptyGroups ([     "Groups:=", ["Group1", "Group2", "Group3"] ])</pre>

<b>VB Syntax</b>	DeleteEmptyGroups <Parameters>
<b>VB Example</b>	<pre>oEditor.DeleteEmptyGroups Array("Groups:=", Array("Group1", "Group2", "Group3"))</pre>

## DeleteLastOperation

Deletes the last operation performed on the specified object(s).

<b>UI Access</b>	Modeler > Delete Last Operation.		
<b>Parameters</b>	Name	Type	Description <i>&lt;SelectionsArray&gt;</i> Structured array. See: <a href="#">SelectionsArray</a> .
<b>Return Value</b>	None.		

<b>Python Syntax</b>	DeleteLastOperation( <i>&lt;SelectionsArray&gt;</i> )
----------------------	-------------------------------------------------------

<b>Python Example</b>	<pre><code>oEditor.DeleteLastOperation([     "NAME:Selections",     "Selections:=", "Box3",     "NewPartsModelFlag:=", "Model" ])</code></pre>
-----------------------	------------------------------------------------------------------------------------------------------------------------------------------------

<b>VB Syntax</b>	<code>DeleteLastOperation &lt;SelectionsArray&gt;</code>
<b>VB Example</b>	<code>oEditor.DeleteLastOperation Array("NAME:Selections", "Selections:=", "Box3", "NewPartsModelFlag:=", "Model")</code>

## DeleteOperation

Deletes specified operation performed on a selected object.

<b>UI Access</b>	Select an operation in the project tree, then press <b>Delete</b> on the keyboard.		
<b>Parameters</b>	Name <code>&lt;ParamsArray&gt;</code>	Type Array	Description Structured array.  <code>Array("NAME:Parameters",        Array("NAME:PartOperations",              Array("NAME:&lt;PartName&gt;",                    "OperationIndices:=", &lt;array of operation                    indices&gt;),              Array("NAME:UDMOperations")))</code>

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

<b>Python Syntax</b>	DeleteOperation(<ParamsArray>)
<b>Python Example</b>	<pre>oEditor.DeleteOperation( [     "NAME:Parameters",     [         "NAME:PartOperations",         [             "NAME:Coil_0",             "OperationIndices:=", [1]         ]     ],     ["NAME:UDMOperations"] ])</pre>

<b>VB Syntax</b>	DeleteOperation <ParamsArray>
<b>VB Example</b>	<pre>oEditor.DeleteOperation _     Array("NAME:Parameters", _</pre>

```
Array ("NAME:PartOperations", _
      Array ("NAME:Coil_0", _
             "OperationIndices:=", Array(1))), _
      Array ("NAME:UDMOperations"))
```

## DetachEdges

Detaches the specified edge(s) from an object.

UI Access	Modeler > Edge > Detach Edges.		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: <a href="#">SelectionsArray</a> .
	<Parameters>	Array	Structured array.  Array ("NAME:Parameters",  <DetachEdgesArray>)
<DetachEdgesArray>	Array	Structured array.  Array ("NAME:DetachEdgesToParameters",  "EdgesToDelete:=" , <array containing integer edge IDs>)	
Return Value	None.		

Python Syntax	DetachEdges(<SelectionsArray>, <Parameters>)
Python Example	<pre>oEditor.DetachEdges (     ["NAME:Selections",      "Selections:=", "Rectangle1",</pre>

```

    "NewPartsModelFlag:=", "Model"
    ],
    [
        "NAME:Parameters",
        [
            "NAME:DetachEdgesToParameters",
            "EdgesToDetach:=", [18,17]
        ]
    ]
)

```

<b>VB Syntax</b>	DetachEdges <SelectionsArray> <Parameters>
<b>VB Example</b>	<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.		
<b>Parameters</b>	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: <a href="#">SelectionsArray</a> .

	<code>&lt;DetachFacesArray&gt;</code>		<code>&lt;DetachFacesArray&gt;)</code>
	<code>&lt;DetachFacesArray&gt;</code>	Array	Structured array.  Array ("NAME:DetachFacesToParameters", "FacesToDetach:=", <array containing integer face IDs>)
<b>Return Value</b>	None.		

<b>Python Syntax</b>	<code>DetachFaces(&lt;SelectionsArray&gt;, &lt;Parameters&gt;)</code>
<b>Python Example</b>	<pre> oEditor.DetachFaces (     [ "NAME:Selections",       "Selections:=", "Box3",       "NewPartsModelFlag:=", "Model"     ],     [ "NAME:Parameters",       [ "NAME:DetachFacesToParameters",         "FacesToDetach:=", [ 68, 67 ]       ]     ] ) </pre>

<b>VB Syntax</b>	<code>DetachFaces &lt;SelectionsArray&gt; &lt;Parameters&gt;</code>
<b>VB</b>	<code>oEditor.DetachFaces Array("NAME:Selections", "Selections:=", "Box3",</code>

<b>Example</b>	<pre>"NewPartsModelFlag:=", "Model") Array("NAME:Parameters", Array("NAME:DetachFacesToParameters", "FacesToDetach:=", [68,67]))</pre>
----------------	----------------------------------------------------------------------------------------------------------------------------------------

## EditEntityList

Modifies an entity list.

UI Access	<b>Modeler &gt; List &gt; Reassign.</b>		
<b>Parameters</b>	Name <i>&lt;SelectionsArray&gt;</i>	Type Array	Description Structured array. See: <a href="#">SelectionsArray</a> .
	<i>&lt;Parameters&gt;</i>	Array	Structured array.  Array ("NAME:GeometryEntityListParameters", "EntityType:=" , <string "Object" or "Face">, "EntityList:=" , <string list> )
<b>Return Value</b>	None.		

<b>Python Syntax</b>	EditEntityList(<SelectionsArray>, <Parameters>)
<b>Python Example</b>	<pre>oEditor.EditEntityList(     [         "NAME:Selections",         "Selections:=" ,         "Objectlist1"     ])</pre>

```

        ],
        [
        "NAME:GeometryEntityListParameters",
        "EntityType:=" , "Object",
        "EntityList:=" , "Box1, Box2, Box3"
    ]
)

```

<b>VB Syntax</b>	EditEntityList <SelectionsArray> <Parameters>
<b>VB Example</b>	<pre> oEditor.EditEntityList Array("NAME:Selections",       "Selections:=", "Objectlist1") Array("NAME:GeometryEntityListParameters",       "EntityType:=", "Object",       "EntityList:=", "Box1, Box2, Box3") </pre>

## EditFaceCS

Recreates an existing face coordinate system. See: [CreateFaceCS](#).

<b>UI Access</b>	<b>Modeler &gt; Coordinate System &gt; Edit.</b>								
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;Parameters&gt;</td> <td>Array</td> <td> <p>Structured array.</p> <pre> Array("NAME:FaceCSParameters",       &lt;OriginArray&gt;,       "MoveToEnd:=", &lt;boolean&gt;,       ... ) </pre> </td> </tr> </tbody> </table>	Name	Type	Description	<Parameters>	Array	<p>Structured array.</p> <pre> Array("NAME:FaceCSParameters",       &lt;OriginArray&gt;,       "MoveToEnd:=", &lt;boolean&gt;,       ... ) </pre>		
Name	Type	Description							
<Parameters>	Array	<p>Structured array.</p> <pre> Array("NAME:FaceCSParameters",       &lt;OriginArray&gt;,       "MoveToEnd:=", &lt;boolean&gt;,       ... ) </pre>							

			<pre> "FaceID:=" , &lt;integer&gt;, &lt;AxisPosnArray&gt;, "WhichAxis:=" , &lt;string "X", "Y", or "Z"&gt;, "ZRotationAngle:=" , &lt;string&gt;, "XOffset:=" , &lt;string&gt;, "YOffset:=" , &lt;string&gt;, "AutoAxis:=" , &lt;boolean&gt; </pre>
	<i>&lt;OriginArray&gt;</i>	Array	<p>Structured array.</p> <pre> Array("NAME:Origin",       "IsAttachedToEntity:=" , &lt;boolean&gt;,       "EntityID:=" , &lt;integer&gt;,       "FacetedBodyTriangleIndex:=" , &lt;integer&gt;,       "TriangleVertexIndex:=" , &lt;integer&gt;,       "PositionType:=" , &lt;string "FaceCenter",       "EdgeCenter", "OnVertex", "OnEdge", or "OnFace"&gt;,       "UParam:=" , &lt;float between 0 and 1 representing       the relative position of the point on the edge or       face&gt;,       "VParam:=" , &lt;float between 0 and 1 representing       the relative position of the point on the edge or       face&gt;,       "XPosition:=" , &lt;string&gt;, </pre>

			<pre>"YPosition:=" , &lt;string&gt;, "ZPosition:=" , &lt;string&gt;)  IsAttachedToEntity specifies whether the point is anchored to a vertex, edge, or face. If True, provide UParam and VParam. If False, provide XPosition, YPosition, and ZPosition to provide fixed position. Pass "0" for unused parameters.</pre>
	<b>&lt;AxisPosnArray&gt;</b>	Array	<p>Structured array.</p> <pre>Array("NAME:AxisPosn",       "IsAttachedToEntity:=" , &lt;boolean&gt;,       "EntityID:=" , &lt;integer&gt;,       "FacetedBodyTriangleIndex:=" , &lt;integer&gt;,       "TriangleVertexIndex:=" , &lt;integer&gt;,       "PositionType:=" , &lt;string "FaceCenter", "EdgeCenter", "OnVertex", "OnEdge", or "OnFace"&gt;,       "UParam:=" , &lt;float&gt;,       "VParam:=" , &lt;float&gt;,       "XPosition:=" , &lt;string&gt;,       "YPosition:=" , &lt;string&gt;,       "ZPosition:=" , &lt;string&gt;)</pre>
	<b>&lt;AttributesArray&gt;</b>	Array	Structured array. See: <a href="#">AttributesArray</a> . Use to select the coordinate system to edit.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	EditFaceCS (<Parameters>, <AttributesArray>)
----------------------	----------------------------------------------

**Python Example**

```
oEditor.EditFaceCS(  
    [ "NAME:FaceCSParameters",  
        [ "NAME:Origin",  
            "IsAttachedToEntity:=" , True,  
            "EntityID:=" , 12,  
            "FacetedBodyTriangleIndex:=" , -1,  
            "TriangleVertexIndex:=" , -1,  
            "PositionType:=" , "FaceCenter",  
            "UParam:=" , 0,  
            "VParam:=" , 0,  
            "XPosition:=" , "0",  
            "YPosition:=" , "0",  
            "ZPosition:=" , "0"  
        ],  
        "MoveToEnd:=" , False,  
        "FaceID:=" , 12,  
        [ "NAME:AxisPosn",  
            "IsAttachedToEntity:=" , True,  
            "EntityID:=" , 12,  
            "FacetedBodyTriangleIndex:=" , -1,
```

```

        "TriangleVertexIndex:=" , -1,
        "PositionType:=" , "OnFace",
        "UParam:=" , 0.62951717774066,
        "VParam:=" , 0.226514925559344,
        "XPosition:=" , "1200mm",
        "YPosition:=" , "-354.697014888131mm",
        "ZPosition:=" , "125.903435548132mm"
    ],
    "WhichAxis:=" , "X",
    "ZRotationAngle:=" , "0deg",
    "XOffset:=" , "0mm",
    "YOffset:=" , "0mm",
    "AutoAxis:=" , False
],
[ "NAME:Attributes",
  "Name:=" , "FaceCS1",
  "PartName:=" , "Box1"
])

```

<b>VB Syntax</b>	EditFaceCS <Parameters> <AttributesArray>
<b>VB Example</b>	oEditor.EditFaceCS

```
Array("NAME:FaceCSParameters",
      Array("NAME:Origin",
            "IsAttachedToEntity:=" , True,
            "EntityID:="           , 12,
            "FacetedBodyTriangleIndex:=" , -1,
            "TriangleVertexIndex:=" , -1,
            "PositionType:="        , "FaceCenter",
            "UParam:="              , 0,
            "VParam:="              , 0,
            "XPosition:="           , "0",
            "YPosition:="           , "0",
            "ZPosition:="           , "0"),
      "MoveToEnd:="           , False,
      "FaceID:="              , 12,
      Array("NAME:AxisPosn",
            "IsAttachedToEntity:=" , True,
            "EntityID:="           , 12,
            "FacetedBodyTriangleIndex:=" , -1,
            "TriangleVertexIndex:=" , -1,
            "PositionType:="        , "OnFace",
```

```

    "UParam:=" , 0.62951717774066,
    "VParam:=" , 0.226514925559344,
    "XPosition:=" , "1200mm",
    "YPosition:=" , "-354.697014888131mm",
    "ZPosition:=" , "125.903435548132mm"),
    "WhichAxis:=" , "X",
    "ZRotationAngle:=" , "0deg",
    "XOffset:=" , "0mm",
    "YOffset:=" , "0mm",
    "AutoAxis:=" , False)

Array("NAME:Attributes",
      "Name:=" , "FaceCS1",
      "PartName:=" , "Box1")

```

## EditObjectCS

Edits an existing object coordinate system. See: [CreateObjectCS](#).

UI Access	Modeler > Coordinate System > Edit.								
Parameters	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;Parameters&gt;</td> <td>Array</td> <td>Structured array. Array ("NAME:ObjectCSParameters",        &lt;OriginArray&gt;        "MoveToEnd:=" , &lt;boolean&gt;,</td> </tr> </tbody> </table>	Name	Type	Description	<Parameters>	Array	Structured array. Array ("NAME:ObjectCSParameters", <OriginArray> "MoveToEnd:=" , <boolean>,		
Name	Type	Description							
<Parameters>	Array	Structured array. Array ("NAME:ObjectCSParameters", <OriginArray> "MoveToEnd:=" , <boolean>,							

			<pre> "ReverseXAxis:=" , &lt;boolean&gt;, "ReverseYAxis:=" , &lt;boolean&gt;, &lt;xAxisArray / xAxisPosArray&gt; &lt;yAxisArray / yAxisPosArray&gt; </pre> <p><b>Note:</b> xAxisArray and xAxisPosArray differ. Use xAxisArray for absolute position and xAxisPosArray for relative position. Do the same for yAxisArray and yAxisPosArray.</p>
	<i>&lt;OriginArray&gt;</i>	Array	<p>Structured array.</p> <pre> Array("NAME:Origin",       "IsAttachedToEntity:=" , &lt;boolean&gt;,       "EntityID:=" , &lt;integer&gt;,       "FacetedBodyTriangleIndex:=" , &lt;integer&gt;,       "TriangleVertexIndex:=" , &lt;integer&gt;,       "PositionType:=" , &lt;string "OnVertex",       "EdgeCenter", "FaceCenter", "OnEdge", or "AbsolutePosition"&gt;,       "UParam:=" , &lt;float between 0 and 1 representing       the relative position of the point on the edge or       face&gt;,       "VParam:=" , &lt;float between 0 and 1 representing       the relative position of the point on the edge or       face&gt;,       "XPosition:=" , &lt;string&gt;,       "YPosition:=" , &lt;string&gt;,       "ZPosition:=" , &lt;string&gt;) </pre>

		<pre>"YPosition:=" , &lt;string&gt;, "ZPosition:=" , &lt;string&gt;)  IsAttachedToEntity specifies whether the point is anchored. If True, provide UParam and VParam. If False, provide XPosition, YPosition, and ZPosition to provide fixed position. Pass "0" for unused parameters.</pre>
<xAxisArray>	Array	<p>Structured array for absolute position:</p> <pre>Array ("NAME:xAxis",       "DirectionType:=" , "AbsoluteDirection",       "EdgeID:=" , &lt;integer&gt;,       "FaceID:=" , &lt;integer&gt;,       "xDirection:=" , &lt;string&gt;,       "yDirection:=" , &lt;string&gt;,       "zDirection:=" , &lt;string&gt;,       "UParam:=" , &lt;float&gt;,       "VParam:=" , &lt;float&gt;)</pre>
<xAxisPosArray>	Array	<p>Structured array for relative position:</p> <pre>Array ("NAME:xAxisPos",       "IsAttachedToEntity:=" , &lt;boolean&gt;,       "EntityID:=" , &lt;integer&gt;,       "FacetedBodyTriangleIndex:=" , &lt;integer&gt;,       "TriangleVertexIndex:=" , &lt;integer&gt;,       "PositionType:=" , &lt;string "OnVertex",       "EdgeCenter", "FaceCenter", or "OnEdge"&gt;,</pre>

			<pre>"UParam:=" , &lt;float&gt;, "VParam:=" , &lt;float&gt;, "XPosition:=" , &lt;string&gt;, "YPosition:=" , &lt;string&gt;, "ZPosition:=" , &lt;string&gt;)</pre>
	<code>&lt;yAxisArray&gt;</code>	Array	<p>Structured array for absolute position:</p> <pre>Array ("NAME:yAxis",       "DirectionType:=" , "AbsoluteDirection",       "EdgeID:=" , &lt;integer&gt;,       "FaceID:=" , &lt;integer&gt;,       "xDirection:=" , &lt;string&gt;,       "yDirection:=" , &lt;string&gt;,       "zDirection:=" , &lt;string&gt;,       "UParam:=" , &lt;float&gt;,       "VParam:=" , &lt;float&gt;)</pre>
	<code>&lt;yAxisPosArray&gt;</code>	Array	<p>Structured array for relative position:</p> <pre>Array ("NAME:yAxisPos",       "IsAttachedToEntity:=" , &lt;boolean&gt;,       "EntityID:=" , &lt;integer&gt;,       "FacetedBodyTriangleIndex:=" , &lt;integer&gt;,       "TriangleVertexIndex:=" , &lt;integer&gt;,</pre>

			<pre>"PositionType:=" , &lt;string "OnVertex", "EdgeCenter", "FaceCenter", or "OnEdge"&gt;, "UParam:=" , &lt;float&gt;, "VParam:=" , &lt;float&gt;, "XPosition:=" , &lt;string&gt;, "YPosition:=" , &lt;string&gt;, "ZPosition:=" , &lt;string&gt;)</pre>
	<b>&lt;AttributesArray&gt;</b>	Array	Structured array. See: <a href="#">AttributesArray</a> . Use to select the coordinate system to edit.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	EditObjectCS(<Parameters>,<AttributesArray>)
<b>Python Example</b>	<pre>oEditor.EditObjectCS( [ "NAME:ObjectCSPARAMETERS",   [ "NAME:Origin",     "IsAttachedToEntity:=" , True,     "EntityID:=" , 59,     "FacetedBodyTriangleIndex:=" , -1,     "TriangleVertexIndex:=" , -1,     "PositionType:=" , "OnVertex",     "UParam:=" , 0,     "VParam:=" , 0,</pre>

```
        "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"
])

```

<b>VB Syntax</b>	EditObjectCS <Parameters> <AttributesArray>
<b>VB Example</b>	<pre> oEditor&gt;EditObjectCS Array("NAME:ObjectCSParameters", Array("NAME:Origin", "IsAttachedToEntity:=", _ false, "EntityID:=", -1, "FacetedBodyTriangleIndex:=", -1, "TriangleVertexIndex:=", _  -1, "PositionType:=", "AbsolutePosition", "UParam:=", 0, "VParam:=", 0, "XPosition:=", _  "0mm", "YPosition:=", "0mm", "ZPosition:=", "0mm"), "MoveToEnd:=", false, "ReverseXAxis:=", _  false, "ReverseYAxis:=", false, Array("NAME:xAxisPos", "IsAttachedToEntity:=", true, </pre>

```

"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")

```

## EditRelativeCS

Edits an existing Relative Coordinate System. See: [CreateRelativeCS](#).

UI Access	Modeler > Coordinate System > Edit.								
Parameters	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;Parameters&gt;</td> <td>Array</td> <td> <p>Structured array.</p> <p>Array ("NAME:RelativeCSPARAMETERS",</p> <p>"Mode:=" , "Axis/Position",</p> <p>"OriginX:=" , &lt;string&gt;,</p> <p>"OriginY:=" , &lt;string&gt;,</p> <p>"OriginZ:=" , &lt;string&gt;,</p> </td></tr> </tbody> </table>	Name	Type	Description	<Parameters>	Array	<p>Structured array.</p> <p>Array ("NAME:RelativeCSPARAMETERS",</p> <p>"Mode:=" , "Axis/Position",</p> <p>"OriginX:=" , &lt;string&gt;,</p> <p>"OriginY:=" , &lt;string&gt;,</p> <p>"OriginZ:=" , &lt;string&gt;,</p>		
Name	Type	Description							
<Parameters>	Array	<p>Structured array.</p> <p>Array ("NAME:RelativeCSPARAMETERS",</p> <p>"Mode:=" , "Axis/Position",</p> <p>"OriginX:=" , &lt;string&gt;,</p> <p>"OriginY:=" , &lt;string&gt;,</p> <p>"OriginZ:=" , &lt;string&gt;,</p>							

			<pre>"XAxisXvec:=", &lt;string&gt;, "XAxisYvec:=", &lt;string&gt;, "XAxisZvec:=", &lt;string&gt;, "YAxisXvec:=", &lt;string&gt;, "YAxisYvec:=", &lt;string&gt;, "YAxisZvec:=", &lt;string&gt;)</pre>
	<b>&lt;AttributesArray&gt;</b>	Array	Structured array. See: <a href="#">AttributesArray</a> . Use to select the coordinate system to edit.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	EditRelativeCS(<Parameters>,<AttributesArray>)
<b>Python Example</b>	<pre>oEditor.EditRelativeCS( ["NAME:RelativeCSParameters",  "Mode:=", "Axis/Position",  "OriginX:=", "0.62mm",  "OriginY:=", "-0.7mm",  "OriginZ:=", "0mm",  "XAxisXvec:=", "1mm",  "XAxisYvec:=", "0mm",  "XAxisZvec:=", "0mm",  "YAxisXvec:=", "0mm",  "YAxisYvec:=", "1mm"])</pre>

```
    "YAxisZvec:="           , "0mm"
],
[ "NAME:Attributes",
  "Name:="                , "RelativeCS1"
])
```

<b>VB Syntax</b>	EditRelativeCS <Parameters> <AttributesArray>
<b>VB Example</b>	<pre>oEditor.EditRelativeCS Array("NAME:RelativeCSParameters", "Mode:=", "Axis/Position", "OriginX:=", _ "0mm", "OriginY:=", "0mm", "OriginZ:=", "0mm", "XAxisXvec:=", "0.66mm", "XAxisYvec:=", - "0.28mm", "XAxisZvec:=", "0mm", "YAxisXvec:=", "0.06mm", "YAxisYvec:=", "0.14mm", "YAx- isZvec:=", _ "0mm"), Array("NAME:Attributes", "Name:=", "RelativeCS1")</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	<b>Modeler &gt; Export.</b>		
<b>Parameters</b>	Name <i>&lt;Parameters&gt;</i>	Type Array	Description <p>Structured array.</p> <pre>Array ("NAME:ExportParameters",       "AllowRegionDependentPartSelectionForPMLCreation:=",       &lt;boolean&gt;,       "AllowRegionSelectionForPMLCreation:=",       &lt;boolean&gt;,       "Selections:=" , &lt;string list&gt;,       "File Name:=" , &lt;string filepath&gt;,       "Major Version:=" , &lt;integer (-1 if not applicable)&gt;,       "Minor Version:=" , &lt;integer (-1 if not applicable)&gt;)</pre>
<b>Return Value</b>	None.		

Python Syntax	Export( <i>&lt;Parameters&gt;</i> )
<b>Python Example</b>	<pre>oEditor.Export(     [ "NAME:ExportParameters",       "AllowRegionDependentPartSelectionForPMLCreation:=", True,       "AllowRegionSelectionForPMLCreation:=", True,       "Selections:=" , "Box1,Box2,Box3",</pre>

```
"File Name:="           , "C:/Users/jdoe/Desktop/export.sab",
"Major Version:="      , 25,
"Minor Version:="      , 0
])
```

VB Syntax	Export <Parameters>
VB Example	<pre>oEditor.Export  Array("NAME:ExportParameters",       "AllowRegionDependentPartSelectionForPMLCreation:=", True,       "AllowRegionSelectionForPMLCreation:=", True,       "Selections:="          , "Box1,Box2,Box3",       "File Name:="           , "C:/Users/jdoe/Desktop/export.sab",       "Major Version:="       , 25,       "Minor Version:="       , 0)</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 Ensight use \*.avz. For export to Ensight in -ng mode, the corresponding version of Ensight must be installed. On Linux, it might need manual set environment variable AWP\_ROOT212 to its installation path, e.g. AWP\_ROOT212-2=/installations/ansys\_inc/v212/ for AnsysEDT v21.2 and Ensight 21.2.

ExportModelImageToFile supports export overlay of polar plot 3D with existing transformation (scaling, rotation and translation) in -ng (non-graphical) mode.

UI Access	<b>Modeler &gt; Export.</b>																		
<b>Parameters</b>	<table border="1"> <thead> <tr> <th data-bbox="460 267 1022 300">Name</th> <th data-bbox="1022 267 1106 300">Type</th> <th data-bbox="1106 267 1888 300">Description</th> </tr> </thead> <tbody> <tr> <td data-bbox="460 300 1022 332">&lt;path&gt;</td><td data-bbox="1022 300 1106 332">String</td><td data-bbox="1106 300 1888 332">Full file path including extension.</td></tr> <tr> <td data-bbox="460 332 1022 365">&lt;width&gt;</td><td data-bbox="1022 332 1106 365">Integer</td><td data-bbox="1106 332 1888 365">Width in pixels (use 0 for default).</td></tr> <tr> <td data-bbox="460 365 1022 398">&lt;height&gt;</td><td data-bbox="1022 365 1106 398">Integer</td><td data-bbox="1106 365 1888 398">Height in pixels (use 0 for default).</td></tr> <tr> <td data-bbox="460 398 1022 1135">&lt;Parameters&gt;</td><td data-bbox="1022 398 1106 1135" style="vertical-align: top;">Array</td><td data-bbox="1106 398 1888 1135" style="vertical-align: top;"> <p>Structured array.</p> <pre data-bbox="1121 479 1854 1122">         Array("NAME:SaveImageParams",               "ShowAxis:=" , &lt;string containing               boolean&gt;,               "ShowGrid:=" , &lt;string containing               boolean&gt;,               "ShowRuler:=" , &lt;string containing               boolean&gt;,               "ShowRegion:=" , &lt;string&gt;,               "Selections:=" , &lt;string&gt;,               "FieldPlotSelections:=", &lt;string&gt;' # Comma delimited string. #Use to set which field plot to show.               "Orientation:=" , &lt;string&gt;               "ShowOrientationGadget:=" , &lt;False&gt; </pre> </td></tr> </tbody> </table>	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	<p>Structured array.</p> <pre data-bbox="1121 479 1854 1122">         Array("NAME:SaveImageParams",               "ShowAxis:=" , &lt;string containing               boolean&gt;,               "ShowGrid:=" , &lt;string containing               boolean&gt;,               "ShowRuler:=" , &lt;string containing               boolean&gt;,               "ShowRegion:=" , &lt;string&gt;,               "Selections:=" , &lt;string&gt;,               "FieldPlotSelections:=", &lt;string&gt;' # Comma delimited string. #Use to set which field plot to show.               "Orientation:=" , &lt;string&gt;               "ShowOrientationGadget:=" , &lt;False&gt; </pre>			
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	<p>Structured array.</p> <pre data-bbox="1121 479 1854 1122">         Array("NAME:SaveImageParams",               "ShowAxis:=" , &lt;string containing               boolean&gt;,               "ShowGrid:=" , &lt;string containing               boolean&gt;,               "ShowRuler:=" , &lt;string containing               boolean&gt;,               "ShowRegion:=" , &lt;string&gt;,               "Selections:=" , &lt;string&gt;,               "FieldPlotSelections:=", &lt;string&gt;' # Comma delimited string. #Use to set which field plot to show.               "Orientation:=" , &lt;string&gt;               "ShowOrientationGadget:=" , &lt;False&gt; </pre>																	
<b>Return Value</b>	None.																		

<b>Python Syntax</b>	ExportModelImageToFile(<path> <width> <height> <Parameters>)
<b>Python Example</b>	oEditor.ExportModelImageToFile

```
("C:/Users/jdoe/Desktop/export.bmp",
1920,
1080,
[
    "NAME:SaveImageParams",
    "ShowAxis:=" , "True",
    "ShowGrid:=" , "True",
    "ShowRuler:=" , "True",
    "ShowRegion:=" , "Default",
    "Selections:=" , "",
    "FieldPlotSelections:=" , "",
    "FitToSelections:=" , "",
    "FitToFieldPlotSelections:=" , "",
    "Orientation:=" , ""
])
])
```

<b>VB Syntax</b>	ExportModelImageToFile <path> <width> <height> <Parameters>
<b>VB Example</b>	<pre>oEditor.ExportModelImageToFile     "C:/Users/jdoe/Desktop/export.bmp",     1383,</pre>

```

512,
Array("NAME:SaveImageParams",
      "ShowAxis:=", "True",
      "ShowGrid:=", "True",
      "ShowRuler:=", "True",
      "ShowRegion:=", "Default",
      "Selections:=", "",
      "FitToFieldPlotSelections:=", "",
      "Orientation:=", ""
)

```

## ExportModelMeshToFile

Exports geometry model to a 3D model file (e.g. \*.obj, \*.wrl, etc.).

<b>UI Access</b>	N/A		
<b>Parameters</b>	Name	Type	Description
	<filePath>	String	Full file path, including extension *.obj, *.wrl, etc
<b>Return Value</b>	None.		

<b>Python Syntax</b>	ExportModelMeshToFile <filePath>, <selections>)
<b>Python Example</b>	<pre> oEditor.ExportModelMeshToFile("E:/MyDir/scriptRun/2Selected-ng.obj", ["BotCover-", ,"AveragingVolumeAtPeakRMSEfieldLocation"]) </pre>

<b>VB Syntax</b>	ExportModelMeshToFile <filePath>, <selections>
<b>VB Example</b>	<pre>oEditor.ExportModelMeshToFile("E:/MyDir/scriptRun/2Selected-ng.obj", ["BotCover-", "AveragingVolumeAtPeakRMSEfieldLocation"])</pre>

## Fillet

Performs a fillet on specified edge(s).

UI Access	<b>Modeler &gt; Fillet.</b>		
	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: <a href="#">SelectionsArray</a> .
Parameters	<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.		

<b>Python Syntax</b>	<code>Fillet(&lt;SelectionsArray&gt;, &lt;Parameters&gt;)</code>
----------------------	------------------------------------------------------------------

	<pre> oEditor.Fillet(     ["NAME:Selections",         "Selections:=" , "Box1",         "NewPartsModelFlag:=" , "Model"     ],     ["NAME:Parameters",         ["NAME:FilletParameters",             "Edges:=" , [13],             "Vertices:=" , [],             "Radius:=" , "1mm",             "Setback:=" , "0mm"         ]     ] ) </pre>
--	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

<b>VB Syntax</b>	Fillet <SelectionsArray> <Parameters>
<b>VB Example</b>	<pre> oEditor.Fillet     Array("NAME:Selections",         "Selections:=" , "Box1",         "NewPartsModelFlag:=" , "Model")     Array("NAME:Parameters",         Array("NAME:FilletParameters", </pre>

	"Edges:=" , Array(13), "Vertices:=" , Array(), "Radius:=" , "1mm", "Setback:=" , "0mm") )
--	----------------------------------------------------------------------------------------------------

## FlattenGroup

Flattens a specified history tree group.

UI Access	Modeler > Group > Flatten.		
Parameters	Name <i>&lt;GroupID&gt;</i>	Type Array	Description Structured array. Array ("Groups:=", Array(<string list of group IDs>))
Return Value	None.		

Python Syntax	FlattenGroup (<GroupID>)
Python Example	oEditor.FlattenGroup ( ["Groups:=", ["Group1"] ] )

VB Syntax	FlattenGroup <GroupID>
VB Example	oEditor.FlattenGroup Array("Groups:=", Array("Group1"))

## GenerateHistory

Generates the history for specified 1D object(s).

<b>UI Access</b>	<b>Modeler &gt; Generate History.</b>		
<b>Parameters</b>	Name <SelectionsArray>	Type Array	Description Structured array. See: <a href="#">SelectionsArray</a> .
<b>Return Value</b>	None.		

<b>Python Syntax</b>	GenerateHistory (<SelectionsArray>)
<b>Python Example</b>	<pre> oEditor.GenerateHistory(     [ "NAME:Selections",         "Selections:="          , "Polyline1",         "NewPartsModelFlag:="    , "Model",         "UseCurrentCS:="         , True     ] ) </pre>

<b>VB Syntax</b>	GenerateHistory <SelectionsArray>
------------------	-----------------------------------

<b>VB Example</b>	<pre>oEditor.GenerateHistory Array("NAME:Selections",                                "Selections:=" , "Polyline1",                                "NewPartsModelFlag:=" , "Model",                                "UseCurrentCS:=" , True)</pre>
-------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

## GetActiveCoordinateSystem

Returns the active coordinate system.

<b>UI Access</b>	None.
<b>Parameters</b>	None.
<b>Return Value</b>	String name of active coordinate system.

<b>Python Syntax</b>	GetActiveCoordinateSystem()
<b>Python Example</b>	<pre>oEditor.GetActiveCoordinateSystem()</pre>

<b>VB Syntax</b>	GetActiveCoordinateSystem
<b>VB Example</b>	<pre>dim csName csName = oEditor.GetActiveCoordinateSystem</pre>

## GetActiveCoordinateSystemTransform

Returns the transformation information of active coordinate system.

<b>UI Access</b>	None.
<b>Parameters</b>	None.
<b>Return Value</b>	Array of strings containing transformation information.

<b>Python Syntax</b>	GetActiveCoordinateSystemTransform()
<b>Python Example</b>	<code>oEditor.GetActiveCoordinateSystemTransform()</code>

<b>VB Syntax</b>	GetActiveCoordinateSystemTransform
<b>VB Example</b>	<code>oEditor.GetActiveCoordinateSystemTransform</code>

## GetCoordinateSystems

Returns the names of coordinate systems in the design.

<b>UI Access</b>	None.
<b>Parameters</b>	None.
<b>Return Value</b>	Array containing string names of coordinate systems.

<b>Python Syntax</b>	GetCoordinateSystems()
<b>Python Example</b>	<code>oEditor.GetCoordinateSystems ()</code>

<b>VB Syntax</b>	GetCoordinateSystems()
<b>VB Example</b>	<code>dim csNames</code>

	csNames = oEditor.GetCoordinateSystems
--	----------------------------------------

## HealObject

Heals an imported object.

UI Access	Modeler > Model Preparation > Heal.											
Parameters	<table border="1"> <tr> <td>Name</td> <td>Type</td> <td>Description</td> </tr> <tr> <td>&lt;SelectionsArray&gt;</td> <td>Array</td> <td>Structured array. See: <a href="#">SelectionsArray</a>.</td> </tr> <tr> <td>&lt;Parameters&gt;</td> <td>Array</td> <td> <p>Structured array.</p> <pre>Array ("NAME:ObjectName", "Version:=" , &lt;integer&gt;, "AutoHeal:=" , &lt;boolean&gt;, "TolerantStitch:=" , &lt;boolean&gt;, "SimplifyGeom:=" , &lt;boolean&gt;, "TightenGaps:=" , &lt;boolean&gt;, "HealToSolid:=" , &lt;boolean&gt;, "StopAfterFirstStitchError:=", &lt;boolean&gt;, "MaxStitchTol:=" , &lt;float&gt;, "ExplodeAndStitch:=" , &lt;boolean&gt;, "GeomSimplificationTol:=" , &lt;integer&gt;, "MaximumGeneratedRadiusForSimplification:=" , &lt;integer&gt;),</pre> </td> </tr> </table>	Name	Type	Description	<SelectionsArray>	Array	Structured array. See: <a href="#">SelectionsArray</a> .	<Parameters>	Array	<p>Structured array.</p> <pre>Array ("NAME:ObjectName", "Version:=" , &lt;integer&gt;, "AutoHeal:=" , &lt;boolean&gt;, "TolerantStitch:=" , &lt;boolean&gt;, "SimplifyGeom:=" , &lt;boolean&gt;, "TightenGaps:=" , &lt;boolean&gt;, "HealToSolid:=" , &lt;boolean&gt;, "StopAfterFirstStitchError:=", &lt;boolean&gt;, "MaxStitchTol:=" , &lt;float&gt;, "ExplodeAndStitch:=" , &lt;boolean&gt;, "GeomSimplificationTol:=" , &lt;integer&gt;, "MaximumGeneratedRadiusForSimplification:=" , &lt;integer&gt;),</pre>		
Name	Type	Description										
<SelectionsArray>	Array	Structured array. See: <a href="#">SelectionsArray</a> .										
<Parameters>	Array	<p>Structured array.</p> <pre>Array ("NAME:ObjectName", "Version:=" , &lt;integer&gt;, "AutoHeal:=" , &lt;boolean&gt;, "TolerantStitch:=" , &lt;boolean&gt;, "SimplifyGeom:=" , &lt;boolean&gt;, "TightenGaps:=" , &lt;boolean&gt;, "HealToSolid:=" , &lt;boolean&gt;, "StopAfterFirstStitchError:=", &lt;boolean&gt;, "MaxStitchTol:=" , &lt;float&gt;, "ExplodeAndStitch:=" , &lt;boolean&gt;, "GeomSimplificationTol:=" , &lt;integer&gt;, "MaximumGeneratedRadiusForSimplification:=" , &lt;integer&gt;),</pre>										

			<pre> "SimplifyType:=" , &lt;integer&gt;, "TightenGapsWidth:=" , &lt;float&gt;, "RemoveSliverFaces:=" , &lt;boolean&gt;, "RemoveSmallEdges:=" , &lt;boolean&gt;, "RemoveSmallFaces:=" , &lt;boolean&gt;, "SliverFaceTol:=" , &lt;integer&gt;, "SmallEdgeTol:=" , &lt;integer&gt;, "SmallFaceAreaTol:=" , &lt;integer&gt;, "BoundingBoxScaleFactor:=" , &lt;integer&gt;, "RemoveHoles:=" , &lt;boolean&gt;, "RemoveChamfers:=" , &lt;boolean&gt;, "RemoveBlends:=" , &lt;boolean&gt;, "HoleRadiusTol:=" , &lt;integer&gt;, "ChamferWidthTol:=" , &lt;integer&gt;, "BlendRadiusTol:=" , &lt;integer&gt;, "AllowableSurfaceAreaChange:=" , &lt;integer&gt;, "AllowableVolumeChange:=" , &lt;integer&gt;) </pre>
<b>Return Value</b>	None.		

<b>Python Syntax</b>	HealObject(<SelectionsArray>, <Parameters>)
----------------------	---------------------------------------------

**Python Example**

```
oEditor.HealObject(  
    ["NAME:Selections",  
        "Selections:=" , "Box1",  
        "NewPartsModelFlag:=" , "Model"  
    ],  
    ["NAME:ObjectHealingParameters",  
        "Version:=" , 1,  
        "AutoHeal:=" , True,  
        "TolerantStitch:=" , True,  
        "SimplifyGeom:=" , True,  
        "TightenGaps:=" , True,  
        "HealToSolid:=" , False,  
        "StopAfterFirstStitchError:=", False,  
        "MaxStitchTol:=" , 0.001,  
        "ExplodeAndStitch:=" , True,  
        "GeomSimplificationTol:=", -1,  
        "MaximumGeneratedRadiusForSimplification:=", -1,  
        "SimplifyType:=" , 2,  
        "TightenGapsWidth:=" , 1E-06,  
        "RemoveSliverFaces:=" , False,
```

```

    "RemoveSmallEdges:="      , False,
    "RemoveSmallFaces:="      , False,
    "SliverFaceTol:="        , 0,
    "SmallEdgeTol:="          , 0,
    "SmallFaceAreaTol:="      , 0,
    "BoundingBoxScaleFactor:=", 1250,
    "RemoveHoles:="           , False,
    "RemoveChamfers:="        , False,
    "RemoveBlends:="          , False,
    "HoleRadiusTol:="         , 0,
    "ChamferWidthTol:="       , 0,
    "BlendRadiusTol:="        , 0,
    "AllowableSurfaceAreaChange:=", 5,
    "AllowableVolumeChange:=", 5
  ])
)

```

<b>VB Syntax</b>	HealObject <SelectionsArray> <Parameters>
<b>VB Example</b>	<pre> oEditor.HealObject Array("NAME:Selections", "Selections:=", "Box2", "NewPartsModelFlag:=", _ "Model"), Array("NAME:ObjectHealingParameters", "Version:=", 1, "AutoHeal:=", false, "TolerantStitch:=", _ </pre>

```

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)

```

## 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.

		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>, "SourceFile:=" , <string>)
<b>Return Value</b>	None.	

Python Syntax	Import(<Parameters>)
<b>Python Example</b>	<pre> oEditor.Import(     [         "NAME:NativeBodyParameters",         "HealOption:=" , 0,         "Options:=" , "-1",         "FileType:=" , "UnRecognized",         "MaxStitchTol:=" , -1,         "ImportFreeSurfaces:=" , False,         "GroupByAssembly:=" , False,         "CreateGroup:=" , True,         "STLFileUnit:=" , "Auto",         "MergeFacesAngle:=" , 0.02,         "HealSTL:=" , False,         "ReduceSTL:=" , False,         "ReduceMaxError:=" , 0,         "ReducePercentage:=" , 100,         "PointCoincidenceTol:=" , 1E-06,         "CreateLightweightPart:=" , False,         "ImportMaterialNames:=" , False,         "SeparateDisjointLumps:=" , False,     ] ) </pre>

```

    "SourceFile:="           , "C:\\Users\\jdoe\\Desktop\\export.model"
]
)

```

VB Syntax	Import <Parameters>
VB Example	<pre> oEditor.Import Array ("NAME:NativeBodyParameters",                      "HealOption:="           , 0,                      "Options:="              , "-1",                      "FileType:="             , "UnRecognized",                      "MaxStitchTol:="         , -1,                      "ImportFreeSurfaces:="   , False,                      "GroupByAssembly:="      , False,                      "CreateGroup:="           , True,                      "STLFileUnit:="           , "Auto",                      "MergeFacesAngle:="       , 0.02,                      "HealSTL:="               , False,                      "ReduceSTL:="              , False,                      "ReduceMaxError:="         , 0,                      "ReducePercentage:="       , 100,                      "PointCoincidenceTol:="   , 1E-06,                      "CreateLightweightPart:="  , False,                      "ImportMaterialNames:="    , False, ) </pre>

	<pre>"SeparateDisjointLumps:=", False, "SourceFile:=" , "C:\\Users\\jdoe\\Desktop\\export.model")</pre>
--	---------------------------------------------------------------------------------------------------------

## ImportDXF [Modeler]

Imports a DXF model file.

UI Access	Modeler > Import.		
	Name	Type	Description
Parameters	<Parameters>	Array	<p>Structured array.</p> <pre>Array ("NAME:options",       "FileName:=" , &lt;string&gt;,       "Scale:=" , &lt;float&gt;,       "AutoDetectClosed:=" , &lt;boolean&gt;,       "SelfStitch:=" , &lt;boolean&gt;,       "DefeatureGeometry:=" , &lt;boolean&gt;,       "DefeatureDistance:=" , &lt;integer&gt;,       "RoundCoordinates:=" , &lt;boolean&gt;,       "RoundNumDigits:=" , &lt;integer&gt;,       "WritePolyWithWidthAsFilledPoly:=" , &lt;boolean&gt;,       "ImportMethod:=" , &lt;integer&gt;,       "2DSheetBodies:=" , &lt;boolean&gt;,       &lt;layersArray&gt;)</pre>

	<code>&lt;layersArray&gt;</code>	Array	Structured array.  Array ("NAME:LayerInfo", <layer>, <layer>, <layer>, ...)
	<code>&lt;layer&gt;</code>	Array	Structured array.  Array ("NAME:<layerName>", "source:=" , <string>, "display_source:=" , <string>, "import:=" , <boolean>, "dest:=" , <string>, "dest_selected:=" , <boolean>, "layer_type:=" , <string>)
<b>Return Value</b>	None.		

<b>Python Syntax</b>	<code>ImportDXF(&lt;Parameters&gt;)</code>
<b>Python Example</b>	<pre>oEditor.ImportDXF(                   [ "NAME:options",                   "FileName:=", "C:/Users/jdoe/Desktop/export.dxf",                   "Scale:="                  , 0.001,                   "AutoDetectClosed:="    , True,                   "SelfStitch:="        , True,                   "DefeatureGeometry:="  , False,</pre>

```
"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"
]
])
```

<b>VB Syntax</b>	ImportDXF <Parameters>
<b>VB Example</b>	<pre> oEditor.ImportDXF Array("NAME:options", "FileName:=", _ "C:/Users/jdoe/Desktop/export.dxf", "Scale:=", 0.001, "AutoDetectClosed:=", _ true, "SelfStitch:=", true, "DefeatureGeometry:=", false, "DefeatureDistance:=", _ 0, "RoundCoordinates:=", false, "RoundNumDigits:=", 4, "WritePolyWithWidthAsFilledPoly:=", _ false, "ImportMethod:=", 1, "2DSheetBodies:=", false, Array("NAME:LayerInfo", Array </pre>

```
("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.

<b>UI Access</b>	<b>Modeler &gt; Import From Clipboard.</b>
<b>Parameters</b>	None.
<b>Return Value</b>	None.

<b>Python Syntax</b>	ImportFromClipboard ()
<b>Python Example</b>	oEditor.ImportFromClipboard()

<b>VB Syntax</b>	ImportFromClipboard
<b>VB Example</b>	oEditor.ImportFromClipboard

## ImportGDSII [Modeler]

Imports a GDSII model file.

UI Access	Modeler > Import.		
Parameters	Name <i>&lt;Parameters&gt;</i>	Type Array	Description <p>Structured array.</p> <pre>Array("NAME:options",       "FileName:=" , &lt;string&gt;,       "FlattenHierarchy:=" , &lt;boolean&gt;,       "ImportMethod:=" , &lt;integer&gt;,       &lt;layerMapArray&gt;, &lt;layerMapArray&gt;, &lt;layerMapArray&gt;,...)       "OrderMap:=" ,       [ "entry:=" ,         &lt;entry&gt;, &lt;entry&gt;, &lt;entry&gt;,...]     ])</pre>
	<i>&lt;layerMapArray&gt;</i>	Array	Structured array. <pre>Array("NAME:LayerMapInfo",       "LayerNum:=" , &lt;integer&gt;,       "DestLayer:=" , &lt;string&gt;,       "layer_type:=" , &lt;string&gt;)</pre>
	<i>&lt;entry&gt;</i>	Array	Structured array. <pre>Array("order:=" , &lt;integer LayerNum&gt;, "layer:=" ,</pre>

		<string DestLayer>)
<b>Return Value</b>	None.	

<b>Python Syntax</b>	ImportGDSII(<Parameters>)
<b>Python Example</b>	<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",                 "layer_type:=", "signal"             ],         ],     ], )</pre>

```

        [ "NAME:LayerMapInfo",
          "LayerNum:=" , 14,
          "DestLayer:=" , "Signal14",
          "layer_type:=" , "signal"
        ]
      ],
      "OrderMap:=" ,
      [
        "entry:=" ,
        [ "order:=", 0, "layer:=", "Signal12"] ,
        "entry:=" ,
        [ "order:=", 1, "layer:=", "Signal13"] ,
        "entry:=" ,
        [ "order:=", 2, "layer:=", "Signal14"]
      ]
    )
  )
)

```

<b>VB Syntax</b>	ImportGDSII <Parameters>
<b>VB Example</b>	<pre> oEditor.ImportGDSII Array("NAME:options",   "FileName:=", "C:/export.gds",   "FlattenHierarchy:=", True, </pre>

```

    "ImportMethod:=", 1,
    Array("NAME:LayerMap",
        Array ("NAME:LayerMapInfo",
            "LayerNum:=", 1,
            "DestLayer:=", "Signal1",
            "layer_type:=", "signal")))
    "OrderMap:=", Array(
        "entry:=", Array(
            "order:=", 0,
            "layer:=", "Signal1")))

```

## Imprint

Imprints the geometry of one object upon another.

UI Access	Modeler > Boolean > Imprint....		
<b>Parameters</b>	Name <i>&lt;Selections&gt;</i>	Type Array	Description Structured array.  Array ("NAME:Selections",         "Blank Parts:=", <string, object name>,         "Tool Parts:=", <string, object name>)
	<i>&lt;Parameters&gt;</i>	Array	Structured array.  Array ("NAME:ImprintParameters",

		"KeepOriginals:=", <boolean>)
<b>Return Value</b>	None.	

<b>Python Syntax</b>	Imprint (<Selections>, <Parameters>)
<b>Python Example</b>	<pre> oEditor.Imprint(     ["NAME:Selections",         "Blank Parts:=", "Cylinder1",         "Tool Parts:=", "Box1"],     ["NAME:ImprintParameters",         "KeepOriginals:=", False]) </pre>

<b>VB Syntax</b>	Imprint <Selections>, <Parameters>
<b>VB Example</b>	<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	<b>Modeler &gt; Boolean &gt; Imprint Projection.</b>		
<b>Parameters</b>	Name	Type	Description
	<Selections>	Array	<p>Structured array.</p> <pre>Array("NAME:Selections",       "Selections:=", &lt;string, selected objects&gt;)</pre>
			<p>Structured array.</p> <pre>Array("NAME:ImprintProjectionParameters",       "KeepOriginals:=", &lt;boolean&gt;,       "NormalProjection:=", &lt;boolean&gt;,       "Distance:=", &lt;float&gt;,       "DirectionX:=", &lt;float&gt;,       "DirectionY:=", &lt;float&gt;,       "DirectionZ:=", &lt;float&gt;)</pre>
<b>Return Value</b>	None.		

<b>Python Syntax</b>	ImprintProjection (<Selections>, <Parameters>)
<b>Python Example</b>	<pre> oEditor.ImprintProjection(     ["NAME:Selections",      "Selections:=", "Rectangle1,Cylinder1"],     ["NAME:ImprintProjectionParameters",      ])</pre>

```

    "KeepOriginals:=", False,
    "NormalProjection:=", True,
    "Distance:=", "1.36014705087354mm",
    "DirectionX:=", "0.882257546512569",
    "DirectionY:=", "0.294085848837523",
    "DirectionZ:=", "0.367607311046904"])

```

<b>VB Syntax</b>	ImprintProjection <Selections>, <Parameters>
<b>VB Example</b>	<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.

<b>UI Access</b>	<b>Modeler &gt; Boolean &gt; Intersect.</b>		
<b>Parameters</b>	Name <i>&lt;SelectionsArray&gt;</i>	Type Array	Description Structured array. See: <a href="#">SelectionsArray</a> .
	<i>&lt;Parameters&gt;</i>	Array	Structured array.  Array ("NAME:IntersectParameters", "KeepOriginals:=" , <boolean True to keep original objects; False to delete>)
<b>Return Value</b>	None.		

<b>Python Syntax</b>	Intersect(<SelectionsArray>, <Parameters>)
<b>Python Example</b>	<pre> oEditor.Intersect(     ["NAME:Selections",         "Selections:=", "Rectangle1,Rectangle2"     ],     [ "NAME:IntersectParameters",         "KeepOriginals:=", False     ] ) </pre>

<b>VB Syntax</b>	Intersect <SelectionsArray> <Parameters>
<b>VB Example</b>	<pre> oEditor.Intersect     Array ("NAME:Selections", </pre>

```

    "Selections:=", "Rectangle1,Rectangle2")
Array("NAME:IntersectParameters",
      "KeepOriginals:=", False)

```

## MoveCStoEnd

Moves a specified Object Coordinate System to the end of the History tree.

<b>UI Access</b>	<b>Modeler &gt; Coordinate System &gt; Move CS to End.</b>		
<b>Parameters</b>	Name <i>&lt;SelectionsArray&gt;</i>	Type Array	Description Structured array. See: <a href="#">SelectionsArray</a> .
<b>Return Value</b>	None.		

<b>Python Syntax</b>	MoveCStoEnd (<SelectionsArray>)
<b>Python Example</b>	<pre> oEditor.MoveCSToEnd (     ["NAME:Selections",      "Selections:=", "ObjectCS1"     ]) </pre>

<b>VB Syntax</b>	MoveCStoEnd <SelectionsArray>
<b>VB Example</b>	<pre> oEditor.MoveCSToEnd Array("NAME:Selections", "Selections:=", "ObjectCS1") </pre>

## MoveEntityToGroup

Moves a specified entity or entities to a specified group.

<b>UI Access</b>	Drag item into the group in the history tree.		
<b>Parameters</b>	<b>Name</b> <i>&lt;Objects&gt;</i>	<b>Type</b> Array	<b>Description</b> Structured array. Selects the entity/entities to move. Array("Objects:=", <array containing string object IDs>)
	<i>&lt;MoveEntityToGroup&gt;</i>	Array	Structured array. Array("ParentGroup:=", <string group name>)
<b>Return Value</b>	None.		

<b>Python Syntax</b>	MoveEntityToGroup (<Objects>, <MoveEntityToGroup>)
<b>Python Example</b>	<pre>oEditor.MoveEntityToGroup(     ["Objects:=", ["Box3"]],     ["ParentGroup:=", "Box_Group"])</pre>

<b>VB Syntax</b>	MoveEntityToGroup <Objects> <MoveEntityToGroup>
<b>VB Example</b>	<pre>oEditor.MoveEntityToGroup     Array("Objects:=", Array("Box3"))</pre>

	<code>Array("ParentGroup:=", "Box_Group")</code>
--	--------------------------------------------------

## MoveFaces

Moves the specified faces along normal or along a vector.

UI Access	<b>Modeler &gt; Surface &gt; Move Faces &gt; Along [Normal/Vector].</b>														
	<table border="1"> <tr> <td>Name</td> <td>Type</td> <td>Description</td> </tr> <tr> <td><code>&lt;SelectionsArray&gt;</code></td> <td>Array</td> <td>Structured array. See: <a href="#">SelectionsArray</a>.</td> </tr> <tr> <td><code>&lt;MoveFacesParametersArray&gt;</code></td> <td>Array</td> <td> <p>Structured array.</p> <pre>Array ("NAME:Parameters",       &lt;FacesOfOneObjToMove&gt;, &lt;FacesOfOneObjToMove&gt;, ...)</pre> </td> </tr> <tr> <td><code>&lt;FacesOfOneObjToMove&gt;</code></td> <td>Array</td> <td> <p>Structured array.</p> <pre>Array ("Name:MoveFacesParameters",       "MoveAlongNormalFlag:=", &lt;boolean&gt;,       "OffsetDistance:=", &lt;string&gt;,       "MoveVectorX:=", &lt;string&gt;,       "MoveVectorY:=", &lt;string&gt;,       "MoveVectorZ:=", &lt;string&gt;,       "FacesToMove:=", &lt;array&gt;)</pre> <p><code>MoveAlongNormalFlag</code> specifies whether to move along the face normal or along a vector. If false, provide the <code>MoveVectorX</code>, <code>MoveVectorY</code>, and <code>MoveVectorZ</code> parameters.</p> <p><code>FacesToMove</code> is an array of integers (the IDs of the faces to move).</p> </td> </tr> </table>	Name	Type	Description	<code>&lt;SelectionsArray&gt;</code>	Array	Structured array. See: <a href="#">SelectionsArray</a> .	<code>&lt;MoveFacesParametersArray&gt;</code>	Array	<p>Structured array.</p> <pre>Array ("NAME:Parameters",       &lt;FacesOfOneObjToMove&gt;, &lt;FacesOfOneObjToMove&gt;, ...)</pre>	<code>&lt;FacesOfOneObjToMove&gt;</code>	Array	<p>Structured array.</p> <pre>Array ("Name:MoveFacesParameters",       "MoveAlongNormalFlag:=", &lt;boolean&gt;,       "OffsetDistance:=", &lt;string&gt;,       "MoveVectorX:=", &lt;string&gt;,       "MoveVectorY:=", &lt;string&gt;,       "MoveVectorZ:=", &lt;string&gt;,       "FacesToMove:=", &lt;array&gt;)</pre> <p><code>MoveAlongNormalFlag</code> specifies whether to move along the face normal or along a vector. If false, provide the <code>MoveVectorX</code>, <code>MoveVectorY</code>, and <code>MoveVectorZ</code> parameters.</p> <p><code>FacesToMove</code> is an array of integers (the IDs of the faces to move).</p>		
Name	Type	Description													
<code>&lt;SelectionsArray&gt;</code>	Array	Structured array. See: <a href="#">SelectionsArray</a> .													
<code>&lt;MoveFacesParametersArray&gt;</code>	Array	<p>Structured array.</p> <pre>Array ("NAME:Parameters",       &lt;FacesOfOneObjToMove&gt;, &lt;FacesOfOneObjToMove&gt;, ...)</pre>													
<code>&lt;FacesOfOneObjToMove&gt;</code>	Array	<p>Structured array.</p> <pre>Array ("Name:MoveFacesParameters",       "MoveAlongNormalFlag:=", &lt;boolean&gt;,       "OffsetDistance:=", &lt;string&gt;,       "MoveVectorX:=", &lt;string&gt;,       "MoveVectorY:=", &lt;string&gt;,       "MoveVectorZ:=", &lt;string&gt;,       "FacesToMove:=", &lt;array&gt;)</pre> <p><code>MoveAlongNormalFlag</code> specifies whether to move along the face normal or along a vector. If false, provide the <code>MoveVectorX</code>, <code>MoveVectorY</code>, and <code>MoveVectorZ</code> parameters.</p> <p><code>FacesToMove</code> is an array of integers (the IDs of the faces to move).</p>													
Parameters															

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

<b>Python Syntax</b>	MoveFaces (<SelectionsArray>, <MoveFacesParametersArray>)
<b>Python Example</b>	<pre>oEditor.MoveFaces(     [         "NAME:Selections",         "Selections:="          , "Rectangle1",         "NewPartsModelFlag:="   , "Model"     ],     [         "NAME:Parameters",         [             "NAME:MoveFacesParameters",             "MoveAlongNormalFlag:="   , True,             "OffsetDistance:="       , "1mm",             "MoveVectorX:="          , "0mm",             "MoveVectorY:="          , "0mm",             "MoveVectorZ:="          , "0mm",             "FacesToMove:="          , [183]         ]     ] )</pre>

<b>VB Syntax</b>	MoveFaces <SelectionsArray>, <MoveFacesParametersArray>
<b>VB Example</b>	<pre>oEditor.MoveFaces _</pre> <pre>    Array("NAME:Selections", "Selections:=", _</pre>

```

    "Box2,Box1"), _

    Array("NAME:Parameters", _

    Array("NAME:MoveFacesParameters", _
        "MoveAlongNormalFlag:=", true, _
        "OffsetDistance:=", "1mm", _
        "FacesToMove:=", Array(218)),

    Array("NAME:MoveFacesParameters", _
        "MoveAlongNormalFlag:=", false,_
        "OffsetDistance:=", "1mm", _
        "MoveVectorX:=", "1mm", _
        "MoveVectorY:=", "0mm", _
        "MoveVectorZ:=", "0mm", _
        "FacesToMove:=", Array(185)))

```

## ProjectSheet

Project a sheet object, typically for modeling thin conformal deposits. Typically followed by **Thicken Sheet**.

<b>UI Access</b>	Click <b>Modeler &gt; Surface &gt; Project Sheet</b> .		
<b>Parameters</b>	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: <a href="#">SelectionsArray</a> .
<b>Return Value</b>	None.		

<b>Python Syntax</b>	ProjectSheet (<SelectionsArray>, <Parameters>)
<b>Python Example</b>	<pre>oEditor.ProjectSheet(     ['NAME:Selections',         'Selections:="Box1,Box2,Polyline1"]',     ['NAME:ProjectSheetParameters'] )</pre>

<b>VB Syntax</b>	ProjectSheet <SelectionsArray> <Parameters>
<b>VB Example</b>	<pre>oEditor.ProjectSheet     Array("NAME:Selections",         "Selections:=", "Box1,Box2,Polyline1"),     Array("NAME:ProjectSheetParameters")</pre>

## PurgeHistory

Purges the history of a specified object.

<b>UI Access</b>	Modeler > Purge History.					
<b>Parameters</b>	<table border="1"><tr><td>Name &lt;SelectionsArray&gt;</td><td>Type Array</td><td>Description Structured array. See: <a href="#">SelectionsArray</a>.</td></tr></table>			Name <SelectionsArray>	Type Array	Description Structured array. See: <a href="#">SelectionsArray</a> .
Name <SelectionsArray>	Type Array	Description Structured array. See: <a href="#">SelectionsArray</a> .				

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

<b>Python Syntax</b>	PurgeHistory (<SelectionsArray>)
<b>Python Example</b>	<pre> oEditor.PurgeHistory(     [         "NAME:Selections",         "Selections:=", "Box2",         "NewPartsModelFlag:=", "Model"     ] ) </pre>

<b>VB Syntax</b>	PurgeHistory <SelectionsArray>
<b>VB Example</b>	<pre> oEditor.PurgeHistory     Array("NAME:Selections",         "Selections:=", "Box2",         "NewPartsModelFlag:=", "Model") ) </pre>

## ReplaceWith3DComponent

Replaces the selection with a 3D component.

<b>UI Access</b>	None.									
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;GeometryData&gt;</td> <td>Array</td> <td>Structured array.</td> </tr> <tr> <td>&lt;DesignData&gt;</td> <td>Array</td> <td>Structured array.</td> </tr> </tbody> </table>	Name	Type	Description	<GeometryData>	Array	Structured array.	<DesignData>	Array	Structured array.
Name	Type	Description								
<GeometryData>	Array	Structured array.								
<DesignData>	Array	Structured array.								

	<ImageFile>	Array	Structured array.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	ReplaceWith3DComponent(<GeometryData>, <DesignData>, <ImageFile>)
<b>Python Example</b>	<pre>oEditor.ReplaceWith3DComponent (     ["NAME:ReplaceData",      "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,      "IncludedParts:=", ["outer", "teflon", "inner", "teflon_1"],</pre>

```

    "HiddenParts:=", [],
    "IncludedCS:=", [],
    "ReferenceCS:=", "Global",
    "IncludedParameters:=", ["bend_angle"],
    "ParameterDescription:=", ["bend_angle:=", ""]
],
["NAME:DesignData",
    "Excitations:=", ["1","2"]
],
["NAME:ImageFile",
    "ImageFile:=", ""
]
)

```

<b>VB Syntax</b>	ReplaceWith3DComponent <GeometryData>, <DesignData>, <ImageFile>
<b>VB Example</b>	<pre> oEditor.ReplaceWith3DComponent Array("NAME:CreateData", "ComponentName:=", "ConnectorOnly_wBCs", "Company:="", "", "Company URL:=", "", "Model Number:=", "", "Help URL:=", _  "", "Version:=", "1.0", "Notes:=", "", "IconType:=", "", "Owner:=", _  "MyName", "Email:=", "My@email.com", "Date:=", "3:07:37 PM Jun 25, 2019", _ </pre>

```

"HasLabel:=", false, "IsEncrypted:=", false, "AllowEdit:=", false, _
"SecurityMessage:=", "", "Password:=", "", "EditPassword:=", "", _
"PasswordType:=", "UnknownPassword", "HideContents:=", true, "ReplaceNames:=", true, _
"ComponentOutline:=", "None", "IncludedParts:=", _
Array("pin", "solderl", "ConnectorDielectric", _
"ConnectorOuter", "pcap", "solderr"), "HiddenParts:=", Array(), _
"IncludedCS:=", Array(), "ReferenceCS:=", _
"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: <a href="#">SelectionsArray</a> .
<Parameters>	Array		Structured array.

			Array("NAME:SectionToParameters", "CreateNewObjects:=" , <boolean>, "SectionPlane:=" , <string "XY", "YZ", or "ZX">, "SectionCrossObject:=" , <boolean>)
<b>Return Value</b>	None.		

<b>Python Syntax</b>	Section (<SelectionsArray>, <Parameters>)
<b>Python Example</b>	<pre> oEditor.Section(     ["NAME:Selections",         "Selections:="          , "Cone1",         "NewPartsModelFlag:="    , "Model"     ],     ["NAME:SectionToParameters",         "CreateNewObjects:="     , True,         "SectionPlane:="         , "XY",         "SectionCrossObject:="   , False     ] ) </pre>

<b>VB Syntax</b>	Section <SelectionsArray> <Parameters>
------------------	----------------------------------------

**VB Example**

```
oEditor.Section  
Array("NAME:Selections",  
      "Selections:=" , "Cone1",  
      "NewPartsModelFlag:=" , "Model")  
  
Array("NAME:SectionToParameters",  
      "CreateNewObjects:=" , True,  
      "SectionPlane:=" , "XY",  
      "SectionCrossObject:=" , False)
```

**SeparateBody**

Separates the bodies of specified multi-lump objects.

UI Access	Modeler > Boolean > Separate Bodies.		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: <a href="#">SelectionsArray</a> .
Return Value	None.		

**Python Syntax**

```
SeparateBody (<SelectionsArray>)
```

**Python Example**

```
oEditor.SeparateBody (  
      ["NAME:Selections",  
       "Selections:=", "Rectangle1,Circle1",
```

	<pre>"NewPartsModelFlag:=", "Model" ])</pre>
--	----------------------------------------------

<b>VB Syntax</b>	SeparateBody <SelectionsArray>
<b>VB Example</b>	<pre>oEditor.SeparateBody Array("NAME:Selections",       "Selections:=", "Rectangle1,Circle1",       "NewPartsModelFlag:=", "Model")</pre>

## SetModelUnits

Sets the model units.

UI Access	Modeler > Units.						
Parameters	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;Parameters&gt;</td> <td>Array</td> <td> <p>Structured array.</p> <pre>Array("NAME:Units Parameter",       "Units:=", &lt;string&gt;,       "Rescale:=", &lt;boolean True to rescale model; else False&gt;)</pre> <p>To see valid unit strings, select <b>Modeler &gt; Units</b>.</p> </td> </tr> </tbody> </table>	Name	Type	Description	<Parameters>	Array	<p>Structured array.</p> <pre>Array("NAME:Units Parameter",       "Units:=", &lt;string&gt;,       "Rescale:=", &lt;boolean True to rescale model; else False&gt;)</pre> <p>To see valid unit strings, select <b>Modeler &gt; Units</b>.</p>
Name	Type	Description					
<Parameters>	Array	<p>Structured array.</p> <pre>Array("NAME:Units Parameter",       "Units:=", &lt;string&gt;,       "Rescale:=", &lt;boolean True to rescale model; else False&gt;)</pre> <p>To see valid unit strings, select <b>Modeler &gt; Units</b>.</p>					
Return Value	None.						

<b>Python Syntax</b>	SetModelUnits (<Parameters>)
<b>Python Example</b>	<pre>oEditor.SetModelUnits(     [ "NAME:Units Parameter",         "Units:=", "km",         "Rescale:=", False     ] )</pre>

<b>VB Syntax</b>	SetModelUnits <Parameters>
<b>VB Example</b>	<pre>oEditor.SetModelUnits Array("NAME:Units Parameter",     "Units:=", "km",     "Rescale:=", False)</pre>

## SetWCS

Sets the working coordinate system.

<b>UI Access</b>	<b>Modeler &gt; Coordinate System &gt; Set Working CS.</b>		
<b>Parameters</b>	Name <Parameters>	Type Array	Description Structured array.  Array ("NAME:SetWCS Parameter", "Working Coordinate System:=", <string CS name>,

			"RegionDepCSOk:=" , <boolean True if region-dependent; else False)
<b>Return Value</b>	None.		

<b>Python Syntax</b>	SetWCS (<Parameters>)
<b>Python Example</b>	<pre>oEditor.SetWCS (     [ "NAME:SetWCS Parameter",         "Working Coordinate System:=", "Global",         "RegionDepCSOk:=", False     ] )</pre>

<b>VB Syntax</b>	SetWCS <Parameters>
<b>VB Example</b>	<pre>oEditor.SetWCS Array("NAME:SetWCS Parameter",     "Working Coordinate System:=", "Global",     "RegionDepCSOk:=", False)</pre>

## ShowWindow

Opens the active 3D Modeler window.

<b>UI Access</b>	None.
------------------	-------

<b>Parameters</b>	None.
<b>Return Value</b>	None.

<b>Python Syntax</b>	ShowWindow
<b>Python Example</b>	<code>oEditor.ShowWindow()</code>

<b>VB Syntax</b>	ShowWindow
<b>VB Example</b>	<code>oEditor.ShowWindow</code>

## Simplify

Converts a complex MCAD object into simpler primitives which are easy to mesh and solve.

UI Access	Modeler > Model Preparation > Simplify.		
<b>Parameters</b>	Name <code>&lt;Selections&gt;</code>	Type Array	Description Structured array. See: <a href="#">SelectionsArray</a> .

<b>Parameters</b>	<code>&lt;Parameters&gt;</code>	Array	Structured array.  <code>Array ("NAME:SimplifyParameters",         "Type:=", &lt;string fit type&gt;,         "ExtrusionAxis:=", &lt;string&gt;,         "Cleanup:=", &lt;boolean&gt;,         "Splitting:=", &lt;boolean&gt;),</code>
-------------------	---------------------------------	-------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

			<pre>"SeparateBodies:=", &lt;boolean&gt;, "CloneBody:=", &lt;boolean&gt;, "Generate Primitive History:=", &lt;boolean&gt;, "NumberPointsCurve:=", &lt;integer&gt;, "LengthThresholdCurve:=", &lt;integer&gt;)</pre>
	<b>&lt;CreateGroup&gt;</b>	Array	<p>Structured array.</p> <pre>Array("CreateGroupsForNewObjects:=",&lt;boolean&gt;)</pre>
<b>Return Value</b>	None.		

<b>Python Syntax</b>	Simplify (<Selections>, <Parameters>, <CreateGroup>)
<b>Python Example</b>	<pre>oEditor.Simplify( ["NAME:Selections",  "Selections:=", "Cylinder1,Box2",  "NewPartsModelFlag:=", "Model"], ["NAME:SimplifyParameters",  "Type:=", "Polygon Fit",  "ExtrusionAxis:=", "Auto",  "Cleanup:=", True,  "Splitting:=", True,  "SeparateBodies:=", False,  "CloneBody:=", False,</pre>

```
    "Generate Primitive History:=", True,  
    "NumberPointsCurve:=", 3,  
    "LengthThresholdCurve:=", 20],  
    ["CreateGroupsForNewObjects:=", True]  
)
```

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,       "CloneBody:=", false,       "Generate Primitive History:=", true,</pre>

```

    "NumberPointsCurve:=", 3,
    "LengthThresholdCurve:=", 20),
Array("CreateGroupsForNewObjects:=", true)

```

## Split

Splits the specified object(s) along a plane.

UI Access	<b>Modeler &gt; Boolean &gt; Split.</b>											
Parameters	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;SelectionsArray&gt;</td> <td>Array</td> <td>Structured array. See: <a href="#">SelectionsArray</a>.</td> </tr> <tr> <td>&lt;Parameters&gt;</td> <td>Array</td> <td> <p>Structured array.</p> <pre>         Array ("NAME:SplitToParameters",                "SplitPlane:=", &lt;string "XY", "YZ", "ZX", or                "Dummy"&gt;,                "WhichSide:=", &lt;string "PositiveOnly", "Neg-                ativeOnly" or "Both"&gt;,                "ToolType:=", &lt;string "PlaneTool" or                "FaceTool"&gt;,                "ToolEntityID:=", &lt;-1 if using SplitPlane; else                FaceID&gt;,                "SplitCrossingObjectsOnly:=", &lt;boolean&gt;,                "DeleteInvalidObjects:=", &lt;boolean&gt;) </pre> <p>You can split an object either along an existing plane , or by creating a plane using an edge.</p> <p>For existing plane:</p> </td> </tr> </tbody> </table>	Name	Type	Description	<SelectionsArray>	Array	Structured array. See: <a href="#">SelectionsArray</a> .	<Parameters>	Array	<p>Structured array.</p> <pre>         Array ("NAME:SplitToParameters",                "SplitPlane:=", &lt;string "XY", "YZ", "ZX", or                "Dummy"&gt;,                "WhichSide:=", &lt;string "PositiveOnly", "Neg-                ativeOnly" or "Both"&gt;,                "ToolType:=", &lt;string "PlaneTool" or                "FaceTool"&gt;,                "ToolEntityID:=", &lt;-1 if using SplitPlane; else                FaceID&gt;,                "SplitCrossingObjectsOnly:=", &lt;boolean&gt;,                "DeleteInvalidObjects:=", &lt;boolean&gt;) </pre> <p>You can split an object either along an existing plane , or by creating a plane using an edge.</p> <p>For existing plane:</p>		
Name	Type	Description										
<SelectionsArray>	Array	Structured array. See: <a href="#">SelectionsArray</a> .										
<Parameters>	Array	<p>Structured array.</p> <pre>         Array ("NAME:SplitToParameters",                "SplitPlane:=", &lt;string "XY", "YZ", "ZX", or                "Dummy"&gt;,                "WhichSide:=", &lt;string "PositiveOnly", "Neg-                ativeOnly" or "Both"&gt;,                "ToolType:=", &lt;string "PlaneTool" or                "FaceTool"&gt;,                "ToolEntityID:=", &lt;-1 if using SplitPlane; else                FaceID&gt;,                "SplitCrossingObjectsOnly:=", &lt;boolean&gt;,                "DeleteInvalidObjects:=", &lt;boolean&gt;) </pre> <p>You can split an object either along an existing plane , or by creating a plane using an edge.</p> <p>For existing plane:</p>										

			<ul style="list-style-type: none"> <li>• SplitPlane is "XY", "YZ", or "ZX"</li> <li>• ToolType is "PlaneTool"</li> <li>• ToolEntityID is -1</li> </ul> <p>For an edge:</p> <ul style="list-style-type: none"> <li>• SplitPlane is "Dummy"</li> <li>• ToolType is "EdgeTool"</li> <li>• ToolEntityID is the face ID</li> </ul>
<b>Return Value</b>	None.		

<b>Python Syntax</b>	Split(<SelectionsArray>, <Parameters>)
<b>Python Example</b>	<pre> oEditor.Split(     [         "NAME:Selections",         "Selections:=",         "Box1",         "NewPartsModelFlag:=",         "Model"     ],     [         "NAME:SplitToParameters",         "SplitPlane:=",         "XY",         "WhichSide:=",         "PositiveOnly",         "ToolType:=",         "PlaneTool",         "ToolEntityID:=",         -1     ] ) </pre>

```

    "SplitCrossingObjectsOnly:=", False,
    "DeleteInvalidObjects:=", True
]
)

```

<b>VB Syntax</b>	Split <SelectionsArray> <Parameters>
<b>VB Example</b>	<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.

<b>UI Access</b>	<b>Modeler &gt; Model Preparation &gt; Stitch Sheets.</b>		
<b>Parameters</b>	Name <Selections>	Type Array	Description Structured array. See: <a href="#">SelectionsArray</a> .

	<i>&lt;Parameters&gt;</i>	Array	Structured array.  Array ("NAME:StitchParameters", "MaxTol:=", <integer maximum tolerance>)
<b>Return Value</b>	None.		

<b>Python Syntax</b>	Stitch (<Selections>, <Parameters>)
<b>Python Example</b>	<pre>oEditor.Stitch(     ["NAME:Selections",      "Selections:=", "Rectangle3,Rectangle4",      "NewPartsModelFlag:=", "Model"],     ["NAME:StitchParameters",      "MaxTol:=", -1] )</pre>

<b>VB Syntax</b>	Stitch <Selections>, <Parameters>
<b>VB Example</b>	<pre>oEditor.Stitch Array("NAME:Selections",       "Selections:=", "Rectangle3,Rectangle4",       "NewPartsModelFlag:=", "Model"),</pre>

	<pre>Array("NAME:StitchParameters",       "MaxTol:=", -1)</pre>

## Subtract

Subtracts the specified object(s).

UI Access	Modeler > Boolean > Subtract.		
<b>Parameters</b>	Name <i>&lt;Selections&gt;</i>	Type Array	Description Structured array.  Array("NAME:Selections",       "Blank Parts:=", <string object name(s)>,       "Tool Parts:=", <string object name(s)>)  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.  Array("NAME:SubtractParameters",       "KeepOriginals:=", <boolean>)
<b>Return Value</b>	None.		

<b>Python Syntax</b>	<code>Subtract(&lt;Selections&gt;, &lt;Parameters&gt;)</code>
<b>Python Example</b>	<code>oEditor.Subtract(</code>

```
[ "NAME:Selections",
    "Blank Parts:=", "Rectangle1",
    "Tool Parts:=", "Rectangle2"
],
[ "NAME:SubtractParameters",
    "KeepOriginals:=", False
])
```

<b>VB Syntax</b>	Subtract <Selections> <Parameters>
<b>VB Example</b>	<pre>oEditor.Subtract  Array("NAME:Selections",       "Blank Parts:=", "Rectangle1",       "Tool Parts:=", "Rectangle2")  Array("NAME:SubtractParameters",       "KeepOriginals:=", false)</pre>

## SweepFacesAlongNormal

Sweep the specified face(s) along normal.

<b>UI Access</b>	Modeler > Surface > Sweep Faces Along Normal		
<b>Parameters</b>	Name <SelectionsArray>	Type Array	Description Structured array. See: <a href="#">SelectionsArray</a> .

	<code>&lt;parameters&gt;</code>	Array	Structured array.  Array ("NAME:Parameters", "NAME:SweepFaceAlongNormalToParameters", "FacesToDetach:=", <faceIDarray>, "LengthOfSweep:=", "<value><units>")
<b>Return Value</b>	None		

<b>Python Syntax</b>	<code>SweepFacesAlongNormal(&lt;SelectionsArray&gt; &lt;parameters&gt;)</code>
<b>Python Example</b>	<pre>oEditor.SweepFacesAlongNormal (     ["NAME:Selections",         "Selections:=", "Rectangle1",         "NewPartsModelFlag:=", "Model"],     ["NAME:Parameters",         "NAME:SweepFaceAlongNormalToParameters",         "FacesToDetach:=", [183],         "LengthOfSweep:=", "0.1mm"] )</pre>

<b>VB Syntax</b>	<code>SweepFacesAlongNormal &lt;SelectionsArray&gt; &lt;parameters&gt;</code>
<b>VB Example</b>	<code>oEditor.SweepFacesAlongNormal</code>

```
        Array ("NAME:Selections",
            "Selections:=", "Rectangle1",
            "NewPartsModelFlag:=", "Model"),
        Array ("NAME:Parameters",
            "NAME:SweepFaceAlongNormalToParameters",
            "FacesToDetach:=", Array(57),
            "LengthOfSweep:=", "0.5mm")
    )
```

## ThickenSheet

Thickens a sheet object to convert it to a 3D object.

UI Access	Modeler > Surface > Thicken Sheet		
<b>Parameters</b>	Name <i>&lt;SelectionsArray&gt;</i>	Type Array	Description Structured array. See: <a href="#">SelectionsArray</a> .
	<i>&lt;parameters&gt;</i>	Array	Structured array.  Array ("NAME:SheetThickenParameters",         "Thickness:=" , <string>,         "BothSides:=" , <boolean>)
<b>Return Value</b>	None.		

<b>Python Syntax</b>	ThickenSheet (<SelectionsArray> <parameters>)
<b>Python Example</b>	<pre> oEditor.ThickenSheet(     [         "NAME:Selections",         "Selections:=" , "Rectangle3",         "NewPartsModelFlag:=" , "Model"     ],     [         "NAME:SheetThickenParameters",         "Thickness:=" , "0.01mm",         "BothSides:=" , False     ] ) </pre>

<b>VB Syntax</b>	ThickenSheet <SelectionsArray> <parameters>
<b>VB Example</b>	<pre> oEditor.ThickenSheet     Array("NAME:Selections",         "Selections:=" , "Rectangle3",         "NewPartsModelFlag:=" , "Model")     Array("NAME:SheetThickenParameters",         "Thickness:=" , "0.01mm",         "BothSides:=" , false) ) </pre>

## UncoverFaces

Uncovers the specified face(s).

UI Access	Modeler > Surface > Uncover Faces		
	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: <a href="#">SelectionsArray</a> .
Parameters	<parameters>	Array	Structured array.  Array ("NAME:Parameters",  Array ("NAME:UncoverFacesParameters",  "FacesToUncover:=" , <array of face IDs> ) )
Return Value	None.		

Python Syntax	UncoverFaces(<SelectionsArray> <parameters>)
Python Example	<pre>oEditor.UncoverFaces (     [ "NAME:Selections",         "Selections:="          , "Box1",         "NewPartsModelFlag:="    , "Model"     ],     [ "NAME:Parameters",         [ "NAME:UncoverFacesParameters", </pre>

```

        "FacesToUncover:="      , [12,16,18]
    ]
])

```

<b>VB Syntax</b>	UncoverFaces <SelectionsArray> <parameters>
<b>VB Example</b>	<pre> oEditor.UncoverFaces  Array("NAME:Selections",       "Selections:="      , "Box1",       "NewPartsModelFlag:=" , "Model")  Array("NAME:Parameters",       Array("NAME:UncoverFacesParameters",             "FacesToUncover:="      , Array(12,16,18))) </pre>

## Unite

Unites the specified objects.

UI Access	Modeler > Boolean > Unite		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: <a href="#">SelectionsArray</a> .
	<parameters>	Array	Structured array. Array ("NAME:UniteParameters", "KeepOriginals:=" , <boolean>)

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

<b>Python Syntax</b>	Unite(<SelectionsArray>, <parameters>)
<b>Python Example</b>	<pre>oEditor.Unite(     [         "NAME:Selections",         "Selections:=", "Rectangle1,Rectangle2"     ],     [         "NAME:UniteParameters",         "KeepOriginals:=", False     ] )</pre>

<b>VB Syntax</b>	Unite <SelectionsArray> <parameters>
<b>VB Example</b>	<pre>oEditor.Unite     Array("NAME:Selections",         "Selections:=", "Rectangle1,Rectangle2")     Array("NAME:UniteParameters",         "KeepOriginals:=", false)</pre>

## Ungroup

Ungroups a specified history tree group.

<b>UI Access</b>	<b>Modeler &gt; Group &gt; Ungroup</b>		
<b>Parameters</b>	Name <i>&lt;Groups&gt;</i>	Type Array	Description Structured array. Array("Groups:=", <array of group names to ungroup>)
<b>Return Value</b>	None		

<b>Python Syntax</b>	Ungroup(<Groups>)
<b>Python Example</b>	<code>oEditor.Ungroup(["Groups:=", ["Group1"]])</code>

<b>VB Syntax</b>	Ungroup <Groups>
<b>VB Example</b>	<code>oEditor.Ungroup Array("Groups:=", Array("Group1"))</code>

## WrapSheet

Wraps a sheet object to another object.

<b>UI Access</b>	None.		
<b>Parameters</b>	Name <i>&lt;SelectionsArray&gt;</i>	Type Array	Description Structured array. See: <a href="#">SelectionsArray</a> .
	<i>&lt;Parameters&gt;</i>	Array	Structured array. Array("NAME:WrapSheetParameters", "Imprinted:=", <boolean>)
<b>Return Value</b>	None.		

<b>Python Syntax</b>	WrapSheet (<SelectionsArray>, <Parameters>)
<b>Python Example</b>	<pre>oEditor.WrapSheet(     [         "NAME:Selections",         "Selections:=", "Rectangle1,Box1"     ],     [         "NAME:WrapSheetParameters",         "Imprinted:=", True     ] )</pre>

<b>VB Syntax</b>	WrapSheet <SelectionsArray> <Parameters>
<b>VB Example</b>	<pre>oEditor.WrapSheet     Array("NAME:Selections",         "Selections:=", "Rectangle1,Box1"),     Array("NAME:WrapSheetParameters",         "Imprinted:=", true)</pre>

## Other oEditor Commands

[AddDefinitionFromBlock](#)

---

[AddDefinitionFromLibFile](#)[AddViewOrientation](#)[BreakUDMConnection](#)[ChangeProperty](#)[Delete](#)[FitAll](#)[GeometryCheckAndAutofix](#)[GetBodyNamesByPosition](#)[GetChildNames \[Modeler\]](#)[GetChildObject \[Modeler\]](#)[GetChildTypes \[Modeler\]](#)[GetEdgeByPosition](#)[GetEdgeIDsFromObject](#)[GetEdgeIDsFromFace](#)[GetEntityListIDByName](#)[GetExtendedDefinitionObject](#)[GetFaceArea](#)[GetFaceByPosition](#)[GetFaceCenter](#)[GetFaceIDs](#)[GetGeometryModelerMode](#)

[GetMatchedObjectName](#)

[GetModelBoundingBox](#)

[GetModelUnits](#)

[GetNumObjects](#)

[GetObjectIDByName](#)

[GetObjectName](#)

[GetObjectByNameByFaceID](#)

[GetObjectsByMaterial](#)

[GetObjectsInGroup](#)

[GetObjectVolume](#)

[GetPropertyValue](#)

[GetPropEvaluatedValue](#)

[GetPropNames](#)

[GetPropSIValue](#)

[GetPropValue](#)

[GetSelections](#)

[GetUserPosition](#)

[GetVertexIDsFromEdge](#)

[GetVertexIDsFromFace](#)

[GetVertexIDsFromObject](#)

[GetVertexPosition](#)[OpenExternalEditor](#)[PageSetup](#)[RenamePart](#)[SetPropValue \[Modeler\]](#)

## AddDefinitionFromBlock

Adds a material definition from block text (same definition format as would be contained in the material library file) by library type (using definition folder name). This scripting command directly supports the .AMAT (or .ASURF) definition formats.

<b>UI Access</b>	N/A		
<b>Parameters</b>	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
<b>Return Value</b>	A property scripting object for the definition.		

<b>P- y- t- h- o- n S-</b>	AddDefinitionFromBlock(<defBlock>,<defFolderName>,<newTimeStamp>,<replaceExisting>)
----------------------------------------------------	-------------------------------------------------------------------------------------

<b>y-</b> <b>nt-</b> <b>ax</b>	
<b>P-</b> <b>yt-</b> <b>h-</b> <b>o-</b> <b>n</b> <b>E-</b> <b>x-</b> <b>a-</b> <b>m-</b> <b>pl-</b> <b>e</b>	<pre>oProject = oDesktop.NewProject() oProject.InsertDesign("HFSS", "HFSSDesign1", "DrivenModal", "") oDesign = oProject.SetActiveDesign("HFSSDesign1") oEditor = oDesign.SetActiveEditor("3D Modeler") oEditor.CreateBox()  [      "NAME:BoxParameters",     "XPosition:=" , "-0.4mm",     "YPosition:=" , "-1mm",     "ZPosition:=" , "0mm",     "XSize:=" , "1.4mm",     "YSIZE:=" , "1.6mm",     "ZSize:=" , "0.6mm" ], [</pre>

```
"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()

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  
        ]  
    ]
```

```
[ ]  
]  
])
```

## AddDefinitionFromLibFile

Adds a material definition from a library file (e.g. AMAT file), by name and library type (using definition folder name) . This scripting command directly supports the .AMAT (or .ASURF) definition formats.

<b>UI Access</b>	N/A		
<b>Parameters</b>	Name	Type	Description
	<FilePath>	String	Path of the library file (i.e. AMAT file or ASURF file)
	<defName>	String	Which definition to use, required because a lib file can have multiple definitions.
	<defFolderName>	String	Library type (by definition folder name).
	<newTimeStamp>	String	New timestamp string (time_t as integer, number of seconds since 1/1/1970 12:00am), default is current time
	<replaceExisting>	Boolean	True to replace existing, False to choose a new unique name if an existing definition is found, default is False
<b>Return Value</b>	Property scripting object for the definition.		

<b>Python Syntax</b>	AddDefinitionFromLibFile(<FilePath>, <defName>, <defFolderName>, <newTimeStamp>,<replaceExisting>)
<b>Python Example</b>	<pre>oProject = oDesktop.NewProject() oProject.InsertDesign("HFSS", "HFSSDesign1", "DrivenModal", "")</pre>

```
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",
        "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  
])
```



```
        "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: <a href="#">SelectionsArray</a> .

<b>Name</b>	<b>Type</b>	<b>Description</b>
<SelectionsArray>	Array	Structured array. See: <a href="#">SelectionsArray</a> .
<AttributesArray>	Array	Structured array. See: <a href="#">AttributesArray</a> .

This script supports the following attributes:

- MaterialValue
- SolveInside

			<ul style="list-style-type: none"><li>• ShellElement</li><li>• ShellElementThickness</li><li>• IsMaterialEditable</li><li>• UseMaterialAppearance</li><li>• IsLightweight</li></ul>
<b>Return Value</b>	None.		

<b>Python Syntax</b>	AssignSurfaceMaterial(<SelectionsArray>, <AttributesArray>)
----------------------	-------------------------------------------------------------

**Python Example**

```
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  
    ])
```

**VB Syntax**

```
AssignSurfaceMaterial <SelectionsArray> <AttributesArray>
```

<b>VB Example</b>	<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 SpaceClaim.

<b>UI Access</b>	Break Connection.						
<b>Parameters</b>	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;SelectionsArray&gt;</td> <td>Array</td> <td>Structured array. See: <a href="#">SelectionsArray</a>.</td> </tr> </tbody> </table>	Name	Type	Description	<SelectionsArray>	Array	Structured array. See: <a href="#">SelectionsArray</a> .
Name	Type	Description					
<SelectionsArray>	Array	Structured array. See: <a href="#">SelectionsArray</a> .					
<b>Return Value</b>	None.						

<b>Python Syntax</b>	BreakUDMConnection (<SelectionsArray>)
<b>Python Example</b>	<pre>oEditor.BreakUDMConnection (     [         "NAME:Selections",         "Selections:=" , "SpaceClaim1"     ] )</pre>

<b>VB Syntax</b>	BreakUDMConnection <SelectionsArray>
<b>VB Example</b>	<pre>oEditor.BreakUDMConnection Array("NAME:Selections", "Selections:=", "")</pre>

## ChangeProperty

Changes the properties of an object in the history tree.

<b>UI Access</b>	Right-click an object in the History Tree and select <b>Properties</b> .		
<b>Parameters</b>	Name <propertyArgs>	Type Array	Description Structured array. The properties vary depending on the object. Due to the number of potential configurations, it is recommended that you generate this script using the UI's <b>Automation</b> tab.
<b>Return Value</b>	None.		

<b>Python Syntax</b> <pre>ChangeProperty(&lt;propertyArgs&gt;)</pre>	<b>Example Changing the Position of a Box:</b> <pre>oEditor.ChangeProperty( [     "NAME:AllTabs",     [         "NAME:Geometry3DCmdTab",         [             "NAME:PropServers" ,             "Box1&gt;CreateBox:1"         ],         [             "NAME:ChangedProps",             [                 "NAME:Position",                 "X:="           , "0.35in",                 "Y:="           , "0.55in",                 "Z:="           , "0in"             ]         ]     ] ))</pre>
<b>Python Example</b> <pre>oEditor.ChangeProperty( [     "NAME:AllTabs",     [         "NAME:Geometry3DAttributeTab",         [             "NAME:PropServers" ,             "Box1"</pre>	<b>Example Changing a Box's Material and Wireframe Display:</b> <pre>oEditor.ChangeProperty( [     "NAME:AllTabs",     [         "NAME:Geometry3DAttributeTab",         [             "NAME:PropServers" ,             "Box1"</pre>

```
        ],
        [
            "NAME:ChangedProps",
            [
                "NAME:Material",
                "Value:="      , "\\"vacuum\\"
            ],
            [
                "NAME:Display Wireframe",
                "Value:="      , True
            ]
        ]
    )
)
```

VB Syntax	ChangeProperty <propertyArgs>
<b>VB Example</b>	<p><b>Example Changing the Position of a Box:</b></p> <pre>oEditor.ChangeProperty Array("NAME:AllTabs",     Array("NAME:Geometry3DCmdTab",         Array("NAME:PropServers" ,         "Box1&gt;CreateBox:1"     ),     Array("NAME:ChangedProps",         Array("NAME:Position",             "X:="      , "0.35in",             "Y:="      , "0.55in",             "Z:="      , "0in"         )     ) )</pre>

```
    )  
    )  
Example C  
oEditor.Create  
Array("NA")  
    Array  
        A  
        "  
    )  
    A  
)  
)  
)
```

## 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
                )
            )
        )
    )
)
```

## **CloseAllWindows[Editor]**

Closes all windows belong to current 3D Modeler editor.

<b>UI Access</b>	N/A
<b>Parameters</b>	None.
<b>Return Value</b>	None.

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

<b>VB Syntax</b>	CloseAllWindows
<b>VB Example</b>	<code>oEditor.CloseAllWindows</code>

## Defeature

Removes irrelevant features from a primitive.

<b>UI Access</b>	N/A									
<b>Parameters</b>	<table border="1"><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;Selections&gt;</td><td>Array</td><td>Structured array. See: <a href="#">SelectionsArray</a>.</td></tr><tr><td>&lt;Options&gt;</td><td>Array</td><td>Structured array.  Array ("NAME:options",  "tolerance:=", &lt;double containing tolerance value&gt;,  "fix:=", &lt;boolean, if true, then self-intersections are fixed.&gt;)</td></tr></table>	Name	Type	Description	<Selections>	Array	Structured array. See: <a href="#">SelectionsArray</a> .	<Options>	Array	Structured array.  Array ("NAME:options",  "tolerance:=", <double containing tolerance value>,  "fix:=", <boolean, if true, then self-intersections are fixed.>)
Name	Type	Description								
<Selections>	Array	Structured array. See: <a href="#">SelectionsArray</a> .								
<Options>	Array	Structured array.  Array ("NAME:options",  "tolerance:=", <double containing tolerance value>,  "fix:=", <boolean, if true, then self-intersections are fixed.>)								
<b>Return Value</b>	None.									

<b>Python Syntax</b>	Defeature(<Selections>, <Options>)
<b>Python Example</b>	<code>oEditor.Defeature(</code>

	<pre>[ "NAME:Selections", "Selections:=", "Box1"], [ "NAME:Options", "Tolerance:=", 1E-006, "Fix:=", True])</pre>
--	-------------------------------------------------------------------------------------------------------------------

<b>VB Syntax</b>	Defeature <Selections>, <Options>
<b>VB Example</b>	<pre>oEditor.Defeature Array("NAME:Selections", "Selections:=", "Box1"), Array("NAME:Options", "tolerance:=", 1E-006, "fix:=", true)</pre>

## Delete

Deletes the specified object(s).

<b>UI Access</b>	Edit > Delete.			
<b>Parameters</b>	<table border="1"> <tr> <td>Name &lt;SelectionsArray&gt;</td> <td>Type Array</td> <td>Description Structured array. See: <a href="#">SelectionsArray</a>.</td> </tr> </table>	Name <SelectionsArray>	Type Array	Description Structured array. See: <a href="#">SelectionsArray</a> .
Name <SelectionsArray>	Type Array	Description Structured array. See: <a href="#">SelectionsArray</a> .		
<b>Return Value</b>	None.			

<b>Python Syntax</b>	Delete(<SelectionsArray>)
<b>Python Example</b>	<pre>oEditor.Delete(     [ "NAME:Selections",       "Selections:=", "Rectangle1,Rectangle2"     ] )</pre>

<b>VB Syntax</b>	Delete <SelectionsArray>
<b>VB Example</b>	<code>oEditor.Delete Array("NAME:Selections", "Selections:=", "Rectangle1,Rectangle2")</code>

## FitAll

Fits the design to the modeling area.

<b>UI Access</b>	<b>View &gt; Fit All &gt; All Views.</b>
<b>Parameters</b>	None.
<b>Return Value</b>	None.

<b>Python Syntax</b>	<code>FitAll()</code>
<b>Python Example</b>	<code>oEditor.FitAll()</code>

<b>VB Syntax</b>	<code>FitAll()</code>
<b>VB Example</b>	<code>Set oEditor = oDesign.SetActiveEditor("3D Modeler") oEditor.FitAll()</code>

## GenerateAllUserDefinedModels

Generates all user defined models.

<b>UI Access</b>	N/A
<b>Parameters</b>	None.
<b>Return Value</b>	Array of models generated.

<b>Python Syntax</b>	GenerateAllUserDefinedModels ()
<b>Python Example</b>	<code>oEditor.GenerateAllUserDefinedModels ()</code>

<b>VB Syntax</b>	GenerateAllUserDefinedModels
<b>VB Example</b>	<code>oEditor.GenerateAllUserDefinedModels</code>

## GenerateUserDefinedModel

Generates specified user defined model(s).

<b>UI Access</b>	N/A		
<b>Parameters</b>	Name <i>&lt;SelectionsArray&gt;</i>	Type Array	Description Structured array. See: <a href="#">SelectionsArray</a> .
<b>Return Value</b>	Array of models generated.		

<b>Python Syntax</b>	GenerateUserDefinedModel (<SelectionsArray>)
<b>Python Example</b>	<code>oEditor.GenerateUserDefinedModel (["NAME:Selections", "Selections:=", "model1,</code>

	model2" ] )
--	-------------

<b>VB Syntax</b>	GenerateAllUserDefinedModels
<b>VB Example</b>	<code>oEditor.GenerateUserDefinedModel Array ("NAME:Selections", "Selections:=", "model1, model2")</code>

## 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 <i>&lt;positionParameters&gt;</i>	Type Array	Description Structured array containing position coordinates for active coordinate system.  Array ("NAME:Parameters", "XPosition:=", <string>, "YPosition:=", <string>, "ZPosition:=", <string>)
Return Value	Array containing string object names.		

### Python Syntax

```
GetBodyNamesByPosition (<positionParameters>)
```

<b>Python Example</b>	<pre><code>oEditor.GetBodyNamesByPosition(     [         "NAME:Parameters",         "XPosition:=", "0mm",         "YPosition:=", "15mm",         "ZPosition:=", "0mm"     ] )</code></pre>
-----------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

<b>VB Syntax</b>	<b>GetBodyNamesByPosition &lt;positionParameters&gt;</b>
<b>VB Example</b>	<pre><code>oEditor.GetBodyNamesByPosition Array("NAME:Parameters", _     "XPosition:=", "0mm", _     "YPosition:=", "15mm", _     "ZPosition:=", "0mm")</code></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.

<b>UI Access</b>	N/A
------------------	-----

	Name	Type	Description
Parameters	<category>	String	<p><i>Optional.</i> Passing no input returns the list of possible strings:</p> <ul style="list-style-type: none"> <li>• "AllParts" – Returns the names of all parts.</li> <li>• "CoordinateSystems" – Returns the names of all coordinate systems.</li> <li>• "Groups" – Returns the names of all groups.</li> <li>• "Lists" – Returns the names of all lists.</li> <li>• "ModelParts" – Returns names of model parts.</li> <li>• "NonModelParts" – Returns the names of non-model parts.</li> <li>• "Planes" – Returns the names of all planes.</li> <li>• "Points" – Returns the names of all points.</li> <li>• "SubmodelDefinitions" – Returns the names of sub-model definitions.</li> </ul>
Return Value	Array containing object names in the selected category.		

Python Syntax	GetChildNames(<category>)
Python Example	<p><b>Standalone Example:</b></p> <pre>oEditor.GetChildNames ("ModelParts") oEditor.GetChildNames ("Points")</pre> <p><b>Example used in Conjunction with <a href="#">GetChildObject</a> and <a href="#">GetPropNames</a>:</b></p>

```

oDesign = oProject.GetActiveDesign()
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".



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".

```

<b>VB Syntax</b>	GetChildNames <category>
<b>VB Example</b>	<p><b>Standalone Example:</b></p> <pre> oEditor.GetChildNames "ModelParts" oEditor.GetChildNames "Points"</pre> <p><b>Example used in Conjunction with <a href="#">GetChildObject</a> and <a href="#">GetPropNames</a>:</b></p>

```
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/SIVValue", "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.

<b>UI Access</b>	N/A		
<b>Parameters</b>	Name <code>&lt;object&gt;</code>	Type String	Description In this case, "3D Modeler".
<b>Return Value</b>	Returns the 3D Modeler object. In the examples below, it is assigned to the variable oEditor.		

<b>Python Syntax</b>	<code>GetChildObject(&lt;object&gt;)</code>
	<p><b>Standalone Example:</b></p> <pre>oDesign = oProject.GetActiveDesign() oEditor = oDesign.GetChildObject("3D Modeler")</pre> <p><b>Example used in Conjunction with <a href="#">GetChildNames</a> and <a href="#">GetPropNames</a>:</b></p>
<b>Python Example</b>	<pre>oDesign = oProject.GetActiveDesign() oModel = oDesign.GetChildObject("3D Modeler") oModel.GetChildTypes()</pre> <ul style="list-style-type: none"> <li>This returns an array containing strings: "ModelParts", "AllParts", "NonModelParts", "CoordinateSystems", "Points", "Planes", "SubmodelDefinitions", "Groups", and "Lists".</li> </ul>

```
oModel.GetChildNames ("ModelParts")
    • This returns an array containing string model parts. For example: "WG_Interior", "WG", "MT_Interior",
      "MT", "Box2", "Cylinder1".
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", "Mater-
      ial/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>
<b>VB Example</b>	<p><b>Standalone Example:</b></p> <pre>Dim oAnsoftApp Dim oDesktop Dim oProject Dim oDesign Dim oEditor Set oDesign = oProject.GetActiveDesign() Set oEditor = oDesign.GetChildObject "3D Modeler"</pre>

**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", "Mater-  
      ial/SIValue", "Material/EvaluatedValue", "Solve Inside", "Orientation", "Orientation/Choices", "Model", "Dis-  
      play 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.

<b>UI Access</b>	N/A
<b>Parameters</b>	None.
<b>Return Value</b>	Array containing the names of child types.

<b>Python Syntax</b>	GetChildTypes()
<b>Python Example</b>	<p><b>Standalone Example:</b></p> <pre>oDesign = oProject.GetActiveDesign() oEditor = oDesign.SetActiveEditor("3D Modeler") oEditor.GetChildTypes()</pre> <ul style="list-style-type: none"><li>• This returns an array containing strings: "ModelParts", "AllParts", "NonModelParts", "CoordinateSystems", "Points", "Planes", "SubmodelDefinitions", "Groups", and "Lists".</li></ul> <p><b>Example Used in Conjunction with <a href="#">GetChildNames</a>:</b></p> <pre>oProject = oDesktop.GetActiveProject()</pre>

	<pre><code>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".</code></pre> <p><code>oDesign.GetChildTypes()</code></p> <ul style="list-style-type: none"> <li>• This returns an array containing the child types. For example: "Module", "Editor", and "Variable".</li> </ul>
--	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

VB Syntax	GetChildTypes()
VB Example	<p><b>Standalone Example:</b></p> <pre><code>Dim oDesign Dim oProject Dim oEditor Set oDesign = oProject.GetActiveDesign() Set oEditor = oDesign.SetActiveEditor "3D Modeler" oEditor.GetChildTypes()</code></pre> <ul style="list-style-type: none"> <li>• This returns an array containing strings: "ModelParts", "AllParts", "NonModelParts", "CoordinateSystems", "Points", "Planes", "SubmodelDefinitions", "Groups", and "Lists".</li> </ul> <p><b>Example Used in Conjunction with <a href="#">GetChildNames</a>:</b></p> <pre><code>Dim oDesign Dim oProject</code></pre>

```
Dim oDesign  
  
Set oDesign = oProject.GetActiveDesign()  
  
Set 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".  
  
Set oDesign.GetChildTypes()  
  
• 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 <i>&lt;positionParameters&gt;</i>	Type Array	Description Structured array.  Array ("NAME:EdgeParameters", "BodyName:=", <string object name>, "Xposition:=", <string>, "YPosition:=", <string>, "ZPosition:=", <string>)  <b>Note:</b>

		For 2D XY Designs, ZPosition should be set to "0". For 2D RZ Designs, YPosition should be set to "0".
<b>Return Value</b>	Array containing string edge IDs.	

<b>Python Syntax</b>	GetEdgeByPosition (<positionParameters>)
<b>Python Example</b>	<pre> oEditor.GetEdgeByPosition(     [         "NAME:EdgeParameters",         "BodyName:=", "Box1",         "Xposition:=", "10mm",         "YPosition:=", "0mm",         "ZPosition:=", "10mm"     ] ) </pre>

<b>VB Syntax</b>	GetEdgeByPosition <positionParameters>
<b>VB Example</b>	<pre> oEditor.GetEdgeByPosition Array("NAME:EdgeParameters",     "BodyName:=", "Box1",     "Xposition:=", "10mm",     "YPosition:=", "0mm",     "ZPosition:=", "10mm") </pre>

## GetEdgeIDFromNameForFirstOperation

Gets edge ID from first operation of a part.

<b>UI Access</b>	N/A		
<b>Parameters</b>	Name	Type	Description
	<PartName>	String	Name of specified part.
<b>Return Value</b>	Integer edge ID		

<b>Python Syntax</b>	GetEdgeIDFromNameForFirstOperation(<PartName>, <EdgeName>)
<b>Python Example</b>	oEditor.GetEdgeIDFromNameForFirstOperation("Coil_1", "Edge_4510")

<b>VB Syntax</b>	GetEdgeIDFromNameForFirstOperation <PartName>, <EdgeName>
<b>VB Example</b>	oEditor.GetEdgeIDFromNameForFirstOperation "Coil_1", "Edge_4510"

## GetEdgeIDsFromFace

Returns the edge IDs for a specified Face ID.

<b>UI Access</b>	N/A		
<b>Parameters</b>	Name	Type	Description
	<faceID>	Integer	ID of the specified face.

<b>Return Value</b>	Array containing string edge IDs.
---------------------	-----------------------------------

<b>Python Syntax</b>	GetEdgeIDsFromFace (<faceID>)
<b>Python Example</b>	<code>oEditor.GetEdgeIDsFromFace (20)</code>

<b>VB Syntax</b>	GetEdgeIDsFromFace <faceID>
<b>VB Example</b>	<code>oEditor.GetEdgeIDsFromFace 20</code>

## GetEdgeIDsFromObject

Returns the edge IDs for a specified object.

<b>UI Access</b>	N/A								
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;object&gt;</td> <td>String</td> <td>Object name.</td> </tr> </tbody> </table>			Name	Type	Description	<object>	String	Object name.
Name	Type	Description							
<object>	String	Object name.							
<b>Return Value</b>	Array containing string edge IDs.								

<b>Python Syntax</b>	GetEdgeIDsFromObject (<object>)
<b>Python Example</b>	<code>oEditor.GetEdgeIDsFromObject ("Box1")</code>

<b>VB Syntax</b>	GetEdgeIDsFromObject <object>
------------------	-------------------------------

**VB Example**

```
oEditor.GetEdgeIDsFromObject "Box1"
```

## GetEdgeLength

Returns the length of edges for a specified Face ID.

<b>UI Access</b>	N/A		
<b>Parameters</b>	Name <i>&lt;faceID&gt;</i>	Type Integer	Description ID of the specified face.
<b>Return Value</b>	Integer containing the length of found edges.		

**Python Syntax**

```
GetEdgeLength (<faceID>)
```

**Python Example**

```
oEditor.GetEdgeLength (20)
```

<b>VB Syntax</b>	GetEdgeLength <faceID>		
<b>VB Example</b>	oEditor.GetEdgeLength 20		

## GetEntityListIDByName

Returns a list of IDs in a given entity list (either object or face). See: [CreateEntityList](#).

<b>UI Access</b>	N/A		
<b>Parameters</b>	Name <i>&lt;listName&gt;</i>	Type String	Description List name.

<b>Return Value</b>	Array containing string object IDs.
---------------------	-------------------------------------

<b>Python Syntax</b>	<code>GetEntityListIDByName (&lt;listName&gt;)</code>
<b>Python Example</b>	<code>oEditor.GetEntityListIDByName ("MyList")</code>

<b>VB Syntax</b>	<code>GetEntityListIDByName &lt;listName&gt;</code>
<b>VB Example</b>	<code>oEditor.GetEntityListIDByName "MyList"</code>

## GetExtendedDefinitionObject

Get an object-oriented property scripting object by name and library type (using definition folder name) which also supports getting/setting mod time and getting/setting generic string attributes. This scripting command directly supports the .AMAT (or .ASURF) definition formats.

<b>UI Access</b>	N/A		
<b>Parameters</b>	Name	Type	Description
	<code>&lt;defName&gt;</code>	String	Definition to retrieve.
	<code>&lt;defFolderName&gt;</code>	String	Library type (by definition folder name).
<b>Return Value</b>	An extended material wrapper script object (see material wrapper script object functions above).		

P- yt- h- o-	GetExtendedDefinitionObject(<defName>, <defFolderName>)
-----------------------	---------------------------------------------------------

<b>n s- y- nt- ax</b>	<pre> oProject = oDesktop.GetActiveProject() oDefMgr = 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'" oDefMgr.AddDefinitionFromBlock(defBlock, 'Materials', '10101010', True) oMat = oDefMgr.<b>GetExtendedDefinitionObject</b>('vacuum2', 'Materials') # use extended definition object function(s) to manipulate the definition oMat.SetModTime("1630611487") </pre>
---------------------------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

## GetFaceArea

Returns the area of a specified face.

<b>UI Access</b>	N/A						
<b>Parameters</b>	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 33%;">Name</th> <th style="width: 33%;">Type</th> <th style="width: 33%;">Description</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;"><i>&lt;faceID&gt;</i></td> <td style="text-align: center;">Integer</td> <td>ID of specified face.</td> </tr> </tbody> </table>	Name	Type	Description	<i>&lt;faceID&gt;</i>	Integer	ID of specified face.
Name	Type	Description					
<i>&lt;faceID&gt;</i>	Integer	ID of specified face.					
<b>Return Value</b>	Long integer of face area.						

---

<b>Python Syntax</b>	GetFaceArea (<faceID>)
<b>Python Example</b>	<code>oEditor.GetFaceArea (19)</code>

<b>VB Syntax</b>	GetFaceArea <faceID>
<b>VB Example</b>	<code>oEditor.GetFaceArea 19</code>

## GetFaceCenter

Returns the center position of a specified face.

<b>UI Access</b>	N/A		
<b>Parameters</b>	Name <planarFaceID>	Type Long	Description Face ID
<b>Return Value</b>	Array containing string X, Y, Z coordinates.		

<b>Python Syntax</b>	GetFaceCenter (<planarFaceID>)
<b>Python Example</b>	<code>oEditor.GetFaceCenter (12)</code>

<b>VB Syntax</b>	GetFaceCenter <planarFaceID>
<b>VB Example</b>	<code>oEditor.GetFaceCenter 12</code>

## GetFaceByPosition

Returns the face ID located at a specified position.

**Note:**

The coordinates must point to exactly one face, not a vertex or edge where two or more faces join.

<b>UI Access</b>	N/A		
<b>Parameters</b>	Name <i>&lt;FaceParameters&gt;</i>	Type Array	Description Structured Array.  Array ("NAME:FaceParameters", "BodyName:=", <string>, "XPosition:=", <string>, "YPosition:=", <string>, "ZPosition:=", <string>)
<b>Return Value</b>	Integer Face ID.		

<b>Python Syntax</b>	GetFaceByPosition( <i>&lt;FaceParameters&gt;</i> )
<b>Python Example</b>	<pre>oEditor.GetFaceByPosition(     [ "NAME:FaceParameters",         "BodyName:=", "Box1",         "XPosition:=", "0.2mm",         "YPosition:=", "-0.2mm",</pre>

	<pre>"ZPosition:=", "0.4mm" ])</pre>
--	--------------------------------------

<b>VB Syntax</b>	<code>GetFaceByPosition &lt;FaceParameters&gt;</code>
<b>VB Example</b>	<pre>dim FaceID  FaceID = oEditor.GetFaceByPosition Array("NAME:FaceParameters",                                          "BodyName:=", "Box1",                                          "XPosition:=", "0.2mm",                                          "YPosition:=", "-0.2mm",                                          "ZPosition:=", "0.4mm")</pre>

## GetFaceIDFromNameForFirstOperation

Gets face ID from first operation of a part.

<b>UI Access</b>	N/A		
<b>Parameters</b>	Name	Type	Description
	<PartName>	String	Name of specified part.
<b>Return Value</b>	Integer face ID		

<b>Python Syntax</b>	<code>GetFaceIDFromNameForFirstOperation(&lt;PartName&gt;, &lt;FaceName&gt;)</code>
<b>Python Example</b>	<code>oEditor.GetFaceIDFromNameForFirstOperation("Rotor", "Face_14343"))</code>

<b>VB Syntax</b>	GetFaceIDFromNameForFirstOperation <PartName>, <FaceName>
<b>VB Example</b>	<code>oEditor.GetFaceIDFromNameForFirstOperation "Rotor", "Face_14343"</code>

## GetFaceIDs

Returns the face IDs associated with a specified object.

<b>UI Access</b>	N/A						
<b>Parameters</b>	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>&lt;objectName&gt;</td><td>String</td><td>Object name.</td></tr></tbody></table>	Name	Type	Description	<objectName>	String	Object name.
Name	Type	Description					
<objectName>	String	Object name.					
<b>Return Value</b>	Array containing string face IDs.						

<b>Python Syntax</b>	GetFaceIDs (<objectName>)
<b>Python Example</b>	<code>oEditor.GetFaceIDs ('Box1')</code>

<b>VB Syntax</b>	GetFaceIDs <objectName>
<b>VB Example</b>	<code>oEditor.GetFaceIDs "Box1"</code>

## GetFaceIDsOfSheet

Returns the face IDs associated with a specified sheet object.

<b>UI Access</b>	N/A								
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;sheetName&gt;</td> <td>String</td> <td>Name of specified sheet object.</td> </tr> </tbody> </table>			Name	Type	Description	<sheetName>	String	Name of specified sheet object.
Name	Type	Description							
<sheetName>	String	Name of specified sheet object.							
<b>Return Value</b>	Array containing string face IDs.								

<b>Python Syntax</b>	GetFaceIDsOfSheet(<sheetName>)
<b>Python Example</b>	<code>oEditor.GetFaceIDsOfSheet ('Sheet1')</code>

<b>VB Syntax</b>	GetFaceIDsOfSheet <sheetName>
<b>VB Example</b>	<code>oEditor.GetFaceIDsOfSheet "Sheet1"</code>

## GetMatchedObjectName

Returns all object names containing the input text string.

<b>UI Access</b>	N/A								
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;wildcardText&gt;</td> <td>String</td> <td>Text string present in object name(s).</td> </tr> </tbody> </table>			Name	Type	Description	<wildcardText>	String	Text string present in object name(s).
Name	Type	Description							
<wildcardText>	String	Text string present in object name(s).							
<b>Return Value</b>	Array containing string object names.								

<b>Python Syntax</b>	GetMatchedObjectName (<wildcardText>)
<b>Python Example</b>	<code>oEditor.GetMatchedObjectName ('Box*')</code>

	<code>oEditor.GetMatchedObjectName( '?ox?' )</code>
--	-----------------------------------------------------

<b>VB Syntax</b>	<code>GetMatchedObjectName &lt;wildcardText&gt;</code>
------------------	--------------------------------------------------------

<b>VB Example</b>	<code>oEditor.GetMatchedObjectName "Box*"</code>
-------------------	--------------------------------------------------

## GetModelBoundingBox

Returns the bounding box of the current model.

<b>UI Access</b>	N/A
<b>Parameters</b>	None.
<b>Return Value</b>	Array containing string Xmin, Ymin, Zmin, Xmax, Ymax, and Zmax values of the bounding box.

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

<b>Python Example</b>	<code>oEditor.GetModelBoundingBox()</code>
-----------------------	--------------------------------------------

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

<b>VB Example</b>	<code>oEditor.GetModelBoundingBox</code>
-------------------	------------------------------------------

## GetModelUnits

Returns the model's unit of measure.

<b>UI Access</b>	N/A
<b>Parameters</b>	None.
<b>Return Value</b>	String containing unit of measure.

<b>Python Syntax</b>	GetModelUnits()
<b>Python Example</b>	<code>oEditor.GetModelUnits ()</code>

<b>VB Syntax</b>	GetModelUnits
<b>VB Example</b>	<code>oEditor.GetModelUnits</code>

## GetNumObjects

Returns the number of objects in a design.

<b>UI Access</b>	N/A
<b>Parameters</b>	None.
<b>Return Value</b>	Integer number of objects.

<b>Python Syntax</b>	GetNumObjects()
<b>Python Example</b>	<code>oEditor.GetNumObjects ()</code>

<b>VB Syntax</b>	GetNumObjects
<b>VB Example</b>	<code>oEditor.GetNumObjects (</code>

## GetObjectIDByName

Given an object's name, returns its ID. IDs are used with [CreateEntityList](#).

<b>UI Access</b>	N/A						
<b>Parameters</b>	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td><code>&lt;objectName&gt;</code></td><td>String</td><td>Object name.</td></tr></tbody></table>	Name	Type	Description	<code>&lt;objectName&gt;</code>	String	Object name.
Name	Type	Description					
<code>&lt;objectName&gt;</code>	String	Object name.					
<b>Return Value</b>	Integer object ID.						

<b>Python Syntax</b>	<code>GetObjectIDByName(&lt;objectName&gt;)</code>
<b>Python Example</b>	<code>oEditor.GetObjectIDByName ('Box1')</code>

<b>VB Syntax</b>	<code>GetObjectIDByName &lt;objectName&gt;</code>
<b>VB Example</b>	<code>oEditor.GetObjectIDByName "Box1"</code>

## GetObjectName

Returns an object's name from its specified base index (creation order).

<b>UI Access</b>	N/A
------------------	-----

<b>Parameters</b>	Name <i>&lt;index&gt;</i>	Type Integer	Description Base index (where '0' is the first item created)
<b>Return Value</b>	String containing object name.		

<b>Python Syntax</b>	GetObjectName(< <i>index</i> >)
<b>Python Example</b>	<code>oEditor.GetObjectName(3)</code>

<b>VB Syntax</b>	GetObjectName < <i>index</i> >
<b>VB Example</b>	<code>oEditor.GetObjectName 3</code>

## GetObjectNameByEdgeID

Returns an object name given an edge ID.

<b>UI Access</b>	N/A
<b>Parameters</b>	Name <i>&lt;EdgeID&gt;</i>
	Type Integer
	Description The edge ID.
<b>Return Value</b>	String containing object name.

<b>Python Syntax</b>	GetObjectNameByEdgeID(< <i>EdgeID</i> >)
<b>Python Example</b>	<code>oEditor.GetObjectNameByEdgeID(88)</code>

<b>VB Syntax</b>	GetObjectByNameByEdgeID <EdgeID>
<b>VB Example</b>	<code>oEditor.GetObjectByNameByEdgeID 10</code>

## GetObjectByNameByFaceID

Returns an object name given a face ID.

<b>UI Access</b>	N/A								
<b>Parameters</b>	<table border="1"><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;FaceID&gt;</td><td>Integer</td><td>The face ID.</td></tr></table>			Name	Type	Description	<FaceID>	Integer	The face ID.
Name	Type	Description							
<FaceID>	Integer	The face ID.							
<b>Return Value</b>	String containing object name.								

<b>Python Syntax</b>	GetObjectByNameByFaceID (<FaceID>)
<b>Python Example</b>	<code>oEditor.GetObjectByNameByFaceID (88)</code>

<b>VB Syntax</b>	GetObjectByNameByFaceID <FaceID>
<b>VB Example</b>	<code>oEditor.GetObjectByNameByFaceID 10</code>

## GetObjectByNameByID

Returns an object name given an ID.

<b>UI Access</b>	N/A
------------------	-----

<b>Parameters</b>	<table border="1"> <tr> <td>Name</td><td>Type</td><td>Description</td></tr> <tr> <td>&lt;ObjID&gt;</td><td>Integer</td><td>The object ID.</td></tr> </table>	Name	Type	Description	<ObjID>	Integer	The object ID.
Name	Type	Description					
<ObjID>	Integer	The object ID.					
<b>Return Value</b>	String containing object name.						

<b>Python Syntax</b>	GetObjectByNameByID(<ObjID>)
<b>Python Example</b>	<code>oEditor.GetObjectNameByID(88)</code>

<b>VB Syntax</b>	GetObjectByNameByID <ObjID>
<b>VB Example</b>	<code>oEditor.GetObjectNameByID 10</code>

## GetObjectByNameByVertexID

Returns an object name given a vertex ID.

<b>UI Access</b>	N/A						
<b>Parameters</b>	<table border="1"> <tr> <td>Name</td> <td>Type</td> <td>Description</td> </tr> <tr> <td>&lt;VertexID&gt;</td> <td>Integer</td> <td>The vertex ID.</td> </tr> </table>	Name	Type	Description	<VertexID>	Integer	The vertex ID.
Name	Type	Description					
<VertexID>	Integer	The vertex ID.					
<b>Return Value</b>	String containing object name.						

<b>Python Syntax</b>	GetObjectByNameByVertexID(<VertexID>)
<b>Python Example</b>	<code>oEditor.GetObjectNameByVertexID(88)</code>

<b>VB Syntax</b>	GetObjectByNameByVertexID <VertexID>
<b>VB Example</b>	<code>oEditor.GetObjectByNameByVertexID 10</code>

## GetObjectsByMaterial

Returns a list of objects of a specified material.

<b>UI Access</b>	N/A						
<b>Parameters</b>	<table border="1"><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;material&gt;</td><td>String</td><td>Material name.</td></tr></table>	Name	Type	Description	<material>	String	Material name.
Name	Type	Description					
<material>	String	Material name.					
<b>Return Value</b>	Array containing string object names.						

<b>Python Syntax</b>	GetObjectsByMaterial (<material>)
<b>Python Example</b>	<code>oEditor.GetObjectsByMaterial('copper')</code>

<b>VB Syntax</b>	GetObjectsByMaterial <material>
<b>VB Example</b>	<code>oEditor.GetObjectsByMaterial "copper"</code>

## GetObjectShapeType

Returns the shape type of a specified object.

<b>UI Access</b>	N/A
------------------	-----

Parameters	Name	Type	Description
	<objName>	String	Name of specified object.
	<csName>	String	Name of coordinate system of the object.
<b>Return Value</b>	String containing shape type.		

<b>Python Syntax</b>	GetObjectShapeType(<objName>, <csName>)
<b>Python Example</b>	<code>oEditor.GetObjectShapeType ("box1", "Global")</code>

<b>VB Syntax</b>	GetObjectShapeType <objName>, <csName>
<b>VB Example</b>	<code>oEditor.GetObjectShapeType "box1", "Global"</code>

## GetObjectsInGroup

Returns a list of objects in a specified group.

<b>UI Access</b>	N/A								
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;group&gt;</td> <td>String</td> <td>           Group name.            One of:  <ul style="list-style-type: none"> <li>• "&lt;materialName&gt;"</li> <li>• "&lt;assignmentName&gt;"</li> <li>• "Non Model"</li> <li>• "Solids"</li> </ul> </td> </tr> </tbody> </table>			Name	Type	Description	<group>	String	Group name. One of: <ul style="list-style-type: none"> <li>• "&lt;materialName&gt;"</li> <li>• "&lt;assignmentName&gt;"</li> <li>• "Non Model"</li> <li>• "Solids"</li> </ul>
Name	Type	Description							
<group>	String	Group name. One of: <ul style="list-style-type: none"> <li>• "&lt;materialName&gt;"</li> <li>• "&lt;assignmentName&gt;"</li> <li>• "Non Model"</li> <li>• "Solids"</li> </ul>							

			<ul style="list-style-type: none"><li>• "Unclassified"</li><li>• "Sheets"</li><li>• "Lines"</li></ul>
<b>Return Value</b>	Array containing string object names.		

<b>Python Syntax</b>	GetObjectsInGroup (<group>)
<b>Python Example</b>	<code>oEditor.GetObjectsInGroup ('Sheets')</code>

<b>VB Syntax</b>	GetObjectsInGroup <group>
<b>VB Example</b>	<code>oEditor.GetObjectsInGroup "Sheets"</code>

## GetObjectVolume

Returns an object's volume from its name).

<b>UI Access</b>	N/A								
<b>Parameters</b>	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>&lt;name&gt;</td><td>String</td><td>Object name</td></tr></tbody></table>			Name	Type	Description	<name>	String	Object name
Name	Type	Description							
<name>	String	Object name							
<b>Return Value</b>	.Real								

---

<b>Python Syntax</b>	GetObjectVolume(<name>)
<b>Python Example</b>	<code>oEditor.GetObjectVolume("Box1")</code>

<b>VB Syntax</b>	GetObjectVolume <name>
<b>VB Example</b>	<code>oEditor.GetObjectVolume "Box1"</code>

### GetObjPath [Editor]

Obtains the path to the 3D modeler.

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

<b>Python Syntax</b>	GetObjPath()
<b>Python Example</b>	<code>oEditor.GetObjPath ()</code>

<b>VB Syntax</b>	GetObjPath
<b>VB Example</b>	<code>oEditor.GetObjPath</code>

### GetPartsForUserDefinedModel

Obtains parts from a specified user defined model.

<b>UI Access</b>	N/A								
<b>Parameters</b>	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>&lt;udmName&gt;</td><td>String</td><td>Name of user defined model.</td></tr></tbody></table>			Name	Type	Description	<udmName>	String	Name of user defined model.
Name	Type	Description							
<udmName>	String	Name of user defined model.							
<b>Return Value</b>	Array of strings containing associated parts.								

<b>Python Syntax</b>	GetPartsForUserDefinedModel (<udmName>)
<b>Python Example</b>	<code>oEditor.GetPartsForUserDefinedModel ("OnDieSpiralInductor1")</code>

<b>VB Syntax</b>	GetPartsForUserDefinedModel <udmName>
<b>VB Example</b>	<code>oEditor.GetPartsForUserDefinedModel "OnDieSpiralInductor1"</code>

## GetPoints [3D Modeler Editor]

Returns all the points defined in current 3D modeler.

<b>UI Access</b>	N/A
<b>Parameters</b>	None.
<b>Return Value</b>	Array of strings containing names of points.

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

<b>Python Example</b>	<code>oEditor.GetPoints()</code>
-----------------------	----------------------------------

<b>VB Syntax</b>	<code>GetPoints</code>
<b>VB Example</b>	<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.

<b>UI Access</b>	N/A								
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><code>&lt;PropName&gt;</code></td> <td>String</td> <td>Name of the property.</td> </tr> </tbody> </table>			Name	Type	Description	<code>&lt;PropName&gt;</code>	String	Name of the property.
Name	Type	Description							
<code>&lt;PropName&gt;</code>	String	Name of the property.							
<b>Return Value</b>	String value of the evaluated value.								

<b>Python Syntax</b>	<code>GetPropEvaluatedValue (&lt;PropName&gt;)</code>
<b>Python Example</b>	<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		
<b>Parameters</b>	Name <i>&lt;PropTab&gt;</i>	Type String	Description One of the following, where tab titles are shown in parentheses: <ul style="list-style-type: none"><li>• PassedParameterTab ("Parameter Values")</li><li>• DefinitionParameterTab (Parameter Defaults")</li><li>• LocalVariableTab ("Variables" or "Local Variables")</li><li>• ProjectVariableTab ("Project variables")</li><li>• ConstantsTab ("Constants")</li><li>• BaseElementTab ("Symbol" or "Footprint")</li><li>• ComponentTab ("General")</li><li>• Component("Component")</li><li>• CustomTab ("Intrinsic Variables")</li><li>• Quantities ("Quantities")</li><li>• Signals ("Signals")</li></ul>
	<i>&lt;PropServer&gt;</i>	String	An object identifier, generally returned from another script method, such as CompInst@R;2;3

	<i>&lt;PropName&gt;</i>	String	Name of the property.
<b>Return Value</b>	String value of the property.		

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

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

## 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.

<b>UI Access</b>	N/A		
<b>Parameters</b>	Name <i>&lt;IncludeReadOnly&gt;</i>	Type Boolean	Description Optional. <ul style="list-style-type: none"><li>• <b>True</b> - includes all properties.</li><li>• <b>False</b> - returns only property names that can be changed.</li></ul> True by default.
<b>Return Value</b>	Returns property names of the current 3D Model object.		

<b>Python Syntax</b>	GetPropNames( <i>&lt;IncludeReadOnly&gt;</i> )
<b>Python Example</b>	<code>oEditor.GetPropNames (True)</code>

<b>VB Syntax</b>	GetPropNames( <i>&lt;IncludeReadOnly&gt;</i> )
<b>VB Example</b>	<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.

<b>UI Access</b>	N/A						
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;PropName&gt;</td> <td>String</td> <td>Name of the property.</td> </tr> </tbody> </table>	Name	Type	Description	<PropName>	String	Name of the property.
Name	Type	Description					
<PropName>	String	Name of the property.					
<b>Return Value</b>	Property value as a double floating value, or NAN if the property value cannot be converted to double floating point.						

<b>Python Syntax</b>	GetPropSIValue (<PropName>)
<b>Python Example</b>	<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 [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.

<b>UI Access</b>	N/A		
<b>Parameters</b>	Name	Type	Description
	<propPath>		Optional. Child object's property path. See: <a href="#">Object Property Function Summary</a> .
<b>Return Value</b>	The property value.		

<b>Python Syntax</b>	GetPropValue(<propPath>)
<b>Python Example</b>	Rh1 = oDesign.GetChildObject('Box1') Rh1.GetPropValue('Orientation') Returns the specified Property Value: 'Global'

<b>VB Syntax</b>	GetPropValue(<propPath>)
<b>VB Example</b>	Rh1 = oDesign.GetChildObject("Box1")

	<code>GetPropValue("Orientation")</code>
--	------------------------------------------

## GetRelativeCoordinateSystems

Returns the relative coordinate systems in the current 3D modeler.

<b>UI Access</b>	N/A
<b>Parameters</b>	None.
<b>Return Value</b>	Array containing name of relative coordinate systems.

<b>Python Syntax</b>	<code>GetRelativeCoordinateSystems()</code>
<b>Python Example</b>	<code>oEditor.GetRelativeCoordinateSystems ()</code>

<b>VB Syntax</b>	<code>GetRelativeCoordinateSystems</code>
<b>VB Example</b>	<code>oEditor.GetRelativeCoordinateSystems</code>

## GetSelections [Model Editor]

Returns an array of currently selected objects.

<b>UI Access</b>	N/A
<b>Parameters</b>	None.
<b>Return Value</b>	Array containing object IDs

<b>Python Syntax</b>	GetSelections()
<b>Python Example</b>	<code>oEditor.GetSelections ()</code>

<b>VB Syntax</b>	GetSelections
<b>VB Example</b>	<code>oEditor.GetSelections</code>

## GetSubGroupsInGroup

Returns subgroup names in a specified group.

<b>UI Access</b>	N/A		
<b>Parameters</b>	Name <i>&lt;group&gt;</i>	Type String	Description Group name.  One of: <ul style="list-style-type: none"><li>• "&lt;<i>materialName</i>&gt;"</li><li>• "&lt;<i>assignmentName</i>&gt;"</li><li>• "Non Model"</li><li>• "Solids"</li><li>• "Unclassified"</li><li>• "Sheets"</li><li>• "Lines"</li></ul>
<b>Return Value</b>	Array containing subgroup names.		

---

<b>Python Syntax</b>	<code>GetSubGroupsInGroup(&lt;group&gt;)</code>
<b>Python Example</b>	<code>oEditor.GetSubGroupsInGroup ('Sheets')</code>

<b>VB Syntax</b>	<code>GetSubGroupsInGroup&lt;group&gt;</code>
<b>VB Example</b>	<code>oEditor.GetSubGroupsInGroup "Sheets"</code>

## GetUserPosition

Returns a user's current coordinates in the 3D Modeler window.

<b>UI Access</b>	N/A		
<b>Parameters</b>	Name <i>&lt;prompt&gt;</i>	Type String	Description "Enter a point." Then click a point in the 3D Modeler window.
<b>Return Value</b>	Array containing X, Y, Z coordinates.		

<b>Python Syntax</b>	<code> GetUserPosition(&lt;prompt&gt;)</code>
<b>Python Example</b>	<code>oEditor.GetUserPosition("Enter a point.")</code>

<b>VB Syntax</b>	<code> GetUserPosition &lt;prompt&gt;</code>
<b>VB Example</b>	<code>oEditor.GetUserPosition "Enter a point."</code>

## GetVertexIDFromNameForFirstOperation

Returns vertex ID associated with a specified vertex belongs to the first operation of a part.

<b>UI Access</b>	N/A		
<b>Parameters</b>	Name	Type	Description
	<PartName>	String	Name of specified part.
<b>Return Value</b>	Integer representing vertex ID.		

<b>Python Syntax</b>	GetVertexIDFromNameForFirstOperation(<PartName>, <VertexName>)
<b>Python Example</b>	oEditor.GetVertexIDFromNameForFirstOperation("Part1", "Vertex1")

<b>VB Syntax</b>	GetVertexIDFromNameForFirstOperation <PartName>, <VertexName>
<b>VB Example</b>	oEditor.GetVertexIDFromNameForFirstOperation "Part1", "Vertex1"

## GetVertexIDsFromEdge

Returns vertex IDs associated with a specified edge.

<b>UI Access</b>	N/A		
<b>Parameters</b>	Name	Type	Description
	<edgeID>	Integer	Edge ID.

<b>Return Value</b>	Array containing string vertex IDs.
---------------------	-------------------------------------

<b>Python Syntax</b>	GetVertexIDsFromEdge(<edgeID>)
<b>Python Example</b>	<code>oEditor.GetVertexIDsFromEdge(10)</code>

<b>VB Syntax</b>	GetVertexIDsFromEdge <edgeID>
<b>VB Example</b>	<code>oEditor.GetVertexIDsFromEdge 10</code>

## GetVertexIDsFromFace

Returns vertex IDs associated with a specified face.

<b>UI Access</b>	N/A								
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;faceID&gt;</td> <td>Integer</td> <td>Face ID.</td> </tr> </tbody> </table>			Name	Type	Description	<faceID>	Integer	Face ID.
Name	Type	Description							
<faceID>	Integer	Face ID.							
<b>Return Value</b>	Array containing string vertex IDs.								

<b>Python Syntax</b>	GetVertexIDsFromFace(<faceID>)
<b>Python Example</b>	<code>oEditor.GetVertexIDsFromFace(10)</code>

<b>VB Syntax</b>	GetVertexIDsFromFace <faceID>
------------------	-------------------------------

**VB Example**

```
oEditor.GetVertexIDsFromFace 10
```

**GetVertexIDsFromObject**

Returns vertex IDs associated with a specified object.

<b>UI Access</b>	N/A								
<b>Parameters</b>	<table border="1"><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;objectName&gt;</td><td>String</td><td>Object name.</td></tr></table>			Name	Type	Description	<objectName>	String	Object name.
Name	Type	Description							
<objectName>	String	Object name.							
<b>Return Value</b>	Array containing string vertex IDs.								

**Python Syntax**

```
GetVertexIDsFromObject(<objectName>)
```

**Python Example**

```
oEditor.GetVertexIDsFromObject ('Box1')
```

<b>VB Syntax</b>	GetVertexIDsFromObject <objectName>
<b>VB Example</b>	oEditor.GetVertexIDsFromObject "Box1"

**GetVertexPosition**

Returns an array of coordinates for a specified vertex.

<b>UI Access</b>	N/A								
<b>Parameters</b>	<table border="1"><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;vertexID&gt;</td><td>Integer</td><td>Vertex ID.</td></tr></table>			Name	Type	Description	<vertexID>	Integer	Vertex ID.
Name	Type	Description							
<vertexID>	Integer	Vertex ID.							

<b>Return Value</b>	Array containing X, Y, Z coordinates.
---------------------	---------------------------------------

<b>Python Syntax</b>	GetVertexPosition(<vertexID>)
<b>Python Example</b>	<code>oEditor.GetVertexPosition(1)</code>

<b>VB Syntax</b>	GetVertexPosition <vertexID>
<b>VB Example</b>	<code>oEditor.GetVertexPosition 1</code>

## GetWireBodyNames

Returns the wire body names in current 3D modeler.

<b>UI Access</b>	N/A
<b>Parameters</b>	None.
<b>Return Value</b>	Array containing string of names.

<b>Python Syntax</b>	GetWireBodyNames()
<b>Python Example</b>	<code>oEditor.GetWireBodyNames ()</code>

<b>VB Syntax</b>	GetWireBodyNames
<b>VB Example</b>	<code>oEditor.GetWireBodyNames</code>

## OpenExternalEditor

Launches a SpaceClaim session.

<b>UI Access</b>	<b>Modeler &gt; SpaceClaim Link &gt; Connect to Active Session.</b>						
<b>Parameters</b>	<table border="1"><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;SelectionsArray&gt;</td><td>Array</td><td>Structured array. See: <a href="#">SelectionsArray</a>.</td></tr></table>	Name	Type	Description	<SelectionsArray>	Array	Structured array. See: <a href="#">SelectionsArray</a> .
Name	Type	Description					
<SelectionsArray>	Array	Structured array. See: <a href="#">SelectionsArray</a> .					
<b>Return Value</b>	None.						

<b>Python Syntax</b>	OpenExternalEditor (<SelectionsArray>)
<b>Python Example</b>	<pre>oEditor.OpenExternalEditor (     [         "NAME:Selections",         "Selections:=" , "SpaceClaim1"     ])</pre>

<b>VB Syntax</b>	OpenExternalEditor <SelectionsArray>
<b>VB Example</b>	<pre>oEditor.OpenExternalEditor Array("NAME:Selections",     "Selections:=", "")</pre>

## PageSetup

Specifies Page Setup settings for printing.

---

UI Access	File > Page Setup.		
<b>Parameters</b>	Name	Type	Description
	<Parameters>	Array	<p>Structured array.</p> <pre>[     "NAME : PageSetupData",     "margins :=", &lt;SetupArray&gt; ]</pre>
<b>Return Value</b>			None.

<b>Python Syntax</b>	PageSetup(<Parameters>)
<b>Python Example</b>	<pre>oEditor.PageSetup [     "NAME : PageSetupData",     "margins :=",</pre>

```
[ "left:=", "10mm",
  "right:=", "10mm",
  "top:=", "10mm",
  "bottom:=", "10mm"]
])
```

<b>VB Syntax</b>	PageSetup <Parameters>
<b>VB Example</b>	<pre>oEditor.PageSetup Array("NAME:PageSetupData",   "margins:=",   Array("left:=", "10mm",         "right:=", "10mm",         "top:=", "10mm",         "bottom:=", "10mm"       )     )</pre>

## RemoveBadEdges

Removes bad edges from specified list.

<b>UI Access</b>	N/A								
<b>Parameters</b>	<table border="1"><tr><th>Name</th><th>Type</th><th>Description</th></tr><tr><td>&lt;EdgeList&gt;</td><td>Array</td><td>Array containing edge IDs.</td></tr></table>			Name	Type	Description	<EdgeList>	Array	Array containing edge IDs.
Name	Type	Description							
<EdgeList>	Array	Array containing edge IDs.							

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

<b>Python Syntax</b>	RemoveBadEdges (<EdgeList>)
<b>Python Example</b>	<code>oEditor.RemoveBadEdges ([18, 30])</code>

<b>VB Syntax</b>	RemoveBadEdges <EdgeList>
<b>VB Example</b>	<code>oEditor.RemoveBadEdges Array(18, 30)</code>

## RemoveBadFaces

Removes bad faces from specified list.

<b>UI Access</b>	N/A		
<b>Parameters</b>	Name <FaceList>	Type Array	Description Array containing face IDs.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	RemoveBadFaces (<FaceList>)
<b>Python Example</b>	<code>oEditor.RemoveBadFaces ([328, 333])</code>

<b>VB Syntax</b>	RemoveBadFaces <FaceList>
------------------	---------------------------

<b>VB Example</b>	<code>oEditor.RemoveBadFaces Array(328, 333)</code>
-------------------	-----------------------------------------------------

## RemoveBadVertices

Removes bad vertices from specified list.

<b>UI Access</b>	N/A		
<b>Parameters</b>	Name <code>&lt;VertexList&gt;</code>	Type Array	Description Array containing vertex IDs.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	<code>RemoveBadVertices (&lt;VertexList&gt;)</code>
<b>Python Example</b>	<code>oEditor.RemoveBadVertices ([324, 325])</code>

<b>VB Syntax</b>	<code>RemoveBadVertices &lt;VertexList&gt;</code>
<b>VB Example</b>	<code>oEditor.RemoveBadVertices Array(324, 325)</code>

## RenamePart

Renames an object.

<b>UI Access</b>	Enter new name in <b>Name</b> field.		
<b>Parameters</b>	Name <code>&lt;renameParametersArray&gt;</code>	Type Array	Description Structured array.

			Array("NAME:Rename Data", "Old Name:=", <string>, "New Name:=", <string> )
<b>Return Value</b>	None.		

<b>Python Syntax</b>	oEditor.RenamePart(<renameParametersArray>)
<b>Python Example</b>	<pre>oEditor.RenamePart( [   'NAME:Rename Data',   'Old Name:=' , 'partname',   'New Name:=' , 'newpartname', ] )</pre>

<b>VB Syntax</b>	oEditor.RenamePart <RenameParametersArray>
<b>VB Example</b>	<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 <b>Properties</b> on History Tree objects		
Parameters	Name	Type	Description
	<propPath>	String	Child object's property path. See: <a href="#">Object Property Function Summary</a> .
	<value>	Varies	New property value.
Return Value	<p>Boolean:</p> <ul style="list-style-type: none"><li>• <b>True</b> – property found.</li><li>• <b>False</b> – property not found.</li></ul>		

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.

<b>UI Access</b>	N/A		
<b>Parameters</b>	Name <i>&lt;RelativeCS&gt;</i>	Type String	Description Name of relative coordinate system
<b>Return Value</b>	None.		

<b>Python Syntax</b>	SetTopDownViewDirectionForActiveView (<RelativeCS>)
<b>Python Example</b>	<code>oEditor.SetTopDownViewDirectionForActiveView("Global")</code>

<b>VB Syntax</b>	SetTopDownViewDirectionForActiveView <RelativeCS>
<b>VB Example</b>	<code>oEditor.SetTopDownViewDirectionForActiveView "Global"</code>

## **SetTopDownViewDirectionForAllViews**

Sets all views to top-down view direction.

<b>UI Access</b>	N/A		
<b>Parameters</b>	Name <i>&lt;RelativeCS&gt;</i>	Type String	Description Name of relative coordinate system
<b>Return Value</b>	None.		

<b>Python Syntax</b>	SetTopDownViewDirectionForAllViews (<RelativeCS>)
<b>Python Example</b>	<code>oEditor.SetTopDownViewDirectionForAllViews ("Global")</code>

<b>VB Syntax</b>	SetTopDownViewDirectionForAllViews <RelativeCS>
<b>VB Example</b>	<code>oEditor.SetTopDownViewDirectionForAllViews "Global"</code>

## UpdatePriorityList

Updates specified priority lists.

<b>UI Access</b>	N/A		
<b>Parameters</b>	Name <Lists>	Type Array	Description Array of priority list names.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	UpdatePriorityList (<Lists>)
<b>Python Example</b>	<code>oEditor.UpdatePriorityList(["PriorityList1", "PriorityList2"])</code>

<b>VB Syntax</b>	UpdatePriorityList <Lists>
<b>VB Example</b>	<code>oEditor.UpdatePriorityList Array("PriorityList1", "PriorityList2")</code>

## UpgradeVersion

Upgrades legacy geometry to current version.

<b>UI Access</b>	Right-click on an operation icon in the history tree, select <b>Upgrade Version</b> .						
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;Selections&gt;</td> <td>Array</td> <td> <p>Structured array.</p> <pre>Array("NAME:Parameters",       Array("NAME:PartOperations",             Array("NAME:&lt;string&gt;",),             "OperationIndices:=", &lt;array of integers&gt;),       Array("NAME:UDMOperations"))</pre> </td> </tr> </tbody> </table>	Name	Type	Description	<Selections>	Array	<p>Structured array.</p> <pre>Array("NAME:Parameters",       Array("NAME:PartOperations",             Array("NAME:&lt;string&gt;",),             "OperationIndices:=", &lt;array of integers&gt;),       Array("NAME:UDMOperations"))</pre>
Name	Type	Description					
<Selections>	Array	<p>Structured array.</p> <pre>Array("NAME:Parameters",       Array("NAME:PartOperations",             Array("NAME:&lt;string&gt;",),             "OperationIndices:=", &lt;array of integers&gt;),       Array("NAME:UDMOperations"))</pre>					
<b>Return Value</b>	None.						

<b>Python Syntax</b>	UpgradeVersion (<Selections>)
<b>Python Example</b>	<pre>oEditor.UpgradeVersion([     "NAME:Parameters",     [ "NAME:PartOperations",         [ "NAME:source",             "OperationIndices:=", [0]],     ],     [ "NAME:UDMOperations"]]</pre>

	])
--	----

<b>VB Syntax</b>	UpgradeVersion <Lists>
<b>VB Example</b>	<pre>oEditor.UpgradeVersion Array(     "NAME:Parameters",     Array("NAME:PartOperations",         Array("NAME:source",             "OperationIndices:=", Array(0))     ),     Array"NAME:UDMOperations" )</pre>

## Validate3DComponent

Validates a 3D component.

<b>UI Access</b>	N/A									
<b>Parameters</b>	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>&lt;componentPath&gt;</td><td>String</td><td>Path to a 3D component.</td></tr><tr><td>&lt;password&gt;</td><td>String</td><td>Optional. Password to access the component.</td></tr></tbody></table>	Name	Type	Description	<componentPath>	String	Path to a 3D component.	<password>	String	Optional. Password to access the component.
Name	Type	Description								
<componentPath>	String	Path to a 3D component.								
<password>	String	Optional. Password to access the component.								
<b>Return Value</b>	Boolean: <ul style="list-style-type: none"><li>• 1 - component is valid.</li></ul>									

	<ul style="list-style-type: none"> <li>• <b>0</b> - component is not valid.</li> </ul>
--	----------------------------------------------------------------------------------------

<b>Python Syntax</b>	Validate3DComponent (<componentPath>, <password>)
<b>Python Example</b>	<pre>oEditor.Validate3DComponent (     "C:/temp/component.3dcomp", "")</pre>

<b>VB Syntax</b>	Validate3DComponent <componentPath>, <password>
<b>VB Example</b>	<pre>oEditor.Validate3DComponent     "C:/temp/component.3dcomp", "</pre>

## WriteHistoryTreeLayoutForTest

Writes history tree layout to a file.

<b>UI Access</b>	N/A		
<b>Parameters</b>	Name	Type	Description
	<filePath>	String	Path to the file.
	<OrgByMaterial>	Integer	<ul style="list-style-type: none"> <li>• <b>1</b> - organize objects by material.</li> <li>• <b>0</b> - do not organize by material.</li> </ul>
	<OrgByAssignment>	Integer	<ul style="list-style-type: none"> <li>• <b>1</b> - organize sheets by assignment.</li> <li>• <b>0</b> - do not organize by assignment.</li> </ul>
	<OrgByCompDefinition>	Integer	<ul style="list-style-type: none"> <li>• <b>1</b> - organize components by definition.</li> <li>• <b>0</b> - do not organize by definition.</li> </ul>
	<DonotOrgUnderGroup>	Integer	<ul style="list-style-type: none"> <li>• <b>1</b> - do not organize within group.</li> </ul>

			<ul style="list-style-type: none"><li>• <b>0</b> - organize within group.</li></ul>
<ShowGroup>	Integer	Optional.	<ul style="list-style-type: none"><li>• <b>1</b> - show group.</li><li>• <b>0</b> - do not show.</li></ul>
<b>Return Value</b>	None.		

<b>Python Syntax</b>	WriteHistoryTreeLayoutForTest (<filePath>, <OrgByMaterial>, <OrgByAssignment>, <OrgByCompDefinition>, <DoNotOrgUnderGroup>, <>ShowGroup>)
<b>Python Example</b>	<pre>oEditor.WriteHistoryTreeLayoutForTest ('C:\\temp', 1, 0, 0, 0, 1)</pre>

<b>VB Syntax</b>	WriteHistoryTreeLayoutForTest <filePath>, <OrgByMaterial>, <OrgByAssignment>, <OrgByCompDefinition>, <DoNotOrgUnderGroup>, <>ShowGroup>
<b>VB Example</b>	<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**.

<b>UI Access</b>	<b>3D Components&gt;Cables &gt; Edit Cables</b>
<b>Parameters</b>	None
<b>Return Value</b>	None

<b>Python Syntax</b>	AddCableToBundle(<parameters>))
<b>Python Example</b>	<pre>oModule.AddCableToBundle("bundle1", "tpwire2", 1, [      "NAME:CableInstParams",     "XPos:=" , "0mm",     "YPos:=" , "0mm",     "RotX:=" , "0deg" ] , [     "NAME:CableInstAttribs",     "Name:=" , "tpwire2" ] )</pre>

<b>VB Syntax</b>	AddCableToBundle <name> Array(<parameters> )
<b>VB Example</b>	<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....**

<b>UI Access</b>	3D Components>Cables > Edit Cables
<b>Parameters</b>	None
<b>Return Value</b>	None

<b>Python Syntax</b>	CreateCableBundle([<parameters>]), [<name>])
<b>Python Example</b>	<pre> oModule.CreateCableBundle(     [         "NAME:BundleParams",         "AutoPack:=" , True,         [             "NAME:InsulationJacketParams", </pre>

```
        "InsThickness:="           , "0.25mm",
        "JacketMaterial:="        , "PVC plastic",
        "InnerDiameter:="         , "2.5mm"
    ],
],
[
    "NAME:BundleAttrs",
    "Name:="                  , "bundle1"
]
)
```

<b>VB Syntax</b>	CreateCableBundle Array(<parameters> , <name>)
<b>VB Example</b>	<pre>oModule.CreateCableBundle Array("NAME:BundleParams", "AutoPack:=", true, Array("NAME:InsulationJacketParams", "InsThickness:=", _ "0.25mm", "JacketMaterial:=", "PVC plastic", "InnerDiameter:=", "2.5mm")), Array("NAME:BundleAttrs",_ "Name:=", _ "bundle3")</pre>

## 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**.

---

<b>UI Access</b>	<b>3D Components&gt;Cables &gt; New Cable Harness...</b>
<b>Parameters</b>	None
<b>Return Value</b>	None

<b>Python Syntax</b>	CreateCableHarness(<parameters>))
	<pre>oModule.CreateCableHarness (     [         "NAME:HarnessParams",         "Bundle:=" , "bundle1",         "TwistAlongPath:=" , "720deg",         "Route:=" , "Polyline1",         "AutoOrient:=" , False,         "Origin:=" , [ "0mm", "0mm", "0mm" ],         "XAxisEnd:=" , [ "-6mm", "0mm", "0mm" ],         "PlaneFlip:=" , True,         [             "NAME:RefConductors",             "tpwire2/w1"         ],         [ </pre>
<b>Python Example</b>	

```

        "NAME:InputTerminations",
        "tpwire2/w1:=" , [
        "tpwire2/w2:=" , [
    ] ,
    [
        "NAME:OutputTerminations",
        "tpwire2/w1:=" , [
        "tpwire2/w2:=" , [
    ]
],
[
        "NAME:HarnessAttrs",
        "Name:=" , "MyHarness"
    ]
)

```

<b>VB Syntax</b>	CreateCableHarness Array(<parameters> )
<b>VB Example</b>	<pre> oModule.CreateCableHarness Array("NAME:HarnessParams", "Bundle:=", "bundle1", "TwistAlongPath:=", _ "360deg", "Route:=", "Polyline2", "AutoOrient:=", false, "Origin:=", Array( _ </pre>

```

"-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:HarnessAttribs",
"Name:=", _
"MyNewHarness")

```

## 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**.

<b>UI Access</b>	<b>3D Components&gt;Cables &gt; Edit Cables</b>
<b>Parameters</b>	None
<b>Return Value</b>	None

<b>Python Syntax</b>	CreateClockSource(<parameters>))
<b>Python Example</b>	<pre> oModule.CreateClockSource( [     "NAME:ClockSignalParams",     "Period:="           , "35us", ] ) </pre>

```
        "LowPulseVal:="           , "0V",
        "HighPulseVal:="          , "1V",
        "Risetime:="              , "5us",
        "Falltime:="              , "5us",
        "PulseWidth:="             , "20us"
    ],
[
    "NAME:TDSourceAttrs",
    "Name:="                  , "clock1"
])
```

<b>VB Syntax</b>	CreateClockSourceArray(<parameters> )
<b>VB Example</b>	<pre>oModule.CreateClockSource Array("NAME:ClockSignalParams", "Period:=", "35us", "LowPulseVal:=", _ "0V", "HighPulseVal:=", "1V", "Risetime:=", "5us", "Falltime:=", "5us", "PulseWidth:=", _ "20us"), Array("NAME:TDSourceAttrs", "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**.

<b>UI Access</b>	<b>3D Components&gt;Cables &gt; Edit Cable Sources...</b>
<b>Parameters</b>	None
<b>Return Value</b>	None

<b>Python Syntax</b>	CreatePWLSource(<parameters>))
<b>Python Example</b>	<pre> oModule.CreatePWLSource(     [         "NAME:PWLSignalParams",         [             "NAME:SignalValues",             "0V",             "0.5V",             "0V"         ],         [             "NAME:TimeValues",             "0ns", </pre>

```
        "1ns",
        "2ns"
    ],
[
    "NAME:TDSourceAttrs",
    "Name:="           , "pwl3"
])
```

<b>VB Syntax</b>	CreatePWLSourceArray(<parameters> )
<b>VB Example</b>	<pre>oModule.CreatePWLSource Array("NAME:PWLSignalParams", Array("NAME:SignalValues", "0V", _ "0.5V", "0V"), Array("NAME:TimeValues", "0ns", "1ns", "2ns")), Array ("NAME:TDSourceAttrs", "Name:=", _ "pwl1")</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**.

<b>UI Access</b>	3D Components>Cables > Edit Cables
------------------	------------------------------------

<b>Parameters</b>	None
<b>Return Value</b>	None

<b>Python Syntax</b>	CreateStraightWireCable([<parameters>], [<name>])
<b>Python Example</b>	<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"     ],     [         "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	<b>3D Components&gt;Cables &gt; Edit Cables</b>
Parameters	None
Return Value	None

Python Syntax	CreateTwisted Pair Cable([<parameters>]), [<name>])
Python Example	<pre>oModule.CreateTwistedPairCable("stwire1", [     "NAME:TwistedPairParams",</pre>

```

[

    "NAME:VirtualJacketParams",
    "JacketMaterial:=" , "",
    "InnerDiameter:=" , "0"

] ,
    "IsLayLengthSpecified:=", False,
    "LayLength:=" , "13.88888888889mm",
    "TurnsPerMeter:=" , "72"

] ,
[

    "NAME:TwistedPairAttribs",
    "Name:=" , "tpwire1"

] )

```

<b>VB Syntax</b>	CreateTwistedPairCable "<name>" , Array(<parameters>)
<b>VB Example</b>	<pre> oModule.CreateTwistedPairCable "stwire1", Array("NAME:TwistedPairParams", _ Array("NAME:VirtualJacketParams", "JacketMaterial:=", _ "", "InnerDiameter:=", "0"), "IsLayLengthSpecified:=", false, "LayLength:=", _ "13.88888888889mm", "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.

<b>UI Access</b>	<b>3D Components&gt;Cables &gt; Export Cable Library...</b>
<b>Parameters</b>	None
<b>Return Value</b>	None

<b>Python Syntax</b>	ExportCableLibrary(<parameters>))
<b>Python Example</b>	<pre>oModule = oDesign.GetModule("CableSetup") oModule.ExportCableLibrary("D:\\Users\\JaneDoe\\PersonalLib\\MyCables")</pre>

<b>VB Syntax</b>	ExportCableLibrary(<parameters> )
<b>VB Example</b>	<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.

---

<b>UI Access</b>	<b>3D Components&gt;Cables &gt; Import Cable Library...</b>
<b>Parameters</b>	None
<b>Return Value</b>	None

<b>Python Syntax</b>	ImportCableLibrary(<parameters>))
<b>Python Example</b>	<pre>oModule = oDesign.GetModule("CableSetup") oModule.ExportCableLibrary("D:\\\\Users\\\\JaneDoe\\\\PersonalLib\\\\MyCables")</pre>

<b>VB Syntax</b>	ImportCableLibrary(<parameters> )
<b>VB Example</b>	<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.

<b>UI Access</b>	<b>3D Components&gt;Cables &gt; Edit Cables...</b>
<b>Parameters</b>	None
<b>Return Value</b>	None

<b>Python Syntax</b>	RemoveCable(<parameters>))
<b>Python Example</b>	<pre>oModule = oDesign.GetModule("CableSetup") oModule.RemoveCable ("bundle4")</pre>

<b>VB Syntax</b>	ImportCableLibrary(<parameters> )
<b>VB Example</b>	<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**.

<b>UI Access</b>	<b>3D Components&gt;Cables &gt; New Cable Harness...</b>
<b>Parameters</b>	None
<b>Return Value</b>	None

<b>Python Syntax</b>	UpdateCableHarness(<parameters>))
<b>Python Example</b>	<pre>oModule.UpdateCableHarness (     [         "NAME:HarnessParams", </pre>

```
"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:="

                    "Imped:="

                ] ,
[

    "NAME:OutputTerminations",
    "tpwire2/w1:=" , [
                    "Imped:="

                    "Source:="

                ]
]
```

```

        ],
        [
            "NAME:HarnessAttrs",
            "Name:="           , "MyHarness"
        ]
    )

```

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:=", _ "50ohm"), "tpwire2/w2:=", Array("Imped:=", "50ohm))), Array("NAME:HarnessAttrs", "Name:=", _ "MyNewHarness") </pre>

# 11 - Output Variable Script Commands

Output variable commands should be executed by the "OutputVariable" module.

First, obtain the output variable module from oDesign and use it for output variable commands:

```
Set oModule = oDesign.GetModule("OutputVariable")
oModule.<CommandName><args>
```

The commands are:

[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.		

Pyth-on Syn-tax	CreateOutputVariable (<OutputVarName>, <Expression>, <SolutionName>, <reportType   solutionType>, <contextArray   domainArray>)
Pyth-on Exam-ple	<p><b>HFSS Example:</b></p> <pre>oModule.CreateOutputVariable("Var" &amp; OutputQuantity, OutputQuantity, Solution, "Far Fields", ["Context:=", InfiniteSphere, "Domain:=", "Sweep"])</pre>

**Layout Editor Example :**

```
Set oModule = oDesign.GetModule("OutputVariable")
oModule.CreateOutputVariable("test","mag(S(WavePort1,WavePort1))", "Setup1 : LastAdaptive",
", "Modal Solution Data", ["Domain:=", "Sweep"])
```

<b>VB Syntax</b>	CreateOutputVariable <OutputVarName>, <Expression>, <SolutionName>, <reportType   solutionType>, <contextArray   domainArray>
<b>VB Example</b>	<p><b>HFSS Example:</b></p> <pre>oModule.CreateOutputVariable "Var_" &amp; OutputQuantity, OutputQuantity, Solution, "Far Fields", Array("Context:=", InfiniteSphere, "Domain:=", "Sweep")</pre> <p><b>Layout Editor Example :</b></p> <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.

<b>UI Access</b>	HFSS 3D Layout > Results > Output Variables. In the <b>Output Variables</b> window, click <b>Delete</b> .						
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;OutputVarName&gt;</td> <td>String</td> <td>Name of the output variable.</td> </tr> </tbody> </table>	Name	Type	Description	<OutputVarName>	String	Name of the output variable.
Name	Type	Description					
<OutputVarName>	String	Name of the output variable.					

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

<b>Python Syntax</b>	DeleteOutputVariable (<OutputVarName>)
<b>Python Example</b>	<pre>oModule = oDesign.GetModule("OutputVariable") oModule.DeleteOutputVariable ("testNew")</pre>

<b>VB Syntax</b>	DeleteOutputVariable <OutputVarName>
<b>VB Example</b>	<pre>Set oModule = oDesign.GetModule("OutputVariable") oModule.DeleteOutputVariable "testNew"</pre>

## DoesOutputVariableExist

Verifies whether or not a named output variable exists.

<b>UI Access</b>	N/A						
<b>Parameters</b>	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>&lt;outputVariableName&gt;</td><td>String</td><td>The output variable name.</td></tr></tbody></table>	Name	Type	Description	<outputVariableName>	String	The output variable name.
Name	Type	Description					
<outputVariableName>	String	The output variable name.					
<b>Return Value</b>	Boolean True if the variable exists; False if it does not.						

<b>Python Syntax</b>	DoesOutputVariableExist(<outputVariableName>)
----------------------	-----------------------------------------------

<b>Python Example</b> <pre>oProject = oDesktop.GetActiveProject() oDesign = oProject.GetActiveDesign() oModule = oDesign.GetModule("OutputVariable") oModule.DoesOutputVariableExist("MyTestVar")</pre>
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

<b>VB Syntax</b> <pre>DoesOutputVariableExist(&lt;outputVariableName&gt;)</pre>
<b>VB Example</b> <pre>Set oModule = oDesign.GetModule "OutputVariable" oModule.DoesOutputVariableExist "MyTestVar"</pre>

## EditOutputVariable

Changes the name or expression of an existing output variable.

<b>UI Access</b> N/A																								
<b>Parameters</b>	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;OrigVarName&gt;</td> <td>String</td> <td>Name of the original output variable.</td> </tr> <tr> <td>&lt;NewExpression&gt;</td> <td>String</td> <td>New value to assign to the variable.</td> </tr> <tr> <td>&lt;NewVarName&gt;</td> <td>String</td> <td>New name of the variable if any, else pass empty string.</td> </tr> <tr> <td>&lt;SolutionName&gt;</td> <td>String</td> <td>Name of the solution as seen in the output variable UI.  For example, "Setup1 : Last Adaptive".</td> </tr> <tr> <td>&lt;ReportType&gt; or &lt;SolutionType&gt;</td> <td>String</td> <td>The name of the report type as seen in the output variable UI.  For Layout Editor, use the solution type.</td> </tr> <tr> <td>&lt;ContextArray&gt; or &lt;DomainArray&gt;</td> <td>Array</td> <td>Structured array containing context for which the out-</td> </tr> </tbody> </table>	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-		
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-																						

		<p>put variable expression is being evaluated.</p> <pre>Array("Context:=", &lt;string&gt;)</pre> <p>For Layout Editor, use Domain array:</p> <pre>Array("Domain:=", &lt;string&gt;)</pre>
<b>Return Value</b>	None	

<b>Python Syntax</b>	<pre>EditOutputVariable (&lt;OrigVarName&gt;, &lt;NewExpression&gt;, &lt;NewVarName&gt;, &lt;SolutionName&gt;, &lt;ReportType   SolutionType&gt;, &lt;ContextArray   DomainArray&gt;)</pre>
<b>Python Example</b>	<pre>oModule = oDesign.GetModule("OutputVariable") oModule.EditOutputVariable ("test", "normalize(R1_0.V)", "testNew", "TR", "Standard", [])</pre>

<b>VB Syntax</b>	<pre>EditOutputVariable &lt;OrigVarName&gt;, &lt;NewExpression&gt;, &lt;NewVarName&gt;, &lt;SolutionName&gt;, &lt;ReportType   SolutionType&gt;, &lt;ContextArray   DomainArray&gt;</pre>
<b>VB Example</b>	<pre>Set oModule = oDesign.GetModule("OutputVariable") oModule.EditOutputVariable "test", "dB(S(WavePort1,WavePort1)) ", "testNew", "Setup1 : LastAdaptive", "Modal Solution Data", Array("Domain:=", "Sweep")</pre>

## ExportOutputVariables

Exports output variables to a file.

<b>UI Access</b>	Click on <b>Export</b> in the <b>Output Variables</b> dialog.						
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;FileName&gt;</td> <td>String</td> <td>Name of the file include path.</td> </tr> </tbody> </table>	Name	Type	Description	<FileName>	String	Name of the file include path.
Name	Type	Description					
<FileName>	String	Name of the file include path.					
<b>Return Value</b>	<p>Boolean:</p> <ul style="list-style-type: none"> <li>• <b>True</b> - output variable successfully exported.</li> <li>• <b>False</b> - error when export output variables.</li> </ul>						

<b>Python Syntax</b>	ExportOutputVariables(<FileName>)
<b>Python Example</b>	<code>oModule.ExportOutputVariables ("C:/output_var.aoutvar")</code>

<b>VB Syntax</b>	ExportOutputVariables <FileName>
<b>VB Example</b>	<code>oModule.ExportOutputVariables "C:/output_var.aoutvar"</code>

## GetOutputVariables

Returns the list of output variables.

<b>UI Access</b>	N/A
<b>Parameters</b>	None.
<b>Return Value</b>	Array containing all output variables.

<b>Python Syntax</b>	GetOutputVariables()
<b>Python Example</b>	<code>oDesign.GetOutputVariables ()</code>

<b>VB Syntax</b>	GetOutputVariables
<b>VB Example</b>	<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.

<b>UI Access</b>	N/A		
<b>Parameters</b>	<b>Name</b>	<b>Type</b>	<b>Description</b>
	<code>&lt;OutputVarName&gt;</code>	String	Name of the output variable.
	<code>&lt;IntrinsicVariation&gt;</code>	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>
	<code>&lt;SolutionName&gt;</code>	String	Name of the solution as listed in the output variable UI. For example, <code>"Setup1 : Last Adaptive"</code> .
	<code>&lt;ReportType&gt;</code>	String	The name of the report type as seen in the output variable UI.  Possible values in Layout/Schematic Editor are: <ul style="list-style-type: none"><li>• "Standard"</li><li>• "Load Pull"</li></ul>

			<ul style="list-style-type: none"> <li>• "Constellation"</li> <li>• "Data Table"</li> <li>• "Eye Diagram"</li> <li>• "Statistical"</li> </ul>
	<ContextArray>	Array	<p>Structured array containing context for which the output variable expression is being evaluated. Can be empty.</p> <pre>Array("Context:=", &lt;string&gt;)</pre>
<b>Return Value</b>	Double value of the output variable.		

<b>Python Syntax</b>	GetOutputVariableValue(<OutputVarName>, <IntrinsicVariation>, <SolutionName>, <ReportTypeName>, <ContextArray>)
<b>Python Example</b>	<pre>Val = oDesign.GetOutputVariableValue("test", "Freq = '20Ghz' Theta='20deg' Phi='30deg'", "TR", "Standard", [])</pre>

<b>VB Syntax</b>	GetOutputVariableValue <OutputVarName>, <IntrinsicVariation>, <SolutionName>, <ReportTypeName>, <ContextArray>
<b>VB Example</b>	<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 <b>Import</b> in the <b>Output Variables</b> dialog.						
Parameters	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>&lt;FileName&gt;</td><td>String</td><td>Name of the file include path.</td></tr></tbody></table>	Name	Type	Description	<FileName>	String	Name of the file include path.
Name	Type	Description					
<FileName>	String	Name of the file include path.					
Return Value	Boolean: <ul style="list-style-type: none"><li>• <b>True</b> - output variable successfully imported.</li><li>• <b>False</b> - error when import output variables.</li></ul>						

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 VerifEye 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

# 12 - 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](#)

[AddVerifyEyeAnalysis](#)

[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

<b>UI Access</b>	Right-click the report and select <b>Trace Characteristics &gt; Add All Eye Measurements</b>								
<b>Parameters</b>	<table border="1"><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;ReportName&gt;</td><td>String</td><td>Name of the report.</td></tr></table>			Name	Type	Description	<ReportName>	String	Name of the report.
Name	Type	Description							
<ReportName>	String	Name of the report.							
<b>Return Value</b>	None.								

<b>Python Syntax</b>	AddAllEyeMeasurements(<ReportName>)
<b>Python Example</b>	oModule.AddAllEyeMeasurements ("DQS_Eye")

<b>VB Syntax</b>	AddAllEyeMeasurements <ReportName>
<b>VB Example</b>	oModule.AddAllEyeMeasurements "DQS_Eye"

## AddCartesianLimitLine

Adds a limit line to a report on the X axis.

<b>UI Access</b>	<b>Report2D &gt; Add Limit Line&gt; Specify Points...</b>											
<b>Parameters</b>	<table border="1"><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;ReportName&gt;</td><td>String</td><td>Name of the report.</td></tr><tr><td>&lt;Def&gt;</td><td>Array</td><td>Structured array:</td></tr></table>			Name	Type	Description	<ReportName>	String	Name of the report.	<Def>	Array	Structured array:
Name	Type	Description										
<ReportName>	String	Name of the report.										
<Def>	Array	Structured array:										

		<pre>Array("NAME:CartesianLimitLine",       Array("NAME:XValues", &lt;integer X values&gt;),       "XUnits:=", &lt;string unit of measure for X&gt;,       Array("NAME:YValues", &lt;integer Y values&gt;),       "YUnits:=", "&lt;string unit of measure for Y&gt;",       "YAxis:=", &lt;string name of associated Y axis&gt;     )</pre>
<b>Return Value</b>	None.	

<b>Python Syntax</b>	AddCartesianLimitLine (< <i>ReportName</i> >, < <i>Def</i> >)
<b>Python Example</b>	<pre>oModule.AddCartesianLimitLine("Project Outputs",                                ["NAME:CartesianLimitLine",                                 ["NAME:XValues", 0, 2, 5, 7, 10, 15],                                 "XUnits:=", "s",                                 ["NAME:YValues", 0.05, 0.3, 0.65, 0.825, 0.95, 1],                                 "YUnits:=", "mV", "YAxis:=", "Y1"                                ]                              )</pre>

<b>VB Syntax</b>	AddCartesianLimitLine < <i>ReportName</i> >, < <i>Def</i> >
------------------	-------------------------------------------------------------

<b>VB Example</b> <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...											
<b>Parameters</b>	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;ReportName&gt;</td> <td>String</td> <td>Name of the report.</td> </tr> <tr> <td>&lt;LimitLineParams&gt;</td> <td>String</td> <td>Structured array.             Array ("NAME:CartesianLimitLineFromCurve",           "TraceName:=", &lt;string name of selected trace&gt;,           "CurveName:=", &lt;string name of selected curve&gt;,           "Start:=", "&lt;value&gt;&lt;unit&gt;",           "Stop:=", "&lt;value&gt;&lt;unit&gt;",           "YAxis:=", &lt;integer Y-Axis number&gt;,           "YOffset:=", &lt;value offset from Y-Axis&gt;,         </td> </tr> </tbody> </table>	Name	Type	Description	<ReportName>	String	Name of the report.	<LimitLineParams>	String	Structured array.  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>,		
Name	Type	Description										
<ReportName>	String	Name of the report.										
<LimitLineParams>	String	Structured array.  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:=", "&lt;AboveCurve   BelowCurve   Above and Below Curve&gt;",  "YShiftPercent:=", &lt;value percentage to shift from Y&gt;)</pre>
<b>Return Value</b>	None.		

<b>Python Syntax</b>	AddCartesianLimitLineFromCurve(<ReportName>, <LimitLineParams>)
<b>Python Example</b>	<pre>oModule.AddCartesianLimitLineFromCurve("Variables Plot 2", [      "NAME:CartesianLimitLineFromCurve",     "TraceName:="           , "Phase",     "CurveName:="           , "",     "Start:="                , "0deg",     "Stop:="                 , "375deg",     "YAxis:="                , 1,     "YOffset:="              , 0,     "CreateMode:="           , "AboveCurve",     "YShiftPercent:="        , 10 ])</pre>

<b>VB Syntax</b>	AddCartesianLimitLineFromCurve<ReportName>, <LimitLineParams>
------------------	---------------------------------------------------------------

**VB Example**

```

oModule.AddCartesianLimitLineFromCurve _
    "Variables Plot 2", _
    Array("NAME:CartesianLimitLineFromCurve", _
        "TraceName:=", "Phase", "CurveName:=", "", _
        "Start:=", "0deg", "Stop:=", "375deg", _
        "YAxis:=", 1, "YOffset:=", 0, _
        "CreateMode:=", "AboveCurve", _
        "YShiftPercent:=", 10)

```

**AddCartesianLimitLineFromEquation**

Adds a limit line to a report from a specified equation.

UI Access	Report2D > Add Limit Line > Specify Equation...		
	Name <i>&lt;ReportName&gt;</i>	Type String	Description Name of the report.
Parameters	<i>&lt;LimitLineParams&gt;</i>	Array	<p>Structured array.</p> <pre> Array("NAME:CartesianLimitLineFromEquation",       "YAxis:=", &lt;integer Y-Axis number&gt;,       "Start:=", &lt;string start frequency with unit&gt;,       "Stop:=", &lt;string end frequency with unit&gt;,       "Step:=", &lt;string frequency resolution with unit&gt;,       "Equation:=", &lt;string specified equation&gt; ) </pre>

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

<b>Python Syntax</b>	AddCartesianLimitLineFromEquation(<ReportName>, <LimitLineParams>)
<b>Python Example</b>	<pre> oModule.AddCartesianLimitLineFromEquation("S Parameter Plot 1", [     "NAME:CartesianLimitLineFromEquation",     "YAxis:=" , 1,     "Start:=" , "9GHz",     "Stop:=" , "11GHz",     "Step:=" , "0.2GHz",     "Equation:=" , "x+1" ]) </pre>

<b>VB Syntax</b>	AddCartesianLimitLineFromEquation <ReportName>, <LimitLineParams>
<b>VB Example</b>	<pre> oModule.AddCartesianLimitLineFromEquation _      "S Parameter Plot 1", _     Array("NAME:CartesianLimitLineFromEquation", _         "YAxis:=" , 1, _         "Start:=" , "9GHz", "Stop:=" , "11GHz", "Step:=" , "0.2GHz", _         "Equation:=" , "x+1") </pre>

## AddCartesianXMarker

Adds a marker to a report on the X axis.

<b>UI Access</b>	<b>Report2D &gt; Marker &gt; Add X Marker.</b>		
<b>Parameters</b>	Name	Type	Description
	<ReportName>	String	Name of report.
	<MarkerName>	String	Marker name, including any trailing number.
<XValue>	Double	X coordinate.	
<b>Return Value</b>	None		

<b>Python Syntax</b>	AddCartesianXMarker (<ReportName>, <MarkerName>, <XValue>)
<b>Python Example</b>	oModule.AddCartesianXMarker ("XY Plot 1", "MX1", 0)

<b>VB Syntax</b>	AddCartesianXMarker <ReportName>, <MarkerName>, <XValue>
<b>VB Example</b>	oModule.AddCartesianXMarker "XY Plot1", "MX1", 0

## AddCartesianYMarker

Adds a marker to a report on the Y axis.

<b>UI Access</b>	<b>Report2D &gt; Marker &gt; Add Y Marker.</b>
------------------	------------------------------------------------

<b>Parameters</b>	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.
<b>Return Value</b>	None		

<b>Python Syntax</b>	AddCartesianYMarker (<ReportName>, <MarkerName>, <AxisName>, <YValue>, <CurveName>)
<b>Python Example</b>	oModule.AddCartesianYMarker("XY Plot 1", "MY1", "Y1", 0, "dB() : Setup1 : Sweep1")

<b>VB Syntax</b>	AddCartesianYMarker <ReportName>, <MarkerName>, <AxisName>, <YValue>, <CurveName>
<b>VB Example</b>	oModule.AddCartesianYMarker "XY Plot 1", "MY1", "Y1", 0, "dB() : Setup1 : Sweep1"

## AddCartesianYMarkerToStack

Adds a marker to a stacked report on the Y axis.

<b>UI Access</b>	N/A		
<b>Parameters</b>	Name	Type	Description
	<ReportName>	String	Name of report.

	<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.
<b>Return Value</b>	None		

<b>Python Syntax</b>	AddCartesianYMarkerToStack (<ReportName>, <MarkerName>, <AxisName>, <YValue>, <CurveName>, <StackOption>)
<b>Python Example</b>	<pre>oModule.AddCartesianYMarker("XY Plot 1", "MY1", "Y1", 0, "dB() : Setup1 : Sweep1", ["All"])</pre>

<b>VB Syntax</b>	AddCartesianYMarkerToStack <ReportName>, <MarkerName>, <AxisName>, <YValue>, <CurveName>, <StackOption>
<b>VB Example</b>	<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.

<b>UI Access</b>	Report2D > Marker > Add Delta Marker.
------------------	---------------------------------------

<b>Parameters</b>	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.
<b>Return Value</b>	None		

<b>Python Syntax</b>	AddDeltaMarker(<ReportName>, <MarkerName1>, <CurveName1>, <PrimarySweepValue1>, <MarkerName2>, <CurveName2>, <PrimarySweepValue2>)
<b>Python Example</b>	<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>

<b>VB Syntax</b>	AddDeltaMarker <ReportName>, <MarkerName1>, <CurveName1>, <PrimarySweepValue1>, <MarkerName2>, <CurveName2>, <PrimarySweepValue2>
<b>VB Example</b>	<pre> oModule.AddDeltaMarker "S Parameter Plot 1", "m1", "dB(S(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.
Return Value	<PrimarySweepValue>	String	Primary sweep value, including unit.
	None		

Python Syntax	AddMarker( <ReportName>, <MarkerName>, <CurveName>, <PrimarySweepValue>)
Python Example	<pre>oModule.AddMarker("XY Plot 1", "m5", "GS1.VAL : TR4 : Cartesian", "3.6159999999997s")</pre>

<b>VB Syntax</b>	AddMarker <ReportName>, <MarkerName>, <CurveName>, <PrimarySweepValue>
<b>VB Example</b>	<pre>Set oModule = oDesign.GetModule("ReportSetup") oModule.AddMarker "XY Plot1", "m1", "mag(S(Port1 Port1)) : Setup1: LastAdaptive : Cartesian", "0.3in"</pre>

## AddNote

Adds a note at a specified location to a given report.

<b>UI Access</b>	Right-click on the plot and select <b>Add Note</b> .		
<b>Parameters</b>	Name <ReportName> <NotedataArray>  <NoteArray>	Type String Array  Array	Description Name of report Structured array.  Structured array:  "NAME:<StringDataName>", <NoteArray>  "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>

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

<b>Python Syntax</b>	AddNote (<ReportName>, <NotedataArray>)
<b>Python Example</b>	<pre>oModule.AddNote("XY Plot1",     [         "NAME:NoteDataSource",         "SourceName:=", "Note1",         "HaveDefaultPos:=", False,         "DefaultXPos:=", 1996,         "DefaultYPos:=", 3177,         "String:=", "This is a note."     ] )</pre>

<b>VB Syntax</b>	AddNote <ReportName> <NotedataArray>
<b>VB Example</b>	<pre>oModule.AddNote "XY Plot1", _     Array("NAME:NoteDataSource", _         "SourceName:=", "Note1", _         "HaveDefaultPos:=", false, _         "DefaultXPos:=", 1996, _</pre>

	<pre>"DefaultYPos:=", 3177, _ "String:=", "This is a note.")</pre>
--	--------------------------------------------------------------------

## AddTraceCharacteristics

Adds a trace characteristics field to the legend on a report.

UI Access	Report2D > Trace Characteristics > All. This opens the <b>Add Trace Characteristics</b> dialog box.		
	Name	Type	Description
	<ReportName>	String	Name of report.
Parameters	<FunctionName>	String	The function name. See the <b>Functions</b> column of the <b>Add Trace Characteristics</b> dialog box.
	<FunctionArgs>	Array	<p>Array containing string values for any function arguments. Pass empty array if no arguments exist.</p> <p>To see which argument(s) a function takes, see the bottom of the <b>Add Trace Characteristics</b> dialog box.</p> <p>For function with one argument:</p> <pre>Array(&lt;value&gt;)</pre> <p>For function with multiple arguments:</p> <pre>Array(&lt;value&gt;, &lt;value&gt;, ...)</pre>
	<RangeArgs>	Array	<p>Required. Array containing either string "Full" for a full sweep range, or "Specified" and strings containing the start and end values for the frequency range.</p> <p>For example:</p> <pre>Array("Specified", "19.5GHz", "24.4GHz")</pre>
Return Value	None.		

<b>Python Syntax</b>	AddTraceCharacteristics (<ReportName>, <FunctionName>, <FunctionArgs>, <RangeArgs>)
<b>Python Example</b>	<pre> oModule.AddTraceCharacteristics("XY Plot 2", "delaytime", ["0"], ["Full"]) oModule.AddTraceCharacteristics("Differential S-parameters", "prms", ["0", "0"], ["Full"]) oModule.AddTraceCharacteristics("Rept2DRectFreq", "distortion", ["0"], ["Specified", "19.5GHz", "20.4GHz"]) </pre>

<b>VB Syntax</b>	AddTraceCharacteristics <ReportName>, <FunctionName>, <FunctionArgs>, <RangeArgs>
<b>VB Example</b>	<pre> oModule.AddTraceCharacteristics "XY Plot 2", "delaytime", Array("0"), Array("Full") oModule.AddTraceCharacteristics "Differential S-parameters", "prms", Array("0", "0"), Array("Full") oModule.AddTraceCharacteristics "Rept2DRectFreq", "distortion", Array("0"), Array( "Specified", "19.5GHz", "20.4GHz") </pre>

## AddTraces

Creates a new trace and adds it to the specified report.

<b>UI Access</b>	<b>Modify Report &gt; Add Trace.</b>		
<b>Parameters</b>	Name	Type	Description
	<ReportName>	String	Name of Report
	<SolutionName>	String	Name of the solution as listed in the <b>Modify Report</b> 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:=", &lt;DomainType&gt;)       &lt;DomainType&gt; ex. "Sweep" or "Time"       Array("Context:=", &lt;GeometryType&gt;)       &lt;GeometryType&gt; ex. "Infinite Spheren",       "Spheren", "Polylinen"</pre>
	< <i>FamiliesArray</i> >	Array  Contains sweep definitions for the report.  <pre>Array("&lt;VariableName&gt;:= ", &lt;ValueArray&gt;)       &lt;ValueArray&gt;       Array("All") or Array("Value1", "Value2",       ..."Valuen")       examples of &lt;VariableName&gt; "Freq", "Theta",       "Distance"</pre>
	< <i>ReportdataArray</i> >	Array  This array contains the report quantity and X, Y, and (Z) axis definitions.  <pre>Array("X Component:=", &lt;VariableName&gt;,       "Y Component:=", &lt;VariableName&gt;   &lt;ReportQuantityArray&gt;)       &lt;ReportQuantityArray&gt; ex. Array("dB(S(Port1,       Port1))")</pre>
<b>Return Value</b>	None	

<b>Python Syntax</b>	Add Traces(< <i>ReportName</i> >, < <i>SolutionName</i> >, < <i>ContextArray</i> >, < <i>FamiliesArray</i> >, < <i>ReportdataArray</i> >)
<b>Python Example</b>	<code>oModule.AddTraces("XY Plot1", "Setup1 : Sweep1",</code>

```

["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)"]], [])
```

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>

	Array()
--	---------

## ApplyReportTemplate

Applies settings to a report from a template file.

<b>UI Access</b>	Right-click on a report, select <b>Report Templates &gt; Apply Settings</b> .												
<b>Parameters</b>	<table border="1"> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> <tr> <td>&lt;ReportName&gt;</td> <td>String</td> <td>Name of the report to apply settings to.</td> </tr> <tr> <td>&lt;TemplateFile&gt;</td> <td>String</td> <td>Template file name with path.</td> </tr> <tr> <td>&lt;PropertyType&gt;</td> <td>String</td> <td>Property types to apply. ("Graphical"   "Data"   "All")</td> </tr> </table>	Name	Type	Description	<ReportName>	String	Name of the report to apply settings to.	<TemplateFile>	String	Template file name with path.	<PropertyType>	String	Property types to apply. ("Graphical"   "Data"   "All")
Name	Type	Description											
<ReportName>	String	Name of the report to apply settings to.											
<TemplateFile>	String	Template file name with path.											
<PropertyType>	String	Property types to apply. ("Graphical"   "Data"   "All")											
<b>Return Value</b>	None.												

<b>Python Syntax</b>	ApplyReportTemplate(<ReportName>, <TemplateFile>, <PropertyType>)
<b>Python Example</b>	oModule.ApplyReportTemplate("Variables Plot 1", "C:/MyTemplate.rpt", "Graphical")

<b>VB Syntax</b>	ApplyReportTemplate <ReportName>, <TemplateFile>, <PropertyType>
<b>VB Example</b>	oModule.ApplyReportTemplate "Variables Plot 1", "C:/MyTemplate.rpt", "Graphical"

## ChangeProperty[ReportSetup]

Change the properties for a Report.

<b>UI Access</b>	Double-click on a report to change its properties.						
<b>Parameters</b>	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td><i>&lt;PropertyArray&gt;</i></td><td>Array</td><td>Varies, depending on the properties associated with the select object command.</td></tr></tbody></table>	Name	Type	Description	<i>&lt;PropertyArray&gt;</i>	Array	Varies, depending on the properties associated with the select object command.
Name	Type	Description					
<i>&lt;PropertyArray&gt;</i>	Array	Varies, depending on the properties associated with the select object command.					
<b>Return Value</b>	None.						

<b>Python Syntax</b>	ChangeProperty( <i>&lt;PropertyArray&gt;</i> )
<b>Python Example</b>	<pre>oModule = oDesign.GetModule("ReportSetup") oModule.ChangeProperty( [     "NAME:AllTabs",     [         "NAME:General",         [             "NAME:PropServers",             "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
            ]
        ]
    ]
]
```

	]
	]
	)

<b>VB Syntax</b>	ChangeProperty <PropertyArray>
<b>VB Example</b>	<pre> oModule.ChangeProperty Array("NAME:AllTabs", Array("NAME:Cartesian", Array ("NAME:PropServers", _ "XY Plot 1:CartesianDisplayTypeProperty"), Array("NAME:ChangedProps", Array("NAME&gt;Show X Scrollbar", "Value:=", _ true))) oModule.ChangeProperty Array("NAME:AllTabs", Array("NAME:General", Array("NAME:PropServ- ers", _ "XY Plot 1:General"), Array("NAME:ChangedProps", Array("NAME:Use Scientific Notation", "Value:=", _ true))) </pre>

## ClearAllMarkers

Clears all markers from a report.

<b>UI Access</b>	<b>Report2D &gt; Markers &gt; Clear All.</b>								
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;ReportName&gt;</td> <td>String</td> <td>Name of Report</td> </tr> </tbody> </table>			Name	Type	Description	<ReportName>	String	Name of Report
Name	Type	Description							
<ReportName>	String	Name of Report							
<b>Return Value</b>	None								

<b>Python Syntax</b>	ClearAllMarkers(<ReportName>)
<b>Python Example</b>	<code>oModule.ClearAllMarkers ("XY Plot 1")</code>

<b>VB Syntax</b>	ClearAllMarkers <ReportName>
<b>VB Example</b>	<code>oModule.ClearAllMarkers "XY Plot 1"</code>

## ClearAllTraceCharacteristics

Clears all trace characteristics from the legend in a report.

<b>UI Access</b>	Report2D > Trace Characteristics > Clear All.		
<b>Parameters</b>	Name <PlotName>	Type String	Description Name of the plot
<b>Return Value</b>	None		

<b>Python Syntax</b>	ClearAllTraceCharacteristics(<PlotName>)
<b>Python Example</b>	<code>oModule.ClearAllTraceCharacteristics ("XY Plot 1")</code>

<b>VB Syntax</b>	ClearAllTraceCharacteristics <PlotName>
<b>VB Example</b>	oModule.ClearAllTraceCharacteristics "XY Plot 1"

## CloneReportsFromDatasetSolution

Clones a report for a solved solution from a dataset solution.

<b>UI Access</b>	Right-click the Project tree on the report and choose <b>Clone from Dataset Solution</b> > [Dataset_SolutionName]		
<b>Parameters</b>	Name	Type	Description
	<ReportsToClone>	Array	Array of report names to clone.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	CloneReportsFromDatasetSolution(<ReportsToClone>, <SoluNameToUse>)
<b>Python Example</b>	oModule.CloneReportsFromDatasetSolution(["Rectangular Plot1"], "DatasetSolution_rsm_5812")

<b>VB Syntax</b>	CloneReportsFromDatasetSolution <ReportsToClone>, <SoluNameToUse>
<b>VB Example</b>	oModule.CloneReportsFromDatasetSolution _ Array("Rectangular Plot1"), "DatasetSolution_rsm_5812"

## CopyPlotSettings

Copies settings of a specified plot.

UI Access	Right-click a report, select <b>Copy</b>		
Parameters	Name <i>&lt;ReportName&gt;</i>	Type String	Description Name of specified report.
Return Value	None.		

Python Syntax	CopyPlotSettings( <i>&lt;ReportName&gt;</i> )
Python Example	<code>oModule.CopyPlotSettings ("Plot 1")</code>

VB Syntax	CopyPlotSettings < <i>ReportName</i> >
VB Example	<code>oModule.CopyPlotSettings "Plot 1"</code>

## CopyReportDefinitions

Copy the definition of a report for paste operations.

UI Access	Select a report in the Project tree, right-click and select <b>Copy Definition</b>		
Parameters	Name <i>&lt;ReportsArray&gt;</i>	Type Array	Description Names of reports from which to copy the definitions
Return Value	None		

<b>Python Syntax</b>	CopyReportDefinitions(<ReportsArray>)
<b>Python Example</b>	<code>oModule.CopyReportDefinitions(["Transmission", "Reflection"])</code>

<b>VB Syntax</b>	CopyReportDefinitions <ReportsArray>
<b>VB Example</b>	<code>oModule.CopyReportDefinitionsv _</code> <code>Array("Transmission", "Reflection")</code>

## CopyReportsData

Copy all data corresponding to the specified reports.

<b>UI Access</b>	Select a report in the Project tree, right-click and select <b>Copy Data</b>						
<b>Parameters</b>	<table border="1"> <tr> <td>Name</td> <td>Type</td> <td>Description</td> </tr> <tr> <td>&lt;ReportsArray&gt;</td> <td>Array</td> <td>Names of reports from which to copy data</td> </tr> </table>	Name	Type	Description	<ReportsArray>	Array	Names of reports from which to copy data
Name	Type	Description					
<ReportsArray>	Array	Names of reports from which to copy data					
<b>Return Value</b>	None						

<b>Python Syntax</b>	CopyReportsData (<ReportsArray>)
<b>Python Example</b>	<code>oModule.CopyReportsData ([ "Transmission", "Reflection" ])</code>

<b>VB Syntax</b>	CopyReportsData <ReportsArray>
<b>VB Example</b>	<code>oModule.CopyReportsData _</code>

	Array("Transmission", "Reflection")
--	-------------------------------------

## CopyTraceDefinitions

Copy trace definitions for a paste operation.

<b>UI Access</b>	Select a trace in the Project tree, right-click and select <b>Copy Definition</b>		
<b>Parameters</b>	Name	Type	Description
	<ReportName>	String	Name of Report
<b>Return Value</b>	None		

<b>Python Syntax</b>	CopyTraceDefinitions(<ReportName>, <TracesArray>)
<b>Python Example</b>	oModule.CopyTraceDefinitions ("Transmission", ["mag(S(Port1,Port2))"])

<b>VB Syntax</b>	CopyTraceDefinitions <ReportName>, <TracesArray>
<b>VB Example</b>	oModule.CopyTraceDefinitions "Transmission", _ Array("mag(S(Port1,Port2))")

## CopyTracesData

Copies trace data for a paste operation.

<b>UI Access</b>	Select a trace in the Project tree, right-click and select <b>Copy Data</b> .		
<b>Parameters</b>	Name	Type	Description
	<ReportName>	String	Name of Report.
<b>Return Value</b>	Trace definitions from which to copy corresponding data.		

<b>Python Syntax</b>	CopyTracesData(<ReportName>, <TracesArray>)
<b>Python Example</b>	<pre>oModule.CopyTracesData ("Transmission", ["mag(S(Port1,Port2))"])</pre>

<b>VB Syntax</b>	CopyTracesData <ReportName>, <TracesArray>
<b>VB Example</b>	<pre>oModule.CopyTracesData _ "C11", Array("mag(S(Port1,Port2))")</pre>

y, l

## CreateReport

Creates a new report with a single trace and adds it to the **Results** branch in the project tree.

<b>UI Access</b>	Right-click on <b>Results</b> > <b>Create [Type] Report</b>		
<b>Parameters</b>	Name	Type	Description
	<ReportName>	String	Name of report.
	<ReportType>	String	Type of report.

		<p>Possible values are:</p> <p>"Modal S Parameters" - Only for Driven Modal solution-type problems with ports.</p> <p>"Terminal S Parameters" - Only for Driven Terminal solution-type problems with ports.</p> <p>"Eigenmode Parameters" - Only for Eigenmode solution-type problems.</p> <p>"Fields"</p> <p>"Far Fields" - Only for problems with radiation or PML boundaries.</p> <p>"Near Fields" - Only for problems with radiation or PML boundaries.</p> <p>"Emission Test"</p>
<i>&lt;DisplayType&gt;</i>	String	<p>Type of display.</p> <p>If ReportType is "Modal S Parameters", "Terminal S Parameters", or "Eigenmode Parameters", then set to one of the following:</p> <p>"Rectangular Plot", "Polar Plot", "Radiation Pattern", "Smith Chart", "Data Table", "3D Rectangular Plot", "3D Rectangular Bar Plot" or "3D Polar Plot".</p> <p>If <i>&lt;ReportType&gt;</i> is "Fields", then set to one of the following:</p> <p>"Rectangular Plot", "Polar Plot", "Radiation Pattern", "Rectangular Contour Plot", "Data Table", "3D Rectangular Plot", or "3D Rectangular Bar Plot".</p> <p>If <i>&lt;ReportType&gt;</i> is "Far Fields" or "Near Fields", then set to one of the following:</p>

		<pre>"Rectangular Plot", "Radiation Pattern", "Rectangular Contour Plot" "Data Table", "3D Rectangular Plot", "3D Rectangular Bar Plot" or "3D Polar Plot"  If &lt;ReportType&gt; is "Emission Test", then set to one of the following: "Rectangular Plot" or "Data Table"</pre>
<SolutionName>	String	Name of the solution as listed in the <b>Modify Report</b> dialog box.
<ContextArray>	Array	<p>Context for which the expression is being evaluated. This can be an empty string if there is no context.</p> <pre>Array("Domain:=", &lt;DomainType&gt; &lt;DomainType&gt; ex. "Sweep" or "Time" Array("Context:=", &lt;GeometryType&gt; &lt;GeometryType&gt; ex. "Infinite Spheren", "Spheren", "Polylinen"</pre>
<FamiliesArray>	Array	<p>Contains sweep definitions for the report.</p> <pre>Array("&lt;VariableName&gt;:= ", &lt;ValueArray&gt; &lt;ValueArray&gt; Array("All") or Array("Value1", "Value2", ..."Valuen") examples of &lt;VariableName&gt; "Freq", "Theta", "Distance"</pre>
<ReportdataArray>	Array	<p>This array contains the report quantity and X, Y, and (Z) axis definitions.</p> <pre>Array("X Component:=", &lt;VariableName&gt;, "Y Component:=", &lt;VariableName&gt;   &lt;ReportQuantityArray&gt; &lt;ReportQuantityArray&gt; ex. Array("dB(S(Port1, Port1))")</pre>

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

<b>Python Syntax</b>	CreateReport(<ReportName>, <ReportType>, <DisplayType>, <SolutionName>, <ContextArray>, <FamiliesArray>, <ReportdataArray>, <ExtendedInfo>)
<b>Python Example</b>	<p>Three examples are given below: (nominal, Optimetrics, spectral)</p> <p><b>Nominal Example:</b></p> <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> <p><b>Optimetrics Example:</b></p> <pre>oModule.CreateReport ( "XY Plot 3", "Standard",_                       "Rectangular Plot", "TR", ["NAME:Context",_                       "OptiSetup:=", "ParametricSetup1", "SimValueCor_                       [1, 0, 2, 0, false, false, 0, 1, _]</pre>

```

        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", "K",_
                    false, "0", "MH", false, "100", "TE", false, "4",_
                    "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"  
            ]  
        ]  
    ]  
]  
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
            ]
        ]
    ]
)
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>Show Outline",  
        "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: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))"
    ],
    [
      "NAME:ChangedProps",
      [
        [
```

```
        "NAME:Outline Width",
        "Value:=" , "2"
    ]
]
]
]

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))"
        ],
        [
            "NAME:ChangedProps",
            [
                "NAME:Bar Width",
                "Value:=" , "Wide"
            ]
        ]
    ]
]
```

```
        ] )

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

<b>VB Syntax</b>	<pre>CreateReport &lt;ReportName&gt;, &lt;ReportType&gt;, &lt;DisplayType&gt;, &lt;SolutionName&gt;, &lt;ContextArray&gt;, &lt;FamiliesArray&gt;, &lt;ReportdataArray&gt;, &lt;ExtendedInfo&gt;</pre>
<b>VB Example</b>	<pre>oModule.CreateReport "3D Cartesian Plot1", "Far Fields", _     "3D Cartesian Plot", "Setup1 : LastAdaptive", _     Array("Context:=", "Infinite Sphere1", "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:=",</pre>

```
"Infinite Sphere1"), Array("_u:=", Array("All"), "_v:=", Array("All"), "Freq:=",
Array( _
    "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()
```

## CreateReport [Designer]

*Use:*Creates a new report with a single trace and adds it to the **Results** branch in the project tree.

*Command:*Product Menu>Results>Create <type> Report

*Syntax:*CreateReport <ReportName> <ReportType> <DisplayType> <SolutionName> <ContextArray> <FamiliesArray> <ReportdataArray>

*Return Value:*None

*Parameters:*<ReportName>

*Type:* <string>

Name of Report.

<ReportType>

*Type*: <string>

Possible values are:

"Standard" - For most plot types.

"Load Pull" - For load pull plots.

"Constellation" - For constellation plots.

"Data table" - For data tables.

"Eye Diagram" - For eye diagrams.

"Statistical" - For statistical plots.

<DisplayType>

*Type*: <string>

Possible values are:

"Rectangular Plot", "Polar Plot", "Radiation Pattern", "Smith Chart", "Data Table", "3D Rectangular Plot", "3D Polar Plot", or  
"Rectangular Stacked Plot".

<SolutionName>

*Type*: <string>

Name of the solution as listed in the **Modify Report** dialog box.

For example: "Setup1 : Last Adaptive"

<ContextArray>

*Type: Array of strings*

Context for which the expression is being evaluated. This can be an empty string if there is no context.

Array("Domain:=", <DomainType>)

<DomainType>

ex. "Sweep" or "Time"

Array ("Context:=", <SimValueContext>)

Context for the trace. For more information see [SimValueContext](#).

<FamiliesArray>

*Type: Array of strings*

Contains sweep definitions for the report.

Array("<VariableName>:= ", <ValueArray>)

<ValueArray>

Array("All") or Array("Value1", "Value2", ... "Valuen")

examples of <VariableName>

"Freq", "Theta", "Distance"

<ReportdataArray>

*Type: Array of strings*

This array contains the report quantity and X, Y, and (Z) axis definitions.

Array("X Component:=", <VariableName>, "Y Component:=", <VariableName> | <ReportQuantityArray>)

<ReportQuantityArray>

ex. Array("dB(S(Port1, Port1))")

*VB Example:*

```
oModule.CreateReport "XY Stacked Plot 1", "Standard", "Rectangular Stacked Plot", _  
"LinearFrequency", Array("NAME:Context", "SimValueContext:=", Array(3, 0, 2, 0, _  
false, false, -1, 1, 0, 1, 1, "", 0, 0)), Array("F:=", Array("All")), Array("X Component:=", _  
"F", "Y Component:=", Array("dB(S(Port1,Port1))", "dB(S(Port1,Port2))", _  
"dB(S(Port2,Port1))", "dB(S(Port2,Port2))")), Array()
```

*VB Example:*

```
oModule.CreateReport "Data Table 1", "Standard", "Data Table", "LinearFrequency", _  
Array("NAME:Context", "SimValueContext:=", Array( _
```

```
3, 0, 2, 0, false, false, -1, 1, 0, 1, 1, "", 0, 0)), Array("F:=", Array("All")), Array("X Component:=", _
"F", "Y Component:=", Array("dB(S(Port1,Port1))")), Array()
```

*VB Example:*

```
oModule.CreateReport "3D Rectangular Plot 1", "Standard", "3D Rectangular Plot", _
"LinearFrequency", Array("NAME:Context", "SimValueContext:=", Array(3, 0, 2, 0, _
false, false, -1, 1, 0, 1, 1, "", 0, 0)), Array("F:=", Array("All")), Array("X Component:=", _
"F", "Y Component:=", "F", "Z Component:=", Array("dB(S(Port1,Port1))")), Array()
```

*VB Example:*

```
oModule.CreateReport "3D Rectangular Plot 2", "Standard", "3D Rectangular Plot", _
"LinearFrequency", Array("NAME:Context", "SimValueContext:=", Array(3, 0, 2, 0, _
false, false, -1, 1, 0, 1, 1, "", 0, 0)), Array("F:=", Array("All")), Array("X Component:=", _
"F", "Y Component:=", "F", "Z Component:=", Array("dB(S(Port1,Port1))")), Array()
```

Python Syntax	CreateReport(<data_block>)
<b>Python Example</b>	<pre>oModule = oDesign.GetModule("ReportSetup") oModule.CreateReport("QuickEye Voltage Plot 1", "Eye Diagram", "Rectangular Plot", "QuickEyeAnalysis", [</pre>

---

```
"NAME:Context",
"SimValueContext:=" , [1,0,2,0,False,False,
-1,1,0,1,1,"",0,0,"DE",False,"0","DP",False,
"20000000","DT",False,"0.001","NUMLEVELS",False,"1","PCID",
False,"3","PID",False,"0",
"WE",False,"49.99995us","WM",False,"49.99995us","WN",False,
"0ps","WS",False,"0ps"]

],
[
  "Time:=" , ["All"]
],
[
  [
    "Component:=" , ["aeyeprobe3"]
],
[
  [
    "Unit Interval:=" , "1e-009s",
    "Offset:=" , "0",
    "Auto Delay:=" , True,
    "Manual Delay:=" , "0ps",
    "AutoCompCrossAmplitude:=" , True,
    "CrossingAmplitude:=" , "0mV",
```

```
    "AutoCompEyeMeasurementPoint:=", True,  
    "EyeMeasurementPoint:=" , "5e-010s"  
  ] )
```

## CreateReportFromTemplate

Creates a report from a saved template.

<b>UI Access</b>	<b>[product] &gt; Results &gt; Report Templates &gt; PersonalLib &gt; [TemplateName]</b>								
<b>Parameters</b>	<table border="1"><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;TemplatePath&gt;</td><td>String</td><td>Path to report template</td></tr></table>			Name	Type	Description	<TemplatePath>	String	Path to report template
Name	Type	Description							
<TemplatePath>	String	Path to report template							
<b>Return Value</b>	None								

<b>Python Syntax</b>	CreateReportFromTemplate( <TemplatePath>)
<b>Python Example</b>	oModule.CreateReportFromTemplate( "C:/MyHFSSProjects/PersonalLib/ReportTemplates/TestTemplate.rpt")

<b>VB Syntax</b>	CreateReportFromTemplate <TemplatePath>
<b>VB Example</b>	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	<table border="1"> <thead> <tr> <th>Name</th><th>Type</th><th>Description</th></tr> </thead> <tbody> <tr> <td>&lt;ReportType&gt;</td><td>String</td><td>Report type name as input parameter</td></tr> <tr> <td>&lt;DisplayType&gt;</td><td>String</td><td>Display type name as input parameter</td></tr> <tr> <td>&lt;SolutionName&gt;</td><td>String</td><td>Solution name as input parameter</td></tr> <tr> <td>&lt;SimValueCtxt&gt;</td><td>String</td><td>A context name, or array of string that encoded the context(I).</td></tr> <tr> <td>&lt;CategoryName&gt;</td><td>String</td><td>Category name as input parameter</td></tr> <tr> <td>&lt;PointSet&gt;</td><td>Array</td><td>Array of strings(II)</td></tr> <tr> <td>&lt;CommonComponentsOfTraces&gt;</td><td>Array</td><td>Array of strings(III)</td></tr> <tr> <td>&lt;ExtTraceInfo&gt;</td><td>Array</td><td>Array of strings(IV)</td></tr> </tbody> </table>	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)
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	<pre>oModule.CreateReportOfAllQuantities("Smith Chart all", "Modal Solution Data", "Smith Chart", "Setup1 : LastAdaptive", [], "S Parameter", ["Freq:=", ["All"], "offset:=", ["All"], "a:=", ["Nominal"], "b:=", ["Nominal"]], [], [])</pre>

<b>VB Syntax</b>	CreateReportOfAllQuantities < <i>ReportNameArg</i> >, < <i>ReportType</i> >, < <i>DisplayType</i> >, < <i>SolutionName</i> >, < <i>SimValueCtx</i> >, < <i>CategoryName</i> >, < <i>PointSet</i> >, < <i>CommonComponentsOfTraces</i> >, < <i>ExtTraceInfo</i> >
<b>VB Example</b>	<pre> oModule.CreateReportOfAllQuantities "Smith Chart all", _ "Modal Solution Data", "Smith Chart", "Setup1 : LastAdaptive", _ Array(), "S Parameter", _ Array("Freq:=", Array("All"), "offset:=", Array("All"), _ "a:=", Array("Nominal"), "b:=", Array("Nominal")), _ Array(), Array() </pre>

## DeleteMarker

**Use:** Deletes the specified marker.

<b>UI Access</b>	[product] > Fields > Fields > Marker > Delete Marker.						
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;<i>MarkerName</i>&gt;</td> <td>String</td> <td>Name of the marker.</td> </tr> </tbody> </table>	Name	Type	Description	< <i>MarkerName</i> >	String	Name of the marker.
Name	Type	Description					
< <i>MarkerName</i> >	String	Name of the marker.					
<b>Return Value</b>	None.						

<b>Python Syntax</b>	DeleteMarker(< <i>MarkerName</i> >)
<b>Python Example</b>	<pre> oModule.DeleteMarker ("m1") </pre>

<b>VB Syntax</b>	DeleteMarker <MarkerName>
<b>VB Example</b>	<code>oModule.DeleteMarker "m1"</code>

## DeleteAllReports

Deletes all existing reports.

<b>UI Access</b>	Right-click the report to delete in the project tree, and then click <b>Delete All Reports</b> on the shortcut menu.
<b>Parameters</b>	None.
<b>Return Value</b>	None.

<b>Python Syntax</b>	<code>DeleteAllReports()</code>
<b>Python Example</b>	<code>oModule.DeleteAllReports ()</code>

<b>VB Syntax</b>	<code>DeleteAllReports</code>
<b>VB Example</b>	<code>oModule.DeleteAllReports</code>

## DeleteReports

Deletes an existing report or reports.

<b>UI Access</b>	Right-click the report to delete in the project tree, and then click <b>Delete</b> on the shortcut menu
------------------	---------------------------------------------------------------------------------------------------------

<b>Parameters</b>	Name <i>&lt;ReportNameArray&gt;</i>	Type Array	Description Array of report names to be deleted
<b>Return Value</b>	None.		

<b>Python Syntax</b>	DeleteReports( <i>&lt;ReportNameArray&gt;</i> )
<b>Python Example</b>	<code>oModule.DeleteReports ([ "Rept2DRectFreq" ])</code>

<b>VB Syntax</b>	DeleteReports <i>&lt;ReportNameArray&gt;</i>
<b>VB Example</b>	<code>oModule.DeleteReports Array("Rept2DRectFreq")</code>

## DeleteTraceCharacteristics

Deletes a trace characteristics field from a report

<b>UI Access</b>	N/A		
<b>Parameters</b>	Name <i>&lt;ReportName&gt;</i>	Type String	Description Name of the report to delete from.
	<i>&lt;TraceCharsNames&gt;</i>	Array	Array of trace characteristics to delete.
<b>Return Value</b>	None.		

<b>Python</b>	DeleteTraceCharacteristics( <i>&lt;ReportName&gt;</i> , <i>&lt;TraceCharsNames&gt;</i> )
---------------	------------------------------------------------------------------------------------------

<b>Syntax</b>	
<b>Python Example</b>	<code>oModule.DeleteTraceCharacteristics("Variables Plot 1", ["lowercutoff", "gain-crossover"])</code>

<b>VB Syntax</b>	<code>DeleteTraceCharacteristics &lt;ReportName&gt;, &lt;TraceCharsNames&gt;</code>
<b>VB Example</b>	<code>oModule.DeleteTraceCharacteristics "Variables Plot 1", _ Array("lowercutoff", "gaincrossover")</code>

## DeleteTraces

Deletes an existing traces or traces.

<b>UI Access</b>	Right-click the Trace to delete in the project tree, and then click <b>Delete</b> on the shortcut menu		
<b>Parameters</b>	Name	Type	Description
	<code>&lt;TraceSelection&gt;</code>	Array	Structured array define selections.  <code>Array ("&lt;ReportName&gt;:=", &lt;TracesArray&gt;, &lt;TracesArray&gt;, ... )</code>
	<code>&lt;ReportName&gt;</code>	String	Name of Report
	<code>&lt;TracesArray&gt;</code>	Array	Contains the traces to delete within a report  <code>Array (&lt;Trace&gt;, &lt;Trace&gt;, ... )</code>
	<code>&lt;Trace&gt;</code>	String	A specific trace that the user wishes to delete
<b>Return Value</b>	None.		

<b>Python Syntax</b>	<code>DeleteTraces(&lt;TraceSelection&gt;)</code>
----------------------	---------------------------------------------------

**Python Example**

```
oModule.DeleteTraces (["XY Plot 1:=",  
                      ["dB(S(LumpPort1,LumpPort1))"],  
                      "XY Plot 2:=", ["Mag_E"]])
```

**VB Syntax**

DeleteTraces <TraceSelection>

**VB Example**

```
oModule.DeleteTraces Array("XY Plot 1:=", _  
                      Array("dB(S(LumpPort1,LumpPort1))"), _  
                      "XY Plot 2:=", Array("Mag_E"))
```

## DoesSupportTraceCharacteristics

Determines whether trace characteristics is supported in a specified display type.

<b>UI Access</b>	N/A								
<b>Parameters</b>	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>&lt;DisplayType&gt;</td><td>String</td><td>Specified display type to check.</td></tr></tbody></table>			Name	Type	Description	<DisplayType>	String	Specified display type to check.
Name	Type	Description							
<DisplayType>	String	Specified display type to check.							
<b>Return Value</b>	<p>Integer</p> <ul style="list-style-type: none"><li>• <b>1</b> - trace characteristics is supported.</li><li>• <b>0</b> - trace characteristics not supported.</li></ul>								

<b>Python Syntax</b>	DoesSupportTraceCharacteristics(<DisplayType>)
<b>Python Example</b>	<code>oModule.DoesSupportTraceCharacteristics ("Rectangular Plot")</code>

<b>VB Syntax</b>	DoesSupportTraceCharacteristics <DisplayType>
<b>VB Example</b>	<code>oModule.DoesSupportTraceCharacteristics "Rectangular Plot"</code>

## DumpAllReportsData

Dumps all reports data to an Ansoft report data file.

<b>UI Access</b>	N/A						
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;FileName&gt;</td> <td>String</td> <td>File name with path.</td> </tr> </tbody> </table>	Name	Type	Description	<FileName>	String	File name with path.
Name	Type	Description					
<FileName>	String	File name with path.					
<b>Return Value</b>	None.						

<b>Python Syntax</b>	DumpAllReportsData(<FileName>)
<b>Python Example</b>	<code>oModule.DumpAllReportsData ("C:/ReportsData.rdat")</code>

<b>VB Syntax</b>	DumpAllReportsData <FileName>
<b>VB Example</b>	<code>oModule.DumpAllReportsData "C:/ReportsData.rdat"</code>

## EditCartesianXMarker

Edits an XMarker value.

<b>UI Access</b>	N/A		
<b>Parameters</b>	Name	Type	Description
	<ReportName>	String	Name of report.
	<MarkerName>	String	Marker name, including any trailing number.
<XValue>	Double	X coordinate.	
<b>Return Value</b>	None.		

<b>Python Syntax</b>	EditCartesianXMarker (<ReportName>, <MarkerName>, <XValue>)
<b>Python Example</b>	oModule.EditCartesianXMarker ("XY Plot 1", "MX1", 0)

<b>VB Syntax</b>	EditCartesianXMarker <ReportName>, <MarkerName>, <XValue>
<b>VB Example</b>	oModule.EditCartesianXMarker "XY Plot1", "MX1", 0

## EditCartesianYMarker

Edits a YMarker value.

<b>UI Access</b>	N/A
------------------	-----

<b>Parameters</b>	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.
<b>Return Value</b>	None		

<b>Python Syntax</b>	EditCartesianYMarker (<ReportName>, <MarkerName>, <AxisName>, <YValue>, <CurveName>)
<b>Python Example</b>	oModule.EditCartesianYMarker("XY Plot 1", "MY1", "Y1", 0, "dB() : Setup1 : Sweep1")

<b>VB Syntax</b>	EditCartesianYMarker <ReportName>, <MarkerName>, <AxisName>, <YValue>, <CurveName>
<b>VB Example</b>	oModule.EditCartesianYMarker "XY Plot 1", "MY1", "Y1", 0, "dB() : Setup1 : Sweep1"

## EditMarker

Edits a marker on a report.

<b>UI Access</b>	N/A		
<b>Parameters</b>	Name	Type	Description
	<ReportName>	String	Name of report.

	<code>&lt;CurveName&gt;</code>	String	Full trace name.
	<code>&lt;PrimarySweepValue&gt;</code>	String	Primary sweep value, including unit.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	<code>EditMarker( &lt;ReportName&gt;, &lt;MarkerName&gt;, &lt;CurveName&gt;, &lt;PrimarySweepValue&gt;)</code>
<b>Python Example</b>	<pre>oModule.EditMarker("XY Plot 1", "m5", "GS1.VAL : TR4 : Cartesian", "3.6159999999997s")</pre>

<b>VB Syntax</b>	<code>EditMarker &lt;ReportName&gt;, &lt;MarkerName&gt;, &lt;CurveName&gt;, &lt;PrimarySweepValue&gt;</code>
<b>VB Example</b>	<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.

<b>UI Access</b>	N/A								
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><code>&lt;ReportName&gt;</code></td> <td>String</td> <td>Name of specified report.</td> </tr> </tbody> </table>			Name	Type	Description	<code>&lt;ReportName&gt;</code>	String	Name of specified report.
Name	Type	Description							
<code>&lt;ReportName&gt;</code>	String	Name of specified report.							

	<code>&lt;ExportFileName&gt;</code>	String	File name to export to.
<b>Return Value</b>	N/A		

<b>Python Syntax</b>	<code>ExportEyeMaskViolation(&lt;ReportName&gt;, &lt;ExportFileName&gt;)</code>
<b>Python Example</b>	<code>oModule.ExportEyeMaskViolation("Variables Plot 1", "C:/eyemask1.csv")</code>

<b>VB Syntax</b>	<code>ExportEyeMaskViolation &lt;ReportName&gt;, &lt;ExportFileName&gt;</code>
<b>VB Example</b>	<code>oModule.ExportEyeMaskViolation "Variables Plot 1", "C:/eyemask1.csv"</code>

## ExportImageToFile [Reporter]

Exports a report image in a specified format. In Release 23.1, this command is fully supports -ng (non-graphical) mode.

<b>UI Access</b>	N/A																	
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><code>&lt;ReportName&gt;</code></td> <td>String</td> <td>Name of report to be exported.</td> </tr> <tr> <td><code>&lt;FileName&gt;</code></td> <td>String</td> <td>Full path of the exported image file name; with extension of jpg, gif, tiff, tif, bmp, or wrl.</td> </tr> <tr> <td><code>&lt;Width&gt;</code></td> <td>Integer</td> <td>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.</td> </tr> <tr> <td><code>&lt;Height&gt;</code></td> <td>Integer</td> <td>Image height in pixels; if width or height is less or equal to zero, use the report window height, or 500 pixels report window is closed.</td> </tr> </tbody> </table>			Name	Type	Description	<code>&lt;ReportName&gt;</code>	String	Name of report to be exported.	<code>&lt;FileName&gt;</code>	String	Full path of the exported image file name; with extension of jpg, gif, tiff, tif, bmp, or wrl.	<code>&lt;Width&gt;</code>	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.	<code>&lt;Height&gt;</code>	Integer	Image height in pixels; if width or height is less or equal to zero, use the report window height, or 500 pixels report window is closed.
Name	Type	Description																
<code>&lt;ReportName&gt;</code>	String	Name of report to be exported.																
<code>&lt;FileName&gt;</code>	String	Full path of the exported image file name; with extension of jpg, gif, tiff, tif, bmp, or wrl.																
<code>&lt;Width&gt;</code>	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.																
<code>&lt;Height&gt;</code>	Integer	Image height in pixels; if width or height is less or equal to zero, use the report window height, or 500 pixels report window is closed.																
<b>Return Value</b>	None.																	

<b>Python Syntax</b>	ExportImageToFile(<ReportName>, <FileName>, <Width>, <Height>)
<b>Python Example</b>	<pre> oModule.ExportImageToFile(     "Rectangular Contour Plot 1",     "D:/work/2015/UV-Export/ppl.gif",     0, 0) </pre>

<b>VB Syntax</b>	ExportImageToFile <ReportName>, <FileName>, <Width>, <Height>
<b>VB Example</b>	<pre> oModule.ExportImageToFile _     "Rectangular Contour Plot 1", _     "D:/work/2015/UV-Export/ppl.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 Ensight use \*.avz. For export to Ensight in -ng mode, the corresponding version of Ensight must be installed. On Linux, it might need manual set environment variable AWP\_ROOT212 to its installation path, e.g. AWP\_ROOT212-2=/installations/ansys\_inc/v212/ for AnsysEDT v21.2 and Ensight 21.2.

ExportModelImageToFile supports export overlay of polar plot 3D with existing transformation (scaling, rotation and translation) in -ng (non-graphical) mode.

<b>UI Access</b>	<b>Modeler &gt; Export.</b>		
<b>Parameters</b>	Name <path>	Type String	Description Full file path including extension.

	<table border="1"> <tr> <td><code>&lt;width&gt;</code></td><td>Integer</td><td>Width in pixels (use 0 for default).</td></tr> <tr> <td><code>&lt;height&gt;</code></td><td>Integer</td><td>Height in pixels (use 0 for default).</td></tr> <tr> <td><code>&lt;Parameters&gt;</code></td><td>Array</td><td> <p>Structured array.</p> <pre>         Array("NAME:SaveImageParams",               "ShowAxis:=" , &lt;string containing boolean&gt;,               "ShowGrid:=" , &lt;string containing boolean&gt;,               "ShowRuler:=" , &lt;string containing boolean&gt;,               "ShowRegion:=" , &lt;string&gt;,               "Selections:=" , &lt;string&gt;,               "FieldPlotSelections:=", &lt;string&gt;' # Comma delimited string. #Use to set which field plot to show.               "Orientation:=" , &lt;string&gt;               "ShowOrientationGadget:=" , &lt;False&gt;             )       </pre> </td></tr> </table>	<code>&lt;width&gt;</code>	Integer	Width in pixels (use 0 for default).	<code>&lt;height&gt;</code>	Integer	Height in pixels (use 0 for default).	<code>&lt;Parameters&gt;</code>	Array	<p>Structured array.</p> <pre>         Array("NAME:SaveImageParams",               "ShowAxis:=" , &lt;string containing boolean&gt;,               "ShowGrid:=" , &lt;string containing boolean&gt;,               "ShowRuler:=" , &lt;string containing boolean&gt;,               "ShowRegion:=" , &lt;string&gt;,               "Selections:=" , &lt;string&gt;,               "FieldPlotSelections:=", &lt;string&gt;' # Comma delimited string. #Use to set which field plot to show.               "Orientation:=" , &lt;string&gt;               "ShowOrientationGadget:=" , &lt;False&gt;             )       </pre>
<code>&lt;width&gt;</code>	Integer	Width in pixels (use 0 for default).								
<code>&lt;height&gt;</code>	Integer	Height in pixels (use 0 for default).								
<code>&lt;Parameters&gt;</code>	Array	<p>Structured array.</p> <pre>         Array("NAME:SaveImageParams",               "ShowAxis:=" , &lt;string containing boolean&gt;,               "ShowGrid:=" , &lt;string containing boolean&gt;,               "ShowRuler:=" , &lt;string containing boolean&gt;,               "ShowRegion:=" , &lt;string&gt;,               "Selections:=" , &lt;string&gt;,               "FieldPlotSelections:=", &lt;string&gt;' # Comma delimited string. #Use to set which field plot to show.               "Orientation:=" , &lt;string&gt;               "ShowOrientationGadget:=" , &lt;False&gt;             )       </pre>								
<b>Return Value</b>	None.									

<b>Python Syntax</b>	<code>ExportModelImageToFile(&lt;path&gt; &lt;width&gt; &lt;height&gt; &lt;Parameters&gt;)</code>
<b>Python Example</b>	<pre> oEditor.ExportModelImageToFile   ("C:/Users/jdoe/Desktop/export.bmp",    1920,    1080,   ) </pre>

```
[  
    "NAME:SaveImageParams",  
    "ShowAxis:=" , "True",  
    "ShowGrid:=" , "True",  
    "ShowRuler:=" , "True",  
    "ShowRegion:=" , "Default",  
    "Selections:=" , "",  
    "FieldPlotSelections:=" , "",  
    "FitToSelections:=" , "",  
    "FitToFieldPlotSelections:=" , "",  
    "Orientation:=" , ""  
])
```

VB Syntax	ExportModelImageToFile <path> <width> <height> <Parameters>
<b>VB Example</b>	<pre>oEditor.ExportModelImageToFile     "C:/Users/jdoe/Desktop/export.bmp",     1383,     512,     Array ("NAME:SaveImageParams",         "ShowAxis:=" , "True",</pre>

```

    "ShowGrid:=", "True",
    "ShowRuler:=", "True",
    "ShowRegion:=", "Default",
    "Selections:=", "",
    "FitToFieldPlotSelections:=", "",
    "Orientation:=", ""

```

## ExportModelMeshToFile

Exports geometry model to a 3D model file (e.g. \*.obj, \*.wrl, etc.).

<b>UI Access</b>	N/A		
<b>Parameters</b>	Name	Type	Description
	<filePath>	String	Full file path, including extension *.obj, *.wrl, etc
<b>Return Value</b>	None.		

<b>Python Syntax</b>	ExportModelMeshToFile <filePath>, <selections>
<b>Python Example</b>	<code>oEditor.ExportModelMeshToFile("E:/MyDir/scriptRun/2Selected-ng.obj", ["BotCover-", "AveragingVolumeAtPeakRMSEfieldLocation"])</code>

<b>VB Syntax</b>	ExportModelMeshToFile <filePath>, <selections>
<b>VB Example</b>	<code>oEditor.ExportModelMeshToFile("E:/MyDir/scriptRun/2Selected-ng.obj", ["BotCover-", "AveragingVolumeAtPeakRMSEfieldLocation"])</code>

## 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-out- put/DirectivityPlot1.case")</pre>

---

```
oModule.ExportPlot3DToFile("Gain Plot 1", "D:/projects/test2-output/GainPlot1.case")
```

<b>VB Syntax</b>	ExportPlot3DToFile <plotName> <Path>
<b>VB Example</b>	oModule.ExportPlot3DToFile("rE Plot 1", "D:/projects/test2-output/rEPlot1.case")

## ExportReport

**Note:**

The ExportReport script command has been replaced by the script command [ExportToFile](#). ExportReport remains in order to retain backward compatibility for existing scripts, but it is strongly recommended that you now use [ExportToFile](#).

Export a report to a data file.

*Command:* None

*Syntax:* ExportReport <ReportName>, <FileName>, <FileExtension>

*Return Value:* None

*Parameters:* <ReportName>

Type: string

<Filename>

Type: string

<FileExtension>

Type: string

*VB Example:*

```
Dim oAnsoftApp  
Dim oDesktop  
Dim oProject  
Dim oDesign  
Dim oEditor  
Dim oModule  
  
Set oAnsoftApp = CreateObject("Ansoft.ElectronicsDesktop")  
Set oDesktop = oAnsoftApp.GetAppDesktop()  
oDesktop.RestoreWindow  
  
Set oProject = oDesktop.SetActiveProject("BJTinverter")  
Set oDesign = oProject.SetActiveDesign("Nexxim1")  
oDesign.ExportReport "Data Table 1", "table_test", "csv"
```

<b>Python Syntax</b>	ExportReport( <ReportName>, <FileName>)
<b>Python Example</b>	<pre>oDesign.ExportReport(     "Plot1", "c:\report1.dat")</pre>

## ExportReportDataToFile

Exports report data to a file.

<b>UI Access</b>	N/A		
<b>Parameters</b>	Name	Type	Description
	<ReportName>	String	Name of specified report.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	ExportReportDataToFile(<ReportName>, <ExportFileName>)
<b>Python Example</b>	oModule.ExportReportDataToFile("Plot 1", "C:/Plot1data.rdat")

<b>VB Syntax</b>	ExportReportDataToFile <ReportName>, <ExportFileName>
<b>VB Example</b>	oModule.ExportReportDataToFile "Plot 1", "C:/Plot1data.rdat"

## ExportTableToFile

Exports a marker table from a report to a file.

<b>UI Access</b>	Right-click on a plot, select <b>Marker</b> > <b>Export Marker Table...</b> or <b>Export Delta Marker Table...</b>		
<b>Parameters</b>	Name	Type	Description
	<ReportName>	String	Name of specified report.

	<i>&lt;TableType&gt;</i>	String	Type of marker table to export. "Marker" or "DeltaMarker".
<b>Return Value</b>	None.		

<b>Python Syntax</b>	ExportTableToFile(<ReportName>, <ExportFileName>, <TableType>)
<b>Python Example</b>	oModule.ExportTableToFile("Plot 1", "C:/Marker.csv", "Marker")

<b>VB Syntax</b>	ExportTableToFile <ReportName>, <ExportFileName>, <TableType>
<b>VB Example</b>	oModule.ExportTableToFile "Plot 1", "C:/Marker.csv", "Marker"

## ExportToFile [Reporter]

From a data table or plot, generates text format, comma delimited, tab delimited, .dat, or .rdat type output files.

<b>UI Access</b>	Right-click on report name in the Project tree and select <b>Export</b> .		
<b>Parameters</b>	Name	Type	Description
<b>UI Access</b>	Right-click on report name in the Project tree and select <b>Export</b> .		
<b>Parameters</b>	Name	Type	Description
	<ReportName>	String	Name of the report
	<FileName>	String	Path and File Name  Supported formats
	.txt		Post processor format file
	.csv		Comma-delimited data file
	.tab		Tab-separated file

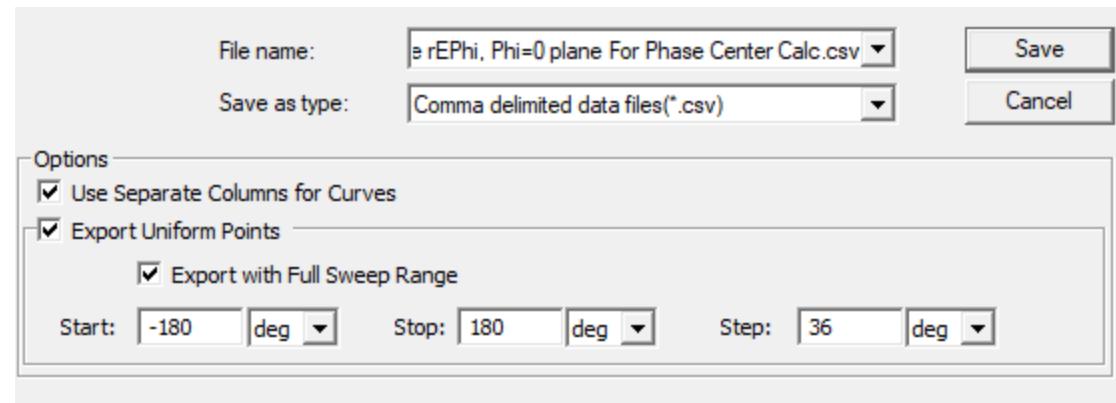
			.dat	Ansys plot data file
			.rdat	Ansys report data file
<b>Return Value</b>	None			

<b>Python Syntax</b>	ExportToFile (<ReportName>, <FileName>)
<b>Python Example</b>	<pre>oModule.ExportToFile(     "Plot1", "c:\report1.dat")</pre>

<b>VB Syntax</b>	ExportToFile <ReportName>, <FileName>
<b>VB Example</b>	<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.



<b>UI Access</b>	Right-click on report name in the project tree and select <b>Export Data</b> .		
<b>Parameters</b>	Name	Type	Description
	<ReportName>	String	Name of report to be exported.
	<FileName>	String	Full path of the exported image file name; with extension of .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"

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

<b>Python Syntax</b>	ExportUniformPointsToFile(<ReportName>, <FileName>, <SweepRange>)
<b>Python Example</b>	<pre>oModule.ExportUniformPointsToFile("S Parameter Table 2", "D:/MyFiles/cftt.csv", "4GHz", "5GHz", "1GHz", False, " kV, MHz, yd ", False)</pre>

<b>VB Syntax</b>	ExportUniformPointsToFile <ReportName>, <FileName>
<b>VB Example</b>	<pre>oModule.ExportUniformPointsToFile "rETheta and rEPhi [dB]", "C:/MyFiles/rETheta and rEPhi [dB].csv", "-180deg", "180deg", "36deg"</pre>

## GetAllCategories

Get all available category names (not including variable and output-variables) in a solution for a report type and display type, returned as an array of text strings.

<b>UI Access</b>	NA															
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;ReportType&gt;</td> <td>String</td> <td>Report type name.</td> </tr> <tr> <td>&lt;DisplayType&gt;</td> <td>String</td> <td>Name of display type.</td> </tr> <tr> <td>&lt;SolutionName&gt;</td> <td>String</td> <td>Name of solution.</td> </tr> <tr> <td>&lt;SimValueCtx&gt;</td> <td>Array</td> <td>A context name, or array of strings that encode the contexts.</td> </tr> </tbody> </table>	Name	Type	Description	<ReportType>	String	Report type name.	<DisplayType>	String	Name of display type.	<SolutionName>	String	Name of solution.	<SimValueCtx>	Array	A context name, or array of strings that encode the contexts.
Name	Type	Description														
<ReportType>	String	Report type name.														
<DisplayType>	String	Name of display type.														
<SolutionName>	String	Name of solution.														
<SimValueCtx>	Array	A context name, or array of strings that encode the contexts.														
<b>Return Value</b>	Array of text strings															

<b>Python Syntax</b>	GetAllCategories(<ReportType>, <DisplayType>, <SolutionName>, <SimValueCtxt>)
<b>Python Example</b>	<pre>oModule.GetAllCategories("Far Fields", "Rectangular Plot", "Setup1 : LastAdaptive", "Infinite Sphere1")</pre>

<b>VB Syntax</b>	GetAllCategories <ReportType>, <DisplayType>, <SolutionName>, <SimValueCtxt>
<b>VB Example</b>	<pre>oModule.GetAllCategories _ "Far Fields", "Rectangular Plot", _ "Setup1 : LastAdaptive", "Infinite Sphere1"</pre>

## GetAllQuantities

Gets all available quantity names in category, returned as an array of text strings.

<b>UI Access</b>	NA																		
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;ReportType&gt;</td> <td>String</td> <td>Report type name.</td> </tr> <tr> <td>&lt;DisplayType&gt;</td> <td>String</td> <td>Display type name.</td> </tr> <tr> <td>&lt;SolutionName&gt;</td> <td>String</td> <td>Name of solution.</td> </tr> <tr> <td>&lt;SimValueCtxt&gt;</td> <td>Array</td> <td>A context name, or array of string that encoded the contexts(l).</td> </tr> <tr> <td>&lt;CategoryName&gt;</td> <td>String</td> <td>A category name as input parameter. a category name returned in GetAllCategories() or "Variables", or "Output Variables"</td> </tr> </tbody> </table>	Name	Type	Description	<ReportType>	String	Report type name.	<DisplayType>	String	Display type name.	<SolutionName>	String	Name of solution.	<SimValueCtxt>	Array	A context name, or array of string that encoded the contexts(l).	<CategoryName>	String	A category name as input parameter. a category name returned in GetAllCategories() or "Variables", or "Output Variables"
Name	Type	Description																	
<ReportType>	String	Report type name.																	
<DisplayType>	String	Display type name.																	
<SolutionName>	String	Name of solution.																	
<SimValueCtxt>	Array	A context name, or array of string that encoded the contexts(l).																	
<CategoryName>	String	A category name as input parameter. a category name returned in GetAllCategories() or "Variables", or "Output Variables"																	
<b>Return Value</b>	Array of text strings																		

<b>Python Syntax</b>	GetAllQuantities(<ReportType>,<DisplayType>, <SolutionName>, <SimValueCtxt>, <CategoryName>)
<b>Python Example</b>	<pre> oModule.GetAllQuantities(     "Far Fields", "Rectangular Plot",     "Setup1 : LastAdaptive",     "Infinite Spherel", "Gain") </pre>

<b>VB Syntax</b>	GetAllQuantities <ReportType>,<DisplayType>, <SolutionName>, <SimValueCtxt>, <CategoryName>
<b>VB Example</b>	<pre> oModule.GetAllQuantities _     "Far Fields", "Rectangular Plot", _     "Setup1 : LastAdaptive", )     "Infinite Spherel", "Gain" </pre>

## GetAllReportNames

Gets the names of existing reports in a design

<b>UI Access</b>	N/A
<b>Parameters</b>	None.
<b>Return Value</b>	Array of report names

<b>Python Syntax</b>	GetAllReportNames()
<b>Python Example</b>	<code>oModule.GetAllReportNames ()</code>

<b>VB Syntax</b>	GetAllReportNames
<b>VB Example</b>	<code>oModule.GetAllReportNames</code>

## GetAvailableDisplayTypes

Retrieves all supported display types in report type as an array of text strings.

<b>UI Access</b>	N/A						
<b>Parameters</b>	<table border="1"><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><code>&lt;ReportType&gt;</code></td><td>String</td><td>Report type name.</td></tr></table>	Name	Type	Description	<code>&lt;ReportType&gt;</code>	String	Report type name.
Name	Type	Description					
<code>&lt;ReportType&gt;</code>	String	Report type name.					
<b>Return Value</b>	Array of text strings						

<b>Python Syntax</b>	GetAvailableDisplayTypes(<ReportType>)
<b>Python Example</b>	<code>oModule.GetAvailableDisplayTypes ("Far Fields")</code>

<b>VB Syntax</b>	GetAvailableDisplayTypes <ReportType>
<b>VB Example</b>	<code>oModule.GetAvailableDisplayTypes "Far Fields"</code>

## GetAvailableReportTypes

Retrieves all available report types in the current Design as an array of text string.

<b>UI Access</b>	N/A
<b>Parameters</b>	None.
<b>Return Value</b>	Array of text strings

<b>Python Syntax</b>	GetAvailableReportTypes()
<b>Python Example</b>	<code>oModule.GetAvailableReportTypes ()</code>

<b>VB Syntax</b>	GetAvailableReportTypes
<b>VB Example</b>	<code>oModule.GetAvailableReportTypes</code>

## GetAvailableSolutions

Gets all available solutions in report type as an array of text strings.

<b>UI Access</b>	N/A		
<b>Parameters</b>	Name <i>&lt;ReportType&gt;</i>	Type String	Description Report type name.
<b>Return Value</b>	Array of text strings		

<b>Python Syntax</b>	GetAvailableSolutions(<ReportType>)
<b>Python Example</b>	oModule.GetAvailableSolutions ("Far Fields")

<b>VB Syntax</b>	GetAvailableSolutions <ReportType>
<b>VB Example</b>	oModule.GetAvailableSolutions "Far Fields"

## GetChildNames [Report Setup]

Gets a list of report names.

<b>UI Access</b>	N/A
<b>Parameters</b>	None.
<b>Return Value</b>	Array of strings containing all report names.

<b>Python Syntax</b>	GetChildNames()
<b>Python Example</b>	oModule.GetChildNames ()

<b>VB Syntax</b>	GetChildNames
<b>VB Example</b>	oModule.GetChildNames

## 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.

<b>UI Access</b>	NA						
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;ObjectPath&gt;</td> <td>String</td> <td>Report Name or <a href="#">an object path</a> beginning with a report name.</td> </tr> </tbody> </table>	Name	Type	Description	<ObjectPath>	String	Report Name or <a href="#">an object path</a> beginning with a report name.
Name	Type	Description					
<ObjectPath>	String	Report Name or <a href="#">an object path</a> beginning with a report name.					
<b>Return Value</b>	A ReportSetup(Results) Module Child Object, [ <a href="#">ReportSetup(Results) Module Child Objects</a> ]						

<b>Python Syntax</b>	GetChildObject(<ObjectPath>)
<b>Python Example</b>	<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>

<b>VB Syntax</b>	GetChildObject <ObjectPath>
<b>VB Example</b>	<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.

<b>UI Access</b>	N/A
<b>Parameters</b>	None.
<b>Return Value</b>	Array of strings containing child object types.

<b>Python Syntax</b>	GetChildTypes ()
<b>Python Example</b>	<code>oModule.GetChildTypes ()</code>

<b>VB Syntax</b>	GetChildTypes
<b>VB Example</b>	<code>oModule.GetChildTypes</code>

## GetCurvePropServerName

Gets the PropServer (the owner of the properties, or the list containing them) name of a curve.

<b>UI Access</b>	N/A		
<b>Parameters</b>	Name	Type	Description
	<code>&lt;ReportName&gt;</code>	String	Name of specified report.
	<code>&lt;TraceName&gt;</code>	String	Name of specified trace.

<b>Return Value</b>	Array of string containing the PropServer name.
---------------------	-------------------------------------------------

<b>Python Syntax</b>	GetCurvePropServerName(<ReportName>, <TraceName>)
<b>Python Example</b>	oModule.GetCurvePropServerName ("Plot 1", "Phase")

<b>VB Syntax</b>	GetCurvePropServerName <ReportName>, <TraceName>
<b>VB Example</b>	oModule.GetCurvePropServerName "Plot 1", "Phase"

## GetDisplayType

Gets the display type of a specified report.

<b>UI Access</b>	NA						
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;ReportName&gt;</td> <td>String</td> <td>Report name</td> </tr> </tbody> </table>	Name	Type	Description	<ReportName>	String	Report name
Name	Type	Description					
<ReportName>	String	Report name					
<b>Return Value</b>	String containing display type.						

<b>Python Syntax</b>	GetDisplayType(<ReportName>)
<b>Python Example</b>	oModule.GetDisplayType ("Design Plot 1")

<b>VB Syntax</b>	GetDisplayType < <i>ReportName</i> >
<b>VB Example</b>	<code>oModule.GetDisplayType "Design Plot 1"</code>

## GetDynLinkIntrinsicVariables

Gets variable names from a trace included in dynamic link outputs.

<b>UI Access</b>	N/A						
<b>Parameters</b>	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>&lt;<i>TraceName</i>&gt;</td><td>String</td><td>Trace name with its report name.</td></tr></tbody></table>	Name	Type	Description	< <i>TraceName</i> >	String	Trace name with its report name.
Name	Type	Description					
< <i>TraceName</i> >	String	Trace name with its report name.					
<b>Return Value</b>	Array of variable names						

<b>Python Syntax</b>	GetDynLinkIntrinsicVariables(< <i>TraceName</i> >)
<b>Python Example</b>	<code>oModule.GetDynLinkIntrinsicVariables ("Plot 1:Trace")</code>

<b>VB Syntax</b>	GetDynLinkIntrinsicVariables < <i>TraceName</i> >
<b>VB Example</b>	<code>oModule.GetDynLinkIntrinsicVariables "Plot 1:Trace")</code>

## GetDynLinkQtyValueState

Gets the state of the quantity values from a source dynamic linked trace.

<b>UI Access</b>	N/A
------------------	-----

<b>Parameters</b>	Name	Type	Description
	<TraceName>	String	Name of specified source trace.
	<QtyName>	String	Name of specified quantity value.
<b>Return Value</b>	Array of strings containing states.		

<b>Python Syntax</b>	GetDynLinkQtyValueState(<TraceName>, <QtyName>)
<b>Python Example</b>	oModule.GetDynLinkQtyValueState("Plot 1:Trace1", "")

<b>VB Syntax</b>	GetDynLinkQtyValueState <TraceName>, <QtyName>
<b>VB Example</b>	oModule.GetDynLinkQtyValueState "Plot 1:Trace1", ""

## GetDynLinkTraces

Gets the names of the dynamic linked traces.

<b>UI Access</b>	N/A								
<b>Parameters</b>	<table border="1"> <tr> <td>Name</td> <td>Type</td> <td>Description</td> </tr> <tr> <td>&lt;SoluName&gt;</td> <td>String</td> <td>Name of the source solution. If empty, refer to current solution.</td> </tr> </table>			Name	Type	Description	<SoluName>	String	Name of the source solution. If empty, refer to current solution.
Name	Type	Description							
<SoluName>	String	Name of the source solution. If empty, refer to current solution.							
<b>Return Value</b>	Array of strings containing trace names.								

<b>Python Syntax</b>	GetDynLinkTraces(<SoluName>)
<b>Python Example</b>	oModule.GetDynLinkTraces("")

<b>VB Syntax</b>	GetDynLinkTraces <SolName>
<b>VB Example</b>	<code>oModule.GetDynLinkTraces ""</code>

## GetDynLinkVariableValues

Gets the values of a variable from a dynamic linked trace.

<b>UI Access</b>	N/A									
<b>Parameters</b>	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>&lt;TraceName&gt;</td><td>String</td><td>Name of specified dynamic linked trace, with its report name.</td></tr><tr><td>&lt;VarName&gt;</td><td>String</td><td>Name of specified variable.</td></tr></tbody></table>	Name	Type	Description	<TraceName>	String	Name of specified dynamic linked trace, with its report name.	<VarName>	String	Name of specified variable.
Name	Type	Description								
<TraceName>	String	Name of specified dynamic linked trace, with its report name.								
<VarName>	String	Name of specified variable.								
<b>Return Value</b>	Array of strings containing variable values.									

<b>Python Syntax</b>	GetDynLinkVariableValues(<TraceName>, <VarName>)
<b>Python Example</b>	<code>oModule.GetDynLinkVariableValues("Plot 1:Trace", "Var1")</code>

<b>VB Syntax</b>	GetDynLinkVariableValues <TraceName>, <VarName>
<b>VB Example</b>	<code>oModule.GetDynLinkVariableValues "Plot 1:Trace", "Var1"</code>

## GetName

Returns the name and ID of the active design. If the design is a sub-circuit, returns the parent design also.

<b>UI Access</b>	N/A
<b>Parameters</b>	None.
<b>Return Value</b>	String indicating the name and ID of the active design and parent design (if the active design is a sub-circuit).

<b>Python Syntax</b>	GetName()
<b>Python Example</b>	<code>design_name = oDesign.GetName()</code>

<b>VB Syntax</b>	GetName
<b>VB Example</b>	<code>design_name = oDesign.GetName</code>

## GetObjPath [Design]

Obtains the path to the design.

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

<b>Python Syntax</b>	GetObjPath()
<b>Python Example</b>	<code>oDesign.GetObjPath()</code>

<b>VB Syntax</b>	GetObjPath
<b>VB Example</b>	<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.

<b>UI Access</b>	N/A		
<b>Parameters</b>	Name <i>&lt;PropTab&gt;</i>	Type String	Description One of the following, where tab titles are shown in parentheses: <ul style="list-style-type: none"><li>• PassedParameterTab ("Parameter Values")</li><li>• DefinitionParameterTab (Parameter Defaults")</li><li>• LocalVariableTab ("Variables" or "Local Variables")</li><li>• ProjectVariableTab ("Project variables")</li><li>• ConstantsTab ("Constants")</li></ul>

		<ul style="list-style-type: none"> <li>• BaseElementTab ("Symbol" or "Footprint")</li> <li>• ComponentTab ("General")</li> <li>• Component("Component")</li> <li>• CustomTab ("Intrinsic Variables")</li> <li>• Quantities ("Quantities")</li> <li>• Signals ("Signals")</li> </ul>
	<PropServer>	String An object identifier, generally returned from another script method, such as CompInst@R;2;3
	<PropName>	String Name of the property.
<b>Return Value</b>	String value of the property.	

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

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

	...
--	-----

## GetPropNames [Reporter]

Report setup module does not have its own property, this function always returns empty array.

<b>UI Access</b>	N/A
<b>Parameters</b>	None.
<b>Return Value</b>	Empty array.

<b>Python Syntax</b>	GetPropNames()
<b>Python Example</b>	<code>oModule.GetPropNames ()</code>

<b>VB Syntax</b>	GetPropNames
<b>VB Example</b>	<code>oModule.GetPropNames</code>

## GetPropertyValue [Report Setup]

Gets the property value for a Report, or reports' child object.

<b>UI Access</b>	N/A
------------------	-----

<b>Parameters</b>	Name <i>&lt;PropPath&gt;</i>	Type String	Description A child object's property path. See <a href="#">property path discussion here</a> .
<b>Return Value</b>	String containing the property value.		

<b>Python Syntax</b>	GetPropValue(<PropPath>)
<b>Python Example</b>	<code>oModule.GetPropValue("S Parameter Plot 1/Display Type")</code>

<b>VB Syntax</b>	GetPropValue <PropPath>
<b>VB Example</b>	<code>oModule.GetPropValue "S Parameter Plot 1/Display Type"</code>

## GetQtyExpressionsForSourceTrace

Gets the quantity expressions from a specified source trace.

<b>UI Access</b>	N/A		
<b>Parameters</b>	Name <i>&lt;SourceTraceName&gt;</i>	Type String	Description Name of specified source trace.
<b>Return Value</b>	Array of strings containing quantity expressions.		

<b>Python Syntax</b>	GetQtyExpressionsForSourceTrace(<SourceTraceName>)
<b>Python Example</b>	<code>oModule.GetQtyExpressionsForSourceTrace("Plot 1:Trace1")</code>

<b>VB Syntax</b>	GetQtyExpressionsForSourceTrace <SourceTraceName>
<b>VB Example</b>	<code>oModule.GetQtyExpressionsForSourceTrace "Plot 1:Trace1"</code>

## GetReportTraceNames

Gets the names of existing trace names in a plot.

<b>UI Access</b>	N/A						
<b>Parameters</b>	<table border="1"><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;PlotName&gt;</td><td>String</td><td>Name of specified plot.</td></tr></table>	Name	Type	Description	<PlotName>	String	Name of specified plot.
Name	Type	Description					
<PlotName>	String	Name of specified plot.					
<b>Return Value</b>	Array of strings containing trace names.						

<b>Python Syntax</b>	GetReportTraceNames(<PlotName>)
<b>Python Example</b>	<code>oModule.GetReportTraceNames ("SParameter Plot 1")</code>

<b>VB Syntax</b>	GetReportTraceNames <PlotName>
<b>VB Example</b>	<code>oModule.GetReportTraceNames "SParameter Plot 1"</code>

## GetReportSummaryForRegressionTesting

Gets report summary from a dumped report file.

<b>UI Access</b>	N/A
------------------	-----

<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th><th>Type</th><th>Description</th></tr> </thead> <tbody> <tr> <td>&lt;ReportName&gt;</td><td>String</td><td>Name of a specified dumped report.</td></tr> </tbody> </table>	Name	Type	Description	<ReportName>	String	Name of a specified dumped report.
Name	Type	Description					
<ReportName>	String	Name of a specified dumped report.					
<b>Return Value</b>	String containing report summary.						

<b>Python Syntax</b>	GetReportSummaryForRegressionTesting(<ReportName>)
<b>Python Example</b>	<code>oModule.GetReportSummaryForRegressionTesting("C:/report.rdat")</code>

<b>VB Syntax</b>	GetReportSummaryForRegressionTesting <ReportName>
<b>VB Example</b>	<code>oModule.GetReportSummaryForRegressionTesting "C:/report.rdat"</code>

## GetSolutionContexts

Gets all available solution context names in a solution as an array of text strings.

<b>UI Access</b>	N/A												
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;ReportType&gt;</td> <td>String</td> <td>Report type name.</td> </tr> <tr> <td>&lt;DisplayType&gt;</td> <td>String</td> <td>Display type name.</td> </tr> <tr> <td>&lt;SolutionName&gt;</td> <td>String</td> <td>Name of solution.</td> </tr> </tbody> </table>	Name	Type	Description	<ReportType>	String	Report type name.	<DisplayType>	String	Display type name.	<SolutionName>	String	Name of solution.
Name	Type	Description											
<ReportType>	String	Report type name.											
<DisplayType>	String	Display type name.											
<SolutionName>	String	Name of solution.											
<b>Return Value</b>	Array of text strings												

<b>Python Syntax</b>	GetSolutionContexts(<ReportType>, <DisplayType>, <SolutionName>)
----------------------	------------------------------------------------------------------

<b>Python Example</b>	<pre>oModule.GetSolutionContexts(     "Far Fields", "Rectangular Plot", "Setup1:LastAdaptive")</pre>
-----------------------	----------------------------------------------------------------------------------------------------------

<b>VB Syntax</b>	GetSolutionContexts < <i>ReportType</i> >, < <i>DisplayType</i> >, < <i>SolutionName</i> >
<b>VB Example</b>	<pre>oModule.GetSolutionContexts _     "Far Fields", "Rectangular Plot", _     "Setup1:LastAdaptive"</pre>

## GetSolutionDataPerVariation

Obtains solution data for a given report type and solution. You must have already run a simulation.

<b>UI Access</b>	N/A																		
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;<i>reportTypeArg</i>&gt;</td> <td>String</td> <td>Report type name as input parameter.</td> </tr> <tr> <td>&lt;<i>solutionNameArg</i>&gt;</td> <td>String</td> <td>Solution name as input parameter.</td> </tr> <tr> <td>&lt;<i>simValueCtxtArg</i>&gt;</td> <td>Structured Array</td> <td>Same as ContextArray values created in the relevant <a href="#">CreateReport</a> script.</td> </tr> <tr> <td>&lt;<i>familiesArg</i>&gt;</td> <td>Array of Strings</td> <td>Same as FamiliesArray values created in the relevant <a href="#">CreateReport</a> script.</td> </tr> <tr> <td>&lt;<i>expressionArg</i>&gt;</td> <td>String or Array of Strings</td> <td>Text string or array of text strings; valid expression, may validate it as the data-table Y-component.</td> </tr> </tbody> </table>	Name	Type	Description	< <i>reportTypeArg</i> >	String	Report type name as input parameter.	< <i>solutionNameArg</i> >	String	Solution name as input parameter.	< <i>simValueCtxtArg</i> >	Structured Array	Same as ContextArray values created in the relevant <a href="#">CreateReport</a> script.	< <i>familiesArg</i> >	Array of Strings	Same as FamiliesArray values created in the relevant <a href="#">CreateReport</a> script.	< <i>expressionArg</i> >	String or Array of Strings	Text string or array of text strings; valid expression, may validate it as the data-table Y-component.
Name	Type	Description																	
< <i>reportTypeArg</i> >	String	Report type name as input parameter.																	
< <i>solutionNameArg</i> >	String	Solution name as input parameter.																	
< <i>simValueCtxtArg</i> >	Structured Array	Same as ContextArray values created in the relevant <a href="#">CreateReport</a> script.																	
< <i>familiesArg</i> >	Array of Strings	Same as FamiliesArray values created in the relevant <a href="#">CreateReport</a> script.																	
< <i>expressionArg</i> >	String or Array of Strings	Text string or array of text strings; valid expression, may validate it as the data-table Y-component.																	
<b>Return Value</b>	<p>ARRAY of ISolutionDataResultComInterface objects, containing:</p> <ul style="list-style-type: none"> <li>• <a href="#">GetSweepNames()</a></li> </ul>																		

- [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].

<b>Python Syntax</b>	GetSolutionDataPerVariation(reportTypeArg, solutionNameArg, simValueCtxtArg, familiesArg, expressionArg)
<b>Python Example</b>	<pre> oModule = oDesign.GetModule("ReportSetup") arr = oModule.GetSolutionDataPerVariation('Modal Solution Data', 'Setup1 : Sweep', ['Domain:=' , 'Sweep'], ['Freq:=' , ['All']] , 'offset:=' , ['All']] , ['S (Port1,Port1)', 'dB(S(Port1,Port3))']) </pre>

<b>VB Syntax</b>	GetSolutionDataPerVariation(reportTypeArg, solutionNameArg, simValueCtxtArg, familiesArg, expressionArg)
<b>VB Example</b>	<pre> set solutionData = oModule.GetSolutionDataPerVariation(     "Modal Solution Data",     "Setup1 : Sweep", </pre>

```

        Array("Domain:=", "Sweep"),
        Array("Freq:=", Array("All"), "offset:=", Array("All")),
        Array("S(Port1,Port1)", "dB(S(Port1,Port3))")
    )
)

```

## GetDataExpressions

Returns data expressions.

**Important:**

This script does not function on its own, but as part of [GetSolutionDataPerVariation](#).

<b>UI Access</b>	N/A
<b>Parameters</b>	N/A
<b>Return Value</b>	Array of text strings

<b>Python Syntax</b>	GetDataExpressions()
<b>Python Example</b>	expressions = oModule.GetDataExpressions()

<b>VB Syntax</b>	GetDataExpressions()
<b>VB Example</b>	<pre> dim expressions expressions = oModule.GetDataExpressions() </pre>

## GetDataUnits

Returns text string containing units.

**Important:**

This script does not function on its own, but as part of [GetSolutionDataPerVariation](#).

<b>UI Access</b>	N/A		
<b>Parameters</b>	Name <expressionString>	Type String	Description Can be returned from <a href="#">GetDataExpressions()</a>
<b>Return Value</b>	Text string of units; empty if no units		

<b>Python Syntax</b>	GetDataUnits(<expressionString>)
<b>Python Example</b>	<code>oModule.GetDataUnits(expressions)</code>

<b>VB Syntax</b>	GetDataUnits(<expressionString>)
<b>VB Example</b>	<code>oModule.GetDataUnits(expressions)</code>

## GetDesignVariableNames

Returns array of strings containing design variable names.

**Important:**

This script does not function on its own, but as part of [GetSolutionDataPerVariation](#).

<b>UI Access</b>	NA
<b>Parameters</b>	NA
<b>Return Value</b>	Array of strings

<b>Python Syntax</b>	<code>GetDesignVariableNames()</code>
<b>Python Example</b>	<code>names = oModule.GetDesignVariableNames()</code>

<b>VB Syntax</b>	<code>GetDesignVariableNames()</code>
<b>VB Example</b>	<code>dim names names = oModule.GetDesignVariableNames()</code>

## GetDesignVariableUnits

Returns array of strings containing design variable units.

**Important:**

This script does not function on its own, but as part of [GetSolutionDataPerVariation](#).

<b>UI Access</b>	N/A						
<b>Parameters</b>	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>&lt;varName&gt;</td><td>String</td><td>Can be returned from <a href="#">GetDesignVariableNames()</a></td></tr></tbody></table>	Name	Type	Description	<varName>	String	Can be returned from <a href="#">GetDesignVariableNames()</a>
Name	Type	Description					
<varName>	String	Can be returned from <a href="#">GetDesignVariableNames()</a>					
<b>Return Value</b>	Text string of units; empty if no units.						

<b>Python Syntax</b>	GetDesignVariableUnits(<varName>)
<b>Python Example</b>	units = oModule.GetDesignVariableUnits('Variable Name')

<b>VB Syntax</b>	GetDesignVariableUnits(<varName>)
<b>VB Example</b>	dim units units = oModule.GetDesignVariableUnits("Variable Name")

## GetDesignVariableValue

Returns a design variable's value.

**Important:**

This script does not function on its own, but as part of [GetSolutionDataPerVariation](#).

<b>UI Access</b>	N/A
------------------	-----

Parameters	Name	Type	Description
	<varName>	String	Can be returned from <a href="#">GetDesignVariableNames()</a>
	<siValue>	Boolean	True to return SI-value; False to return by <a href="#">GetDesignVariableUnits()</a>
Return Value	Double value		

<b>Python Syntax</b>	<code>GetDesignVariableValue(&lt;varName&gt;, &lt;siValue&gt;)</code>
<b>Python Example</b>	<code>value = oModule.GetDesignVariableValue('varName', 1)</code>

<b>VB Syntax</b>	<code>GetDesignVariableValue(&lt;varName&gt;, &lt;siValue&gt;)</code>
<b>VB Example</b>	<code>dim value value = oModule.GetDesignVariableValue("varName", True)</code>

## GetDesignVariationKey

Returns a design's Variation Key.

**Important:**

This script does not function on its own, but as part of [GetSolutionDataPerVariation](#).

<b>UI Access</b>	N/A
<b>Parameters</b>	N/A
<b>Return Value</b>	String containing variation key.

<b>Python Syntax</b>	GetDesignVariationKey()
<b>Python Example</b>	<code>oModule.GetDesignVariationKey()</code>

<b>VB Syntax</b>	GetDesignVariationKey()
<b>VB Example</b>	<pre>dim key set key = oModule.GetDesignVariationKey()</pre>

## GetImagDataValues

Returns array of imaginary data values.

**Important:**

This script does not function on its own, but as part of [GetSolutionDataPerVariation](#).

<b>UI Access</b>	N/A		
<b>Parameters</b>	Name	Type	Description
	<expressionString>	String	Can be returned from <a href="#">GetDataExpressions()</a>
<b>Return Value</b>	Array of doubles		

<b>Python Syntax</b>	GetImagDataValues(<expressionString>,<siValue>)
<b>Python Example</b>	imaginaryvalues = oModule.GetImagDataValues('expression',1)

<b>VB Syntax</b>	GetImagDataValues(<expressionString>,<siValue>)
<b>VB Example</b>	dim imaginaryvalues imaginaryvalues = oModule.GetImagDataValues("expression",True)

## GetPerQuantityPrimarySweepValues

Returns per quantity primary sweep values.

**Important:**

This script does not function on its own, but as part of [GetSolutionDataPerVariation](#).

<b>UI Access</b>	N/A		
<b>Parameters</b>	Name	Type	Description
	<expressionString>	String	Can be returned from <a href="#">GetDataExpressions()</a> .
<b>Return Value</b>	Array of doubles if <a href="#">IsPerQuantityPrimarySweep()</a> returned True; error if returned False		

<b>Python Syntax</b>	GetPerQuantitySweepValues(<expressionString>, <siValue>)
<b>Python Example</b>	<pre>sweepvalues = oModule.GetPerQuantitySweepValues('0.111,0.201,0.345,0.231', 1)</pre>

<b>VB Syntax</b>	GetPerQuantitySweepValues(<expressionString>, <siValue>)
<b>VB Example</b>	<pre>dim sweepvalues sweepvalues = oModule.GetPerQuantitySweepValues("0.111,0.201,0.345,0.231", True)</pre>

## GetRealDataValues

Returns array of real data values.

**Important:**

This script does not function on its own, but as part of [GetSolutionDataPerVariation](#).

<b>UI Access</b>	N/A		
<b>Parameters</b>	Name	Type	Description
	<expressionString>	String	Can be returned from <a href="#">GetDataExpressions()</a>
<b>Return Value</b>	Array of doubles		

<b>Python Syntax</b>	GetRealDataValues(<expressionString>,<siValue>)
<b>Python Example</b>	<pre>realvalues = oModule.GetRealDataValues('expression',1)</pre>

<b>VB Syntax</b>	GetRealDataValues(<expressionString>,<siValue>)
<b>VB Example</b>	<pre>dim realvalues realvalues = oModule.GetRealDataValues("expression",True)</pre>

## GetSweepNames

Returns array of text strings containing primary sweep name(s).

**Important:**

This script does not function on its own, but as part of [GetSolutionDataPerVariation](#).

<b>UI Access</b>	N/A
<b>Parameters</b>	N/A
<b>Return Value</b>	Array of text strings

<b>Python Syntax</b>	GetSweepNames()
<b>Python Example</b>	<pre>sweepnames = oModule.GetSweepNames()</pre>

<b>VB Syntax</b>	GetSweepNames()
<b>VB Example</b>	dim sweepnames sweepnames = oModule.GetSweepNames ()

## GetSweepUnits

Returns text string containing units.

### Important:

This script does not function on its own, but as part of [GetSolutionDataPerVariation](#).

<b>UI Access</b>	N/A		
<b>Parameters</b>	Name	Type	Description
	<sweepName>	String	Primary sweep name
<b>Return Value</b>	Text string containing units		

<b>Python Syntax</b>	GetSweepUnits(<sweepName>)
<b>Python Example</b>	sweepunits = oModule.GetSweepUnits ('Sweep 1')

<b>VB Syntax</b>	GetSweepUnits(<sweepName>)
------------------	----------------------------

<b>VB Example</b>	<pre>dim sweepunits sweepunits = oModule.GetSweepUnits("Sweep 1")</pre>
-------------------	-------------------------------------------------------------------------

## GetSweepValues

Returns sweep values.

### Important:

This script does not function on its own, but as part of [GetSolutionDataPerVariation](#).

<b>UI Access</b>	N/A									
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;sweepName&gt;</td> <td>String</td> <td>Primary sweep name</td> </tr> <tr> <td>&lt;siValue&gt;</td> <td>Boolean</td> <td>True to return SI-value; False to return by <a href="#">GetSweepUnits()</a>.</td> </tr> </tbody> </table>	Name	Type	Description	<sweepName>	String	Primary sweep name	<siValue>	Boolean	True to return SI-value; False to return by <a href="#">GetSweepUnits()</a> .
Name	Type	Description								
<sweepName>	String	Primary sweep name								
<siValue>	Boolean	True to return SI-value; False to return by <a href="#">GetSweepUnits()</a> .								
<b>Return Value</b>	Array of doubles									

<b>Python Syntax</b>	GetSweepValues(<sweepName>, <siValue>)
<b>Python Example</b>	sweepvalues = oModule.GetSweepValues('Sweep 1', True)

<b>VB Syntax</b>	GetSweepValues(<sweepName>, <siValue>)
<b>VB Example</b>	<pre>dim sweepvalues sweepvalues = oModule.GetSweepValues("Sweep 1", True)</pre>

## IsDataComplex

Returns whether an expression is complex.

**Important:**

This script does not function on its own, but as part of [GetSolutionDataPerVariation](#).

<b>UI Access</b>	NA						
<b>Parameters</b>	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>&lt;expressionString&gt;</td><td>String</td><td>Can be returned from <a href="#">GetDataExpressions()</a>.</td></tr></tbody></table>	Name	Type	Description	<expressionString>	String	Can be returned from <a href="#">GetDataExpressions()</a> .
Name	Type	Description					
<expressionString>	String	Can be returned from <a href="#">GetDataExpressions()</a> .					
<b>Return Value</b>	Boolean (True if expression is Complex data; False if not)						

<b>Python Syntax</b>	<code>IsDataComplex(&lt;expressionString&gt;)</code>
<b>Python Example</b>	<code>oModule.IsDataComplex('.001,.234,.455,.434')</code>

<b>VB Syntax</b>	<code>IsDataComplex(&lt;expressionString&gt;)</code>
<b>VB Example</b>	<code>oModule.IsDataComplex('.001,.234,.455,.434')</code>

## IsPerQuantityPrimarySweep

Returns whether data expressions have different primary sweep values.

**Important:**

This script does not function on its own, but as part of [GetSolutionDataPerVariation](#).

<b>UI Access</b>	N/A
<b>Parameters</b>	N/A
<b>Return Value</b>	Boolean (True if data expressions have different primary sweep values)

<b>Python Syntax</b>	<code>IsPerQuantityPrimarySweep()</code>
<b>Python Example</b>	<code>var = oModule.IsPerQuantityPrimarySweep()</code>

<b>VB Syntax</b>	<code>IsPerQuantityPrimarySweep()</code>
<b>VB Example</b>	<code>dim var var = oModule.IsPerQuantityPrimarySweep()</code>

## Release Data

Releases all cached data. After this function is called, all subsequent function calls to the object will fail.

**Important:**

This script does not function on its own, but as part of [GetSolutionDataPerVariation](#).

<b>UI Access</b>	N/A
<b>Parameters</b>	N/A
<b>Return Value</b>	N/A

<b>Python Syntax</b>	ReleaseData()
<b>Python Example</b>	<code>oModule.ReleaseData()</code>

<b>VB Syntax</b>	ReleaseData()
<b>VB Example</b>	<code>oModule.ReleaseData()</code>

## GroupPlotCurvesByGroupingStrategy

Groups curves in a Stacked Plot automatically based on a curve grouping strategy.

<b>UI Access</b>	N/A											
<b>Parameters</b>	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td><code>&lt;ReportName&gt;</code></td><td>String</td><td>Name of report.</td></tr><tr><td><code>&lt;GroupStrategy&gt;</code></td><td>String</td><td>Strategy for grouping, "Single", "By Trace" or "By Units".</td></tr></tbody></table>			Name	Type	Description	<code>&lt;ReportName&gt;</code>	String	Name of report.	<code>&lt;GroupStrategy&gt;</code>	String	Strategy for grouping, "Single", "By Trace" or "By Units".
Name	Type	Description										
<code>&lt;ReportName&gt;</code>	String	Name of report.										
<code>&lt;GroupStrategy&gt;</code>	String	Strategy for grouping, "Single", "By Trace" or "By Units".										
<b>Return Value</b>	None.											

<b>Python Syntax</b>	GroupPlotCurvesByGroupingStrategy(<ReportName>, <GroupStrategy>)
<b>Python Example</b>	<code>oModule.GroupPlotCurvesByGroupingStrategy("Transient Plot 1", "By Trace")</code>

<b>VB Syntax</b>	GroupPlotCurvesByGroupingStrategy <ReportName>, <GroupStrategy>
<b>VB Example</b>	<code>oModule.GroupPlotCurvesByGroupingStrategy "Transient Plot 1", "By Trace"</code>

## ImportIntoReport

Imports .tab, .csv, and .dat format files into a report.

<b>UI Access</b>	Right-click on report name in the Project tree and select <b>Import....</b>																		
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;ReportName&gt;</td> <td>String</td> <td>Name of the Report</td> </tr> <tr> <td>&lt;FileName&gt;</td> <td>String</td> <td>Path and File Name</td> </tr> <tr> <td></td> <td>.csv</td> <td>Comma-delimited data file</td> </tr> <tr> <td></td> <td>.tab</td> <td>Tab-separated file</td> </tr> <tr> <td></td> <td>.dat</td> <td>Ansys plot data file</td> </tr> </tbody> </table>	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
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																	
<b>Return Value</b>	None																		

<b>Python Syntax</b>	ImportIntoReport (<ReportName>, <FileName>)
<b>Python Example</b>	<code>oDesign.ImportIntoReport ("Plot1", "c:\report1.dat")</code>

<b>VB Syntax</b>	ImportIntoReport <ReportName>, <FileName>
<b>VB Example</b>	<code>oDesign.ImportIntoReport "Plot1", "c:\report1.dat"</code>

## ImportReportDataIntoReport

Imports report data from a file into a specified report.

<b>UI Access</b>	N/A									
<b>Parameters</b>	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>&lt;ReportName&gt;</td><td>String</td><td>Name of specified report.</td></tr><tr><td>&lt;FileName&gt;</td><td>String</td><td>Name of specified data file. File extenstion "rdat" is expected.</td></tr></tbody></table>	Name	Type	Description	<ReportName>	String	Name of specified report.	<FileName>	String	Name of specified data file. File extenstion "rdat" is expected.
Name	Type	Description								
<ReportName>	String	Name of specified report.								
<FileName>	String	Name of specified data file. File extenstion "rdat" is expected.								
<b>Return Value</b>	None.									

<b>Python Syntax</b>	ImportReportDataIntoReport(<ReportName>, <FileName>)
<b>Python Example</b>	<code>oModule.ImportReportDataIntoReport ("Plot 1", "C:/Plot1data.rdat")</code>

<b>VB Syntax</b>	ImportReportDataIntoReport <ReportName>, <FileName>
<b>VB Example</b>	<code>oModule.ImportReportDataIntoReport "Plot 1", "C:/Plot1data.rdat"</code>

## MovePlotCurvesToGroup

In a Stacked Plot move curve(s) from its stack(s) to an existing stack. Here term 'group' is synonymous to 'stack' in the context of cartesian stacked plot.

<b>UI Access</b>	N/A												
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;ReportName&gt;</td> <td>String</td> <td>Name of report.</td> </tr> <tr> <td>&lt;CurveArray&gt;</td> <td>Array</td> <td>Array of curve names to move.</td> </tr> <tr> <td>&lt;StackName&gt;</td> <td>String</td> <td>Name of stack to move to.</td> </tr> </tbody> </table>	Name	Type	Description	<ReportName>	String	Name of report.	<CurveArray>	Array	Array of curve names to move.	<StackName>	String	Name of stack to move to.
Name	Type	Description											
<ReportName>	String	Name of report.											
<CurveArray>	Array	Array of curve names to move.											
<StackName>	String	Name of stack to move to.											
<b>Return Value</b>	None.												

<b>Python Syntax</b>	MovePlotCurvesToGroup(<ReportName>, <CurveArray>, <StackName>)
<b>Python Example</b>	<pre>oModule.MovePlotCurvesToGroup ("XY Stacked Plot 1", ["R2.V : TR", "R2.I : TR"], "Stack 2")</pre>

<b>VB Syntax</b>	MovePlotCurvesToGroup <ReportName>, <CurveArray>, <StackName>
<b>VB Example</b>	<pre>oModule.MovePlotCurvesToGroup _     "XY Stacked Plot 1", _     Array("R2.V : TR", "R2.I : TR"), _     "Stack 2"</pre>

## MovePlotCurvesToNewGroup

Move curve(s) from its stack(s) to a new stack. Here term 'group' is synonymous to 'stack' in the context of Cartesian stacked plot.

<b>UI Access</b>	N/A
------------------	-----

<b>Parameters</b>	Name	Type	Description
	<ReportName>	String	Name of report.
	<CurveArray>	Array	Array of curve names to move.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	MovePlotCurvesToNewGroup (<ReportName>, <CurveArray>)
<b>Python Example</b>	<pre>oModule.MovePlotCurvesToNewGroup(     "XY Stacked Plot 1",     ["R2.V : TR", "R2.I : TR"])</pre>

<b>VB Syntax</b>	MovePlotCurvesToNewGroup <ReportName>, <CurveArray>
<b>VB Example</b>	<pre>oModule.MovePlotCurvesToNewGroup _     "XY Stacked Plot 1", _     Array("R2.V : TR", "R2.I : TR")</pre>

## OpenWindowForAllReports

Opens windows for all reports belong to current design.

<b>UI Access</b>	N/A
<b>Parameters</b>	None.

---

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

<b>Python Syntax</b>	OpenWindowForAllReports()
<b>Python Example</b>	<code>oModule.OpenWindowForAllReports ()</code>

<b>VB Syntax</b>	OpenWindowForAllReports
<b>VB Example</b>	<code>oModule.OpenWindowForAllReports</code>

## OpenWindowForReports

Opens windows for specified reports.

<b>UI Access</b>	N/A		
<b>Parameters</b>	Name <i>&lt;ReportNames&gt;</i>	Type Array	Description Array of strings containing report names.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	OpenWindowForReports( <i>&lt;ReportNames&gt;</i> )
<b>Python Example</b>	<code>oModule.OpenWindowForReports(["Report1", "Report2"])</code>

<b>VB Syntax</b>	OpenWindowForReports <i>&lt;ReportNames&gt;</i>
------------------	-------------------------------------------------

**VB Example**

```
oModule.OpenWindowForReports Array("Report1", "Report2")
```

## PastePlotSettings

Paste plot settings to a specified report.

<b>UI Access</b>	Right-click a report, select <b>Paste</b>		
<b>Parameters</b>	Name	Type	Description
	<ReportName>	String	Name of specified report.
<b>Return Value</b>	None.		

**Python Syntax**

```
PastePlotSettings(<ReportName>, <PropTypeToApply>)
```

**Python Example**

```
oModule.PastePlotSettings("Plot 1", "Graphical")
```

**VB Syntax**

```
PastePlotSettings <ReportName>, <PropTypeToApply>
```

**VB Example**

```
oModule.PastePlotSettings "Plot 1", "Graphical"
```

## PasteReports

Paste copied reports to results in the current project.

**UI Access**

```
Edit > Paste
```

<b>Parameters</b>	None.
<b>Return Value</b>	None.

<b>Python Syntax</b>	PasteReports ()
<b>Python Example</b>	<code>oModule.PasteReports ()</code>

<b>VB Syntax</b>	PasteReports
<b>VB Example</b>	<code>oModule.PasteReports</code>

## PasteReportsWithLegacyNames

Pastes copied reports to results in the current project with legacy name definitions.

<b>UI Access</b>	N/A
<b>Parameters</b>	None.
<b>Return Value</b>	None.

<b>Python Syntax</b>	PasteReportsWithLegacyNames ()
<b>Python Example</b>	<code>oModule.PasteReportsWithLegacyNames ()</code>

<b>VB Syntax</b>	PasteReportsWithLegacyNames
<b>VB Example</b>	<code>oModule.PasteReportsWithLegacyNames</code>

## PasteTraces

Pastes copied traces to a named plot.

<b>UI Access</b>	Paste						
<b>Parameters</b>	<table border="1"><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><code>&lt;ReportName&gt;</code></td><td>String</td><td>Name of plot</td></tr></table>	Name	Type	Description	<code>&lt;ReportName&gt;</code>	String	Name of plot
Name	Type	Description					
<code>&lt;ReportName&gt;</code>	String	Name of plot					
<b>Return Value</b>	None						

<b>Python Syntax</b>	PasteTraces ( <code>&lt;ReportName&gt;</code> )
<b>Python Example</b>	<code>oModule.PasteTraces ("XY Plot1")</code>

<b>VB Syntax</b>	PasteTraces <code>&lt;ReportName&gt;</code>
<b>VB Example</b>	<code>oModule.PasteTraces "XY Plot1"</code>

## PasteTracesWithLegacyNames

Pastes copied traces to a named plot using legacy name definitions.

<b>UI Access</b>	N/A
------------------	-----

<b>Parameters</b>	<table border="1"> <tr> <th>Name</th><th>Type</th><th>Description</th></tr> <tr> <td>&lt;ReportName&gt;</td><td>String</td><td>Name of plot</td></tr> </table>	Name	Type	Description	<ReportName>	String	Name of plot
Name	Type	Description					
<ReportName>	String	Name of plot					
<b>Return Value</b>	None						

<b>Python Syntax</b>	PasteTracesWithLegacyNames (<ReportName>)
<b>Python Example</b>	<code>oModule.PasteTracesWithLegacyNames ("XY Plot1")</code>

<b>VB Syntax</b>	PasteTracesWithLegacyNames <ReportName>
<b>VB Example</b>	<code>oModule.PasteTracesWithLegacyNames "XY Plot1"</code>

## RenameReport

Renames an existing report.

<b>UI Access</b>	Select a report on the Project tree, right-click and select <b>Rename</b>									
<b>Parameters</b>	<table border="1"> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> <tr> <td>&lt;OldReportName&gt;</td> <td>String</td> <td>Old Report Name</td> </tr> <tr> <td>&lt;NewReportName&gt;</td> <td>String</td> <td>New Report Name</td> </tr> </table>	Name	Type	Description	<OldReportName>	String	Old Report Name	<NewReportName>	String	New Report Name
Name	Type	Description								
<OldReportName>	String	Old Report Name								
<NewReportName>	String	New Report Name								
<b>Return Value</b>	None.									

<b>Python Syntax</b>	RenameReport (<OldReportName>, <NewReportName>)
----------------------	-------------------------------------------------

**Python Example**

```
oModule.RenameReport("XY Plot1", "Reflection")
```

**VB Syntax**

```
RenameReport <OldReportName>, <NewReportName>
```

**VB Example**

```
oModule.RenameReport "XY Plot1", "Reflection"
```

## RenameTrace

To rename a trace in a plot

<b>UI Access</b>	N/A												
<b>Parameters</b>	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>&lt;ReportName&gt;</td><td>String</td><td>Name of report.</td></tr><tr><td>&lt;TraceName&gt;</td><td>String</td><td>Name of Trace</td></tr><tr><td>&lt;NewName&gt;</td><td>String</td><td>New trace name.</td></tr></tbody></table>	Name	Type	Description	<ReportName>	String	Name of report.	<TraceName>	String	Name of Trace	<NewName>	String	New trace name.
Name	Type	Description											
<ReportName>	String	Name of report.											
<TraceName>	String	Name of Trace											
<NewName>	String	New trace name.											
<b>Return Value</b>	None.												

**Python Syntax**

```
RenameTrace(<ReportName>, <TraceName>, <NewName>)
```

**Python Example**

```
oModule.RenameTrace ("XY Plot1",
"dB(S(WavePort1,WavePort1))1", "Port1dBs")
```

<b>VB Syntax</b>	RenameTrace <ReportName>, <TraceName>, <NewName>
<b>VB Example</b>	<pre>oModule.RenameTrace "XY Plot1", _ "dB (S (WavePort1, WavePort1)) 1", _ "Port1dbS"</pre>

## ResetPlotSettings

Resets plot settings to defaults.

<b>UI Access</b>	Right-click on a plot, select <b>Edit &gt; Reset Plot Settings</b>						
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;PlotName&gt;</td> <td>String</td> <td>Name of specified plot.</td> </tr> </tbody> </table>	Name	Type	Description	<PlotName>	String	Name of specified plot.
Name	Type	Description					
<PlotName>	String	Name of specified plot.					
<b>Return Value</b>	None.						

<b>Python Syntax</b>	ResetPlotSettings(<PlotName>)
<b>Python Example</b>	oModule.ResetPlotSettings("Differential S-parameters")

<b>VB Syntax</b>	ResetPlotSettings <PlotName>
<b>VB Example</b>	oModule.ResetPlotSettings("Differential S-parameters")

## SavePlotSettingsAsDefault

Saves report plot settings as default.

UI Access	<b>Report Templates &gt; Save Settings as Default</b>		
Parameters	Name <i>&lt;PlotName&gt;</i>	Type String	Description Name of plot to use for plot defaults.
Return Value	None.		

Python Syntax	SavePlotSettingsAsDefault ("<PlotName>")
Python Example	oModule.SavePlotSettingsAsDefault ("XY Plot1")

VB Syntax	SavePlotSettingsAsDefault "<PlotName>"
VB Example	oModule.SavePlotSettingsAsDefault "XY Plot 1"

## SetLinkOutputTraces

Specifies dynamic link output traces from the current design.

UI Access	Right-click the <b>Results</b> , select <b>Link Output...</b>		
Parameters	Name <i>&lt;TraceArray&gt;</i>	Type Array	Description Array of traces to set.  Array("<ReportName>:=", <array of trace names>, "<ReportName>:=", <array of trace names>,...)
Return Value	None.		

<b>Python Syntax</b>	SetLinkOutputTraces(<TraceArray>)
<b>Python Example</b>	<pre> oModule.SetLinkOutputTraces ( [     "Plot 1:=", ["Trace1"],     "Plot 2:=", ["Trace1"] ]) </pre>

<b>VB Syntax</b>	SetLinkOutputTraces <TraceArray>
<b>VB Example</b>	<pre> oModule.SetLinkOutputTraces _ Array("Plot 1:=", Array("Trace1"), _ "Plot 2:=", Array("Trace1")) </pre>

## SetPropertyValue [Report Setup]

Sets the property value for report module child object.

<b>UI Access</b>	Select <b>Edit Properties</b> on Report objects.		
<b>Parameters</b>	Name	Type	Description
	<PropPath>	String	A child object's property path. See <a href="#">property path discussion here</a> .
<b>Return Value</b>	True if the property is found and the new value is valid. Otherwise return False.		

<b>Python Syntax</b>	SetPropValue(<PropPath>, <NewValue>)
<b>Python Example</b>	<pre>oRptModule.SetPropValue("S Parameter Plot 1/Display Type", "DataTable") oRptModule.SetPropValue("S Parameter Plot 1/db(S(port1,port2)/Primary sweep", "Freq")</pre>

<b>VB Syntax</b>	SetPropValue <PropPath>, <NewValue>
<b>VB Example</b>	<pre>oRptModule.SetPropValue "S Parameter Plot 1/Display Type", "DataTable" oRptModule.SetPropValue "S Parameter Plot 1/db(S(port1,port2)/Primary sweep", "Freq"</pre>

## UnGroupPlotCurvesInGroup

From a Stacked Plot, ungroups curves in a stack.

<b>UI Access</b>	N/A									
<b>Parameters</b>	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>&lt;ReportName&gt;</td><td>String</td><td>Name of report.</td></tr><tr><td>&lt;GroupName&gt;</td><td>String</td><td>Stack group name.</td></tr></tbody></table>	Name	Type	Description	<ReportName>	String	Name of report.	<GroupName>	String	Stack group name.
Name	Type	Description								
<ReportName>	String	Name of report.								
<GroupName>	String	Stack group name.								
<b>Return Value</b>	None.									

<b>Python Syntax</b>	UnGroupPlotCurvesInGroup(<ReportName>, <GroupName>)
<b>Python Example</b>	oModule.UnGroupPlotCurvesInGroup("S Parameter Plot 3", "Stack 1")

<b>VB Syntax</b>	UnGroupPlotCurvesInGroup <ReportName>, <GroupName>
<b>VB Example</b>	oModule.UnGroupPlotCurvesInGroup "S Parameter Plot 3", "Stack 1"

## UpdateAllReports

Updates all reports in the **Results** branch in the project tree.

<b>UI Access</b>	Right-click on <b>Results</b> in the project tree, select <b>Update All Reports</b>
<b>Parameters</b>	None
<b>Return Value</b>	None

<b>Python Syntax</b>	UpdateAllReports()
<b>Python Example</b>	oModule.UpdateAllReports ()

<b>VB Syntax</b>	UpdateAllReports
<b>VB Example</b>	oModule.UpdateAllReports

## UpdateReports

Updates specified reports.

UI Access	N/A								
Parameters	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>&lt;ReportNames&gt;</td><td>Array</td><td>Array of strings containing report names.</td></tr></tbody></table>			Name	Type	Description	<ReportNames>	Array	Array of strings containing report names.
Name	Type	Description							
<ReportNames>	Array	Array of strings containing report names.							
Return Value	None.								

Python Syntax	UpdateReports(<ReportNames>)
Python Example	oModule.UpdateReports ([ "XY Plot 1", "XY Plot 4" ])

VB Syntax	UpdateReports <ReportNames>
VB Example	oModule.UpdateReports Array("XY Plot 1", "XY Plot 4")

## 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 <b>Apply Traces</b> button																	
Parameters	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>&lt;ReportName&gt;</td><td>String</td><td>Name of Report.</td></tr><tr><td>&lt;TraceNames&gt;</td><td>Array</td><td>Array of strings containing trace names.</td></tr><tr><td>&lt;SolutionName&gt;</td><td>String</td><td>Name of the solution.</td></tr><tr><td>&lt;ContextArray&gt;</td><td>Array</td><td>Context for which the expression is being evaluated. This can be an empty string if there is no context.</td></tr></tbody></table>			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.
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:=", &lt;DomainType&gt;       &lt;DomainType&gt; ex. "Sweep" or "Time"       Array("Context:=", &lt;GeometryType&gt;             &lt;GeometryType&gt; ex. "Infinite Spheren", "Spheren",             "Polylinen")</pre>
<FamiliesArray>	Array	<p>Contains sweep definitions for the report.</p> <pre>Array("&lt;VariableName&gt;:= ", &lt;ValueArray&gt;       &lt;ValueArray&gt;       Array("All") or Array("Value1", "Value2", ..."ValueN")       examples of &lt;VariableName&gt; "Freq", "Theta", "Distance")</pre>
<ReportdataArray>	Array	<p>This array contains the report quantity and X, Y, and (Z) axis definitions.</p> <pre>Array("X Component:=", &lt;VariableName&gt;,       "Y Component:=", &lt;VariableName&gt;   &lt;ReportQuantityArray&gt;       &lt;ReportQuantityArray&gt; ex. Array("dB(S(Port1, Port1))")</pre>
<ExtTraceInfo>	Array	Optional. Array defines extended trace information.
<b>Return Value</b>	None.	

<b>Python Syntax</b>	UpdateTraces(<ReportName>, <SolutionName>, <ContextArray>, <FamiliesArray>, <ReportdataArray>)
<b>Python Example</b>	<pre>oModule.UpdateTraces ("XY Plot 1", ["NEG1.VAL"], "TR4", [</pre>

```
"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)"]
], [])
```

<b>VB Syntax</b>	UpdateTraces <ReportName>, <SolutionName>, <ContextArray>, <FamiliesArray>, <ReportdataArray>
<b>VB Example</b>	<pre>oModule.UpdateTraces "XY Plot1", Array("dB(S(WavePort1,WavePort1))"), "Setup1 : Sweep1", Array("Domain:=", "Time", "HoldTime:=", 1, "RiseTime:=", 0, "StepTime:=", 0, "Step:=", false,</pre>

```
"WindowWidth:=", 1, "WindowType:=", 0, "KaiserParameter:=", 1, _
"MaximumTime:=", 0), Array("Time:=", Array("All")), _
Array("X Component:=", "Time", _
"Y Component:=", Array("dB(S(WavePort1,WavePort1))")), _
Array()
```

## UpdateTracesContextAndSweeps

Edits sweeps and context of multiple traces without affecting their component expressions.

<b>UI Access</b>	<b>Modify Report</b> with multiple traces selected.		
<b>Parameters</b>	Name	Type	Description
	<ReportName>	String	Name of Report.
	<TraceNames>	Array	Array of strings containing trace names.
	<SolutionName>	String	Name of the solution as listed in the <b>Modify Report</b> dialog box.  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"
	<PointSet>	Array	Point set for the selected traces, for example, X and Y values for the plot.
<b>Return Value</b>	None		

<b>Python Syntax</b>	UpdateTracesContextAndSweeps(<ReportName>, <TraceNames>, <SolutionName>, <ContextArray>, <PointSet>)
<b>Python Example</b>	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"]])
```

<b>VB Syntax</b>	UpdateTracesContextAndSweeps <ReportName>, <TraceNames>, <SolutionName>, <ContextArray>, <PointSet>
<b>VB Example</b>	<pre>oModule.UpdateTracesContextAndSweeps _ "Active S Parameter Quick Report", _ Array("dB(ActiveS(Port1:1))", "dB(ActiveS(Port2:1))"), _ "Setup1 : Sweep1", Array(), _ Array("Freq:=", _</pre>

```
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"))
```

This page intentionally  
left blank.

# 13 - Boundary and Excitation Module Script Commands

Boundary and excitation commands should be executed by the "BoundarySetup" module.

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

## Conventions Used in this Chapter

<BoundName>

Type: string.

Name of a boundary.

<AssignmentObjects>

**Type: Array of strings.**

An array of object names.

<AssignmentFaces>

Type: Array of integers.

An array of face IDs. The ID of a face can be determined through the user interface using the **3D Modeler> Measure> Area** command. The face ID is given in the **Measure Information** dialog box.

<LineEndPoint>

Array(<double>, <double>, <double>)

**The topics for this section include:**

[General Commands Recognized by the Boundary/Excitations Module](#)

[Script Commands for Creating and Modifying Boundaries](#)

[Script Commands for Creating and Modifying PMLs](#)

## Script Commands for Creating and Modifying PMLs

Following are script commands for creating and modifying PMLs that are recognized by the **BoundarySetup** module.

The **PML Setup** wizard allows you to set up one or more PMLs in the model. There is not a single 'Create PML' or 'Edit PML' command that represents the work performed by the **PML Setup** wizard. Instead, a series of geometry and material commands are executed. As a result, when a script is being recorded, a series of geometry and material creation commands is what is actually recorded in the script for a PML setup. This is followed by a script command stating that PMLs have been set up or modified.

[CreatePML](#)

[ModifyPMLGroup](#)

[PMLGroupCreated](#)

[PMLGroupModified](#)

[RecalculatePMLMaterial](#)

### CreatePML

Set up a perfectly matched layer (PML) boundary.

UI Access	HFSS > Boundaries > PML Setup Wizard...		
Parameters	Name <code>&lt;PMLArray&gt;</code>	Type Array	Description Structured array.  For manually created PMLs:  <code>Array ("NAME:PMLCreationSettings", "UserDrawnGroup:=", true, "PMLObj:=", &lt;string, name of the object to use as the PML cover.&gt;,</code>

		<pre> "BaseObj:=", &lt;string, name of the base object touching the PML cover object&gt;, "Thickness:=", &lt;value&gt;, "Orientation:=", &lt;string, orientation of the PML. Possi- ble values are: "XAxis", "YAxis", and "ZAxis"&gt;, "RadDist:=", &lt;value&gt;, "UseFreq:=", &lt;boolean, If true, provide the MinFreq parameter. If false, provide the MinBeta parameter.&gt;, "MinFreq:=", &lt;value&gt;, "MinBeta:=", &lt;double&gt;)  For automatically created PMLs:  Array("NAME:PMLCreationSettings",       "UserDrawnGroup:=", false,       "PMLFaces:=", &lt;AssignmentFaces&gt;,       "CreateJoiningObjs:=", &lt;bool&gt;,       "Thickness:=", &lt;value&gt;,       "RadDist:=", &lt;value&gt;,       "UseFreq:=", &lt;boolean, If true, provide the MinFreq parameter. If false, provide the MinBeta parameter.&gt;,       "MinFreq:=", &lt;value&gt;,       "MinBeta:=", &lt;double&gt;) </pre>
<b>Return Value</b>	None.	

<b>Python Syntax</b>	CreatePML(<PMLArray>)
<b>Python Example</b>	<pre>oModule.CreatePML(     ["NAME:PMLCreationSettings",      "UserDrawnGroup:=", True,      "PMLFaces:=", [120],      "Thickness:=", "1mm",      "CreateJoiningObjs:=", False,      "PMLObj:=", 6,      "BaseObj:=", 15,      "Orientation:=", "ZAxis",      "UseFreq:=", True,      "MinFreq:=", "1GHz",      "MinBeta:=", 20,      "RadDist:=", "0.3mm"    ])</pre>

<b>VB Syntax</b>	CreatePML <PMLArray>
<b>VB Example</b>	<pre>oModule.CreatePML Array("NAME:PMLCreationSettings", _     "UserDrawnGroup:=", false, _</pre>

```

"PMLFaces:=", Array(120), "CreateJoiningObjs:=", true,_
"Thickness:=", "0.33mm", "RadDist:=", "1.6mm",_
"UseFreq:=", true, "MinFreq:=", "1GHz")

oModule.CreatePML Array("NAME:PMLCreationSettings", _
"UserDrawnGroup:=", true, _
"PMLObj:=", "Box1", "BaseObj:=", "Box2", _
"Thickness:=", "0.3mm", "Orientation:=", "ZAxis", _
"RadDist:=", "1.6mm", "UseFreq:=", false, _
"MinBeta:=", "2")

```

## ModifyPMLGroup

Modifies a PML group.

UI Access	N/A								
Parameters	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;PMLGroupArray&gt;</td> <td>Array</td> <td> <p>Structured array.</p> <pre> Array ("NAME:&lt;GroupName&gt;", "RadDist:=", &lt;value&gt;, "UseFreq:=", &lt;boolean&gt;, "MinFreq:=", &lt;value&gt;, "MinBeta:=", &lt;double&gt;) UseFreq </pre> </td></tr> </tbody> </table>	Name	Type	Description	<PMLGroupArray>	Array	<p>Structured array.</p> <pre> Array ("NAME:&lt;GroupName&gt;", "RadDist:=", &lt;value&gt;, "UseFreq:=", &lt;boolean&gt;, "MinFreq:=", &lt;value&gt;, "MinBeta:=", &lt;double&gt;) UseFreq </pre>		
Name	Type	Description							
<PMLGroupArray>	Array	<p>Structured array.</p> <pre> Array ("NAME:&lt;GroupName&gt;", "RadDist:=", &lt;value&gt;, "UseFreq:=", &lt;boolean&gt;, "MinFreq:=", &lt;value&gt;, "MinBeta:=", &lt;double&gt;) UseFreq </pre>							

			If true, provide the MinFreq argument. If false, provide the MinBeta argument.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	ModifyPMLGroup(<PMLGroupArray>)
<b>Python Example</b>	<pre>oModule.ModifyPMLGroup(     ["NAME:PMLGroup1",      "RadDist:=", "1.16666667mm",      "UseFreq:=", False,      "MinBeta:=", 2] )</pre>

<b>VB Syntax</b>	ModifyPMLGroup <PMLGroupArray>
<b>VB Example</b>	<pre>oModule.ModifyPMLGroup Array("NAME:PMLGroup1", _     "RadDist:=", "1.16666667mm", _     "UseFreq:=", false, "MinBeta:=", 2)</pre>

## PMLGroupCreated

Command added by HFSS after a PML has been created. It is not responsible for creating the PML objects and materials. It just contains the information needed by the **PML Setup** wizard for future modification of the PML. This script command is not intended to be modified by you. Removing this command from the script will prevent future modification of the PML through the user interface after the script is played back.

<b>UI Access</b>	N/A								
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;PMLArray&gt;</td> <td>Array</td> <td>Array represents the created PML group.</td> </tr> </tbody> </table>			Name	Type	Description	<PMLArray>	Array	Array represents the created PML group.
Name	Type	Description							
<PMLArray>	Array	Array represents the created PML group.							
<b>Return Value</b>	None.								

<b>Python Syntax</b>	PMLGroupCreated(<PMLArray>)
<b>Python Example</b>	<pre> oModule.PMLGroupCreated(     ["NAME:PMLCreationSettings",      "UserDrawnGroup:=", True,      "PMLFaces:=", [120],      "Thickness:=", "1mm",      "CreateJoiningObjs:=", False,      "PMLObj:=", 6,      "BaseObj:=", 15,      "Orientation:=", "ZAxis",      "UseFreq:=", True, </pre>

```

    "MinFreq:=", "1GHz",
    "MinBeta:=", 20,
    "RadDist:=", "0.3mm"
]
)

```

<b>VB Syntax</b>	PMLGroupCreated <PMLArray>
<b>VB Example</b>	<pre> oModule.PMLGroupCreated Array("NAME:PMLCreationSettings", _ "UserDrawnGroup:=", false, _ "PMLFaces:=", Array(120), "CreateJoiningObjs:=", true, _ "Thickness:=", "0.33mm", "RadDist:=", "1.6mm", _ "UseFreq:=", true, "MinFreq:=", "1GHz") </pre>

## PMLGroupModified

Modifies a defined PML Group.

<b>UI Access</b>	Click <b>Update</b> in the <b>PML Setup</b> wizard.		
<b>Parameters</b>	Name <PMLGroupArray>	Type Array	Description Structured array.  Array ("NAME:<GroupName>", "RadDist:=", <value>, "UseFreq:=", <boolean>,

			<pre>"MinFreq:=", &lt;value&gt;, "MinBeta:=", &lt;double&gt;)  UseFreq</pre> <p>If true, provide the MinFreq argument.</p> <p>If false, provide the MinBeta argument.</p>
<b>Return Value</b>	None.		

<b>Python Syntax</b>	PMLGroupModified(<PMLGroupArray>)
<b>Python Example</b>	<pre>oModule.PMLGroupModified(     ["NAME:PMLGroup1",      "RadDist:=", "2.008333333333in",      "UseFreq:=", True,      "MinFreq:=", "2GHz",      ["NAME:ProjectMaterials"],      ["NAME:InternalPrivateMaterials"]     ] )</pre>

<b>VB Syntax</b>	PMLGroupModified <PMLGroupArray>
<b>VB Example</b>	<pre>oModule.PMLGroupModified Array("NAME:PMLGroup1", _</pre>

```
"RadDist:=", "2.008333333333in", _  
"UseFreq:=", true, "MinFreq:=", "2GHz", _  
Array ("NAME:ProjectMaterials"), _  
Array ("NAME:InternalPrivateMaterials"))
```

## RecalculatePMLMaterials

Updates the PML materials to match the current state of the **PML Setup** wizard data.

<b>UI Access</b>	Click <b>Recalculate Materials</b> in the <b>PML Setup</b> wizard.
<b>Parameters</b>	None.
<b>Return Value</b>	None.

<b>Python Syntax</b>	RecalculatePMLMaterials()
<b>Python Example</b>	oModule.RecalculatePMLMaterials()

<b>VB Syntax</b>	RecalculatePMLMaterials
<b>VB Example</b>	oModule.RecalculatePMLMaterials

# 14 - 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 <SetupName>	Type String	Description Name of the setup.
Return Value	None.		

<b>Python Syntax</b>	CopySetup (<SetupName>)
<b>Python Example</b>	<code>oModule.CopySetup ("OptimizationSetup1")</code>

<b>VB Syntax</b>	CopySetup <SetupName>
<b>VB Example</b>	<code>oModule.CopySetup "OptimizationSetup1"</code>

## DeleteSetups [Optimetrics]

Deletes the specified Optimetrics setups.

<b>UI Access</b>	Right-click the setup in the project tree, and then click <b>Delete</b> on the shortcut menu						
<b>Parameters</b>	<table border="1"><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;NameArray&gt;</td><td>Array of Strings</td><td>An Array of Setup Names</td></tr></table>	Name	Type	Description	<NameArray>	Array of Strings	An Array of Setup Names
Name	Type	Description					
<NameArray>	Array of Strings	An Array of Setup Names					
<b>Return Value</b>	None						

<b>Python Syntax</b>	DeleteSetups (<NameArray>)
<b>Python Example</b>	<code>oModule.DeleteSetups ( ["OptimizationSetup1"] )</code>

<b>VB Syntax</b>	DeleteSetups <NameArray>
<b>VB Example</b>	<code>oModule.DeleteSetups Array("OptimizationSetup1")</code>

## DistributedAnalyzeSetup

Distributes all variable value instances within a parametric sweep to different machines already specified from within the user interface

<b>UI Access</b>	Right-click the parametric setup name in the project tree and select Distribute Analysis.								
<b>Parameters</b>	<table border="1"> <tr> <td>Name</td> <td>Type</td> <td>Description</td> </tr> <tr> <td>&lt;ParametricSetupName&gt;</td> <td>String</td> <td>Name of the Setup</td> </tr> </table>			Name	Type	Description	<ParametricSetupName>	String	Name of the Setup
Name	Type	Description							
<ParametricSetupName>	String	Name of the Setup							
<b>Return Value</b>	None								

<b>Python Syntax</b>	DistributedAnalyzeSetup (<ParametricSetupName>)
<b>Python Example</b>	<code>oModule.DistributedAnalyzeSetup ("ParametricSetup1")</code>

<b>VB Syntax</b>	DistributedAnalyzeSetup <ParametricSetupName>
<b>VB Example</b>	<code>oModule.DistributedAnalyzeSetup "ParametricSetup1"</code>

## EditSetup

Modifies an existing solution setup.

<b>UI Access</b>	Double-click a solution setup in the project tree to modify its settings.											
<b>Parameters</b>	<table border="1"> <tr> <td>Name</td> <td>Type</td> <td>Description</td> </tr> <tr> <td>&lt;SetupName&gt;</td> <td>String</td> <td>Name of the solve setup being edited.</td> </tr> <tr> <td>&lt;Attributes&gt;</td> <td>Array</td> <td>Structured array.</td> </tr> </table>			Name	Type	Description	<SetupName>	String	Name of the solve setup being edited.	<Attributes>	Array	Structured array.
Name	Type	Description										
<SetupName>	String	Name of the solve setup being edited.										
<Attributes>	Array	Structured array.										

		Array ("NAME:<NewSetupName>", <NamedParameters> See the InsertSetup command for details and examples.)
<b>Return Value</b>	None.	

<b>Python Syntax</b>	EditSetup (<SetupName>, <Attributes>)
<b>Python Example</b>	<pre>oModule.EditSetup("Setup1", ["NAME:NewSetup", "AdaptiveFreq:=", "1GHz", "EnableDistribProbTypeOption:=", false, "SaveFields:=", "true", "Enabled:=", true, ["NAME:Cap", "MaxPass:=", 10, "MinPass:=", 1, "MinConvPass:=", 2, "PerError:=", 1, "PerRefine:=", 30, "AutoIncreaseSolutionOrder:=", false, "SolutionOrder:=", "Normal"],</pre>

```
[ "NAME:DC",
    "Residual:=", 1E-005,
    "SolveResOnly:=", false,
    [ "NAME:Cond",
        "MaxPass:=", 10,
        "MinPass:=", 1,
        "MinConvPass:=", 1,
        "PerError:=", 1,
        "PerRefine:=", 30),
    [ "NAME:Mult",
        "MaxPass:=", 1,
        "MinPass:=", 1,
        "MinConvPass:=", 1,
        "PerError:=", 1,
        "PerRefine:=", 30]],
[ "NAME:AC",
    "MaxPass:=", 10,
    "MinPass:=", 1,
    "MinConvPass:=", 2,
    "PerError:=", 1,
    "PerRefine:=", 30]]
```

```
)  
  
oModule.InsertSetup("HfssDrivenAuto",  
    [ "NAME:Setup1",  
        "IsEnabled:=", True,  
        "AutoSolverSetting:=", "Balanced",  
        [ "NAME:Sweeps",  
            [ "NAME:Sweep",  
                "RangeType:=", "LinearStep",  
                "RangeStart:=", "1GHz",  
                "RangeEnd:=", "10GHz",  
                "RangeStep:=", "1GHz"  
            ]  
        ],  
        "SaveRadFieldsOnly:=", False,  
        "SaveAnyFields:=", True,  
        "Type:=", "Discrete"  
    ])  
  
oModule.InsertSetup("HfssDrivenAuto",  
    [  
])
```

```
        "NAME:Setup2",
        "SolveType:="           , "Auto",
        "IsEnabled:="          , True,
        [
            "NAME:MeshLink",
            "ImportMesh:="       , False
        ],
        "AutoSolverSetting:="   , "Balanced",
        [
            "NAME:Sweeps",
            [
                "NAME:Sweep",
                "RangeType:="         , "LinearCount",
                "RangeStart:="        , "0GHz",
                "RangeEnd:="          , "10GHz",
                "RangeCount:="        , 501
            ]
        ],
        "SaveRadFieldsOnly:="   , False,
        "SaveAnyFields:="       , True,
        "InfiniteSphereSetup:=" , "Infinite Sphere1",
```

```
    "ListsForFields:=", ["Objectlist2"],
    "Type:=", "Interpolating"
  ] )

oModule.InsertSetup("HfssDriven",
  ["NAME:Setup3",
   "AdaptMultipleFreqs:=", False,
   "Frequency:=", "5GHz",
   "MaxDeltaS:=", 0.02,
   "PortsOnly:=", False,
   "UseMatrixConv:=", False,
   "MaximumPasses:=", 6,
   "MinimumPasses:=", 1,
   "MinimumConvergedPasses:=", 1,
   "PercentRefinement:=", 30,
   "IsEnabled:=", True,
   "BasisOrder:=", 1,
   "DoLambdaRefine:=", True,
   "DoMaterialLambda:=", True,
   "SetLambdaTarget:=", False,
```

```
"Target:=", 0.3333,  
"UseMaxTetIncrease:=", False,  
"PortAccuracy:=", 2,  
"UseABCOnPort:=", False,  
"SetPortMinMaxTri:=", False,  
"UseDomains:=", True,  
"UseIterativeSolver:=", False,  
"IterativeResidual:=", 1E-06,  
"DDMSolverResidual:=", 0.0001,  
"EnhancedLowFreqAccuracy:=", True,  
"SaveRadFieldsOnly:=", False,  
"SaveAnyFields:=", True,  
"IESolverType:=", "Auto",  
"LambdaTargetForIESolver:=", 0.15,  
"UseDefaultLambdaTgtForIESolver:=", True,  
"SkipPIERegionSolveDuringAdaptivePasses:=", True  
"RayDensityPerWavelength:=", 4,  
"MaxNumberOfBounces:=" , 5,  
"InfiniteSphereSetup:=" , "Infinite Sphere1",  
"SkipSBRSSolveDuringAdaptivePasses:=", True,  
"PTDUTDSimulationSettings:=", "PTD Correction + UTD Rays",
```

```
        "PTDEdgeDensity:="          , 20
    ])
### Edit an SBR+ Setup with Fast Frequency Looping
oModule.EditSetup("HfssDriven",
[
    "NAME:Setup1",
    "IsEnabled:="              , True,
    [
        "NAME:MeshLink",
        "ImportMesh:="           , False
    ],
    "IsSbrRangeDoppler:="     , False,
    "RayDensityPerWavelength:=", 4,
    "MaxNumberOfBounces:="     , 5,
    "IsMonostaticRCS:="       , True,
    "EnableCW Rays:="         , False,
    "RadiationSetup:="        , "",
    "PTDUTDSimulationSettings:=", "None",
    "FastFrequencyLooping:="   , True,
    [

```

```
"NAME:Sweeps",
[
    "NAME:Sweep",
    "RangeType:=" , "LinearStep",
    "RangeStart:=" , "1GHz",
    "RangeEnd:=" , "10GHz",
    "RangeStep:=" , "1GHz"
],
"ComputeFarFields:=" , True
"UseSBREnhancedRadiatedPowerCalculation:=", True,
"IsGOBlockageEnabled:=" , False,
"GOBlockageSurfaceSelfBlock:=" , False
])

##### Edit and RF Discharge Setup for HFSS
import ScriptEnv
ScriptEnv.Initialize("Ansoft.ElectronicsDesktop")
oDesktop.RestoreWindow()
oProject = oDesktop.SetActiveProject("coaxbend_discharge_r212")
oDesign = oProject.SetActiveDesign("HFSSDesign60degBendTeflon")
oModule = oDesign.GetModule("AnalysisSetup")
```

```
oModule.EditSetup("RFDischarge1",
[
    "NAME:RFDischarge1",
    "Enabled:=" , True,
    [
        "NAME:MeshLink",
        "ImportMesh:=" , True,
        "Project:=" , "This Project*",
        "Product:=" , "HFSS",
        "Design:=" , "This Design*",
        "Soln:=" , "Setup1 : Sweep",
        [
            "NAME:Params",
            "bend_angle:=" , "bend_angle"
        ],
        "ForceSourceToSolve:=" , True,
        "PreservePartnerSoln:=" , False,
        "PathRelativeTo:=" , "SourceProduct",
        "ApplyMeshOp:=" , True
    ],
]
```

```
[  
    "NAME:Excitations",  
    [  
        "NAME:1:1",  
        "Magnitude:=" , "1",  
        "Phase:=" , "0deg"  
    ],  
    [  
        "NAME:2:1",  
        "Magnitude:=" , "0",  
        "Phase:=" , "0deg"  
    ],  
    [  
        "NAME:Frequencies",  
        "10GHz"  
    ],  
    [  
        "Minimum Power:=" , "0.01",  
        "Maximum Power:=" , "1000000",  
        "Minimum Pressure:=" , "100pascal",  
        "Maximum Pressure:=" , "101325pascal",  
    ]]
```

```
"Postproc Sampling:=" , 500,  
"Temperature:=" , "0cel",  
"BuiltInGas:=" , "Helium"  
])
```

VB Syntax	EditSetup <SetupName>, <Attributes>
<b>VB Example</b>	<pre>oModule.EditSetup "Setup1", Array("NAME:NewSetup", "AdaptiveFreq:=", "1GHz", "EnableDistribProbTypeOption:=", false, "SaveFields:=", "true", "Enabled:=", true, Array("NAME:Cap", "MaxPass:=", 10, "MinPass:=", 1, "MinConvPass:=", 2, "PerError:=", 1, "PerRefine:=", 30, "AutoIncreaseSolutionOrder:=", false,</pre>

```
    "SolutionOrder:=", "Normal"),
Array("NAME:DC",
      "Residual:=", 1E-005,
      "SolveResOnly:=", false,
      Array("NAME:Cond",
            "MaxPass:=", 10,
            "MinPass:=", 1,
            "MinConvPass:=", 1,
            "PerError:=", 1,
            "PerRefine:=", 30),
      Array("NAME:Mult",
            "MaxPass:=", 1,
            "MinPass:=", 1,
            "MinConvPass:=", 1,
            "PerError:=", 1,
            "PerRefine:=", 30)),
Array("NAME:AC",
      "MaxPass:=", 10,
      "MinPass:=", 1,
      "MinConvPass:=", 2,
      "PerError:=", 1,
```

	"PerRefine:=", 30 )
--	---------------------

## EnableSetup

Enables and disables a defined optimetrics analysis setup.

<b>UI Access</b>	Right-click on a setup in the project tree, select <b>Enable Setup</b> or <b>Disable Setup</b>		
<b>Parameters</b>	Name <i>&lt;SetupName&gt;</i>	Type String	Description Name of specified setup.
	<i>&lt;Enable&gt;</i>	Boolean	Determines whether enable or disable a setup. <ul style="list-style-type: none"><li>• <b>True</b> - enable setup.</li><li>• <b>False</b> - disable setup.</li></ul>
<b>Return Value</b>	None.		

<b>Python Syntax</b>	EnableSetup(<SetupName>, <Enable>)
<b>Python Example</b>	oModule.EnableSetup ("OptimizationSetup1", True)

<b>VB Syntax</b>	EnableSetup <SetupName>, <Enable>
<b>VB Example</b>	oModule.EnableSetup "OptimizationSetup1", true

## ExportDXConfigFile

Create an xml file with the setup information for Design Xplorer

<b>UI Access</b>	Right click on the Design Xplorer setup in the project tree and choose <b>Export External Connector Addin Configuration...</b>									
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;SetupName&gt;</td> <td>String</td> <td>Must be one of existing DesignExplorer setup names</td> </tr> <tr> <td>&lt;FileName&gt;</td> <td>String</td> <td>Must be a valid file path and name</td> </tr> </tbody> </table>	Name	Type	Description	<SetupName>	String	Must be one of existing DesignExplorer setup names	<FileName>	String	Must be a valid file path and name
Name	Type	Description								
<SetupName>	String	Must be one of existing DesignExplorer setup names								
<FileName>	String	Must be a valid file path and name								
<b>Return Value</b>	None									

<b>Python Syntax</b>	ExportDXConfigFile (<SetupName>, <FileName>)
<b>Python Example</b>	<pre>oModule.ExportDXConfigFile ("DesignXplorerSetup1",     "c:/exportdir/DXSetup1.xml")</pre>

<b>VB Syntax</b>	ExportDXConfigFile <SetupName>, <FileName>
<b>VB Example</b>	<pre>oModule.ExportDXConfigFile ("DesignXplorerSetup1",     "c:/exportdir/DXSetup1.xml")</pre>

## ExportOptimetricsProfile

Export Optimetrics profile data

<b>UI Access</b>	Right click on the Optimetrics setup in the project tree and choose <b>View Analysis Result...</b> On the <b>Post Analysis Display</b> dialog box, click the <b>Profile</b> tab and click on the <b>Export</b> button.						
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;SetupName&gt;</td> <td>String</td> <td>Must be one of the existing Parametric, Optimization, Sensitivity, Statistical or</td> </tr> </tbody> </table>	Name	Type	Description	<SetupName>	String	Must be one of the existing Parametric, Optimization, Sensitivity, Statistical or
Name	Type	Description					
<SetupName>	String	Must be one of the existing Parametric, Optimization, Sensitivity, Statistical or					

		DesignXplorer setup names
<FileName>	String	Must be a valid file path and name with extension of csv, tab, dat, or txt
[profileNum]	String	Must be a numeric string. Optional: defaulted to last profile number. It should be a zero indexed profile number.
<b>Return Value</b>	None	

<b>Python Syntax</b>	ExportOptimetricsProfile (<SetupName>, <FileName>, [profileNum])
<b>Python Example</b>	<pre>oModule.ExportOptimetricsProfile ("StatisticalSetup1", "c:/exportdir/test.csv")</pre>

<b>VB Syntax</b>	ExportOptimetricsProfile <SetupName>, <FileName>, [profileNum]
<b>VB Example</b>	<pre>oModule.ExportOptimetricsProfile "StatisticalSetup1", "c:/exportdir/test.csv"</pre>

## ExportOptimetricsResult

Export an existing Optimization, Sensitivity, Statistical or DesignXplorer result. (Does not export Parametric results.)

<b>UI Access</b>	Right click on the desired Optimetrics setup in the project tree and choose <b>View Analysis Result...</b> On the <b>Post Analysis Display</b> dialog box, click the <b>Result</b> tab, then select <b>Table</b> view, and click on the <b>Export</b> button						
<b>Parameters</b>	<table border="1"><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;SetupName&gt;</td><td>String</td><td>Must be one of the existing Optimization, Sensitivity, Statistical, or</td></tr></table>	Name	Type	Description	<SetupName>	String	Must be one of the existing Optimization, Sensitivity, Statistical, or
Name	Type	Description					
<SetupName>	String	Must be one of the existing Optimization, Sensitivity, Statistical, or					

		DesignXplorer setup names
<FileName>	String	Must be a valid file path and name with extension of csv, tab, dat, or txt..
[useFullOutputName]	Boolean	Optional: defaulted to false. If set to true values will be printed with units. This parameter is ignored for Optimization and Statistical results.
<b>Return Value</b>	None	

<b>Python Syntax</b>	ExportOptimetricsResult (<SetupName>, <FileName>, [useFullOutputName])
<b>Python Example</b>	<pre>oModule.ExportOptimetricsResult (     "StatisticalSetup1", "c:/exportdir/test.csv", false)</pre>

<b>VB Syntax</b>	ExportOptimetricsResult <SetupName>, <FileName>, [useFullOutputName]
<b>VB Example</b>	<pre>oModule.ExportOptimetricsResult "StatisticalSetup1", "c:/exportdir/test.csv", false</pre>

## ExportParametricResults

Export existing Parametric results.

<b>UI Access</b>	Right click on the desired Parametric setup in the project tree and choose <b>View Analysis Result...</b> On the <b>Post Analysis Display</b> dialog box, click the <b>Result</b> tab, then select <b>Table</b> view, and click on the <b>Export</b> button.												
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;SetupName&gt;</td> <td>String</td> <td>Must be one of the existing Parametric setup names</td> </tr> <tr> <td>&lt;FileName&gt;</td> <td>String</td> <td>Must be a valid file path and name with extension of csv, tab, dat or txt</td> </tr> <tr> <td>&lt;bOutputUnits&gt;</td> <td>Boolean</td> <td>If set to true, values will be printed with units</td> </tr> </tbody> </table>	Name	Type	Description	<SetupName>	String	Must be one of the existing Parametric setup names	<FileName>	String	Must be a valid file path and name with extension of csv, tab, dat or txt	<bOutputUnits>	Boolean	If set to true, values will be printed with units
Name	Type	Description											
<SetupName>	String	Must be one of the existing Parametric setup names											
<FileName>	String	Must be a valid file path and name with extension of csv, tab, dat or txt											
<bOutputUnits>	Boolean	If set to true, values will be printed with units											

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

<b>Python Syntax</b>	ExportParametricResults (<SetupName>, <FileName>, <bOutputUnits>)
<b>Python Example</b>	<pre>oModule.ExportParametricResults (     "ParametricSetup1", "c:/exportdir/test.csv", False)</pre>

<b>VB Syntax</b>	ExportParametricResults <SetupName>, <FileName>, <bOutputUnits>
<b>VB Example</b>	<pre>oModule.ExportParametricResults     "ParametricSetup1", "c:/exportdir/test.csv", false</pre>

## ExportParametricSetupTable

Exports the parametric setup table as a CSV file.

<b>UI Access</b>	Double-click parametric setup. Select <b>Table</b> tab. Click <b>Export</b> .									
<b>Parameters</b>	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>&lt;SetupName&gt;</td><td>String</td><td>Name of the setup.</td></tr><tr><td>&lt;filePath&gt;</td><td>String</td><td>Full path for file export.</td></tr></tbody></table>	Name	Type	Description	<SetupName>	String	Name of the setup.	<filePath>	String	Full path for file export.
Name	Type	Description								
<SetupName>	String	Name of the setup.								
<filePath>	String	Full path for file export.								
<b>Return Value</b>	None									

<b>Python</b>	ExportParametricSetupTable (<SetupName>, <filePath>)
---------------	------------------------------------------------------

<b>Syntax</b>	
<b>Python Example</b>	<code>oModule.ExportParametricSetupTable('ParametricSetup1', 'E:/Files/ParametricSetup1_Table.csv')</code>

<b>VB Syntax</b>	<code>ExportParametricSetupTable &lt;SetupName&gt;, &lt;filePath&gt;</code>
<b>VB Example</b>	<code>obj.ExportParametricSetupTable "ParametricSetup1", "E:/Files/ParametricSetup1_Table.csv"</code>

## ExportRespSurfaceMinMaxTable

Exports min-max table from a response surface to a file

<b>UI Access</b>	Click on <b>Export...</b> From the <b>Response Surface</b> tab of the Design of Experiments <b>Post Analysis Display</b> dialog.		
<b>Parameters</b>	Name	Type	Description
	<code>&lt;DOEName&gt;</code>	String	Name of the Design of Experiments (DOE) setup.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	<code>ExportRespSurfaceMinMaxTable(&lt;DOEName&gt;, &lt;FileName&gt;)</code>
<b>Python Example</b>	<code>oModule.ExportRespSurfaceMinMaxTable ("DesignOfExperimentsSetup1", "C:/temp/DesignOfExperimentsSetup1_Min-Max_Search.csv")</code>

<b>VB Syntax</b>	ExportRespSurfaceMinMaxTable <DOEName>, <FileName>
<b>VB Example</b>	<pre>oModule.ExportRespSurfaceMinMaxTable _     "DesignOfExperimentsSetup1", _     "C:/temp/DesignOfExperimentsSetup1_Min-Max_Search.csv"</pre>

## ExportRespSurfaceRefinePoints

Exports refinement points table to a file

<b>UI Access</b>	From the <b>Response Surface</b> tab of the Design of Experiments <b>Post Analysis Display</b> dialog box, select <b>Refinement Points</b> option under <b>View</b> , Click on <b>Export....</b>									
<b>Parameters</b>	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>&lt;DOEName&gt;</td><td>String</td><td>Name of the Design of Experiments (DOE) setup.</td></tr><tr><td>&lt;FileName&gt;</td><td>String</td><td>Output file name with path.</td></tr></tbody></table>	Name	Type	Description	<DOEName>	String	Name of the Design of Experiments (DOE) setup.	<FileName>	String	Output file name with path.
Name	Type	Description								
<DOEName>	String	Name of the Design of Experiments (DOE) setup.								
<FileName>	String	Output file name with path.								
<b>Return Value</b>	None.									

<b>Python Syntax</b>	ExportRespSurfaceRefinePoints(<DOEName>, <FileName>)
<b>Python Example</b>	<pre>oModule.ExportRespSurfaceRefinePoints ("DesignOfExperimentsSetup1",     "C:/temp/DesignOfExperimentsSetup1_Refine_Points.csv")</pre>

<b>VB Syntax</b>	ExportRespSurfaceRefinePoints <DOEName>, <FileName>
<b>VB Example</b>	<pre> oModule.ExportRespSurfaceRefinePoints_ "DesignOfExperimentsSetup1", _ "C:/temp/DesignOfExperimentsSetup1_Refine_Points.csv" </pre>

## ExportRespSurfaceResponsePoints

Exports response points table to a file

<b>UI Access</b>	From the <b>Response Surface</b> tab of the Design of Experiments <b>Post Analysis Display</b> dialog box, select <b>Response Points</b> option under <b>View</b> , Click on <b>Export....</b>									
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;DOEName&gt;</td> <td>String</td> <td>Name of the Design of Experiments (DOE) setup.</td> </tr> <tr> <td>&lt;FileName&gt;</td> <td>String</td> <td>Output file name with path.</td> </tr> </tbody> </table>	Name	Type	Description	<DOEName>	String	Name of the Design of Experiments (DOE) setup.	<FileName>	String	Output file name with path.
Name	Type	Description								
<DOEName>	String	Name of the Design of Experiments (DOE) setup.								
<FileName>	String	Output file name with path.								
<b>Return Value</b>	None.									

<b>Python Syntax</b>	ExportRespSurfaceResponsePoints (<DOEName>, <FileName>)
<b>Python Example</b>	<pre> oModule.ExportRespSurfaceResponsePoints ("DesignOfExperimentsSetup1", "C:/temp/DesignOfExperimentsSetup1_Response_Points.csv") </pre>

<b>VB Syntax</b>	ExportRespSurfaceResponsePoints <DOEName>, <FileName>
<b>VB Example</b>	<pre> oModule.ExportRespSurfaceResponsePoints_ "DesignOfExperimentsSetup1", _ </pre>

	"C:/temp/DesignOfExperimentsSetup1_Response_Points.csv"
--	---------------------------------------------------------

## ExportRespSurfaceVerificationPoints

Exports verification points table to a file

<b>UI Access</b>	From the <b>Response Surface</b> tab of the Design of Experiments <b>Post Analysis Display</b> dialog box, select <b>Verification Points</b> option under <b>View</b> , Click on <b>Export....</b>									
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;DOEName&gt;</td> <td>String</td> <td>Name of the Design of Experiments (DOE) setup.</td> </tr> <tr> <td>&lt;FileName&gt;</td> <td>String</td> <td>Output file name with path.</td> </tr> </tbody> </table>	Name	Type	Description	<DOEName>	String	Name of the Design of Experiments (DOE) setup.	<FileName>	String	Output file name with path.
Name	Type	Description								
<DOEName>	String	Name of the Design of Experiments (DOE) setup.								
<FileName>	String	Output file name with path.								
<b>Return Value</b>	None.									

<b>Python Syntax</b>	ExportRespSurfaceVerificationPoints (<DOEName>, <FileName>)
<b>Python Example</b>	<pre>oModule.ExportRespSurfaceVerificationPoints("DesignOfExperimentsSetup1", "C:/temp/DesignOfExperimentsSetup1_Veri_Points.csv")</pre>

<b>VB Syntax</b>	ExportRespSurfaceVerificationPoints <DOEName>, <FileName>
<b>VB Example</b>	<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.

<b>UI Access</b>	NA		
<b>Parameters</b>	Name typeName	Type text string	Description 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".
<b>Return Value</b>	Array of setup names.		

<b>Python Syntax</b>	GetChildNames()
<b>Python Example</b>	<pre>oOptimModule = oDesign.GetChildObject("Optimetrics")</pre>

```
arrAllSetup = oOptimModule.GetChildNames()
arrParmSetup = oOptimModule.GetChildNames("'OptiParametric'")
arrOptimizeSetup = oOptimModule.GetChildNames("'Optimization'")
```

## GetChildObject [Optimetrics]

Gets a Setup Object of the Optimetrics module

<b>UI Access</b>	NA						
<b>Parameters</b>	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>Setup Name</td><td>text string</td><td>A optimetrics setup name, names returned by the GetChildNames().</td></tr></tbody></table>	Name	Type	Description	Setup Name	text string	A optimetrics setup name, names returned by the GetChildNames().
Name	Type	Description					
Setup Name	text string	A optimetrics setup name, names returned by the GetChildNames().					
<b>Return Value</b>	A script object for the setup See discussion of Optimetrics Setup Objects in <a href="#">Object Script Property Function Summary</a> .						

<b>Python Syntax</b>	GetChildObject()
<b>Python Example</b>	<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'].

<b>Python Syntax</b>	GetChildTypes ()
<b>Python Example</b>	<pre>oOptimModule = oDesign.GetChildObject("Optimetrics") arrSetupTypes = oOptimModule.GetChildTypes()</pre>

**GetName**

Returns the name and ID of the active design. If the design is a sub-circuit, returns the parent design also.

<b>UI Access</b>	N/A
<b>Parameters</b>	None.
<b>Return Value</b>	String indicating the name and ID of the active design and parent design (if the active design is a sub-circuit).

<b>Python Syntax</b>	GetName()
<b>Python Example</b>	<pre>design_name = oDesign.GetName()</pre>

<b>VB Syntax</b>	GetName
<b>VB Example</b>	<pre>design_name = oDesign.GetName</pre>

## GetObjPath [Design]

Obtains the path to the design.

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

<b>Python Syntax</b>	GetObjPath()
<b>Python Example</b>	<code>oDesign.GetObjPath()</code>

<b>VB Syntax</b>	GetObjPath
<b>VB Example</b>	<code>oDesign.GetObjPath</code>

## GetOptimetricResult

Returns an Optimetric calculation. The specific calculation is determined by the setup.

<b>UI Access</b>	N/A		
<b>Parameters</b>	Name	Type	Description
	<code>&lt;SetupName&gt;</code>	String	Optimetrics setup name.
	<code>&lt;vars&gt;</code>	Array	Array containing string variable names.

		Use the <b>Sweep Definitions</b> tab in the UI or the <SweepDefs> parameter in the <a href="#">InsertSetup</a> script to determine appropriate inputs.
<values>	Array	<p><i>Optional.</i> Array containing string values.</p> <p>When multiple variables and values are provided, the order must be the same in both the &lt;vars&gt; and &lt;values&gt; arrays. The first variable is paired with the first value, the second variable is paired with the second value, and so on.</p>
<b>Return Value</b>	Calculation result. If the setup contains more than one calculation, the output will be an array of values.	

<b>Python Syntax</b>	GetOptimetricResult(<SetupName>, <vars>, <values>)
<b>Python Example</b>	oModule.GetOptimetricResult('ParametricSetup1', ['AR', 'Re'], ['4.64', '6e+04'])

<b>VB Syntax</b>	GetOptimetricResult <SetupName>, <vars>, <values>
<b>VB Example</b>	<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>

## GetOptimetricsResult

Get an existing Optimization, Sensitivity, Statistical, Parametric or DesignXplorer result.

<b>UI Access</b>	Right click on the desired Optimetrics setup in the project tree and choose <b>View Analysis Result...</b> On the <b>Post Analysis Display</b> dialog box, click the <b>Result</b> tab, then select <b>Table</b> view, and click on the <b>Export</b> button		
<b>Parameters</b>	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..
<b>Return Value</b>	None		

<b>Python Syntax</b>	GetOptimetricsResult (<SetupName>, <FileName>, [useFullOutputName])
<b>Python Example</b>	<pre>oModule.GetOptimetricsResult (     "StatisticalSetup1", "c:/exportdir/test.csv", false)</pre>

<b>VB Syntax</b>	GetOptimetricsResult <SetupName>, <FileName>, [useFullOutputName]
<b>VB Example</b>	<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.

<b>Python Syntax</b>	GetPropNames ()
<b>Python Example</b>	<pre>oOptModule.GetPropNames() oOptModule.GetPropNames(True) oOptModule.GetPropNames(False)</pre>

## GetPropValue [Optimetrics]

Returns the property value for a setup property.

<b>UI Access</b>	NA						
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>property-path</td> <td></td> <td>a child object's property path. See <a href="#">property path discussion here</a>.</td> </tr> </tbody> </table>	Name	Type	Description	property-path		a child object's property path. See <a href="#">property path discussion here</a> .
Name	Type	Description					
property-path		a child object's property path. See <a href="#">property path discussion here</a> .					
<b>Return Value</b>	Returns the value of an setup property.						

<b>Python Syntax</b>	GetPropValue(propPath)
<b>Python Example</b>	<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>

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

<b>VB Example</b>	<pre>GetPropValue("Optimetrics/OptimizationSetup1/Enabled")</pre> <p>Returns True if Enabled, or False if disabled.</p>
-------------------	-------------------------------------------------------------------------------------------------------------------------

## GetSetupNames [Optimetrics]

Gets a list of Optimetrics setup names

<b>UI Access</b>	NA						
<b>Parameters</b>	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>None</td><td></td><td></td></tr></tbody></table>	Name	Type	Description	None		
Name	Type	Description					
None							
<b>Return Value</b>	IAnsoftCollectionObj – a collection of Optimetrics setup names						

<b>Python Syntax</b>	<pre>GetSetupNames()</pre>
<b>Python Example</b>	<pre>oModule = oDesign.GetModule("Optimetrics") setupNames = oModule.GetSetupNames()</pre>

<b>VB Syntax</b>	<pre>GetSetupNames()</pre>
<b>VB Example</b>	<pre>Set oModule_opt = oDesign.GetModule("Optimetrics") Set opt_setup_list = oModule.GetSetupNames() numsetups = setupnames.Count</pre>

## GetSetupNamesByType [Optimetrics]

Gets a list of Optimetrics setup names by type.

<b>UI Access</b>	NA						
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;Optimetrics type&gt;</td> <td>String</td> <td>Examples: parametric, optimization, statistical, sensitivity</td> </tr> </tbody> </table>	Name	Type	Description	<Optimetrics type>	String	Examples: parametric, optimization, statistical, sensitivity
Name	Type	Description					
<Optimetrics type>	String	Examples: parametric, optimization, statistical, sensitivity					
<b>Return Value</b>	Array of Optimetrics setup names of the given type.						

<b>Python Syntax</b>	GetSetupNamesByType (<Optimetrics type>)
<b>Python Example</b>	<pre>for name in oModule.GetSetupNamesByType("optimization")     AddInfoMessage(str(name))</pre>

<b>VB Syntax</b>	GetSetupNamesByType <Optimetrics type>
<b>VB Example</b>	<pre>For each name in oModule.GetSetupNamesByType("optimization")     MsgBox name Next</pre>

## ImportSetup

Import an Optimetric setup from a file.

<b>UI Access</b>	NA									
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;SetupTypeName&gt;</td> <td>String</td> <td>Must be one of "OptiParametric", "OptiOptimization", "OptiSensitivity", "OptiStatistical", or "OptiDesignExplorer".</td> </tr> <tr> <td>&lt;SetupInfo&gt;</td> <td>Array</td> <td>           Array("NAME:&lt;SetupName&gt;", "FilePath")    &lt;SetupName&gt;              Type: &lt;string&gt;              Name of the setup.    &lt;FilePath&gt;              Type : &lt;string: file path&gt;              Must be a valid file path and name.         </td> </tr> </tbody> </table>	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.
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.								
<b>Return Value</b>	None									

<b>Python Syntax</b>	ImportSetup (<SetupTypeName>, <SetupInfo>)
<b>Python Example</b>	<pre> oModule.ImportSetup ("OptiStatistical", ["NAME:StatisticalSetup1", "c:/importdir/mySetupInfoFile"]) </pre>

<b>VB Syntax</b>	ImportSetup <SetupTypeName>, <SetupInfo>
------------------	------------------------------------------

<b>VB Example</b>	<pre>oModule.ImportSetup "OptiStatistical", Array("NAME:StatisticalSetup1", "c:/importdir/mySetupInfoFile")</pre>
-------------------	-------------------------------------------------------------------------------------------------------------------

## InsertSetup

Adds a new solution setup.

UI Access	[product] > Analysis Setup > Add Solution Setup.											
<b>Parameters</b>	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;SetupType&gt;</td> <td>String</td> <td>Type of setup to be inserted.</td> </tr> <tr> <td>&lt;Attributes&gt;</td> <td>Array</td> <td> <p>Structured array.</p> <pre>Array("NAME:&lt;SetupName&gt;", &lt;NamedParameters&gt; &lt;Named Parameters&gt; "AdaptMultipleFreqs:=", &lt;boolean&gt;, "Frequency:=", &lt;string&gt;, "MaxDeltaS:=", &lt;float&gt;, "PortsOnly:=", &lt;boolean&gt;, "UseMatrixConv:=", &lt;boolean&gt;, ### If UseMatrixCov is True, ### [ "NAME:Matrix Convergence", &lt;Convergence Array&gt; ], MaximumPasses:=", &lt;integer&gt;, "MinimumPasses:=", &lt;integer&gt;, "MinimumConvergedPasses:=", &lt;integer&gt;,</pre> </td> </tr> </tbody> </table>	Name	Type	Description	<SetupType>	String	Type of setup to be inserted.	<Attributes>	Array	<p>Structured array.</p> <pre>Array("NAME:&lt;SetupName&gt;", &lt;NamedParameters&gt; &lt;Named Parameters&gt; "AdaptMultipleFreqs:=", &lt;boolean&gt;, "Frequency:=", &lt;string&gt;, "MaxDeltaS:=", &lt;float&gt;, "PortsOnly:=", &lt;boolean&gt;, "UseMatrixConv:=", &lt;boolean&gt;, ### If UseMatrixCov is True, ### [ "NAME:Matrix Convergence", &lt;Convergence Array&gt; ], MaximumPasses:=", &lt;integer&gt;, "MinimumPasses:=", &lt;integer&gt;, "MinimumConvergedPasses:=", &lt;integer&gt;,</pre>		
Name	Type	Description										
<SetupType>	String	Type of setup to be inserted.										
<Attributes>	Array	<p>Structured array.</p> <pre>Array("NAME:&lt;SetupName&gt;", &lt;NamedParameters&gt; &lt;Named Parameters&gt; "AdaptMultipleFreqs:=", &lt;boolean&gt;, "Frequency:=", &lt;string&gt;, "MaxDeltaS:=", &lt;float&gt;, "PortsOnly:=", &lt;boolean&gt;, "UseMatrixConv:=", &lt;boolean&gt;, ### If UseMatrixCov is True, ### [ "NAME:Matrix Convergence", &lt;Convergence Array&gt; ], MaximumPasses:=", &lt;integer&gt;, "MinimumPasses:=", &lt;integer&gt;, "MinimumConvergedPasses:=", &lt;integer&gt;,</pre>										

		<pre>"PercentRefinement:=", &lt;integer&gt;, "IsEnabled:=", &lt;boolean&gt;, "BasisOrder:=", &lt;integer&gt;, "DoLambdaRefine:=", &lt;boolean&gt;, "DoMaterialLambda:=", &lt;boolean&gt;, "SetLambdaTarget:=", &lt;boolean&gt;, "Target:=", &lt;float&gt;,  "PortAccuracy:=", &lt;integer&gt;, SaveRadFieldsOnly:=", &lt;boolean&gt;, "SaveAnyFields:=", &lt;boolean&gt;, "ListsForFields:=", &lt;array of string&gt;,  "IESolverType:=", &lt;string&gt;, "LambdaTargetForIESolver:=", &lt;float&gt;, "UseDefaultLambdaTgtForIESolver:=", &lt;boolean&gt;</pre>
<b>Return Value</b>	None.	

<b>Python Syntax</b>	InsertSetup(<SetupType>, <Attributes>)
<b>Python Example</b>	<pre>oModule.InsertSetup("HfssDrivenAuto", [ "NAME:Setup1",   ".IsEnabled:=", True,   "AutoSolverSetting:=", "Balanced",   [ "NAME:Sweeps",     [ "NAME:Sweep",</pre>

```
        "RangeType:=", "LinearStep",
        "RangeStart:=", "1GHz",
        "RangeEnd:=", "10GHz",
        "RangeStep:=", "1GHz"
    ],
],
"SaveRadFieldsOnly:=", False,
"SaveAnyFields:=", True,
"Type:=", "Discrete"
])

oModule.InsertSetup("HfssDrivenAuto",
[
    "NAME:Setup2",
    "SolveType:=" , "Auto",
    "IsEnabled:=" , True,
[
    "NAME:MeshLink",
    "ImportMesh:=" , False
],
"AutoSolverSetting:=" , "Balanced",
```

```
[  
    "NAME:Sweeps",  
    [  
        "NAME:Sweep",  
        "RangeType:=" , "LinearCount",  
        "RangeStart:=" , "0GHz",  
        "RangeEnd:=" , "10GHz",  
        "RangeCount:=" , 501  
    ],  
    "SaveRadFieldsOnly:=" , False,  
    "SaveAnyFields:=" , True,  
    "InfiniteSphereSetup:=" , "Infinite Sphere1",  
    "ListsForFields:=" , ["Objectlist2"],  
    "Type:=" , "Interpolating"  
])  
  
oModule.InsertSetup("HfssDriven",  
    [ "NAME:Setup3",  
        "AdaptMultipleFreqs:=", False,
```

```
"Frequency:=", "5GHz",
"MaxDeltaS:=", 0.02,
"PortsOnly:=", False,
"UseMatrixConv:=", False,
"MaximumPasses:=", 6,
"MinimumPasses:=", 1,
"MinimumConvergedPasses:=", 1,
"PercentRefinement:=", 30,
".IsEnabled:=", True,
"BasisOrder:=", 1,
"DoLambdaRefine:=", True,
"DoMaterialLambda:=", True,
"SetLambdaTarget:=", False,
"Target:=", 0.3333,
"UseMaxTetIncrease:=", False,
"PortAccuracy:=", 2,
"UseABCOnPort:=", False,
"SetPortMinMaxTri:=", False,
"UseDomains:=", True,
"UseIterativeSolver:=", False,
"IterativeResidual:=", 1E-06,
```

```
        "DDMSolverResidual:=", 0.0001,
        "EnhancedLowFreqAccuracy:=", True,
        "SaveRadFieldsOnly:=", False,
        "SaveAnyFields:=", True,
        "IESolverType:=", "Auto",
        "LambdaTargetForIESolver:=", 0.15,
        "UseDefaultLambdaTgtForIESolver:=", True,
        "SkipIERegionSolveDuringAdaptivePasses:=", True
        "RayDensityPerWavelength:=", 4,
        "MaxNumberOfBounces:=" , 5,
        "InfiniteSphereSetup:=" , "Infinite Sphere1",
        "SkipSBRSoIveDuringAdaptivePasses:=", True,
        "PTDUTDSimulationSettings:=", "PTD Correction + UTD Rays",
        "PTDEdgeDensity:=" , 20
    ] )

### An HFSS with Hybrid and Arrays setup
import ScriptEnv
ScriptEnv.Initialize("Ansoft.ElectronicsDesktop")
oDesktop.RestoreWindow()
```

```
oProject = oDesktop.SetActiveProject("Project1")
oDesign = oProject.SetActiveDesign("HFSSDesign1")
oModule = oDesign.GetModule("AnalysisSetup")
oModule.InsertSetup("HfssDriven",
[
    "NAME:Setup1",
    "SolveType:=" , "Single",
    "Frequency:=" , "5GHz",
    "MaxDeltaE:=" , 0.1,
    "MaximumPasses:=" , 6,
    "MinimumPasses:=" , 1,
    "MinimumConvergedPasses:=", 1,
    "PercentRefinement:=" , 30,
    ".IsEnabled:=" , True,
    [
        "NAME:MeshLink",
        "ImportMesh:=" , False
    ],
    "BasisOrder:=" , 1,
    "DoLambdaRefine:=" , True,
    "DoMaterialLambda:=" , True,
```

```
"SetLambdaTarget:="      , False,
"Target:="                , 0.3333,
"UseMaxTetIncrease:="    , False,
"DrivenSolverType:="     , "Direct Solver",
"EnhancedLowFreqAccuracy:=", False,
"SaveRadFieldsOnly:="    , False,
"SaveAnyFields:="        , True,
"IESolverType:="         , "Auto",
"LambdaTargetForIESolver:=", 0.15,
"UseDefaultLambdaTgtForIESolver:=", True,
"IE Solver Accuracy:="   , "Balanced",
"RayDensityPerWavelength:=", 4,
"MaxNumberOfBounces:="    , 5,
"RadiationSetup:="       , "Infinite Sphere1",
"PTDUTDSimulationSettings:=", "None",
"EnableSBRSelfCoupling:=", False,
"UseSBRAdvOptionsGOBlockage:=", False,
"UseSBRAdvOptionsWedges:=", False,
"SkipSBRSoIveDuringAdaptivePasses:=", False,
"UseSBREnhancedRadiatedPowerCalculation:=", True
```

```
    ] )

### An SBR+ Setup with Fast Frequency Looping
oModule.InsertSetup("HfssDriven",
[
    "NAME:Setup1",
    "IsEnabled:="           , True,
    [
        "NAME:MeshLink",
        "ImportMesh:="      , False
    ],
    "IsSbrRangeDoppler:="   , False,
    "RayDensityPerWavelength:=", 4,
    "MaxNumberOfBounces:="   , 5,
    "IsMonostaticRCS:="     , True,
    "EnableCWRays:="        , False,
    "RadiationSetup:="      , "",
    "PTDUTDSimulationSettings:=", "None",
    "FastFrequencyLooping:=", True,
    [
        "NAME:Sweeps",
```

```
[  
    "NAME:Sweep",  
    "RangeType:=" , "LinearStep",  
    "RangeStart:=" , "1GHz",  
    "RangeEnd:=" , "10GHz",  
    "RangeStep:=" , "1GHz"  
],  
    "ComputeFarFields:=" , True  
)  
  
### SBR+ Setup with Enhanced Radiated Power Calculation  
oModule.InsertSetup("HfssDriven",  
[  
    "NAME:Setup2",  
    ".IsEnabled:=" , True,  
    [  
        "NAME:MeshLink",  
        "ImportMesh:=" , False  
],
```

```
"IsSbrRangeDoppler:=" , False,
"RayDensityPerWavelength:=" , 4,
"MaxNumberOfBounces:=" , 5,
"EnableCWRays:=" , False,
"RadiationSetup:=" , "Infinite Spherel",
"PTDUTDSimulationSettings:=", "None",
"FastFrequencyLooping:=" , False,
"UseSBRAvOptionsGOBlockage:=" , False,
"UseSBRAvOptionsWedges:=" , False,
[
    "NAME:Sweeps",
    [
        "NAME:Sweep",
        "RangeType:=" , "LinearStep",
        "RangeStart:=" , "1GHz",
        "RangeEnd:=" , "10GHz",
        "RangeStep:=" , "1GHz"
    ]
],
"ComputeFarFields:=" , True,
"UseSBREnhancedRadiatedPowerCalculation:=" , True,
```

```
        "IsGOBlockageEnabled:=" , False,
        "GOBlockageSurfaceSelfBlock:=" , False
    ] )

### Insert RF Discharge Setup for HFSS
import ScriptEnv
ScriptEnv.Initialize("Ansoft.ElectronicsDesktop")
oDesktop.RestoreWindow()
oProject = oDesktop.SetActiveProject("coaxbend_discharge_r212")
oDesign = oProject.SetActiveDesign("HFSSDesign60degBendTeflon")
oModule = oDesign.GetModule("AnalysisSetup")
oModule.InsertSetup("HfssRFDischarge",
[
    "NAME:RFDischarge1",
    "Enabled:=" , True,
    [
        "NAME:MeshLink",
        "ImportMesh:=" , True,
        "Project:=" , "This Project*",
        "Product:=" , "HFSS",
```

```
"Design:=" , "This Design*",
"Soln:=" , "Setup1 : Sweep",
[

    "NAME:Params",
    "bend_angle:=" , "bend_angle"
] ,
"ForceSourceToSolve:=" , False,
"PreservePartnerSoln:=" , False,
"PathRelativeTo:=" , "SourceProduct",
"ApplyMeshOp:=" , True
] ,
[
    "NAME:Excitations",
    [
        "NAME:1:1",
        "Magnitude:=" , "1",
        "Phase:=" , "0deg"
] ,
[
    "NAME:2:1",
    "Magnitude:=" , "0",
    "Phase:=" , "0deg"
]
```

```
        "Phase:="           , "0deg"
    ]
],
[
    "NAME:Frequencies",
    "10GHz"
],
"Minimum Power:="      , "0.01",
"Maximum Power:="      , "1000000",
"Minimum Pressure:="   , "100pascal",
"Maximum Pressure:="   , "101325pascal",
"Postproc Sampling:="  , 500,
"Temperature:="         , "0cel",
"BuiltInGas:="          , "Argon"
"Save Electron Density:=", True,
"Is Pulsed Signal:="    , True,
"Duty Cycle:="          , 50
])
#####
Setup with UseMatrixConv is True and Matrix Convergence Entry Specified as All
###
```

```
oModule.InsertSetup("Setup1",
[
    "NAME:Setup1",
    "SolveType:=" , "Single",
    "Frequency:=" , "10GHz",
    "MaxDeltaS:=" , 0.02,
    "UseMatrixConv:=" , True,
    [
        "NAME:Matrix Convergence",
        "AllEntries:=" , True,
        "MagLimit:=" , "0.01",
        "PhaseLimit:=" , "2deg",
        "MagMinThreshold:=" , 0.01
    ],
    "MaximumPasses:=" , 12,
    "MinimumPasses:=" , 1,
    "MinimumConvergedPasses:=" , 1,
    "PercentRefinement:=" , 30,
    "IsEnabled:=" , True,
    [
        "NAME:MeshLink",
```

```
        "ImportMesh:="           , False
    ],
    "BasisOrder:="           , 1,
    "DoLambdaRefine:="       , True,
    "DoMaterialLambda:="     , True,
    "SetLambdaTarget:="      , False,
    "Target:="               , 0.3333,
    "UseMaxTetIncrease:="   , False,
    "PortAccuracy:="         , 2,
    "UseABCOnPort:="         , False,
    "SetPortMinMaxTri:="     , False,
    "DrivenSolverType:="     , "Direct Solver",
    "EnhancedLowFreqAccuracy:=", False,
    "SaveRadFieldsOnly:="    , False,
    "SaveAnyFields:="         , True,
    "IESolverType:="          , "ACA",
    "LambdaTargetForIESolver:=", 0.15,
    "UseDefaultLambdaTgtForIESolver:=", True,
    "IE Solver Accuracy:="   , "Balanced",
    "InfiniteSphereSetup:="   , ""
```

```
        ] )

### Setup with UseMatrixConv as Entries and MeshLink
oModule = oDesign.GetModule("AnalysisSetup")
oModule.InsertSetup("Setup1",
[
    "NAME:Setup1",
    "SolveType:=" , "Single",
    "Frequency:=" , "10GHz",
    "MaxDeltaS:=" , 0.02,
    "UseMatrixConv:=" , True,
    [
        "NAME:Matrix Convergence",
        [
            "NAME:Entries",
            [
                "NAME:Entry",
                "Port1:=" , "A[1,1]P1",
                "Port2:=" , "A[1,1]P1",
                "MagLimit:=" , "0.011",
                "PhaseLimit:=" , "5deg"
            ],
        ],
    ],
]
```

```
[  
    "NAME:Entry",  
    "Port1:=" , "A[4,4]P2",  
    "Port2:=" , "A[1,1]P2",  
    "MagLimit:=" , "0.041",  
    "PhaseLimit:=" , "5deg"  
]  
]  
]  
]  
,"MaximumPasses:=" , 1,  
"MinimumPasses:=" , 1,  
"MinimumConvergedPasses:=" , 1,  
"PercentRefinement:=" , 30,  
"IsEnabled:=" , True,  
[  
    "NAME:MeshLink",  
    "ImportMesh:=" , True,  
    "Project:=" , "This Project*",  
    "Product:=" , "",  
    "Design:=" , "unit",
```

```
"Soln:=" , "Setup1 : LastAdaptive",
[

    "NAME:Params",
    "Airbox_dist:=" , "9.9931mm",
    "Scan_Theta:=" , "0deg",
    "Scan_phi:=" , "0deg",
    "VirtualObject_dist:=" , "2.9979mm",
    "coax_inner_rad:=" , "0.125mm",
    "coax_outer_rad:=" , "0.425mm",
    "cut_off:=" , "1.7mm",
    "feedX:=" , "-0.4mm",
    "feedY:=" , "2mm",
    "feed_length:=" , "1mm",
    "patchX:=" , "9.65mm",
    "rot:=" , "45deg",
    "subH:=" , "0.8mm",
    "subX:=" , "15mm",
    "subY:=" , "15mm"

] ,
"ForceSourceToSolve:=" , False,
"PreservePartnerSoln:=" , False,
```

```
        "PathRelativeTo:="           , "TargetProject",
        "ApplyMeshOp:="            , True,
        "AdaptPort:="              , False
    ],
    "BasisOrder:="             , 1,
    "DoLambdaRefine:="         , False,
    "DoMaterialLambda:="       , True,
    "SetLambdaTarget:="        , False,
    "Target:="                 , 0.3333,
    "UseMaxTetIncrease:="     , False,
    "PortAccuracy:="          , 2,
    "UseABCOnPort:="          , False,
    "SetPortMinMaxTri:="       , False,
    "DrivenSolverType:="       , "Domain Decomposition",
    "IterativeResidual:="      , 0.0001,
    "DDMSolverResidual:="     , 0.0001,
    "EnhancedLowFreqAccuracy:=", False,
    "SaveRadFieldsOnly:="      , False,
    "SaveAnyFields:="          , True,
    "IESolverType:="           , "Auto",
```

```
"LambdaTargetForIESolver:=", 0.15,
"UseDefaultLambdaTgtForIESolver:=", True,
"IE Solver Accuracy:=", "Balanced",
"InfiniteSphereSetup:=", ""

])

### Example from design with UseMatrixConv True for AllDiagEntries

"UseMatrixConv:=", True,
[
    "NAME:Matrix Convergence",
    "AllDiagEntries:=", True,
    "DiagonalMag:=", "0.03",
    "DiagonalPhase:=", "4rad",
    "MagMinThreshold:=", 0.01,
    "AllOffDiagEntries:=", True,
    "OffDiagonalMag:=", "0.05",
    "OffDiagonalPhase:=", "6deg",
    "MagMinThreshold:=", 0.01
],
### Example from Modal design with UseMatrixConv True for Entries

"UseMatrixConv:=", True,
[
```

```
    "NAME:Matrix Convergence",
    [
        "NAME:Entries",
        [
            "NAME:Entry",
            "Port1:=" , "Port1",
            "Port2:=" , "Port1",
            "MagLimit:=" , "0.02",
            "PhaseLimit:=" , "5deg"
        ],
        [
            "NAME:Entry",
            "Port1:=" , "Port3",
            "Port2:=" , "Port2",
            "MagLimit:=" , "0.03",
            "PhaseLimit:=" , "5deg"
        ]
    ],
]
```

```
#### Example from Design with Periodic Ports and UseMatrixConv is True
"UseMatrixConv:="      , True,
[
    "NAME:Matrix Convergence",
    [
        "NAME:Entries",
        [
            "NAME:Entry",
            "Port1:="          , "Incident_Port1",
            "Port2:="          , "Incident_Port1",
            "MagLimit:="       , "0.11",
            "PhaseLimit:="     , "5deg"
        ],
        [
            "NAME:Entry",
            "Port1:="          , "Incident_Port2",
            "Port2:="          , "Incident_Port2",
            "MagLimit:="       , "0.22",
            "PhaseLimit:="     , "5deg"
        ]
    ]
]
```

```
        ] ,  
  
#### Example Terminal Design with UseMatrixConv = True and Expression Cache  
oProject = oDesktop.SetActiveProject("term-2x2_1_parasolid")  
oDesign = oProject.SetActiveDesign("unit")  
oModule = oDesign.GetModule("AnalysisSetup")  
oModule.InsertSetup("Setup1",  
    [  
        "NAME:Setup1",  
        "SolveType:=" , "Single",  
        "Frequency:=" , "10GHz",  
        "MaxDeltaS:=" , 0.02,  
        "UseMatrixConv:=" , True,  
        [  
            "NAME:Matrix Convergence",  
            [  
                "NAME:Entries",  
                [  
                    "NAME:Entry",  
                    "Port1:=" , "P2",  
                    "Port2:=" , "P1",
```

```
        "MagLimit:="      , "0.05",
        "PhaseLimit:="    , "10deg"
    ]
]

],
"MaximumPasses:="      , 15,
"MinimumPasses:="      , 1,
"MinimumConvergedPasses:=", 1,
"PercentRefinement:="   , 30,
".IsEnabled:="          , True,
[
    "NAME:MeshLink",
    "ImportMesh:="          , False
],
"BasisOrder:="          , 1,
"DoLambdaRefine:="       , True,
"DoMaterialLambda:="     , True,
"SetLambdaTarget:="       , False,
"Target:="                , 0.3333,
"UseMaxTetIncrease:="    , False,
"PortAccuracy:="          , 2,
```

```
"UseABCOnPort:=", False,  
"SetPortMinMaxTri:=", False,  
"DrivenSolverType:=", "Iterative Solver",  
"IterativeResidual:=", 0.0001,  
"DDMSolverResidual:=", 0.0001,  
"EnhancedLowFreqAccuracy:=", False,  
"SaveRadFieldsOnly:=", False,  
"SaveAnyFields:=", True,  
[  
    "NAME:ExpressionCache",  
    [  
        "NAME:CacheItem",  
        "Title:=", "dB_AxialRatioValue_1",  
        "Expression:=", "dB(AxialRatioValue)",  
        "Intrinsics:=", "Phi='0deg'\ Theta='0deg'",  
        "ReportType:=", "Far Fields",  
        [  
            "NAME:ExpressionContext",  
            "Context:=", "Infinite Spherel"  
        ]  
    ]
```

```
],  
[  
    "NAME:CacheItem",  
    "Title:="           , "dB_St_Diff1_Diff1__1",  
    "Expression:="     , "dB(St(Diff1,Diff1))",  
    "Intrinsics:="     , "",  
    "ReportType:="     , "Terminal Solution Data",  
    [  
        "NAME:ExpressionContext",  
        "Diff:="             , "Differential Pairs"  
    ]  
],  
[  
    "NAME:CacheItem",  
    "Title:="           , "dB_St_P1_P1__1",  
    "Expression:="     , "dB(St(P1,P1))",  
    "Intrinsics:="     , "",  
    "ReportType:="     , "Terminal Solution Data",  
    [  
        "NAME:ExpressionContext",  
        "Diff:="             , "Terminals"
```

```
        ]
    ],
    [
        "CacheSaveKind:="      , "Delta",
        "ConstantDelta:="     , "0s",
        "IESolverType:="      , "Auto",
        "LambdaTargetForIESolver:=", 0.15,
        "UseDefaultLambdaTgtForIESolver:=", True,
        "IE Solver Accuracy:=" , "Balanced",
        "InfiniteSphereSetup:=" , ""
    )
)
```

VB Syntax	InsertSetup <SetupType>, <Attributes>
VB Example	An HFSS Driven project solution with Auto Solution: <pre>oModule.InsertSetup "HfssDrivenAuto", Array("NAME:Setup1", "IsEnabled:=", true, _ "AutoSolverSetting:=", "Balanced", _ Array("NAME:Sweeps", Array("NAME:Sweep", "RangeType:=", "LinearCount", _ "RangeStart:=", "1GHz", "RangeEnd:=", "10GHz", "RangeCount:=", "451")), _ "SaveRadFieldsOnly:=", false, "SaveAnyFields:=", false, "Type:=", "Interpolating")</pre>

A Driven solution type with a mesh link. References to dependent solve in old scripts are converted to mesh link form.

```
oModule.InsertSetup "HfssDriven",
Array("NAME:Setup1",
"Frequency:=", "1GHz",
"MaxDeltaE:=", 0.1,
"MaximumPasses:=", 6,
"MinimumPasses:=", 1,
"MinimumConvergedPasses:=", 1,
"PercentRefinement:=", 30,
".IsEnabled:=", true,
Array("NAME:MeshLink",
"Project:=", "Tee.aedt",
"Design:=", "TeeModel",
"Soln:=", "Setup1 : LastAdaptive",
Array("NAME:Params", "offset:=", "0in"),
"ForceSourceToSolve:=", false,
"PreservePartnerSoln:=", false,
"PathRelativeTo:=", "SourceProduct",
"ApplyMeshOp:=", true),
"BasisOrder:=", 1,
"UseIterativeSolver:=", false,
```

```
"DoLambdaRefine:=", false,
"DoMaterialLambda:=", true,
"SetLambdaTarget:=", false,
"Target:=", 0.3333,
"UseMaxTetIncrease:=", false,
"MaxTetIncrease:=", 1000000,
"EnableSolverDomains:=", false,
"ThermalFeedback:=", false,
"UsingConstantDelta:=", 0,
"ConstantDelta:=", "0s",
"NumberSolveSteps:=", 1)

A Driven solution type with ports:
oModule.InsertSetup "HfssDriven",
Array("NAME:Setup2",
"Frequency:=", "1GHz",
"PortsOnly:=", false,
"MaxDeltaS:=", 0.02,
"UseMatrixConv:=", false,
"MaximumPasses:=", 6,
"MinimumPasses:=", 1,
```

```
"MinimumConvergedPasses:=", 1,
"PercentRefinement:=", 30,
".IsEnabled:=", true,
"BasisOrder:=", 1,
"UseIterativeSolver:=", true,
"IterativeResidual:=", 0.0001,
"DoLambdaRefine:=", true,
"DoMaterialLambda:=", false,
"SetLambdaTarget:=", false,
"Target:=", 0.3333,
"UseMaxTetIncrease:=", false,
"MaxTetIncrease:=", 1000000,
"PortAccuracy:=", 2,
"UseABCOnPort:=", true,
"SetPortMinMaxTri:=", false,
"EnableSolverDomains:=", false,
"ThermalFeedback:=", false,
"UsingConstantDelta:=", 0,
"ConstantDelta:=", "0s",
"NumberSolveSteps:=", 1)
```

An Eigenmode solution type:

```
oModule.InsertSetup "HfssEigen",
Array("NAME:Setup2",
"MinimumFrequency:=", "1.77347GHz",
"NumModes:=", 1,
"MaxDeltaFreq:=", 10,
"ConvergeOnRealFreq:=", true,
"MaximumPasses:=", 3,
"MinimumPasses:=", 1,
"MinimumConvergedPasses:=", 1,
"PercentRefinement:=", 30,
".IsEnabled:=", true,
"BasisOrder:=", 1,
"UseIterativeSolver:=", false,
"DoLambdaRefine:=", true,
"DoMaterialLambda:=", true,
"SetLambdaTarget:=", false,
"Target:=", 0.2,
"UseMaxTetIncrease:=", false,
"MaxTetIncrease:=", 1000000,
"UsingConstantDelta:=", 0,
```

```
"ConstantDelta:=", "0s",
"NumberSolveSteps:=", 1)

A Driven solution type with ports and matrix convergence:

oModule.InsertSetup "HfssDriven",
Array("NAME:Setup2",
"Frequency:=", "1GHz",
"PortsOnly:=", false,
"MaxDeltaS:=", 0.02,
"UseMatrixConv:=", true,
Array("NAME:ConvergenceMatrix",
"AllDiagEntries:=", true,
"MagMinThreshold:=", 0.01,
"Entry:=", Array("Port1:=", "abc", "ModeNum1:=", 1)),
"MaximumPasses:=", 6,
"MinimumPasses:=", 1,
"MinimumConvergedPasses:=", 1,
"PercentRefinement:=", 30,
".IsEnabled:=", true,
"BasisOrder:=", 1,
"UseIterativeSolver:=", false,
"DoLambdaRefine:=", true,
```

```
"DoMaterialLambda:=", true,
"SetLambdaTarget:=", false,
"Target:=", 0.3333,
"UseMaxTetIncrease:=", false,
"MaxTetIncrease:=", 1000000,
"PortAccuracy:=", 2,
"UseABCOnPort:=", true,
"SetPortMinMaxTri:=", false,
"EnableSolverDomains:=", false,
"ThermalFeedback:=", false,
"UsingConstantDelta:=", 0,
"ConstantDelta:=", "0s",
"NumberSolveSteps:=", 1)

A Driven solution type with Multi-Frequencies specified:
oModule.InsertSetup "HfssDriven", Array("NAME:Setup2", "AdaptMultipleFreqs:=", true, _
Array("NAME:MultipleAdaptiveFreqsSetup", "1.1GHz:=", Array( _
0.02), "10GHz:=", Array(0.02), "11GHz:=", Array(0.02), "12.5GHz:=", Array(0.02),
"15GHz:=", Array( _
0.02), "20GHz:=", Array(0.02)), "MaximumPasses:=", 6, "MinimumPasses:=", 1, "Min-
imumConvergedPasses:=", _
1, "PercentRefinement:=", 30, ".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, "RayDensityPerWavelength:=", 4, "MaxNumberOfBounces:=", 5)

An Driven Solution type with Broadband specified:

oModule.InsertSetup "HfssDriven", Array("NAME:Setup1", "AdaptMultipleFreqs:=", true, _
Array("NAME:MultipleAdaptiveFreqsSetup", Array("NAME:Broadband", "Low:=", _
"2GHz", "High:=", "20GHz")), "MaximumPasses:=", 6, "MinimumPasses:=", 1, "Min-_
imumConvergedPasses:=", _
1, "PercentRefinement:=", 30, ".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, "RayDensityPerWavelength:=", 4, "MaxNumberOfBounces:=", 5)

oModule.InsertFrequencySweep "Setup1", Array("NAME:Sweep", ".IsEnabled:=", true,
"RangeType:=", _

```

```

"LinearCount", "RangeStart:=", "2GHz", "RangeEnd:=", "20GHz", "RangeCount:=", __
451, "Type:=", "Interpolating", "SaveFields:=", false, "SaveRadFields:=", __
false, "InterpTolerance:=", 0.5, "InterpMaxSolns:=", 250, "InterpMinSolns:=", __
0, "InterpMinSubranges:=", 1, "ExtrapToDC:=", false, "InterpUses:=", true, "Inter-
pUsePortImped:=", __
false, "InterpUsePropConst:=", true, "UseDerivativeConvergence:=", false, "Inter-
pDerivTolerance:=", __
0.2, "UseFullBasis:=", true, "EnforcePassivity:=", true, "PassivityErrorTolerance:=",
__
0.0001)

```

## PasteSetup [Optimetrics]

Pastes the specified Optimetrics setup.

<b>UI Access</b>	NA						
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;SetupName&gt;</td> <td>String</td> <td>Name of the Setup</td> </tr> </tbody> </table>	Name	Type	Description	<SetupName>	String	Name of the Setup
Name	Type	Description					
<SetupName>	String	Name of the Setup					
<b>Return Value</b>	None						

<b>Python Syntax</b>	PasteSetup (<SetupName>)
<b>Python Example</b>	oModule.PasteSetup ("OptimizationSetup1")

---

<b>VB Syntax</b>	PasteSetup <SetupName>
<b>VB Example</b>	<code>oModule.PasteSetup "OptimizationSetup1"</code>

## RenameSetup [Optimetrics]

Renames the specified Optimetrics setup.

<b>UI Access</b>	Right-click the setup in the project tree, and then click <b>Rename</b> on the shortcut menu.									
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;OldName&gt;</td> <td>String</td> <td>The name that needs to be replaced</td> </tr> <tr> <td>&lt;NewName&gt;</td> <td>String</td> <td>Replacement name</td> </tr> </tbody> </table>	Name	Type	Description	<OldName>	String	The name that needs to be replaced	<NewName>	String	Replacement name
Name	Type	Description								
<OldName>	String	The name that needs to be replaced								
<NewName>	String	Replacement name								
<b>Return Value</b>	None									

<b>Python Syntax</b>	RenameSetup (<OldName> <NewName>)
<b>Python Example</b>	<code>oModule.RenameSetup ("OptimizationSetup1" "MyOptimization")</code>

<b>VB Syntax</b>	RenameSetup <OldName> <NewName>
<b>VB Example</b>	<code>oModule.RenameSetup "OptimizationSetup1" "MyOptimization"</code>

## SetPropValue [Optimetrics]

Sets the property value for the active Optimetrics setup.

<b>UI Access</b>	Set Property value on Optimetrics objects		
<b>Parameters</b>	Name	Type	Description
	Property path	text string	Setup property path. See discussion of <a href="#">Property Path</a>
<b>Return Value</b>	True if the property is found and the new value is valid. Otherwise return False.		

<b>Python Syntax</b>	SetPropValue(propPath, newValue)
<b>Python Example</b>	<pre>oOptModule.SetPropValue("ParametricSetup1\Enabled", False) //disable ParametricSetup1 oOptModule.SetPropValue("OptimizationSetup1/Optimizer", "Quasi Newton")</pre>

<b>VB Syntax</b>	SetPropValue(propPath, newValue)
<b>VB Example</b>	SetPropValue("ParametricSetup1\Enabled", False)

## SolveAllSetup

Solves all Optimetrics setups

<b>UI Access</b>	Right-click on <b>Optimetrics</b> in Project Manager and select <b>Analyze&gt;All</b> from context menu		
<b>Parameters</b>	Name	Type	Description
	None		

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

<b>Python Syntax</b>	SolveAllSetup()
<b>Python Example</b>	<code>oModule.SolveAllSetup ()</code>

<b>VB Syntax</b>	SolveAllSetup
<b>VB Example</b>	<code>oModule.SolveAllSetup</code>

## SolveSetup [Optimetrics]

Solves the specified Optimetrics setup.

<b>UI Access</b>	Right-click the setup in the project tree, and then click <b>Analyze</b> on the shortcut menu.							
<b>Parameters</b>	<table border="1"> <tr> <td>Name</td> <td>Type</td> <td>Description</td> </tr> <tr> <td><code>&lt;SetupName&gt;</code></td> <td>String</td> <td>Name of the setup to be solved</td> </tr> </table>		Name	Type	Description	<code>&lt;SetupName&gt;</code>	String	Name of the setup to be solved
Name	Type	Description						
<code>&lt;SetupName&gt;</code>	String	Name of the setup to be solved						
<b>Return Value</b>	None							

<b>Python Syntax</b>	SolveSetup (<SetupName>)
<b>Python Example</b>	<code>oModule.SolveSetup ("OptimizationSetup1")</code>

<b>VB Syntax</b>	SolveSetup <SetupName>
<b>VB Example</b>	<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](#)

<b>UI Access</b>	Under <b>Optimetrics</b> in the Project Manager, right-click the < <b>Parametric Setup Name</b> > and then click <b>Validate Parametric Setup</b> on the shortcut menu.						
<b>Parameters</b>	<table border="1"><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;SetupName&gt;</td><td>String</td><td>Name of the parametric setup.</td></tr></table>	Name	Type	Description	<SetupName>	String	Name of the parametric setup.
Name	Type	Description					
<SetupName>	String	Name of the parametric setup.					
<b>Return Value</b>	None						

<b>Python Syntax</b>	ValidateSetup (<SetupName>)
<b>Python Example</b>	<code>oModule.ValidateSetup('ParametricSetup1')</code>

<b>VB Syntax</b>	ValidateSetup <SetupName>
<b>VB Example</b>	<code>oModule.ValidateSetup "ParametricSetup1"</code>

## 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](#)  
[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.

<b>UI Access</b>	NA		
<b>Parameters</b>	Name <code>&lt;SetupName&gt;</code>	Type String	Description Name of the setup.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	<code>CopySetup (&lt;SetupName&gt;)</code>
<b>Python Example</b>	<code>oModule.CopySetup ("OptimizationSetup1")</code>

<b>VB Syntax</b>	CopySetup <SetupName>
<b>VB Example</b>	<code>oModule.CopySetup "OptimizationSetup1"</code>

## DeleteSetups [Optimetrics]

Deletes the specified Optimetrics setups.

<b>UI Access</b>	Right-click the setup in the project tree, and then click <b>Delete</b> on the shortcut menu						
<b>Parameters</b>	<table border="1"> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> <tr> <td>&lt;NameArray&gt;</td> <td>Array of Strings</td> <td>An Array of Setup Names</td> </tr> </table>	Name	Type	Description	<NameArray>	Array of Strings	An Array of Setup Names
Name	Type	Description					
<NameArray>	Array of Strings	An Array of Setup Names					
<b>Return Value</b>	None						

<b>Python Syntax</b>	DeleteSetups (<NameArray>)
<b>Python Example</b>	<code>oModule.DeleteSetups ( ["OptimizationSetup1"] )</code>

<b>VB Syntax</b>	DeleteSetups <NameArray>
<b>VB Example</b>	<code>oModule.DeleteSetups Array("OptimizationSetup1")</code>

## DistributedAnalyzeSetup

Distributes all variable value instances within a parametric sweep to different machines already specified from within the user interface

<b>UI Access</b>	Right-click the parametric setup name in the project tree and select Distribute Analysis.
------------------	-------------------------------------------------------------------------------------------

<b>Parameters</b>	Name	Type	Description
	<ParametricSetupName>	String	Name of the Setup
<b>Return Value</b>	None		

<b>Python Syntax</b>	DistributedAnalyzeSetup (<ParametricSetupName>)
<b>Python Example</b>	<code>oModule.DistributedAnalyzeSetup ("ParametricSetup1")</code>

<b>VB Syntax</b>	DistributedAnalyzeSetup <ParametricSetupName>
<b>VB Example</b>	<code>oModule.DistributedAnalyzeSetup "ParametricSetup1"</code>

## EditSetup

Modifies an existing solution setup.

<b>UI Access</b>	Double-click a solution setup in the project tree to modify its settings.		
<b>Parameters</b>	Name	Type	Description
	<SetupName>	String	Name of the solve setup being edited.
	<Attributes>	Array	Structured array.  Array ("NAME:<NewSetupName>", <NamedParameters>)  See the InsertSetup command for details and examples.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	EditSetup (<SetupName>, <Attributes>)
<b>Python Example</b>	<pre> oModule.EditSetup("Setup1",                   [ "NAME:NewSetup",                     "AdaptiveFreq:=", "1GHz",                     "EnableDistribProbTypeOption:=", false,                     "SaveFields:=", "true",                     "Enabled:=", true,                     [ "NAME:Cap",                       "MaxPass:=", 10,                       "MinPass:=", 1,                       "MinConvPass:=", 2,                       "PerError:=", 1,                       "PerRefine:=", 30,                       "AutoIncreaseSolutionOrder:=", false,                       "SolutionOrder:=", "Normal"],                     [ "NAME:DC",                       "Residual:=", 1E-005,                       "SolveResOnly:=", false,                       [ "NAME:Cond",                         "MaxPass:=", 10, </pre>

```
        "MinPass:=", 1,
        "MinConvPass:=", 1,
        "PerError:=", 1,
        "PerRefine:=", 30),
        [ "NAME:Mult",
            "MaxPass:=", 1,
            "MinPass:=", 1,
            "MinConvPass:=", 1,
            "PerError:=", 1,
            "PerRefine:=", 30]],
        [ "NAME:AC",
            "MaxPass:=", 10,
            "MinPass:=", 1,
            "MinConvPass:=", 2,
            "PerError:=", 1,
            "PerRefine:=", 30]]
    )
oModule.InsertSetup("HfssDrivenAuto",
[ "NAME:Setup1",
    "IsEnabled:=", True,
```

```
"AutoSolverSetting:=", "Balanced",
[ "NAME:Sweeps",
    [ "NAME:Sweep",
        "RangeType:=", "LinearStep",
        "RangeStart:=", "1GHz",
        "RangeEnd:=", "10GHz",
        "RangeStep:=", "1GHz"
    ]
],
"SaveRadFieldsOnly:=", False,
"SaveAnyFields:=", True,
"Type:=", "Discrete"
])

oModule.InsertSetup("HfssDrivenAuto",
[
    "NAME:Setup2",
    "SolveType:=" , "Auto",
    "IsEnabled:=" , True,
[
    "NAME:MeshLink",
```

```
        "ImportMesh:="           , False
    ],
    "AutoSolverSetting:="   , "Balanced",
    [
        "NAME:Sweeps",
        [
            "NAME:Sweep",
            "RangeType:="          , "LinearCount",
            "RangeStart:="          , "0GHz",
            "RangeEnd:="             , "10GHz",
            "RangeCount:="           , 501
        ]
    ],
    "SaveRadFieldsOnly:="   , False,
    "SaveAnyFields:="       , True,
    "InfiniteSphereSetup:=" , "Infinite Sphere1",
    "ListsForFields:="      , ["Objectlist2"],
    "Type:="                 , "Interpolating"
])
```

```
oModule.InsertSetup("HfssDriven",
["NAME:Setup3",
 "AdaptMultipleFreqs:=", False,
 "Frequency:=", "5GHz",
 "MaxDeltaS:=", 0.02,
 "PortsOnly:=", False,
 "UseMatrixConv:=", False,
 "MaximumPasses:=", 6,
 "MinimumPasses:=", 1,
 "MinimumConvergedPasses:=", 1,
 "PercentRefinement:=", 30,
 "IsEnabled:=", True,
 "BasisOrder:=", 1,
 "DoLambdaRefine:=", True,
 "DoMaterialLambda:=", True,
 "SetLambdaTarget:=", False,
 "Target:=", 0.3333,
 "UseMaxTetIncrease:=", False,
 "PortAccuracy:=", 2,
 "UseABCOnPort:=", False,
 "SetPortMinMaxTri:=", False,
```

```
"UseDomains:=", True,  
"UseIterativeSolver:=", False,  
"IterativeResidual:=", 1E-06,  
"DDMSolverResidual:=", 0.0001,  
"EnhancedLowFreqAccuracy:=", True,  
"SaveRadFieldsOnly:=", False,  
"SaveAnyFields:=", True,  
"IESolverType:=", "Auto",  
"LambdaTargetForIESolver:=", 0.15,  
"UseDefaultLambdaTgtForIESolver:=", True,  
"SkipPIERegionSolveDuringAdaptivePasses:=", True  
"RayDensityPerWavelength:=", 4,  
"MaxNumberOfBounces:=" , 5,  
"InfiniteSphereSetup:=" , "Infinite Sphere1",  
"SkipSBRSSolveDuringAdaptivePasses:=", True,  
"PTDUTDSimulationSettings:=", "PTD Correction + UTD Rays",  
"PTDEdgeDensity:=" , 20  
])  
### Edit an SBR+ Setup with Fast Frequency Looping  
oModule.EditSetup("HfssDriven",
```

```
[  
    "NAME:Setup1",  
    "IsEnabled:="           , True,  
    [  
        "NAME:MeshLink",  
        "ImportMesh:="          , False  
    ],  
    "IsSbrRangeDoppler:="   , False,  
    "RayDensityPerWavelength:=" , 4,  
    "MaxNumberOfBounces:="   , 5,  
    "IsMonostaticRCS:="     , True,  
    "EnableCW Rays:="       , False,  
    "RadiationSetup:="      , "",  
    "PTDUTDSimulationSettings:=" , "None",  
    "FastFrequencyLooping:=" , True,  
    [  
        "NAME:Sweeps",  
        [  
            "NAME:Sweep",  
            "RangeType:="          , "LinearStep",  
            "RangeStart:="          , "1GHz",  
        ]  
    ]  
]
```

```
        "RangeEnd:=", "10GHz",
        "RangeStep:=", "1GHz"
    ],
],
"ComputeFarFields:=", True
"UseSBREnhancedRadiatedPowerCalculation:=", True,
"IsGOBlockageEnabled:=", False,
"GOBlockageSurfaceSelfBlock:=", False
])

#### Edit and RF Discharge Setup for HFSS
import ScriptEnv
ScriptEnv.Initialize("Ansoft.ElectronicsDesktop")
oDesktop.RestoreWindow()
oProject = oDesktop.SetActiveProject("coaxbend_discharge_r212")
oDesign = oProject.SetActiveDesign("HFSSDesign60degBendTeflon")
oModule = oDesign.GetModule("AnalysisSetup")
oModule.EditSetup("RFDischarge1",
[
    "NAME:RFDischarge1",
    "Enabled:=", True,
```

```
[  
    "NAME:MeshLink",  
    "ImportMesh:="           , True,  
    "Project:="              , "This Project*",  
    "Product:="              , "HFSS",  
    "Design:="               , "This Design*",  
    "Soln:="                 , "Setup1 : Sweep",  
    [  
        "NAME:Params",  
        "bend_angle:="          , "bend_angle"  
    ],  
    "ForceSourceToSolve:="   , True,  
    "PreservePartnerSoln:="  , False,  
    "PathRelativeTo:="        , "SourceProduct",  
    "ApplyMeshOp:="          , True  
],  
[  
    "NAME:Excitations",  
    [  
        "NAME:1:1",  
        "Magnitude:="          , "1",  
    ]]
```

```
        "Phase:="           , "0deg"
    ] ,
    [
        "NAME:2:1",
        "Magnitude:="      , "0",
        "Phase:="           , "0deg"
    ]
],
[
    "NAME:Frequencies",
    "10GHz"
],
[
    "Minimum Power:="   , "0.01",
    "Maximum Power:="   , "1000000",
    "Minimum Pressure:=" , "100pascal",
    "Maximum Pressure:=" , "101325pascal",
    "Postproc Sampling:=" , 500,
    "Temperature:="      , "0cel",
    "BuiltInGas:="       , "Helium"
]
)
```

<b>VB Syntax</b>	EditSetup <SetupName>, <Attributes>
<b>VB Example</b>	<pre> oModule.EditSetup "Setup1",     Array("NAME:NewSetup",         "AdaptiveFreq:=", "1GHz",         "EnableDistribProbTypeOption:=", false,         "SaveFields:=", "true",         "Enabled:=", true,         Array("NAME:Cap",             "MaxPass:=", 10,             "MinPass:=", 1,             "MinConvPass:=", 2,             "PerError:=", 1,             "PerRefine:=", 30,             "AutoIncreaseSolutionOrder:=", false,             "SolutionOrder:=", "Normal"),         Array("NAME:DC",             "Residual:=", 1E-005,             "SolveResOnly:=", false,             Array("NAME:Cond",                 "MaxPass:=", 10, </pre>

```
        "MinPass:=", 1,
        "MinConvPass:=", 1,
        "PerError:=", 1,
        "PerRefine:=", 30),
    Array("NAME:Mult",
          "MaxPass:=", 1,
          "MinPass:=", 1,
          "MinConvPass:=", 1,
          "PerError:=", 1,
          "PerRefine:=", 30)),
    Array("NAME:AC",
          "MaxPass:=", 10,
          "MinPass:=", 1,
          "MinConvPass:=", 2,
          "PerError:=", 1,
          "PerRefine:=", 30))
```

## 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**

<b>Parameters</b>	Name	Type	Description
	<SetupName>	String	Name of specified setup.
	<Enable>	Boolean	Determines whether enable or disable a setup. <ul style="list-style-type: none"><li>• <b>True</b> - enable setup.</li><li>• <b>False</b> - disable setup.</li></ul>
<b>Return Value</b>	None.		

<b>Python Syntax</b>	EnableSetup(<SetupName>, <Enable>)
<b>Python Example</b>	oModule.EnableSetup ("OptimizationSetup1", True)

<b>VB Syntax</b>	EnableSetup <SetupName>, <Enable>
<b>VB Example</b>	oModule.EnableSetup "OptimizationSetup1", true

## ExportDXConfigFile

Create an xml file with the setup information for Design Xplorer

<b>UI Access</b>	Right click on the Design Xplorer setup in the project tree and choose <b>Export External Connector Addin Configuration...</b>											
<b>Parameters</b>	<table border="1"> <tr> <td>Name</td> <td>Type</td> <td>Description</td> </tr> <tr> <td>&lt;SetupName&gt;</td> <td>String</td> <td>Must be one of existing DesignExplorer setup names</td> </tr> <tr> <td>&lt;FileName&gt;</td> <td>String</td> <td>Must be a valid file path and name</td> </tr> </table>			Name	Type	Description	<SetupName>	String	Must be one of existing DesignExplorer setup names	<FileName>	String	Must be a valid file path and name
Name	Type	Description										
<SetupName>	String	Must be one of existing DesignExplorer setup names										
<FileName>	String	Must be a valid file path and name										
<b>Return Value</b>	None											

<b>Python Syntax</b>	ExportDXConfigFile (<SetupName>, <FileName>)
<b>Python Example</b>	<pre>oModule.ExportDXConfigFile ("DesignXplorerSetup1", "c:/exportdir/DXSetup1.xml")</pre>

<b>VB Syntax</b>	ExportDXConfigFile <SetupName>, <FileName>
<b>VB Example</b>	<pre>oModule.ExportDXConfigFile ("DesignXplorerSetup1", "c:/exportdir/DXSetup1.xml")</pre>

## ExportOptimetricsProfile

Export Optimetrics profile data

<b>UI Access</b>	Right click on the Optimetrics setup in the project tree and choose <b>View Analysis Result...</b> On the <b>Post Analysis Display</b> dialog box, click the <b>Profile</b> tab and click on the <b>Export</b> button.												
<b>Parameters</b>	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>&lt;SetupName&gt;</td><td>String</td><td>Must be one of the existing Parametric, Optimization, Sensitivity, Statistical or DesignXplorer setup names</td></tr><tr><td>&lt;FileName&gt;</td><td>String</td><td>Must be a valid file path and name with extension of csv, tab, dat, or txt</td></tr><tr><td>[profileNum]</td><td>String</td><td>Must be a numeric string. Optional: defaulted to last profile number. It should be a zero indexed profile number.</td></tr></tbody></table>	Name	Type	Description	<SetupName>	String	Must be one of the existing Parametric, Optimization, Sensitivity, Statistical or DesignXplorer setup names	<FileName>	String	Must be a valid file path and name with extension of csv, tab, dat, or txt	[profileNum]	String	Must be a numeric string. Optional: defaulted to last profile number. It should be a zero indexed profile number.
Name	Type	Description											
<SetupName>	String	Must be one of the existing Parametric, Optimization, Sensitivity, Statistical or DesignXplorer setup names											
<FileName>	String	Must be a valid file path and name with extension of csv, tab, dat, or txt											
[profileNum]	String	Must be a numeric string. Optional: defaulted to last profile number. It should be a zero indexed profile number.											
<b>Return Value</b>	None												

<b>Python Syntax</b>	ExportOptimetricsProfile (<SetupName>, <FileName>, [profileNum])
<b>Python Example</b>	<pre>oModule.ExportOptimetricsProfile ("StatisticalSetup1", "c:/exportdir/test.csv")</pre>

<b>VB Syntax</b>	ExportOptimetricsProfile <SetupName>, <FileName>, [profileNum]
<b>VB Example</b>	<pre>oModule.ExportOptimetricsProfile "StatisticalSetup1", "c:/exportdir/test.csv"</pre>

## ExportOptimetricsResult

Export an existing Optimization, Sensitivity, Statistical or DesignXplorer result. (Does not export Parametric results.)

<b>UI Access</b>	Right click on the desired Optimetrics setup in the project tree and choose <b>View Analysis Result...</b> On the <b>Post Analysis Display</b> dialog box, click the <b>Result</b> tab, then select <b>Table</b> view, and click on the <b>Export</b> button												
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;SetupName&gt;</td> <td>String</td> <td>Must be one of the existing Optimization, Sensitivity, Statistical, or DesignXplorer setup names</td> </tr> <tr> <td>&lt;FileName&gt;</td> <td>String</td> <td>Must be a valid file path and name with extension of csv, tab, dat, or txt..</td> </tr> <tr> <td>[useFullOutputName]</td> <td>Boolean</td> <td>Optional: defaulted to false. If set to true values will be printed with units. This parameter is ignored for Optimization and Statistical results.</td> </tr> </tbody> </table>	Name	Type	Description	<SetupName>	String	Must be one of the existing Optimization, Sensitivity, Statistical, or DesignXplorer setup names	<FileName>	String	Must be a valid file path and name with extension of csv, tab, dat, or txt..	[useFullOutputName]	Boolean	Optional: defaulted to false. If set to true values will be printed with units. This parameter is ignored for Optimization and Statistical results.
Name	Type	Description											
<SetupName>	String	Must be one of the existing Optimization, Sensitivity, Statistical, or DesignXplorer setup names											
<FileName>	String	Must be a valid file path and name with extension of csv, tab, dat, or txt..											
[useFullOutputName]	Boolean	Optional: defaulted to false. If set to true values will be printed with units. This parameter is ignored for Optimization and Statistical results.											
<b>Return Value</b>	None												

<b>Python Syntax</b>	ExportOptimetricsResult (<SetupName>, <FileName>, [useFullOutputName])
<b>Python Example</b>	<pre>oModule.ExportOptimetricsResult (</pre>

	"StatisticalSetup1", "c:/exportdir/test.csv", false)
--	------------------------------------------------------

<b>VB Syntax</b>	ExportOptimetricsResult <SetupName>, <FileName>, [useFullOutputName]
<b>VB Example</b>	oModule.ExportOptimetricsResult "StatisticalSetup1", "c:/exportdir/test.csv", false

## ExportParametricResults

Export existing Parametric results.

<b>UI Access</b>	Right click on the desired Parametric setup in the project tree and choose <b>View Analysis Result...</b> On the <b>Post Analysis Display</b> dialog box, click the <b>Result</b> tab, then select <b>Table</b> view, and click on the <b>Export</b> button.												
<b>Parameters</b>	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>&lt;SetupName&gt;</td><td>String</td><td>Must be one of the existing Parametric setup names</td></tr><tr><td>&lt;FileName&gt;</td><td>String</td><td>Must be a valid file path and name with extension of csv, tab, dat or txt</td></tr><tr><td>&lt;bOutputUnits&gt;</td><td>Boolean</td><td>If set to true, values will be printed with units</td></tr></tbody></table>	Name	Type	Description	<SetupName>	String	Must be one of the existing Parametric setup names	<FileName>	String	Must be a valid file path and name with extension of csv, tab, dat or txt	<bOutputUnits>	Boolean	If set to true, values will be printed with units
Name	Type	Description											
<SetupName>	String	Must be one of the existing Parametric setup names											
<FileName>	String	Must be a valid file path and name with extension of csv, tab, dat or txt											
<bOutputUnits>	Boolean	If set to true, values will be printed with units											
<b>Return Value</b>	None												

<b>Python Syntax</b>	ExportParametricResults (<SetupName>, <FileName>, <bOutputUnits>)
<b>Python Example</b>	oModule.ExportParametricResults ( "ParametricSetup1", "c:/exportdir/test.csv", False)

<b>VB Syntax</b>	ExportParametricResults <SetupName>, <FileName>, <bOutputUnits>
<b>VB Example</b>	<pre>oModule.ExportParametricResults "ParametricSetup1", "c:/exportdir/test.csv", false</pre>

## ExportParametricSetupTable

Exports the parametric setup table as a CSV file.

<b>UI Access</b>	Double-click parametric setup. Select <b>Table</b> tab. Click <b>Export</b> .									
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;SetupName&gt;</td> <td>String</td> <td>Name of the setup.</td> </tr> <tr> <td>&lt;filePath&gt;</td> <td>String</td> <td>Full path for file export.</td> </tr> </tbody> </table>	Name	Type	Description	<SetupName>	String	Name of the setup.	<filePath>	String	Full path for file export.
Name	Type	Description								
<SetupName>	String	Name of the setup.								
<filePath>	String	Full path for file export.								
<b>Return Value</b>	None									

<b>Python Syntax</b>	ExportParametricSetupTable (<SetupName>, <filePath>)
<b>Python Example</b>	<pre>oModule.ExportParametricSetupTable('ParametricSetup1', 'E:/Files/ParametricSetup1_ Table.csv')</pre>

<b>VB Syntax</b>	ExportParametricSetupTable <SetupName>, <filePath>
<b>VB Example</b>	<pre>obj.ExportParametricSetupTable "ParametricSetup1", "E:/Files/ParametricSetup1_ Table.csv"</pre>

## ExportRespSurfaceMinMaxTable

Exports min-max table from a response surface to a file

<b>UI Access</b>	Click on <b>Export...</b> From the <b>Response Surface</b> tab of the Design of Experiments <b>Post Analysis Display</b> dialog.									
<b>Parameters</b>	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>&lt;DOEName&gt;</td><td>String</td><td>Name of the Design of Experiments (DOE) setup.</td></tr><tr><td>&lt;FileName&gt;</td><td>String</td><td>Output file name with path.</td></tr></tbody></table>	Name	Type	Description	<DOEName>	String	Name of the Design of Experiments (DOE) setup.	<FileName>	String	Output file name with path.
Name	Type	Description								
<DOEName>	String	Name of the Design of Experiments (DOE) setup.								
<FileName>	String	Output file name with path.								
<b>Return Value</b>	None.									

<b>Python Syntax</b>	ExportRespSurfaceMinMaxTable(<DOEName>, <FileName>)
<b>Python Example</b>	<pre>oModule.ExportRespSurfaceMinMaxTable ("DesignOfExperimentsSetup1", "C:/temp/DesignOfExperimentsSetup1_Min-Max_Search.csv")</pre>

<b>VB Syntax</b>	ExportRespSurfaceMinMaxTable <DOEName>, <FileName>
<b>VB Example</b>	<pre>oModule.ExportRespSurfaceMinMaxTable _ "DesignOfExperimentsSetup1", _ "C:/temp/DesignOfExperimentsSetup1_Min-Max_Search.csv"</pre>

## ExportRespSurfaceRefinePoints

Exports refinement points table to a file

<b>UI Access</b>	From the <b>Response Surface</b> tab of the Design of Experiments <b>Post Analysis Display</b> dialog box, select <b>Refinement Points</b> option under <b>View</b> , Click on <b>Export....</b>									
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;DOEName&gt;</td> <td>String</td> <td>Name of the Design of Experiments (DOE) setup.</td> </tr> <tr> <td>&lt;FileName&gt;</td> <td>String</td> <td>Output file name with path.</td> </tr> </tbody> </table>	Name	Type	Description	<DOEName>	String	Name of the Design of Experiments (DOE) setup.	<FileName>	String	Output file name with path.
Name	Type	Description								
<DOEName>	String	Name of the Design of Experiments (DOE) setup.								
<FileName>	String	Output file name with path.								
<b>Return Value</b>	None.									

<b>Python Syntax</b>	ExportRespSurfaceRefinePoints(<DOEName>, <FileName>)
<b>Python Example</b>	<pre>oModule.ExportRespSurfaceRefinePoints ("DesignOfExperimentsSetup1", "C:/temp/DesignOfExperimentsSetup1_Refine_Points.csv")</pre>

<b>VB Syntax</b>	ExportRespSurfaceRefinePoints <DOEName>, <FileName>
<b>VB Example</b>	<pre>oModule.ExportRespSurfaceRefinePoints_ "DesignOfExperimentsSetup1", _ "C:/temp/DesignOfExperimentsSetup1_Refine_Points.csv"</pre>

## ExportRespSurfaceResponsePoints

Exports response points table to a file

<b>UI Access</b>	From the <b>Response Surface</b> tab of the Design of Experiments <b>Post Analysis Display</b> dialog box, select <b>Response Points</b> option under <b>View</b> , Click on <b>Export....</b>						
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;DOEName&gt;</td> <td>String</td> <td>Name of the Design of Experiments (DOE) setup.</td> </tr> </tbody> </table>	Name	Type	Description	<DOEName>	String	Name of the Design of Experiments (DOE) setup.
Name	Type	Description					
<DOEName>	String	Name of the Design of Experiments (DOE) setup.					

	<code>&lt;FileName&gt;</code>	String	Output file name with path.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	<code>ExportRespSurfaceResponsePoints (&lt;DOEName&gt;, &lt;FileName&gt;)</code>
<b>Python Example</b>	<code>oModule.ExportRespSurfaceResponsePoints ("DesignOfExperimentsSetup1", "C:/temp/DesignOfExperimentsSetup1_Response_Points.csv")</code>

<b>VB Syntax</b>	<code>ExportRespSurfaceResponsePoints &lt;DOEName&gt;, &lt;FileName&gt;</code>
<b>VB Example</b>	<code>oModule.ExportRespSurfaceResponsePoints_ "DesignOfExperimentsSetup1", _ "C:/temp/DesignOfExperimentsSetup1_Response_Points.csv"</code>

## ExportRespSurfaceVerificationPoints

Exports verification points table to a file

<b>UI Access</b>	From the <b>Response Surface</b> tab of the Design of Experiments <b>Post Analysis Display</b> dialog box, select <b>Verification Points</b> option under <b>View</b> , Click on <b>Export....</b>											
<b>Parameters</b>	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td><code>&lt;DOEName&gt;</code></td><td>String</td><td>Name of the Design of Experiments (DOE) setup.</td></tr><tr><td><code>&lt;FileName&gt;</code></td><td>String</td><td>Output file name with path.</td></tr></tbody></table>			Name	Type	Description	<code>&lt;DOEName&gt;</code>	String	Name of the Design of Experiments (DOE) setup.	<code>&lt;FileName&gt;</code>	String	Output file name with path.
Name	Type	Description										
<code>&lt;DOEName&gt;</code>	String	Name of the Design of Experiments (DOE) setup.										
<code>&lt;FileName&gt;</code>	String	Output file name with path.										
<b>Return Value</b>	None.											

<b>Python Syntax</b>	ExportRespSurfaceVerificationPoints (<DOEName>, <FileName>)
<b>Python Example</b>	<pre>oModule.ExportRespSurfaceVerificationPoints("DesignOfExperimentsSetup1", "C:/temp/DesignOfExperimentsSetup1_Veri_Points.csv")</pre>

<b>VB Syntax</b>	ExportRespSurfaceVerificationPoints <DOEName>, <FileName>
<b>VB Example</b>	<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.

<b>UI Access</b>	NA		
<b>Parameters</b>	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".
<b>Return Value</b>	Array of setup names.		

<b>Python Syntax</b>	GetChildNames()
<b>Python Example</b>	<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

<b>UI Access</b>	NA		
<b>Parameters</b>	Name	Type	Description

	Setup Name	text string	A optimetrics setup name, names returned by the GetChildNames().
<b>Return Value</b>	A script object for the setup See discussion of Optimetrics Setup Objects in <a href="#">Object Script Property Function Summary</a> .		

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 name and ID of the active design. If the design is a sub-circuit, returns the parent design also.

<b>UI Access</b>	N/A
<b>Parameters</b>	None.
<b>Return Value</b>	String indicating the name and ID of the active design and parent design (if the active design is a sub-circuit).

<b>Python Syntax</b>	<code>GetName()</code>
<b>Python Example</b>	<code>design_name = oDesign.GetName()</code>

<b>VB Syntax</b>	<code>GetName</code>
<b>VB Example</b>	<code>design_name = oDesign.GetName</code>

## GetObjPath [Design]

Obtains the path to the design.

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

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

<b>Python Example</b>	<code>oDesign.GetObjPath()</code>
-----------------------	-----------------------------------

<b>VB Syntax</b>	<code>GetObjPath</code>
<b>VB Example</b>	<code>oDesign.GetObjPath</code>

## GetOptimetricResult

Returns an Optimetric calculation. The specific calculation is determined by the setup.

<b>UI Access</b>	N/A		
<b>Parameters</b>	Name	Type	Description
	<code>&lt;SetupName&gt;</code>	String	Optimetrics setup name.
	<code>&lt;vars&gt;</code>	Array	Array containing string variable names. Use the <b>Sweep Definitions</b> tab in the UI or the <code>&lt;SweepDefs&gt;</code> parameter in the <a href="#">InsertSetup</a> script to determine appropriate inputs.
	<code>&lt;values&gt;</code>	Array	<i>Optional.</i> Array containing string values. When multiple variables and values are provided, the order must be the same in both the <code>&lt;vars&gt;</code> and <code>&lt;values&gt;</code> arrays. The first variable is paired with the first value, the second variable is paired with the second value, and so on.
<b>Return Value</b>	Calculation result. If the setup contains more than one calculation, the output will be an array of values.		

<b>Python Syntax</b>	<code>GetOptimetricResult(&lt;SetupName&gt;, &lt;vars&gt;, &lt;values&gt;)</code>
<b>Python Example</b>	<code>oModule.GetOptimetricResult('ParametricSetup1', ['AR', 'Re'], ['4.64', '6e+04'])</code>

<b>VB Syntax</b>	GetOptimetricResult <SetupName>, <vars>, <values>
<b>VB Example</b>	<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.

<b>Python Syntax</b>	GetPropNames ()
<b>Python Example</b>	<pre>oOptModule.GetPropNames() oOptModule.GetPropNames(True) oOptModule.GetPropNames(False)</pre>

## GetPropValue [Optimetrics]

Returns the property value for a setup property.

<b>UI Access</b>	NA						
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>property-path</td> <td></td> <td>a child object's property path. See <a href="#">property path discussion here</a>.</td> </tr> </tbody> </table>	Name	Type	Description	property-path		a child object's property path. See <a href="#">property path discussion here</a> .
Name	Type	Description					
property-path		a child object's property path. See <a href="#">property path discussion here</a> .					
<b>Return Value</b>	Returns the value of an setup property.						

<b>Python Syntax</b>	GetPropValue(propPath)
<b>Python Example</b>	<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>

<b>VB Syntax</b>	GetPropValue()
<b>VB Example</b>	<pre>GetPropValue("Optimetrics/OptimizationSetup1/Enabled")  Returns True if Enabled, or False if disabled.</pre>

## GetSetupNames [Optimetrics]

Gets a list of Optimetrics setup names

<b>UI Access</b>	NA						
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>None</td> <td></td> <td></td> </tr> </tbody> </table>	Name	Type	Description	None		
Name	Type	Description					
None							
<b>Return Value</b>	IAnsoftCollectionObj – a collection of Optimetrics setup names						

<b>Python Syntax</b>	GetSetupNames()
<b>Python Example</b>	<pre>oModule = oDesign.GetModule("Optimetrics") setupNames = oModule.GetSetupNames()</pre>

<b>VB Syntax</b>	GetSetupNames()
<b>VB Example</b>	<pre>Set oModule_opt = oDesign.GetModule("Optimetrics") Set opt_setup_list = oModule.GetSetupNames() numsetups = setupnames.Count</pre>

## GetSetupNamesByType [Optimetrics]

Gets a list of Optimetrics setup names by type.

<b>UI Access</b>	NA						
<b>Parameters</b>	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>&lt;Optimetrics type&gt;</td><td>String</td><td>Examples: parametric, optimization, statistical, sensitivity</td></tr></tbody></table>	Name	Type	Description	<Optimetrics type>	String	Examples: parametric, optimization, statistical, sensitivity
Name	Type	Description					
<Optimetrics type>	String	Examples: parametric, optimization, statistical, sensitivity					
<b>Return Value</b>	Array of Optimetrics setup names of the given type.						

<b>Python Syntax</b>	GetSetupNamesByType (<Optimetrics type>)
----------------------	------------------------------------------

<b>Python Example</b>	<pre>for name in oModule.GetSetupNamesByType("optimization")     AddInfoMessage(str(name))</pre>
-----------------------	--------------------------------------------------------------------------------------------------

<b>VB Syntax</b>	<b>GetSetupNamesByType &lt;Optimetrics type&gt;</b>
<b>VB Example</b>	For each name in oModule.GetSetupNamesByType("optimization")     Msgbox name     Next

## ImportSetup

Import an Optimetric setup from a file.

<b>UI Access</b>	NA		
<b>Parameters</b>	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.

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

<b>Python Syntax</b>	ImportSetup (<SetupTypeName>, <SetupInfo>)
<b>Python Example</b>	<pre>oModule.ImportSetup ("OptiStatistical", ["NAME:StatisticalSetup1", "c:/importdir/mySetupInfoFile"])</pre>

<b>VB Syntax</b>	ImportSetup <SetupTypeName>, <SetupInfo>
<b>VB Example</b>	<pre>oModule.ImportSetup "OptiStatistical", Array("NAME:StatisticalSetup1", "c:/importdir/mySetupInfoFile")</pre>

## InsertSetup

Adds a new solution setup.

<b>UI Access</b>	[product] > <b>Analysis Setup</b> > <b>Add Solution Setup</b> .		
<b>Parameters</b>	Name <SetupType>	Type String	Description Type of setup to be inserted.
	<Attributes>	Array	Structured array.

		<pre> Array("NAME:&lt;SetupName&gt;", &lt;NamedParameters&gt;  &lt;Named Parameters&gt;  "AdaptMultipleFreqs:=", &lt;boolean&gt;, "Frequency:=", &lt;string&gt;, "MaxDeltaS:=", &lt;float&gt;, "PortsOnly:=", &lt;boolean&gt;, "UseMatrixConv:=", &lt;boolean&gt;, ### If UseMatrixCov is True, ### [ "NAME:Matrix Convergence", &lt;Convergence Array&gt; ], MaximumPasses:=", &lt;integer&gt;, MinimumPasses:=", &lt;integer&gt;, MinimumConvergedPasses:=", &lt;integer&gt;, PercentRefinement:=", &lt;integer&gt;, .IsEnabled:=", &lt;boolean&gt;, BasisOrder:=", &lt;integer&gt;, DoLambdaRefine:=", &lt;boolean&gt;, DoMaterialLambda:=", &lt;boolean&gt;, SetLambdaTarget:=", &lt;boolean&gt;, Target:=", &lt;float&gt;, PortAccuracy:=", &lt;integer&gt;, SaveRadFieldsOnly:=", &lt;boolean&gt;, SaveAnyFields:=", &lt;boolean&gt;, ListsForFields:=", &lt;array of string&gt;, IESolverType:=", &lt;string&gt;, LambdaTargetForIESolver:=", &lt;float&gt;, UseDefaultLambdaTgtForIESolver:=", &lt;boolean&gt; </pre>
<b>Return Value</b>	None.	

Python Syntax	InsertSetup(<SetupType>, <Attributes>)
Python Example	<pre>oModule.InsertSetup ("HfssDrivenAuto", [ "NAME:Setup1",   "IsEnabled:=", True,   "AutoSolverSetting:=", "Balanced",   [ "NAME:Sweeps",     [ "NAME:Sweep",       "RangeType:=", "LinearStep",       "RangeStart:=", "1GHz",       "RangeEnd:=", "10GHz",       "RangeStep:=", "1GHz"     ]   ],   "SaveRadFieldsOnly:=", False,   "SaveAnyFields:=", True,   "Type:=", "Discrete" ] )</pre>

```
oModule.InsertSetup("HfssDrivenAuto",
[
    "NAME:Setup2",
    "SolveType:=" , "Auto",
    "IsEnabled:=" , True,
[
    "NAME:MeshLink",
    "ImportMesh:=" , False
],
"AutoSolverSetting:=" , "Balanced",
[
    "NAME:Sweeps",
[
        "NAME:Sweep",
        "RangeType:=" , "LinearCount",
        "RangeStart:=" , "0GHz",
        "RangeEnd:=" , "10GHz",
        "RangeCount:=" , 501
]
],
"SaveRadFieldsOnly:=" , False,
```

```
        "SaveAnyFields:=" , True,
        "InfiniteSphereSetup:=" , "Infinite Sphere1",
        "ListsForFields:=" , ["Objectlist2"],
        "Type:=" , "Interpolating"
    )

oModule.InsertSetup("HfssDriven",
["NAME:Setup3",
    "AdaptMultipleFreqs:=", False,
    "Frequency:=", "5GHz",
    "MaxDeltaS:=", 0.02,
    "PortsOnly:=", False,
    "UseMatrixConv:=", False,
    "MaximumPasses:=", 6,
    "MinimumPasses:=", 1,
    "MinimumConvergedPasses:=", 1,
    "PercentRefinement:=", 30,
    ".IsEnabled:=", True,
    "BasisOrder:=", 1,
    "DoLambdaRefine:=", True,
```

```
"DoMaterialLambda:=", True,
"SetLambdaTarget:=", False,
"Target:=", 0.3333,
"UseMaxTetIncrease:=", False,
"PortAccuracy:=", 2,
"UseABCOnPort:=", False,
"SetPortMinMaxTri:=", False,
"UseDomains:=", True,
"UseIterativeSolver:=", False,
"IterativeResidual:=", 1E-06,
"DDMSolverResidual:=", 0.0001,
"EnhancedLowFreqAccuracy:=", True,
"SaveRadFieldsOnly:=", False,
"SaveAnyFields:=", True,
"IESolverType:=", "Auto",
"LambdaTargetForIESolver:=", 0.15,
"UseDefaultLambdaTgtForIESolver:=", True,
"SkipIERegionSolveDuringAdaptivePasses:=", True
"RayDensityPerWavelength:=", 4,
"MaxNumberOfBounces:=" , 5,
"InfiniteSphereSetup:=" , "Infinite Sphere1",
```

```
"SkipSBRSoIveDuringAdaptivePasses:=", True,
"PTDUTDSimulationSettings:=", "PTD Correction + UTD Rays",
"PTDEdgeDensity:=" , 20
]

### An HFSS with Hybrid and Arrays setup
import ScriptEnv
ScriptEnv.Initialize("Ansoft.ElectronicsDesktop")
oDesktop.RestoreWindow()
oProject = oDesktop.SetActiveProject("Project1")
oDesign = oProject.SetActiveDesign("HFSSDesign1")
oModule = oDesign.GetModule("AnalysisSetup")
oModule.InsertSetup("HfssDriven",
[
    "NAME:Setup1",
    "SolveType:=" , "Single",
    "Frequency:=" , "5GHz",
    "MaxDeltaE:=" , 0.1,
    "MaximumPasses:=" , 6,
    "MinimumPasses:=" , 1,
```

```
"MinimumConvergedPasses:=", 1,
"PercentRefinement:="      , 30,
".IsEnabled:="            , True,
[
    "NAME:MeshLink",
    "ImportMesh:="           , False
],
"BasisOrder:="             , 1,
"DoLambdaRefine:="          , True,
"DoMaterialLambda:="         , True,
"SetLambdaTarget:="          , False,
"Target:="                  , 0.3333,
"UseMaxTetIncrease:="       , False,
"DrivenSolverType:="         , "Direct Solver",
"EnhancedLowFreqAccuracy:=" , False,
"SaveRadFieldsOnly:="        , False,
"SaveAnyFields:="            , True,
"IESolverType:="             , "Auto",
"LambdaTargetForIESolver:="  , 0.15,
"UseDefaultLambdaTgtForIESolver:=" , True,
"IE Solver Accuracy:="     , "Balanced",
```

```
        "RayDensityPerWavelength:=", 4,
        "MaxNumberOfBounces:=", 5,
        "RadiationSetup:=", "Infinite Sphere1",
        "PTDUTDSimulationSettings:=", "None",
        "EnableSBRSelfCoupling:=", False,
        "UseSBRAvOptionsGOBlockage:=", False,
        "UseSBRAvOptionsWedges:=", False,
        "SkipSBRSoLveDuringAdaptivePasses:=", False,
        "UseSBREnhancedRadiatedPowerCalculation:=", True
    ])

### An SBR+ Setup with Fast Frequency Looping
oModule.InsertSetup("HfssDriven",
[
    "NAME:Setup1",
    "IsEnabled:=", True,
    [
        "NAME:MeshLink",
        "ImportMesh:=", False
    ],

```

```
"IsSbrRangeDoppler:=" , False,
"RayDensityPerWavelength:=" , 4,
"MaxNumberOfBounces:=" , 5,
"IsMonostaticRCS:=" , True,
"EnableCWRays:=" , False,
"RadiationSetup:=" , "",
"PTDUTDSimulationSettings:=" , "None",
"FastFrequencyLooping:=" , True,
[
    "NAME:Sweeps",
    [
        "NAME:Sweep",
        "RangeType:=" , "LinearStep",
        "RangeStart:=" , "1GHz",
        "RangeEnd:=" , "10GHz",
        "RangeStep:=" , "1GHz"
    ],
    "ComputeFarFields:=" , True
])
```

```
### SBR+ Setup with Enhanced Radiated Power Calculation
oModule.InsertSetup("HfssDriven",
[
    "NAME:Setup2",
    "IsEnabled:=" , True,
    [
        "NAME:MeshLink",
        "ImportMesh:=" , False
    ],
    "IsSbrRangeDoppler:=" , False,
    "RayDensityPerWavelength:=" , 4,
    "MaxNumberOfBounces:=" , 5,
    "EnableCWRays:=" , False,
    "RadiationSetup:=" , "Infinite Sphere1",
    "PTDUTDSimulationSettings:=", "None",
    "FastFrequencyLooping:=" , False,
    "UseSBRAdvOptionsGOBlockage:=" , False,
    "UseSBRAdvOptionsWedges:=" , False,
    [
        "NAME:Sweeps",
```

```
[  
    "NAME:Sweep",  
    "RangeType:=" , "LinearStep",  
    "RangeStart:=" , "1GHz",  
    "RangeEnd:=" , "10GHz",  
    "RangeStep:=" , "1GHz"  
]  
,  
"ComputeFarFields:=" , True,  
"UseSBREnhancedRadiatedPowerCalculation:=" , True,  
"IsGOBlockageEnabled:=" , False,  
"GOBlockageSurfaceSelfBlock:=" , False  
])  
  
### Insert RF Discharge Setup for HFSS  
import ScriptEnv  
ScriptEnv.Initialize("Ansoft.ElectronicsDesktop")  
oDesktop.RestoreWindow()  
oProject = oDesktop.SetActiveProject("coaxbend_discharge_r212")  
oDesign = oProject.SetActiveDesign("HFSSDesign60degBendTeflon")  
oModule = oDesign.GetModule("AnalysisSetup")
```

```
oModule.InsertSetup("HfssRFDischarge",
[
    "NAME:RFDischarge1",
    "Enabled:=" , True,
    [
        "NAME:MeshLink",
        "ImportMesh:=" , True,
        "Project:=" , "This Project*",
        "Product:=" , "HFSS",
        "Design:=" , "This Design*",
        "Soln:=" , "Setup1 : Sweep",
        [
            "NAME:Params",
            "bend_angle:=" , "bend_angle"
        ],
        "ForceSourceToSolve:=" , False,
        "PreservePartnerSoln:=" , False,
        "PathRelativeTo:=" , "SourceProduct",
        "ApplyMeshOp:=" , True
    ],
]
```

```
[  
    "NAME:Excitations",  
    [  
        "NAME:1:1",  
        "Magnitude:=" , "1",  
        "Phase:=" , "0deg"  
    ],  
    [  
        "NAME:2:1",  
        "Magnitude:=" , "0",  
        "Phase:=" , "0deg"  
    ],  
    [  
        "NAME:Frequencies",  
        "10GHz"  
    ],  
    ["Minimum Power:=" , "0.01",  
     "Maximum Power:=" , "1000000",  
     "Minimum Pressure:=" , "100pascal",  
     "Maximum Pressure:=" , "101325pascal",  
    ]]
```

```
        "Postproc Sampling:=" , 500,
        "Temperature:=" , "Ocel",
        "BuiltInGas:=" , "Argon"
        "Save Electron Density:=", True,
        "Is Pulsed Signal:=" , True,
        "Duty Cycle:=" , 50
    ])

#### Setup with UseMatrixConv is True and Matrix Convergence Entry Specified as All
###

oModule.InsertSetup("Setup1",
[
    "NAME:Setup1",
    "SolveType:=" , "Single",
    "Frequency:=" , "10GHz",
    "MaxDeltaS:=" , 0.02,
    "UseMatrixConv:=" , True,
    [
        "NAME:Matrix Convergence",
        "AllEntries:=" , True,
        "MagLimit:=" , "0.01",
    ]
])
```

```
    "PhaseLimit:="           , "2deg",
    "MagMinThreshold:="     , 0.01
],
"MaximumPasses:="        , 12,
"MinimumPasses:="        , 1,
"MinimumConvergedPasses:=", 1,
"PercentRefinement:="    , 30,
".IsEnabled:="           , True,
[
    "NAME:MeshLink",
    "ImportMesh:="          , False
],
"BasisOrder:="            , 1,
"DoLambdaRefine:="        , True,
"DoMaterialLambda:="      , True,
"SetLambdaTarget:="       , False,
"Target:="                , 0.3333,
"UseMaxTetIncrease:="    , False,
"PortAccuracy:="          , 2,
"UseABCOnPort:="          , False,
"SetPortMinMaxTri:="      , False,
```

```
        "DrivenSolverType:="      , "Direct Solver",
        "EnhancedLowFreqAccuracy:=", False,
        "SaveRadFieldsOnly:="     , False,
        "SaveAnyFields:="         , True,
        "IESolverType:="          , "ACA",
        "LambdaTargetForIESolver:=", 0.15,
        "UseDefaultLambdaTgtForIESolver:=", True,
        "IE Solver Accuracy:="   , "Balanced",
        "InfiniteSphereSetup:="   , ""

    ])

### Setup with UseMatrixConv as Entries and MeshLink
oModule = oDesign.GetModule("AnalysisSetup")
oModule.InsertSetup("Setup1",
[
    "NAME:Setup1",
    "SolveType:="            , "Single",
    "Frequency:="            , "10GHz",
    "MaxDeltaS:="             , 0.02,
    "UseMatrixConv:="         , True,
    [

```

```
"NAME:Matrix Convergence",
[

    "NAME:Entries",
    [
        "NAME:Entry",
        "Port1:=" , "A[1,1]P1",
        "Port2:=" , "A[1,1]P1",
        "MagLimit:=" , "0.011",
        "PhaseLimit:=" , "5deg"
    ],
    [
        "NAME:Entry",
        "Port1:=" , "A[4,4]P2",
        "Port2:=" , "A[1,1]P2",
        "MagLimit:=" , "0.041",
        "PhaseLimit:=" , "5deg"
    ]
],
"MaximumPasses:=" , 1,
"MinimumPasses:=" , 1,
```

```
"MinimumConvergedPasses:=", 1,  
"PercentRefinement:="      , 30,  
".IsEnabled:="            , True,  
[  
    "NAME:MeshLink",  
    "ImportMesh:="          , True,  
    "Project:="             , "This Project*",  
    "Product:="              , "",  
    "Design:="               , "unit",  
    "Soln:="                , "Setup1 : LastAdaptive",  
    [  
        "NAME:Params",  
        "Airbox_dist:="       , "9.9931mm",  
        "Scan_Theta:="         , "0deg",  
        "Scan_phi:="           , "0deg",  
        "VirtualObject_dist:=" , "2.9979mm",  
        "coax_inner_rad:="     , "0.125mm",  
        "coax_outer_rad:="     , "0.425mm",  
        "cut_off:="             , "1.7mm",  
        "feedX:="               , "-0.4mm",
```

```
        "feedY:="           , "2mm",
        "feed_length:="    , "1mm",
        "patchX:="          , "9.65mm",
        "rot:="             , "45deg",
        "subH:="             , "0.8mm",
        "subX:="             , "15mm",
        "subY:="             , "15mm"
    ],
    "ForceSourceToSolve:=" , False,
    "PreservePartnerSoln:=" , False,
    "PathRelativeTo:="      , "TargetProject",
    "ApplyMeshOp:="         , True,
    "AdaptPort:="           , False
],
"BasisOrder:="           , 1,
"DoLambdaRefine:="        , False,
"DoMaterialLambda:="       , True,
"SetLambdaTarget:="        , False,
"Target:="                 , 0.3333,
"UseMaxTetIncrease:="     , False,
"PortAccuracy:="           , 2,
```

```
"UseABCOnPort:=" , False,
"SetPortMinMaxTri:=" , False,
"DrivenSolverType:=" , "Domain Decomposition",
"IterativeResidual:=" , 0.0001,
"DDMSolverResidual:=" , 0.0001,
"EnhancedLowFreqAccuracy:=", False,
"SaveRadFieldsOnly:=" , False,
"SaveAnyFields:=" , True,
"IESolverType:=" , "Auto",
"LambdaTargetForIESolver:=", 0.15,
"UseDefaultLambdaTgtForIESolver:=", True,
"IE Solver Accuracy:=" , "Balanced",
"InfiniteSphereSetup:=" , ""

])
### Example from design with UseMatrixConv True for AllDiagEntries
"UseMatrixConv:=" , True,
[
    "NAME:Matrix Convergence",
    "AllDiagEntries:=" , True,
    "DiagonalMag:=" , "0.03",
```

```
"DiagonalPhase:=" , "4rad",
"MagMinThreshold:=" , 0.01,
"AllOffDiagEntries:=" , True,
"OffDiagonalMag:=" , "0.05",
"OffDiagonalPhase:=" , "6deg",
"MagMinThreshold:=" , 0.01
],
### Example from Modal design with UseMatrixConv True for Entries
"UseMatrixConv:=" , True,
[
    "NAME:Matrix Convergence",
    [
        "NAME:Entries",
        [
            "NAME:Entry",
            "Port1:=" , "Port1",
            "Port2:=" , "Port1",
            "MagLimit:=" , "0.02",
            "PhaseLimit:=" , "5deg"
        ],
        [
    ]]
```

```
        "NAME:Entry",
        "Port1:="           , "Port3",
        "Port2:="           , "Port2",
        "MagLimit:="        , "0.03",
        "PhaseLimit:="      , "5deg"
    ]
]
],  
  
#### Example from Design with Periodic Ports and UseMatrixConv is True
"UseMatrixConv:="      , True,
[
    "NAME:Matrix Convergence",
    [
        "NAME:Entries",
        [
            "NAME:Entry",
            "Port1:="           , "Incident_Port1",
            "Port2:="           , "Incident_Port1",
            "MagLimit:="        , "0.11",
```

```
        "PhaseLimit:="           , "5deg"
    ],
[
    "NAME:Entry",
    "Port1:="                 , "Incident_Port2",
    "Port2:="                 , "Incident_Port2",
    "MagLimit:="              , "0.22",
    "PhaseLimit:="            , "5deg"
]
],
#### Example Terminal Design with UseMatrixConv = True and Expression Cache
oProject = oDesktop.SetActiveProject("term-2x2_1_parasolid")
oDesign = oProject.SetActiveDesign("unit")
oModule = oDesign.GetModule("AnalysisSetup")
oModule.InsertSetup("Setup1",
[
    "NAME:Setup1",
    "SolveType:="             , "Single",
    "Frequency:="             , "10GHz",
    "MaxDeltaS:="              , 0.02,
```

```
"UseMatrixConv:="      , True,  
[  
    "NAME:Matrix Convergence",  
    [  
        "NAME:Entries",  
        [  
            "NAME:Entry",  
            "Port1:="      , "P2",  
            "Port2:="      , "P1",  
            "MagLimit:="   , "0.05",  
            "PhaseLimit:=" , "10deg"  
        ]  
    ]  
],  
"MaximumPasses:="     , 15,  
"MinimumPasses:="     , 1,  
"MinimumConvergedPasses:=" , 1,  
"PercentRefinement:="  , 30,  
"IsEnabled:="          , True,  
[
```

```
        "NAME:MeshLink",
        "ImportMesh:="           , False
    ],
    "BasisOrder:="           , 1,
    "DoLambdaRefine:="       , True,
    "DoMaterialLambda:="     , True,
    "SetLambdaTarget:="      , False,
    "Target:="               , 0.3333,
    "UseMaxTetIncrease:="   , False,
    "PortAccuracy:="         , 2,
    "UseABCOnPort:="         , False,
    "SetPortMinMaxTri:="     , False,
    "DrivenSolverType:="     , "Iterative Solver",
    "IterativeResidual:="    , 0.0001,
    "DDMSolverResidual:="   , 0.0001,
    "EnhancedLowFreqAccuracy:=", False,
    "SaveRadFieldsOnly:="    , False,
    "SaveAnyFields:="        , True,
[
    "NAME:ExpressionCache",
[

```

```
        "NAME:CacheItem",
        "Title:="                  , "dB_AxialRatioValue_1",
        "Expression:="             , "dB(AxialRatioValue)",
        "Intrinsics:="              , "Phi='0deg'\ Theta='0deg\'",
        "ReportType:="              , "Far Fields",
        [
            "NAME:ExpressionContext",
            "Context:="                , "Infinite Spherel"
        ]
    ],
    [
        "NAME:CacheItem",
        "Title:="                  , "dB_St_Diff1_Diff1__1",
        "Expression:="             , "dB(St(Diff1,Diff1))",
        "Intrinsics:="              , "",
        "ReportType:="              , "Terminal Solution Data",
        [
            "NAME:ExpressionContext",
            "Diff:="                   , "Differential Pairs"
        ]
    ]
]
```

```
        ] ,  
        [  
            "NAME:CacheItem",  
            "Title:=" , "dB_St_P1_P1__1",  
            "Expression:=" , "dB(St(P1,P1))",  
            "Intrinsics:=" , "",  
            "ReportType:=" , "Terminal Solution Data",  
            [  
                "NAME:ExpressionContext",  
                "Diff:=" , "Terminals"  
            ]  
        ]  
    ],  
    "CacheSaveKind:=" , "Delta",  
    "ConstantDelta:=" , "0s",  
    "IESolverType:=" , "Auto",  
    "LambdaTargetForIESolver:=" , 0.15,  
    "UseDefaultLambdaTgtForIESolver:=" , True,  
    "IE Solver Accuracy:=" , "Balanced",  
    "InfiniteSphereSetup:=" , ""  
])
```

VB Syntax	InsertSetup < <i>SetupType</i> >, < <i>Attributes</i> >
VB Example:  An HFSS Driven project solution with Auto Solution:  oModule.InsertSetup "HfssDrivenAuto", Array("NAME:Setup1", "IsEnabled:=", true, _ "AutoSolverSetting:=", "Balanced", _ Array("NAME:Sweeps", Array("NAME:Sweep", "RangeType:=", "LinearCount", _ "RangeStart:=", "1GHz", "RangeEnd:=", "10GHz", "RangeCount:=", "451")), _ "SaveRadFieldsOnly:=", false, "SaveAnyFields:=", false, "Type:=", "Interpolating")  A Driven solution type with a mesh link. References to dependent solve in old scripts are converted to mesh link form.  oModule.InsertSetup "HfssDriven", Array("NAME:Setup1", "Frequency:=", "1GHz", "MaxDeltaE:=", 0.1, "MaximumPasses:=", 6, "MinimumPasses:=", 1, "MinimumConvergedPasses:=", 1, "PercentRefinement:=", 30, "IsEnabled:=", true,	

```
Array("NAME:MeshLink",
"Project:=", "Tee.aedt",
"Design:=", "TeeModel",
"Soln:=", "Setup1 : LastAdaptive",
Array("NAME:Params", "offset:=", "0in"),
"ForceSourceToSolve:=", false,
"PreservePartnerSoln:=", false,
"PathRelativeTo:=", "SourceProduct",
"ApplyMeshOp:=", true),
"BasisOrder:=", 1,
"UseIterativeSolver:=", false,
"DoLambdaRefine:=", false,
"DoMaterialLambda:=", true,
"SetLambdaTarget:=", false,
"Target:=", 0.3333,
"UseMaxTetIncrease:=", false,
"MaxTetIncrease:=", 1000000,
"EnableSolverDomains:=", false,
"ThermalFeedback:=", false,
"UsingConstantDelta:=", 0,
"ConstantDelta:=", "0s",
```

```
"NumberSolveSteps:=", 1)

A Driven solution type with ports:

oModule.InsertSetup "HfssDriven",
Array("NAME:Setup2",
"Frequency:=", "1GHz",
"PortsOnly:=", false,
"MaxDeltaS:=", 0.02,
"UseMatrixConv:=", false,
"MaximumPasses:=", 6,
"MinimumPasses:=", 1,
"MinimumConvergedPasses:=", 1,
"PercentRefinement:=", 30,
".IsEnabled:=", true,
"BasisOrder:=", 1,
"UseIterativeSolver:=", true,
"IterativeResidual:=", 0.0001,
"DoLambdaRefine:=", true,
"DoMaterialLambda:=", false,
"SetLambdaTarget:=", false,
"Target:=", 0.3333,
```

```
"UseMaxTetIncrease:=", false,
"MaxTetIncrease:=", 1000000,
"PortAccuracy:=", 2,
"UseABCOnPort:=", true,
"SetPortMinMaxTri:=", false,
"EnableSolverDomains:=", false,
"ThermalFeedback:=", false,
"UsingConstantDelta:=", 0,
"ConstantDelta:=", "0s",
"NumberSolveSteps:=", 1)

An Eigenmode solution type:

oModule.InsertSetup "HfssEigen",
Array("NAME:Setup2",
"MinimumFrequency:=", "1.77347GHz",
"NumModes:=", 1,
"MaxDeltaFreq:=", 10,
"ConvergeOnRealFreq:=", true,
"MaximumPasses:=", 3,
"MinimumPasses:=", 1,
"MinimumConvergedPasses:=", 1,
"PercentRefinement:=", 30,
```

```
".IsEnabled:=", true,
"BasisOrder:=", 1,
"UseIterativeSolver:=", false,
"DoLambdaRefine:=", true,
"DoMaterialLambda:=", true,
"SetLambdaTarget:=", false,
"Target:=", 0.2,
"UseMaxTetIncrease:=", false,
"MaxTetIncrease:=", 1000000,
"UsingConstantDelta:=", 0,
"ConstantDelta:=", "0s",
"NumberSolveSteps:=", 1)

A Driven solution type with ports and matrix convergence:
oModule.InsertSetup "HfssDriven",
Array("NAME:Setup2",
"Frequency:=", "1GHz",
"PortsOnly:=", false,
"MaxDeltaS:=", 0.02,
"UseMatrixConv:=", true,
Array("NAME:ConvergenceMatrix",
```

```
"AllDiagEntries:=", true,
"MagMinThreshold:=", 0.01,
"Entry:=", Array("Port1:=", "abc", "ModeNum1:=", 1)),
"MaximumPasses:=", 6,
"MinimumPasses:=", 1,
"MinimumConvergedPasses:=", 1,
"PercentRefinement:=", 30,
".IsEnabled:=", true,
"BasisOrder:=", 1,
"UseIterativeSolver:=", false,
"DoLambdaRefine:=", true,
"DoMaterialLambda:=", true,
"SetLambdaTarget:=", false,
"Target:=", 0.3333,
"UseMaxTetIncrease:=", false,
"MaxTetIncrease:=", 1000000,
"PortAccuracy:=", 2,
"UseABCOnPort:=", true,
"SetPortMinMaxTri:=", false,
"EnableSolverDomains:=", false,
"ThermalFeedback:=", false,
```

```
"UsingConstantDelta:=", 0,
"ConstantDelta:=", "0s",
"NumberSolveSteps:=", 1)

A Driven solution type with Multi-Frequencies specified:

oModule.InsertSetup "HfssDriven", Array("NAME:Setup2", "AdaptMultipleFreqs:=", true, _
Array("NAME:MultipleAdaptiveFreqsSetup", "1.1GHz:=", Array( _
0.02), "10GHz:=", Array(0.02), "11GHz:=", Array(0.02), "12.5GHz:=", Array(0.02),
"15GHz:=", Array( _
0.02), "20GHz:=", Array(0.02)), "MaximumPasses:=", 6, "MinimumPasses:=", 1, "Min-
imumConvergedPasses:=", _
1, "PercentRefinement:=", 30, ".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, "RayDensityPerWavelength:=", 4, "MaxNumberOfBounces:=", 5)

An Driven Solution type with Broadband specified:

oModule.InsertSetup "HfssDriven", Array("NAME:Setup1", "AdaptMultipleFreqs:=", true, _
Array("NAME:MultipleAdaptiveFreqsSetup", Array("NAME:Broadband", "Low:=", _
```

```
"2GHz", "High:=", "20GHz")), "MaximumPasses:=", 6, "MinimumPasses:=", 1, "Min-
imumConvergedPasses:=", _
1, "PercentRefinement:=", 30, "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, "RayDensityPerWavelength:=", 4, "MaxNumberOfBounces:=", 5)

oModule.InsertFrequencySweep "Setup1", Array("NAME:Sweep", "IsEnabled:=", true,
"RangeType:=", _
"LinearCount", "RangeStart:=", "2GHz", "RangeEnd:=", "20GHz", "RangeCount:=", _
451, "Type:=", "Interpolating", "SaveFields:=", false, "SaveRadFields:=", _
false, "InterpTolerance:=", 0.5, "InterpMaxSolns:=", 250, "InterpMinSolns:=", _
0, "InterpMinSubranges:=", 1, "ExtrapToDC:=", false, "InterpUseS:=", true, "Inter-
pUsePortImped:=", _
false, "InterpUsePropConst:=", true, "UseDerivativeConvergence:=", false, "Inter-
pDerivTolerance:=", _
0.2, "UseFullBasis:=", true, "EnforcePassivity:=", true, "PassivityErrorTolerance:=", _
-
0.0001)
```

## PasteSetup [Optimetrics]

Pastes the specified Optimetrics setup.

<b>UI Access</b>	NA						
<b>Parameters</b>	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>&lt;SetupName&gt;</td><td>String</td><td>Name of the Setup</td></tr></tbody></table>	Name	Type	Description	<SetupName>	String	Name of the Setup
Name	Type	Description					
<SetupName>	String	Name of the Setup					
<b>Return Value</b>	None						

<b>Python Syntax</b>	PasteSetup (<SetupName>)
<b>Python Example</b>	<code>oModule.PasteSetup ("OptimizationSetup1")</code>

<b>VB Syntax</b>	PasteSetup <SetupName>
<b>VB Example</b>	<code>oModule.PasteSetup "OptimizationSetup1"</code>

## RenameSetup [Optimetrics]

Renames the specified Optimetrics setup.

<b>UI Access</b>	<b>Right-click the setup in the project tree, and then click Rename on the shortcut menu.</b>									
<b>Parameters</b>	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>&lt;OldName&gt;</td><td>String</td><td>The name that needs to be replaced</td></tr><tr><td>&lt;NewName&gt;</td><td>String</td><td>Replacement name</td></tr></tbody></table>	Name	Type	Description	<OldName>	String	The name that needs to be replaced	<NewName>	String	Replacement name
Name	Type	Description								
<OldName>	String	The name that needs to be replaced								
<NewName>	String	Replacement name								
<b>Return Value</b>	None									

<b>Python Syntax</b>	RenameSetup (<OldName> <NewName>)
<b>Python Example</b>	<pre>oModule.RenameSetup ("OptimizationSetup1" "MyOptimization")</pre>

<b>VB Syntax</b>	RenameSetup <OldName> <NewName>
<b>VB Example</b>	<pre>oModule.RenameSetup "OptimizationSetup1" "MyOptimization"</pre>

## SetPropValue [Optimetrics]

Sets the property value for the active Optimetrics setup.

<b>UI Access</b>	Set Property value on Optimetrics objects		
<b>Parameters</b>	Name	Type	Description
	Property path	text string	Setup property path. See discussion of <a href="#">Property Path</a>
<b>Return Value</b>	True if the property is found and the new value is valid. Otherwise return False.		

<b>Python Syntax</b>	SetPropValue(propPath, newValue)
<b>Python Example</b>	<pre>oOptModule.SetPropValue("ParametricSetup1\Enabled", False) //disable ParametricSetup1 oOptModule.SetPropValue("OptimizationSetup1/Optimizer", "Quasi Newton")</pre>

<b>VB Syntax</b>	SetPropValue(propPath, newValue)
<b>VB Example</b>	SetPropValue("ParametricSetup1\Enabled", False)

## SolveAllSetup

Solves all Optimetrics setups

<b>UI Access</b>	Right-click on <b>Optimetrics</b> in Project Manager and select <b>Analyze&gt;All</b> from context menu						
<b>Parameters</b>	<table border="1"><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>None</td><td></td><td></td></tr></table>	Name	Type	Description	None		
Name	Type	Description					
None							
<b>Return Value</b>	None						

<b>Python Syntax</b>	SolveAllSetup()
<b>Python Example</b>	oModule.SolveAllSetup ()

<b>VB Syntax</b>	SolveAllSetup
<b>VB Example</b>	oModule.SolveAllSetup

## SolveSetup [Optimetrics]

Solves the specified Optimetrics setup.

<b>UI Access</b>	Right-click the setup in the project tree, and then click <b>Analyze</b> on the shortcut menu.
------------------	------------------------------------------------------------------------------------------------

<b>Parameters</b>	Name <code>&lt;SetupName&gt;</code>	Type String	Description Name of the setup to be solved
<b>Return Value</b>	None		

<b>Python Syntax</b>	<code>SolveSetup (&lt;SetupName&gt;)</code>
<b>Python Example</b>	<code>oModule.SolveSetup ("OptimizationSetup1")</code>

<b>VB Syntax</b>	<code>SolveSetup &lt;SetupName&gt;</code>
<b>VB Example</b>	<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](#)

<b>UI Access</b>	Under <i>Optimetrics</i> in the Project Manager, right-click the <b>&lt;Parametric Setup Name&gt;</b> and then click <b>Validate Parametric Setup</b> on the shortcut menu.			
<b>Parameters</b>	<table border="1"> <tr> <td>Name <code>&lt;SetupName&gt;</code></td> <td>Type String</td> <td>Description Name of the parametric setup.</td> </tr> </table>	Name <code>&lt;SetupName&gt;</code>	Type String	Description Name of the parametric setup.
Name <code>&lt;SetupName&gt;</code>	Type String	Description Name of the parametric setup.		
<b>Return Value</b>	None			

<b>Python Syntax</b>	ValidateSetup (<SetupName>)
<b>Python Example</b>	oModule.ValidateSetup ('ParametricSetup1')

<b>VB Syntax</b>	ValidateSetup <SetupName>
<b>VB Example</b>	oModule.ValidateSetup "ParametricSetup1"

## Parametric Script Commands

[EditSetup \[Parametric\]](#)

[ExportParametricSetupTable](#)

[GenerateVariationData \[Parametric\]](#)

[InsertSetup \[Parametric\]](#)

[ValidateSetup \[Parametric\]](#)

### EditSetup [Parametric]

Modifies an existing parametric setup

<b>UI Access</b>	Right-click the setup in the project tree, and then click <b>Properties</b> on the shortcut menu.											
<b>Parameters</b>	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>&lt;SetupName&gt;</td><td>String</td><td>Name of the Setup</td></tr><tr><td></td><td></td><td></td></tr></tbody></table>			Name	Type	Description	<SetupName>	String	Name of the Setup			
Name	Type	Description										
<SetupName>	String	Name of the Setup										
<b>Return Value</b>	None											

<b>Python Syntax</b>	EditSetup (<SetupName>, <ParametricParams>)
<b>Python Example</b>	See EditSetup [Optimization]

<b>VB Syntax</b>	EditSetup <SetupName>, <ParametricParams>
<b>VB Example</b>	See EditSetup [Optimization]

## ExportParametricSetupTable

Exports the parametric setup table as a CSV file.

<b>UI Access</b>	Double-click parametric setup. Select <b>Table</b> tab. Click <b>Export</b> .									
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;SetupName&gt;</td> <td>String</td> <td>Name of the setup.</td> </tr> <tr> <td>&lt;filePath&gt;</td> <td>String</td> <td>Full path for file export.</td> </tr> </tbody> </table>	Name	Type	Description	<SetupName>	String	Name of the setup.	<filePath>	String	Full path for file export.
Name	Type	Description								
<SetupName>	String	Name of the setup.								
<filePath>	String	Full path for file export.								
<b>Return Value</b>	None									

<b>Python Syntax</b>	ExportParametricSetupTable (<SetupName>, <filePath>)
<b>Python Example</b>	<code>oModule.ExportParametricSetupTable('ParametricSetup1', 'E:/Files/ParametricSetup1_Table.csv')</code>

<b>VB Syn-</b>	ExportParametricSetupTable <SetupName>, <filePath>
----------------	----------------------------------------------------

<b>tax</b>	
<b>VB Example</b>	obj.ExportParametricSetupTable "ParametricSetup1", "E:/Files/ParametricSetup1_Table.csv"

## GenerateVariationData [Parametric]

Generate variation data before parametric solve for CAD integrated project

*Command:* Right click on the parametric setup in the project tree and choose "Generate Variation Data"

*Syntax:* GenerateVariationData <SetupName>

*Return Value:* None

*Parameters:* <SetupName>

Name of the setup.

*VB Example:*

```
oModule.GenerateVariationData "ParametricSetup1"
```

## InsertSetup [Parametric]

Inserts a new parametric setup.

<b>UI Access</b>	Right-click the <b>Optimetrics</b> folder in the project tree, and then click <b>Add&gt; Parametric</b> on the shortcut menu.		
<b>Parameters</b>	Name <Parametric Params>	Type Array	Description Array("NAME:<SetupName>", "SaveFields:=", <SaveField>, <StartingPoint>, "Sim. Setups:=", <SimSetups>, <SweepDefs>, <SweepOps>,

		Array("NAME:Goals", Array("NAME:Goal", <OptiGoalSpec>), ... Array("NAME:Goal", <OptiGoalSpec>))
<SetupName>	String	Name of the parametric setup.
<SimSetups>	Array of Strings	An array of Twin Builder solution setup names.
<SweepDefs>	Array	Array("NAME:Sweeps", Array("NAME:SweepDefinition", "Variable:=", <VarName>, "Data:=", <SweepData>, "Synchronize:=", <SyncNum>), ... Array("NAME:SweepDefinition", "Variable:=", <VarName>, "Data:=", <SweepData>, "Synchronize:=", <SyncNum>))
<SweepData>	String	"<SweepType>, <StartV>, <StopV>, <StepV>"
<SweepType>	String	The type of sweep data.
<SyncNum>	Integer	SweepDatas with the same value are synchronized.
<SweepOps>		Array("NAME:Sweep Operations", "<OpType>:=, Array(<VarValue>, ..., <VarValue>), ... <OpType>:=, Array(<VarValue>, ..., <VarValue>))
<OpType>	String	The sweep operation type.
<b>Return Value</b>	None	

**Python Syntax**

InsertSetup ("OptiParametric", &lt;ParametricParams&gt;)

**Python Example**

```
oModule.InsertSetup(  
    "OptiParametric",  
    [ "NAME:ParametricSetup1", _  
        "SaveFields:=", true, _  
        [ "NAME:StartingPoint"], _  
        "Sim. Setups:=", [ "Setup1"], _  
        [ "NAME:Sweeps", _  
            [ "NAME:SweepDefinition", _  
                "Variable:=", "$width", _  
                "Data:=", "LIN 12mm 17mm 2.5mm", _  
                "OffsetF1:=", false, _  
                "Synchronize:=", 0 ],  
            [ "NAME:SweepDefinition", _  
                "Variable:=", "$length", _  
                "Data:=", "LIN 8mm 12mm 2mm", _  
                "OffsetF1:=", false, _  
                "Synchronize:=", 0 ]],  
        [ "NAME:Sweep Operations"], _  
        [ "NAME:Goals", _
```

```

["NAME:Goal", _
 "Solution:=", "Setup1 : LastAdaptive", _
 "Calculation:=", "returnloss", _
 "Context:=", "", _
 ["NAME:Ranges", _
 "Range:=", [{"Var:=", "Freq", "Type:=", "s", _
 "Start:=", "8GHz", "Stop:=", "8GHz"}], _
 ["NAME:Goal", _
 "Solution:=", "Setup1 : LastAdaptive", _
 "Calculation:=", "reflect", _
 "Context:=", "", _
 ["NAME:Ranges", _
 "Range:=", [{"Var:=", "Freq", "Type:=", "s", _
 "Start:=", "8GHz", "Stop:=", "8GHz"}]]])

```

<b>VB Syntax</b>	InsertSetup "OptiParametric", <ParametricParams>
<b>VB Example</b>	<pre> oModule.InsertSetup "OptiParametric", Array("NAME:ParametricSetup1", _ "SaveFields:=", true, _ </pre>

```
Array("NAME:StartingPoint"), _  
"Sim. Setups:=", Array("Setup1"), _  
Array("NAME:Sweeps", _  
Array("NAME:SweepDefinition", _  
"Variable:=", "$width", _  
"Data:=", "LIN 12mm 17mm 2.5mm", _  
"OffsetF1:=", false, _  
"Synchronize:=", 0),  
Array("NAME:SweepDefinition", _  
"Variable:=", "$length", _  
"Data:=", "LIN 8mm 12mm 2mm", _  
"OffsetF1:=", 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](#)

<b>UI Access</b>	Under <i>Optimetrics</i> in the Project Manager, right-click the < <b>Parametric Setup Name</b> > and then click <b>Validate Parametric Setup</b> on the shortcut menu.								
<b>Parameters</b>	<table border="1"> <tr> <td>Name</td> <td>Type</td> <td>Description</td> </tr> <tr> <td>&lt;SetupName&gt;</td> <td>String</td> <td>Name of the parametric setup.</td> </tr> </table>			Name	Type	Description	<SetupName>	String	Name of the parametric setup.
Name	Type	Description							
<SetupName>	String	Name of the parametric setup.							
<b>Return Value</b>	None								

### Python Syntax

ValidateSetup (<SetupName>)

<b>Python Example</b>	<code>oModule.ValidateSetup('ParametricSetup1')</code>
-----------------------	--------------------------------------------------------

<b>VB Syntax</b>	<code>ValidateSetup &lt;SetupName&gt;</code>
------------------	----------------------------------------------

<b>VB Example</b>	<code>oModule.ValidateSetup "ParametricSetup1"</code>
-------------------	-------------------------------------------------------

## Optimization Script Commands

[EditSetup \[Optimization\]](#)

[InsertSetup \[Optimization\]](#)

### EditSetup [Optimization]

Modifies an existing optimization setup.

<b>UI Access</b>	Right-click the setup in the project tree, and then click Properties on the shortcut menu									
<b>Parameters</b>	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>&lt;SetupName&gt;</td><td>String</td><td>Name of the Setup</td></tr><tr><td></td><td></td><td></td></tr></tbody></table>	Name	Type	Description	<SetupName>	String	Name of the Setup			
Name	Type	Description								
<SetupName>	String	Name of the Setup								
<b>Return Value</b>	None									

<b>Python Syntax</b>	<code>EditSetup (&lt;SetupName&gt;, &lt;OptimizationParams&gt;)</code>
----------------------	------------------------------------------------------------------------

<b>Python Example</b>	<code>oModule.EditSetup("OptimizationSetup1", [</code>
-----------------------	------------------------------------------------------------

```
"NAME:OptimizationSetup1",
  "UseFastCalculationUpdateAlgo:=", False,
  "FastCalcOptCtrlledByUser:=", False,
  "IsEnabled:=", True,
  "SaveSolutions:=", False,
  [
    "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	<pre> oModule&gt;EditSetup "OptimizationSetup1", Array("NAME:OptimizationSetup1", "UseFastCalculationUpdateAlgo:=", _ false, "FastCalcOptCtrlledByUser:=", false, ".IsEnabled:=", true, "SaveSolutions:=", _ false, Array("NAME:StartingPoint"), "Optimizer:=", "Quasi Newton", Array("NAME:AnalysisStopOptions", "StopForNumIteration:=", _ true, "StopForElapsTime:=", false, "StopForSlowImprovement:=", false, "StopForGrdTolerance:=", _ false, "MaxNumIteration:=", 1000, "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", </pre>

```

"SimValueContext:=", Array( _  

    1, 0, 2, 0, false, false, -1, 1, 0, 1, 1, "", 0, 0)),  

"Calculation:=", "Time", "Name:=", _  

    "Time", Array("NAME:Ranges", "Range:=",  

        Array("Var:=", "Time", "Type:=", "a")), "Condition:=", _  

        "==", Array("NAME:GoalValue", "GoalValueType:=",  

            "Independent", "Format:=", _  

                "Real/Imag", "bG:=", Array("v:=", "[1;]")),  

            "Weight:=", "[1;"])), "Acceptable_Cost:=", _  

            0, "Noise:=", 0.0001, "UpdateDesign:=", false,  

            "UpdateIteration:=", 5, "KeepReportAxis:=", _  

            true, "UpdateDesignWhenDone:=", true)

```

## InsertSetup [Optimization]

*Use:* Inserts a new optimization setup.

UI Access	Right-click the <b>Optimetrics</b> folder in the project tree, and then click <b>Add&gt;Optimization</b> on the shortcut menu.		
Parameters	Name <OptimizationParams>	Type Array	Description Array("NAME:<SetupName>", "SaveFields:=", <SaveField>, <StartingPoint>, "Optimizer:=", <Optimizer>, "MaxIterations:=", <MaxIter>, "PriorPSetup:=", <PriorSetup>, "PreSolvePSetup:=", <Preceed>,

		<pre> &lt;OptimizationVars&gt;, &lt;Constraint&gt;, Array("NAME:Goals", Array("NAME:Goal", &lt;OptiGoalSpec&gt;, &lt;OptimizationGoalSpec&gt;), ... Array("NAME:Goal", &lt;OptiGoalSpec&gt;, &lt;OptimizationGoalSpec&gt;)), "Acceptable_Cost:=", &lt;AcceptableCost&gt;, "Noise:=", &lt;Noise&gt;, "UpdateDesignWhenDone:=", &lt;UpdateDesign&gt; </pre>
<OptimizationVars>	Array	<pre> Array ("NAME:Variables", "VarName:=", Array ("i:=", &lt;IncludeVar&gt;, "Min:=", &lt;MinV&gt;, "Max:=", &lt;MaxV&gt;, "MinStep:=", &lt;MinStepV&gt;, "MaxStep:=", &lt;MaxStepV&gt;), ..... "VarName:=", Array ("i:=", &lt;IncludeVar&gt;, "Min:=", &lt;MinV&gt;, "Max:=", &lt;MaxV&gt;, "MinStep:=", &lt;MinStepV&gt;, "MaxStep:=", &lt;MaxStepV&gt;)) </pre>
<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	<pre> "Condition:=", &lt;OptimizationCond&gt;, Array ("NAME:GoalValue", "GoalValeType:=", &lt;GoalValueType&gt;, "Format:=", &lt;GoalValueFormat&gt;, "bG:=", </pre>

		Array ("v:=", <GoalValue>)), "Weight:=", <Weight>)
<OptimizationCond>	String	Either "<=", "==", or ">="
<GoalValueType>	String	Either "Independent" or "Dependent"
<GoalValueFormat>	String	Either "Real/Imag" or "Mag/Ang".
<GoalValue>	String	Value in string. Value can be a real number, complex number, or expression.
<b>Return Value</b>	None	

**Command:** Right-click the **Optimetrics** folder in the project tree, and then click **Add>Optimization** on the shortcut menu.

**Syntax:** InsertSetup "OptiOptimization", <OptimizationParams>

**Return Value:** None

**Parameters:** <OptimizationParams>

```

Array ("NAME:<SetupName>", "SaveFields:=",
<SaveField>, <StartingPoint>, "Optimizer:=",
<Optimizer>,
"MaxIterations:=", <MaxIter>, "PriorPSetup:=",
<PriorSetup>, "PreSolvePSetup:=", <Preceed>,
<OptimizationVars>, <Constraint>,
Array ("NAME:Goals", Array ("NAME:Goal",
<OptiGoalSpec>, <OptimizationGoalSpec>), ...
Array ("NAME:Goal", <OptiGoalSpec>,
<OptimizationGoalSpec>),
"Acceptable_Cost:=", <AcceptableCost>, "Noise:=",
<Noise>, "UpdateDesignWhenDone:=", <UpdateDesign>

```

```
<OptimizationVars>

    Array("NAME:Variables", "VarName:=", Array("i:=",
        <IncludeVar>, "Min:=", <MinV>, "Max:=", <MaxV>,
        "MinStep:=", <MinStepV>, "MaxStep:=", <MaxStepV>),
        ..... "VarName:=", Array("i:=", <IncludeVar>, "Min:=", <MinV>, "Max:=", <MaxV>,
        "MinStep:=", <MinStepV>, "MaxStep:=", <MaxStepV>) )
```

<MinStepV>

Type : <VarValue>

The minimum step of the variable.

<MaxStepV>

Type: <VarValue>

The maximum step of the variable.

<AcceptableCost>

Type: <double>

The acceptable cost value for the optimizer to stop.

<Noise>

Type: <double>

The noise of the design.

<UpdateDesign>

Type: <bool>

Specifies whether or not to apply the optimal variation to the design after the optimization is done.

<OptimizationGoalSpec>

```
"Condition:=", <OptimizationCond>,
Array("NAME:GoalValue", "GoalValueType:=",
<GoalValueType>,
"Format:=", <GoalValueFormat>, "bG:=",
Array("v:=", <GoalValue>)), "Weight:=", <Weight>)
```

<OptimizationCond>

Type: <string>

Either "<=", "==" , or ">="

<GoalValueType>

Type: <string>

Either "Independent" or "Dependent"

<GoalValueFormat>

Type:<string>

Either "Real/Imag" or "Mag/Ang".

<GoalValue>

Type: <string>

Value in string. Value can be a real number, complex number, or expression.

*VB Example:*

```
oModule.InsertSetup "OptiOptimization",  
    Array("NAME:OptimizationSetup1", _  
        "SaveFields:=", false, _  
        Array("NAME:StartingPoint", "$length:=", "8mm", _  
            "$width:=", "14.5mm"), _  
        "Optimizer:=", "Quasi Newton", _  
        "MaxIterations:=", 100, _  
        "PriorPSetup:=", "ParametricSetup1", _  
        "PreSolvePSetup:=", true, _  
        Array("NAME:Variables", _
```

```
"$length:=", Array("i:=", true, "Min:=", "6mm", _  
"Max:=", "18mm", _  
"MinStep:=", "0.001mm", "MaxStep:=", _  
"1.2mm"), _  
"$width:=", Array("i:=", true, "Min:=", _  
"6.5mm", "Max:=", "19.5mm", _  
"MinStep:=", "0.001mm", "MaxStep:=", _  
"1.3mm")), _  
    Array ("NAME:LCS") , _  
    Array ("NAME:Goals", _  
Array("NAME:Goal", _  
"Solution:=", "Setup1 : LastAdaptive", _  
"Calculation:=", "reflect", _  
"Context:=", "", _  
Array("NAME:Ranges", _  
"Range:=", Array("Var:=", "Freq", _  
"Type:=", "s", _  
"Start:=", "8GHz", "Stop:=", "8GHz")), _  
"Condition:=", "<=", _  
Array("NAME:GoalValue", _  
"GoalValueType:=", "Independent", _
```

```
"Format:=", "Real/Imag", _  
"bG:=", Array("v:=", "[0.0001]), _  
"Weight:=", "[1]),  
"Acceptable_Cost:=", 0.0002, _  
"Noise:=", 0.0001, _  
"UpdateDesign:=", true, _  
"UpdateIteration:=", 5, _  
"KeepReportAxis:=", true, _  
"UpdateDesignWhenDone:=", true)
```

Python Syntax	InsertSetup ("OptiOptimization", <OptimizationParams>)
Python Example	<pre>oModule.InsertSetup(     "OptiOptimization", _     [ "NAME:OptimizationSetup1", _         "SaveFields:=", false, _         [ "NAME:StartingPoint", "\$length:=", "8mm", _             "\$width:=", "14.5mm"], _         "Optimizer:=", "Quasi Newton", _         "MaxIterations:=", 100, _</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)

```

<b>VB Syntax</b>	InsertSetup "OptiOptimization", <OptimizationParams>
<b>VB Example</b>	<pre> oModule.InsertSetup "OptiOptimization", __ Array("NAME:OptimizationSetup1", __ "SaveFields:=", false, __ </pre>

```
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.

<b>UI Access</b>	Right-click the setup in the project tree, and then click Properties on the shortcut menu		
<b>Parameters</b>	Name	Type	Description
	<SetupName>	String	Name of the Setup
<b>Return Value</b>	None		

<b>Python Syntax</b>	EditSetup (<SetupName>, <SensitivityParams>)
<b>Python Example</b>	<pre> oModule.EditSetup ("OptimizationSetup1", [     "NAME:OptimizationSetup1",     "UseFastCalculationUpdateAlgo:=", False,     "FastCalcOptCtrlledByUser:=", False,     "IsEnabled:=", True,     "SaveSolutions:=", False,     [         "NAME:StartingPoint"     ], ] ,</pre>

```
"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
])
```

<b>VB Syntax</b>	EditSetup <SetupName>, <SensitivityParams>
<b>VB Example</b>	oModule.EditSetup("OptimizationSetup1", Array(

```
"NAME:OptimizationSetup1",
"UseFastCalculationUpdateAlgo:=", False,
"FastCalcOptCtrlledByUser:=", False,
".IsEnabled:=", True,
"SaveSolutions:=", False,
Array(
"NAME:StartingPoint"
),
"Optimizer:=", "Quasi Newton",
Array(
"NAME:AnalysisStopOptions",
"StopForNumIteration:=", True,
"StopForElapsTime:=", False,
"StopForSlowImprovement:=", False,
"StopForGrdTolerance:=", False,
"MaxNumIteration:=", 1001,
"MaxSolTimeInSec:=", 3600,
"RelGradientTolerance:=", 0,
"MinNumIteration:=", 10
),
"CostFuncNormType:=", "L2",
```

```
"PriorPSetup:=", "",  
"PreSolvePSetup:=", True,  
Array(  
    "NAME:Variables"  
) ,  
Array(  
    "NAME:LCS"  
) ,  
Array(  
    "NAME:Goals",  
    Array(  
        "NAME:Goal",  
        "ReportType:=", "Standard",  
        "Solution:=", "TR3",  
        Array(  
            "NAME:SimValueContext",  
            "SimValueContext:=",  
            Array(1,0,2,0,False,False,-1,1,0,1,1,"",0,0)  
        ) ,
```

```
"Calculation:=", "mag(DIFF1.VAL)",
"Name:=", "DIFF1.VAL",
Array(
"NAME:Ranges",
"Range:=", Array("Var:=", "Time", "Type:=", "a")
),
"Condition:=", "==",
Array(
"NAME:GoalValue",
"GoalValueType:=", "Independent",
"Format:=", "Real/Imag",
"bG:=", Array("v:=", "Array(1;]")
),
"Weight:=", "Array(1;]"
)
),
"Acceptable_Cost:=", 0,
"Noise:=", 0.0001,
"UpdateDesign:=", False,
"UpdateIteration:=", 5,
"KeepReportAxis:=", True,
```

```
"UpdateDesignWhenDone:=", True
) )
```

## InsertSetup [Sensitivity]

Inserts a new sensitivity setup.

UI Access	Right-click <b>Optimetrics</b> in the project tree, and then click <b>Add&gt;Sensitivity</b> on the shortcut menu.		
<b>Parameters</b>	<b>Name</b>	Type	Description
	<SensitivityParams>	Array	Array("NAME:<SetupName>", "SaveFields:=", <SaveField>, <StartingPoint>, "MaxIterations:=", <MaxIter>, "PriorPSetup:=", <PriorSetup>, "PreSolvePSetup:=", <Preceed>, <SensitivityVars>, <Constraint>, Array("NAME:Goals", Array("NAME:Goal", <OptiGoalSpec>), ..., Array("NAME:Goal", <OptiGoalSpec>)), "Primary Goal:=". <PrimaryGoalID>, "PrimaryError:=", <PrimaryError>)
	<b>SensitivityVars</b>	Array	Array("NAME:Variables", "VarName:=", Array("i:=", <IncludeVar>, "Min:=", <MinV>, "Max:=", <MaxV>, "IDisp:=", <InitialDisp>),... "VarName:=", Array("i:=", <IncludeVar>,

		"Min:=", <MinV>, "Max:=", <MaxV>, "IDisp:=", <InitialDisp>))
<InitialDisp>	VarValue	Index of the Primary goal. Index starts from zero.
<PrimaryError>	Double	Error associated with the Primary goal.
<b>Return Value</b>	None	

<b>Python Syntax</b>	InsertSetup ("OptiSensitivity", <SensitivityParams>)
<b>Python Example</b>	<pre> oModule.InsertSetup (     "OptiSensitivity", _     [ "NAME:SensitivitySetup1", _         "SaveFields:=", true, _         [ "NAME:StartingPoint" ], _         "MaxIterations:=", 20, _         "PriorPSetup:=", "", _         "PreSolvePSetup:=", true, _         [ "NAME:Variables" ], _         [ "NAME:LCS" ], _         "NAME:Goals", _         [ "NAME:Goal", _             "Solution:=", "Setup1 : LastAdaptive", _             "Calculation:=", "returnloss", _ </pre>

```
"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"]]],_
"Primary Goal:=", 1,_
"PrimaryError:=", 0.001])
```

<b>VB Syntax</b>	InsertSetup "OptiSensitivity", <SensitivityParams>
<b>VB Example</b>	oModule.InsertSetup "OptiSensitivity", _

```
Array("NAME:SensitivitySetup1",_
    "SaveFields:=", true,_
    Array("NAME:StartingPoint"), _
    "MaxIterations:=", 20,_
    "PriorPSetup:=", "",_
    "PreSolvePSetup:=", true, _
    Array("NAME:Variables"), _
    Array("NAME:LCS"),_
    Array("NAME:Goals"),_
    Array("NAME:Goal", _
        "Solution:=", "Setup1 : LastAdaptive",_
        "Calculation:=", "returnloss",_
        "Context:=", "",_
        Array("NAME:Ranges",_
            "Range:=", Array("Var:=", "Freq", "_
                Type:=", "s",_
                "Start:=", "8GHz", "Stop:=", "8GHz"))),_
        Array("NAME:Goal",_
            "Solution:=", "Setup1 : LastAdaptive",_
            "Calculation:=", "reflect",_
```

```
"Context:=", "",_
Array("NAME:Ranges",_
"Range:=", Array("Var:=", "Freq",_
"Type:=", "S",_
"Start:=", "8GHz", "Stop:=", "8GHz"))),_
"Primary Goal:=", 1,_
"PrimaryError:=", 0.001)
```

## Statistical Script Commands

[EditSetup \[Statistical\]](#)

[InsertSetup Statistical](#)

### EditSetup [Statistical]

Modifies an existing statistical setup.

<b>UI Access</b>	Right-click the setup in the project tree, and clickProperties on the shortcut menu.	
<b>Parameters</b>	Name  <code>&lt;SetupName&gt;</code>	Type  String
<b>Return Value</b>	None	

<b>Python Syntax</b>	EditSetup (<SetupName>, <StatisticalParams>)
<b>Python Example</b>	See EditSetup [Optimization]

<b>VB Syntax</b>	EditSetup <SetupName>, <StatisticalParams>
<b>VB Example</b>	See EditSetup [Optimization]

*Example:*

```

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

Set oAnsoftApp = CreateObject ("Ansoft.ElectronicsDesktop")
Set oDesktop = oAnsoftApp.GetAppDesktop ()
oDesktop.RestoreWindow
Set oProject = oDesktop.SetActiveProject ("optiguides")
Set oDesign = oProject.SetActiveDesign ("HFSSModel1")
Set oModule = oDesign.GetModule ("Optimetrics")
oModule.EditSetup "StatisticalSetup1", Array ("NAME:StatisticalSetup1",
Array ("NAME:ProdOptiSetupData",

```

```
"SaveFields:=", true, "CopyMesh:=", false),
Array("NAME:StartingPoint", "$length:=", "7.824547736mm",
"$width:=", "14.8570192mm"),
"MaxIterations:=", 50,
"PriorPSetup:=", "",

Array("NAME:Variables",
"$length:=", Array("i:=", true,
"int:=", false,
"Dist:=", "Uniform",
"Tol:=", "10%", "StdD:=", "0.2mm", "Min:=", "-3",
"Max:=", "3", "Shape:=", "1", "Scale:=", "0.04mm",
"Location:=", "0.4mm",
"Dataset:=", "", "LatinHypercube:=", "true", "VarMin:=", "0.2mm", "VarMax:=", "0.6mm", "Prob:=",
"0.01",
"Mean:=", "0.4mm"),

"$width:=", Array("i:=", true,
"int:=", false,
"Dist:=", "Gaussian",
"Tol:=", "10%",
"StdD:=", "0.2mm",
"Min:=", "-3", "Max:=", "3",
"Shape:=", "1",
"Scale:=", "0.04mm",
"Location:=", "0.4mm",
"Dataset:=", "",
"LatinHypercube:=", "true",
"VarMin:=", "0.2mm", "VarMax:=", "0.6mm",
"Prob:=", "0.02",
"Mean:=", "0.4mm")),

Array("NAME:Goals", Array("NAME:Goal",
"ReportType:=", "Modal Solution Data",
```

```

"Solution:=", "Setup1 : PortOnly",
Array("NAME:SimValueContext", "Domain:=", "Sweep"),
"Calculation:=", "returnloss",
"Name:=", "returnloss",

Array("NAME:Ranges",
"Range:=", Array("Var:=", "Freq",
"Type:=", "s",
"Start:=", "8.2GHz", "Stop:=", "0")),

Array("NAME:Goal",
"ReportType:=", "Modal Solution Data",
"Solution:=", "Setup1 : PortOnly",
Array("NAME:SimValueContext",
"Domain:=", "Sweep"),
"Calculation:=", "reflect",
"Name:=", "reflect",
Array("NAME:Ranges",
"Range:=", Array("Var:=", "Freq",
"Type:=", "s",
"Start:=", "8.2GHz", "Stop:=", "0")))
)

```

## InsertSetup [Statistical]

Inserts a new statistical setup.

<b>UI Access</b>	Right-click <b>Optimetrics</b> in the project tree, and then click <b>Add&gt;Statistical</b> on the shortcut menu.		
<b>Parameters</b>	Name <StatisticalParams>	Type Array	Description 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>))
	<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.
<b>Return Value</b>	None	

<b>Python Syntax</b>	InsertSetup ("OptiStatistical", <StatisticalParams>)
<b>Python Example</b>	oModule.InsertSetup (

```
"OptiStatistical", __
  ["NAME:StatisticalSetup1", __
    "SaveFields:=", true, __
    ["NAME:StartingPoint"], __
    "MaxIterations:=", 50, __
    "PriorPSetup:=", "", __
    ["NAME:Variables"], __
    ["NAME:Goals"], __
    ["NAME:Goal", __
      "Solution:=", "Setup1 : LastAdaptive", __
      "Calculation:=", "returnloss", __
      "Context:=", "", __
      ["NAME:Ranges"], __
      "Range:=", [{"Var:=", "Freq", __
        "Type:=", "s", __
        "Start:=", "8GHz", "Stop:=", "8GHz"}]], __
      ["NAME:Goal", __
        "Solution:=", "Setup1 : LastAdaptive", __
        "Calculation:=", "reflect", __
        "Context:=", "", __
```

```
[ "NAME:Ranges", _  
  "Range:=", [ "Var:=", "Freq", "Type:=", _  
    "s", "Start:=", "8GHz", "Stop:=", "8GHz" ] ] ] )
```

VB Syntax	InsertSetup "OptiStatistical", <StatisticalParams>
<b>VB Example</b>	<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", _</pre>

```
"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"))))
```

*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",
"Solutions:=", "Setup1 : LastAdaptive", Array("NAME:SimValueContext"),
"Calculation:=", "Power11",
"Name:=", "Power11",
Array("NAME:Ranges",
"Range:=", Array("Var:=", "Freq", "Type:=", "d",
"DiscreteValues:=", "10GHz"))))
```

### For Q3D Extractor and Circuit the command details are as follows:

Inserts a new statistical setup.

*Command:* Right-click **Optimetrics** in the project tree, and then click **Add>Statistical** on the shortcut menu.

*Syntax:* InsertSetup "OptiStatistical", <StatisticalParams>

*Return Value:* None

*Parameters:* <StatisticalParams>

```
Array("NAME:<SetupName>", "SaveFields:=",
<SaveField>, <StartingPoint>, "MaxIterations:=",
<MaxIter>, "PriorPSetup:=", <PriorSetup>,
"PreSolvePSetup:=", <Preceed>, <StatisticalVars>,
Array("NAME:Goals", Array("NAME:Goal",
<OptiGoalSpec>), ..., Array("NAME:Goal",
```

```
<OptiGoalSpec>)) ) ,  
<StatisticalVars>  
    Array("NAME:Variables",  
        "VarName:=", Array("i:=", <IncludeVar>, "Dist:=",  
<DistType>, "Tol:=", <Tolerance>,  
        "StdD:=", <StdD>, "Min:=", <MinCutoff>, "Max:=",  
<MaxCutoff>, ...  
        "VarName:=", Array("i:=", <IncludeVar>, "Dist:=",  
<DistType>, "Tol:=", <Tolerance>, "StdD:=",  
<StdD>, "Min:=", <MinCutoff>, "Max:=",  
<MaxCutoff>))
```

**Parameters:**

<DistType>

Type : <string>

Distrbution can be "Gaussian" or "Uniform".

<Tolerance>

Type: <VarValue>

The tolerance for the variable when distribution is Uniform.

<StdD>

Type: <VarValue>

The standard deviation for the variable when distribution is Gaussian.

<MinCutoff>

Type: <double>

The minimum cut-off for the variable when distribution is Gaussian.

<MaxCutoff>

Type: <double>

The maximum cut-off for the variable when distribution is Gaussian.

*Example:* oModule.InsertSetup "OptiStatistical", \_

    Array("NAME:StatisticalSetup1", \_

        "SaveFields:=", true, \_

        Array("NAME:StartingPoint"), \_

        "MaxIterations:=", 50, \_

        "PriorPSetup:=", "", \_

        Array("NAME:Variables"), \_

        Array("NAME:Goals", \_

    Array("NAME:Goal", \_

        "Solutions:=", "Setup1 : LastAdaptive", \_

        "Calculation:=", "returnloss", \_

        "Context:=", "", \_

        Array("NAME:Ranges", \_

            "Range:=", Array("Var:=", "Freq", \_

```
"Type:=", "s",_
"Start:=", "8GHz", "Stop:=", "8GHz"))),_
Array("NAME:Goal",_
"Solutions:=", "Setup1 : LastAdaptive",_
"Calculation:=", "reflect",_
"Context:=", "", _
Array("NAME:Ranges",_
"Range:=", Array("Var:=", "Freq", "Type:=", _ 
"s", "Start:=", "8GHz", "Stop:=", "8GHz"))))
```

# 15 - Solutions Module Script Commands

Solutions commands should be executed by the "Solutions" module.

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

```
oModule.CommandName <args>
```

[DeleteSolutionVariation](#)

[ExportEigenmodes](#)

[ExportNetworkData](#)

[GetAvailableVariations](#)

## DeleteImportData

Deletes imported solution or table data in HFSS. Not in HFSS-IE

<b>UI Access</b>	Right-click on <b>Results &gt; Import Solutions</b> , click <b>Delete Selections</b> button.		
<b>Parameters</b>	Name <i>&lt;ImportSpecArray&gt;</i>	Type Array	Description Array of import solutions.  <i>Array (&lt;ImportName:SolnName OR TableName&gt;, ...)</i>
<b>Return Value</b>	None.		

<b>Python Syntax</b>	<code>DeleteImportData(&lt;ImportSpecArray&gt;)</code>
<b>Python Example</b>	<pre> oModule.DeleteImportData([     "Import1:Adaptive_1", "Import2:DataTable" ] ) </pre>

	])
--	----

<b>VB Syntax</b>	DeleteImportData <ImportSpecArray>
<b>VB Example</b>	<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.

<b>UI Access</b>	Right-click on <b>Results</b> , select <b>Browse Solutions...</b> , click <b>Delete</b> button in the dialog.		
<b>Parameters</b>	Name <SoluParams>	Type Array	Description Structured array.  Array (<DataSpecifierArray>, ...)
	<DataSpecifierArray>	Array	Structured array.  Array (<DesignVariationKey>, <SetupName>, <SolnName>)
<b>Return Value</b>	None.		

<b>Python Syntax</b>	DeleteSolutionVariation(<SoluParams>)
<b>Python Example</b>	<pre>oModule.DeleteSolutionVariation ([</pre>

```

        ["width='2in'", "Setup1", "Adaptive_1"],
        ["width='2in'", "Setup1", "Sweep1"]
    )
)

```

<b>VB Syntax</b>	DeleteSolutionVariation <SoluParam>
<b>VB Example</b>	<pre> oModule.DeleteSolutionVariation Array( _     Array("width='2in'", "Setup1", "Adaptive_1"), _     Array("width='2in'", "Setup1", "Sweep1")) </pre>

## EditSources

Indicates which source excitations should be used for fields post-processing.

**For HFSS, the command details are as follows.**

Indicates which source excitations should be used for fields post processing.

**Command: HFSS or HFSS-IE>Fields>Edit Sources**

**Syntax:** EditSources([{"Name": "<portName>" , <keyword>, <value>, ...}...])

**Return Value:** None

**Parameters:**

Each inner block describes the values to be assigned to an individual excitation. The valid keywords are:

Name - Required. This identifies the excitation and uses the same syntax as the edit source dialog box. As opposed to the old command, names can be specified in any order. Unspecified excitations will not be affected, so users can edit one or more excitations without needing to edit them all.

Magnitude - a unit quantity ("1W", "2mV", ...)

Phase - a unit quantity ("1W", "2mV", ...)

For Driven Terminal designs, a terminal may be terminated

Terminated - boolean ("true", "false"). If terminated, Magnitude and Phase are superseded with:

Resistance - a unit quantity ("50ohm", ...)

Reactance - a unit quantity ("50ohm", ...)

For Transient problems, the spectral fields use Magnitude and Phase. The transient fields are controlled by:

Delay- a unit quantity ("3.5ps", ...)

TransientMagnitude - a unit quantity ("1W", ...)

For CMA problems, the special excitation "Modes" specifies the modal excitations. These are then set with the keywords:

Magnitudes - integer vector [ "0", "0", "1", "2.5" ]. The size of the vector will determine how many modes are visible in the dialog.

Phases - unit quantity vector [ "30deg", ... ]. The size of the vector must match the size of the magnitude vector.

For Eigen problems the special excitation "Modes" specifies the modal excitations. These are then set with the keywords:

Magnitudes - integer vector [ "0", "0", "1", "2.5" ]. The size of the vector must match the design,

Phases- unit quantity vector [ "30deg", ... ]. The size of the vector must match the design. :Phases is not needed for Stored Energy fields.

Optional settings that apply to all sources

Incident Power - unit quantity ("2.5W"). Use in system gain calculations

SpecifySystemPower - boolean. If set, the above power is used. Otherwise, the maximum power is calculated.

FieldType - string. One of

---

```

ScatteredFields
TotalFields
IncidentFields
NoIncidentWave
EigenPeakElectricField
EigenStoredEnergy
IncludePortPostProcessing - boolean

```

For Driven Terminal designs, a terminal may be terminated

UseIncidentVoltage - boolean - otherwise use total voltage

For CMA problems:

ExciteModes - boolean. If true use modal excitations, otherwise use terminals. Terminal excitations are specified by their name, magnitude, and phase

For Eigen problems:

The special excitation "Modes" specifies the modal excitations. These are then set with the keywords

For all unit quantities, expressions ("3 \* my\_var", "pwl( freq, ds1 )") are permitted as well.

## ExportEigenmodes

Exports a tab delimited table of Eigenmodes in HFSS. Not in HFSS-IE.

<b>UI Access</b>	N/A		
<b>Parameters</b>	Name	Type	Description
	<SoluName>	String	Name of specified solution.
	<DesignVariation>	String	Design variation key.
	<FileName>	String	Output file name with path.

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

<b>Python Syntax</b>	ExportEigenmodes(<SoluName>, <DesignVariation>, <FileName>)
<b>Python Example</b>	<pre>oModule.ExportEigenmodes (     "Setup1 : LastAdaptive", "",      "C:\mydir\myeigenmode.eig")</pre>

<b>VB Syntax</b>	ExportEigenmodes <SoluName>, <DesignVariation>, <FileName>
<b>VB Example</b>	<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.

<b>UI Access</b>	N/A														
<b>Parameters</b>	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>&lt;DesignVariation&gt;</td><td>String</td><td>Desgin variation key.</td></tr><tr><td>&lt;SoluSelections&gt;</td><td>Array</td><td>Array of selected solutions.</td></tr><tr><td>&lt;SpiceType&gt;</td><td>Integer</td><td>1 - HSpice.</td></tr></tbody></table>			Name	Type	Description	<DesignVariation>	String	Desgin variation key.	<SoluSelections>	Array	Array of selected solutions.	<SpiceType>	Integer	1 - HSpice.
Name	Type	Description													
<DesignVariation>	String	Desgin variation key.													
<SoluSelections>	Array	Array of selected solutions.													
<SpiceType>	Integer	1 - HSpice.													

	<table border="1"> <tr><td><i>&lt;BandWidth&gt;</i></td><td>Integer</td><td>0 - Low (narrow) band width.</td></tr> <tr><td><i>&lt;FWSFile&gt;</i></td><td>String</td><td>Full wave spice file name.</td></tr> <tr><td><i>&lt;LumpedElementFile&gt;</i></td><td>String</td><td>Lumped element file name.</td></tr> <tr><td><i>&lt;PoleZeroSpiceFile&gt;</i></td><td>String</td><td>Pole zero spice file name.</td></tr> <tr><td><i>&lt;PoleZeroMatlabFile&gt;</i></td><td>String</td><td>Pole zero Matlab file name.</td></tr> <tr><td><i>&lt;PartialFractionFile&gt;</i></td><td>String</td><td>Partial fraction file name.</td></tr> <tr><td><i>&lt;FittingError&gt;</i></td><td>Double</td><td>The accuracy to use in fitting the pole zero model.</td></tr> <tr><td><i>&lt;MinimumOrder&gt;</i></td><td>Integer</td><td>Minimum number of poles in rational function expansion.</td></tr> <tr><td><i>&lt;MaximumOrder&gt;</i></td><td>Integer</td><td>Maximum number of poles in rational function expansion.</td></tr> <tr><td><i>&lt;UseCommonGround&gt;</i></td><td>Integer</td><td>Optional. 1 to use common ground, 0 otherwise.</td></tr> <tr><td><i>&lt;EnforcePassivity&gt;</i></td><td>Integer</td><td>Optional. 1 to enforce passivity, 0 otherwise.</td></tr> </table>	<i>&lt;BandWidth&gt;</i>	Integer	0 - Low (narrow) band width.	<i>&lt;FWSFile&gt;</i>	String	Full wave spice file name.	<i>&lt;LumpedElementFile&gt;</i>	String	Lumped element file name.	<i>&lt;PoleZeroSpiceFile&gt;</i>	String	Pole zero spice file name.	<i>&lt;PoleZeroMatlabFile&gt;</i>	String	Pole zero Matlab file name.	<i>&lt;PartialFractionFile&gt;</i>	String	Partial fraction file name.	<i>&lt;FittingError&gt;</i>	Double	The accuracy to use in fitting the pole zero model.	<i>&lt;MinimumOrder&gt;</i>	Integer	Minimum number of poles in rational function expansion.	<i>&lt;MaximumOrder&gt;</i>	Integer	Maximum number of poles in rational function expansion.	<i>&lt;UseCommonGround&gt;</i>	Integer	Optional. 1 to use common ground, 0 otherwise.	<i>&lt;EnforcePassivity&gt;</i>	Integer	Optional. 1 to enforce passivity, 0 otherwise.
<i>&lt;BandWidth&gt;</i>	Integer	0 - Low (narrow) band width.																																
<i>&lt;FWSFile&gt;</i>	String	Full wave spice file name.																																
<i>&lt;LumpedElementFile&gt;</i>	String	Lumped element file name.																																
<i>&lt;PoleZeroSpiceFile&gt;</i>	String	Pole zero spice file name.																																
<i>&lt;PoleZeroMatlabFile&gt;</i>	String	Pole zero Matlab file name.																																
<i>&lt;PartialFractionFile&gt;</i>	String	Partial fraction file name.																																
<i>&lt;FittingError&gt;</i>	Double	The accuracy to use in fitting the pole zero model.																																
<i>&lt;MinimumOrder&gt;</i>	Integer	Minimum number of poles in rational function expansion.																																
<i>&lt;MaximumOrder&gt;</i>	Integer	Maximum number of poles in rational function expansion.																																
<i>&lt;UseCommonGround&gt;</i>	Integer	Optional. 1 to use common ground, 0 otherwise.																																
<i>&lt;EnforcePassivity&gt;</i>	Integer	Optional. 1 to enforce passivity, 0 otherwise.																																
<b>Return Value</b>	None.																																	
<b>Python Syntax</b>	ExportForHSpice(<DesignVariation>, <SoluSelections>, <SpiceType>, <BandWidth>, <FWSFile>, <LumpedElementFile>, <PoleZeroSpiceFile>, <PoleZeroMatlabFile>, <PartialFractionFile>, <FittingError>, <MinimumOrder>, <MaximumOrder>, [Optional <UseCommonGround>], [Optional <EnforcePassivity>])																																	
<b>Python Example</b>	<pre>oModule.ExportForHSpice("width='2in'",                          ["Setup1:Sweep1"], 1, 0,                          "c:\mydir\Sweep1.fws", "", "", "", "",                          .005, 20, 200)</pre>																																	
<b>VB Syntax</b>	ExportForHSpice <DesignVariation>, <SoluSelections>, <SpiceType>, <BandWidth>, <FWSFile>, <LumpedElementFile>, <PoleZeroSpiceFile>, <PoleZeroMatlabFile>, <PartialFractionFile>, <FittingError>, <MinimumOrder>, <MaximumOrder>, [Optional <UseCommonGround>], [Optional <EnforcePassivity>]																																	
<b>VB Example</b>	<pre>oModule.ExportForHSpice "width='2in'", _ Array("Setup1:Sweep1"), 1, 0, _ "c:\mydir\Sweep1.fws", "", "", "", "", _</pre>																																	

	.005, 20, 200
--	---------------

## 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 fre-

		quencies are used.
	<code>&lt;DoRenorm&gt;</code>	Boolean Specifies whether to renormalize the data before export.
	<code>&lt;RenormImped&gt;</code>	Double Real impedance value in ohms, for renormalization. Required in syntax, but ignored if DoRenorm is false.
	<code>&lt;DataType&gt;</code>	String Optional. Type: "S", "Y", or "Z". The matrix to export.
	<code>&lt;pass&gt;</code>	Integer Optional. The pass to export. This is ignored if the sourceName is a frequency sweep. Leaving out this value or specifying -1 gets all passes.
	<code>&lt;ComplexFormat&gt;</code>	Integer Optional. Type: "0", "1", or "2"  The format to use for the exported data.  0 = Magnitude/Phase.  1= Real/Imaginary.  2= db/Phase.
	<code>&lt;Precision&gt;</code>	Integer Optional. Touchstone number of digits precision. Default if not specified is 15.
	<code>&lt;UseExportFreqs&gt;</code>	Boolean Specifies whether to use export frequencies.
	<code>&lt;IncludeGammaComments&gt;</code>	Boolean Specifies whether to include Gamma and Impedance comments.
	<code>&lt;SupportNonStdExport&gt;</code>	Boolean Specifies whether to support non-standard Touchstone extensions for mixed reference impedance.
<b>Return Value</b>	None.	

<b>Python Syntax</b>	ExportNetworkData(<DesignVariationKey>, <SolnSelectionArray>, <SolnSelector>, <FileFormat>, <OutFile>, <FrequenciesArray>, <DoRenorm>, <RenormImped>, [Optional <DataType>], [Optional <pass>], [Optional <ComplexFormat>], [Optional <Precision>], [Optional <UseExportFreqs>], [Optional <IncludeGammaComments>], [Optional <SupportNonStdExport>])
<b>Python Example</b>	<code>oModule.ExportNetworkData("", ["Setup1:Sweep1"], 3, "C://Documents/package_HFSSDesign1.s4p",</code>

	<code>['all'], True, 50, "S", -1, 0, 15, True, True, True</code>
--	------------------------------------------------------------------

<b>VB Syntax</b>	ExportNetworkData < <i>DesignVariationKey</i> >, < <i>SolnSelectionArray</i> >, < <i>SolnSelector</i> >, < <i>FileFormat</i> >, < <i>OutFile</i> >, < <i>FreqsArray</i> >, < <i>DoRenorm</i> >, < <i>RenormImped</i> >, [Optional < <i>DataType</i> >], [Optional < <i>pass</i> >], [Optional < <i>ComplexFormat</i> >], [Optional < <i>Precision</i> >], [Optional < <i>UseExportFreqs</i> >], [Optional < <i>IncludeGammaComments</i> >], [Optional < <i>SupportNonStdExport</i> >]
<b>VB Example</b>	<pre> oModule.ExportNetworkData "width='2in'", _     Array("Setup1:Sweep1"), 2, "c:\mydir\out.tab", _     Array("all"), false, 0  oModule.ExportNetworkData "width='2in'", _     Array("Setup1:Sweep1", "Setup1:Sweep2"), 3, _     "c:\mydir\out.s2p", Array(1.0e9, 1.5e9, 2.0e9), _     true, 50.0 </pre>

## ExportNMFDATA [HFSS]

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.

<b>UI Access</b>	N/A		
<b>Parameters</b>	Name	Type	Description
	< <i>SolnSelectionArray</i> >	Array	Array of selected solutions.
	< <i>OutFile</i> >	String	Full path to the file to write out.

	<b>&lt;FreqsArray&gt;</b>	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.
	<b>&lt;DesignVariationKey&gt;</b>	String	Design variation.
	<b>&lt;DoRenorm&gt;</b>	Boolean	Specifies whether to renormalize the data before export.
	<b>&lt;RenormImped&gt;</b>	Double	Real impedance value in ohms, for renormalization. Required in syntax, but ignored if DoRenorm is false.
	<b>&lt;Pass&gt;</b>	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.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	ExportNMFDATA(<SolnSelectionArray>, <OutFile>, <FreqsArray> <DesignVariationKey>, <DoRenorm>, <RenormImped>)
<b>Python Example</b>	<pre>oModule.ExportNMFDATA(["Setup1:Sweep1"], "c:\mydir\out.nmf", ["all"], "", False, 0)</pre>

<b>VB Syntax</b>	ExportNMFDATA <SolnSelectionArray>, <OutFile>, <FreqsArray> <DesignVariationKey>, <DoRenorm>, <RenormImped>
<b>VB Example</b>	<pre>oModule.ExportNMFDATA Array("Setup1:Sweep1"), _ "c:\mydir\out.nmf", Array("all"), "", false, 0</pre>

## ExportTransientData

Exports transient solution data to a file.

<b>UI Access</b>	<b>HFSS &gt; Results &gt; Solution Data..., then click Export.</b>																		
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;SoluName&gt;</td> <td>String</td> <td>Name of specified solution.</td> </tr> <tr> <td>&lt;Variation&gt;</td> <td>String</td> <td>Design variation key. Pass empty string for the current nominal variation.</td> </tr> <tr> <td>&lt;FileName&gt;</td> <td>String</td> <td>Full path of output file.</td> </tr> <tr> <td>&lt;DataType&gt;</td> <td>String</td> <td>Optional. Type of output data.</td> </tr> <tr> <td>&lt;DataFormat&gt;</td> <td>String</td> <td>           Optional. Format of output data.            .csv - Comma Delimited Data.            .tab - Tab Delimited Data files.            .dat - Ansys Plot Data files            .txt - Post Processing Files         </td> </tr> </tbody> </table>	Name	Type	Description	<SoluName>	String	Name of specified solution.	<Variation>	String	Design variation key. Pass empty string for the current nominal variation.	<FileName>	String	Full path of output file.	<DataType>	String	Optional. Type of output data.	<DataFormat>	String	Optional. Format of output data. .csv - Comma Delimited Data. .tab - Tab Delimited Data files. .dat - Ansys Plot Data files .txt - Post Processing Files
Name	Type	Description																	
<SoluName>	String	Name of specified solution.																	
<Variation>	String	Design variation key. Pass empty string for the current nominal variation.																	
<FileName>	String	Full path of output file.																	
<DataType>	String	Optional. Type of output data.																	
<DataFormat>	String	Optional. Format of output data. .csv - Comma Delimited Data. .tab - Tab Delimited Data files. .dat - Ansys Plot Data files .txt - Post Processing Files																	
<b>Return Value</b>	None.																		

<b>Python Syntax</b>	ExportTransientData(<SoluName>, <Variation>, <FileName>)
<b>Python Example</b>	oModule.ExportTransientData("Setup1:Transient", "", "C:/transient_sol1.csv")

<b>VB Syntax</b>	ExportTransientData <SoluName>, <Variation>, <FileName>
<b>VB Example</b>	oModule.ExportTransientData "Setup1:Transient", "", "C:/transient_sol1.csv"

## FFTOnReport

Performs an FFT on a selected report.

<b>UI Access</b>	Right-click on <b>Results</b> in the project tree, select <b>Perform FFT On Report...</b>												
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;ReportName&gt;</td> <td>String</td> <td>Name of specified report.</td> </tr> <tr> <td>&lt;WindowName&gt;</td> <td>String</td> <td>Name of window to apply for FFT. Possible values are "Rectangular", "Tri", "Van Hann", "Hamming", "Blackman", "Lanczos", "Weber", "Welch".</td> </tr> <tr> <td>&lt;Function&gt;</td> <td>String</td> <td>Function to apply on transformation values. Possible values are "none", "ang_deg", "ang_rad", "arg", "cang_deg", "cang_rad", "dB", "dB1 normalize", "dB2normalize", "dBc", "im", "mag", "normalize", "re".</td> </tr> </tbody> </table>	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", "dB2normalize", "dBc", "im", "mag", "normalize", "re".
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", "dB2normalize", "dBc", "im", "mag", "normalize", "re".											
<b>Return Value</b>	None.												

<b>Python Syntax</b>	FFTOnReport <ReportName>, <WindowName>, <Function>
<b>Python Example</b>	oModule.FFTOnReport("XY Plot 1", "Rectangular", "dB")

<b>VB Syntax</b>	FFTOnReport(<ReportName>, <WindowName>, <Function>)
<b>VB Example</b>	oModule.FFTOnReport "XY Plot 1", "Rectangular", "dB"

## GetAdaptiveFreq

Obtains an adaptive frequency for a specified setup.

<b>UI Access</b>	N/A						
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;SetupName&gt;</td> <td>String</td> <td>Name of specified setup.</td> </tr> </tbody> </table>	Name	Type	Description	<SetupName>	String	Name of specified setup.
Name	Type	Description					
<SetupName>	String	Name of specified setup.					
<b>Return Value</b>	Double frequency value.						

<b>Python Syntax</b>	GetAdaptiveFreq(<SetupName>)
<b>Python Example</b>	<code>oModule.GetAdaptiveFreq("Setup1")</code>

<b>VB Syntax</b>	GetAdaptiveFreq <SetupName>
<b>VB Example</b>	<code>oModule.GetAdaptiveFreq "Setup1"</code>

## GetAdaptiveSettings

Obtains the adaptive frequency settings of a specified setup.

<b>UI Access</b>	N/A		
<b>Parameters</b>	Name <SetupName>	Type String	Description Name of specified setup.
<b>Return Value</b>	Array of strings represents adaptive frequency settings.		

<b>Python Syntax</b>	GetAdaptiveSettings(<SetupName>)
<b>Python Example</b>	<code>oModule.GetAdaptiveSettings ("Setup1")</code>

<b>VB Syntax</b>	GetAdaptiveSettings <SetupName>
------------------	---------------------------------

<b>VB Example</b>	<code>oModule.GetAdaptiveSettings "Setup1"</code>
-------------------	---------------------------------------------------

## GetAllSourceMagnitudes

Obtains the magnitude values of all defined sources.

<b>UI Access</b>	N/A
<b>Parameters</b>	None.
<b>Return Value</b>	Array of strings containing magnitude values.

<b>Python Syntax</b>	<code>GetAllSourceMagnitudes()</code>
<b>Python Example</b>	<code>oModule.GetAllSourceMagnitudes ()</code>

<b>VB Syntax</b>	<code> GetAllSourceMagnitudes</code>
<b>VB Example</b>	<code>oModule.GetAllSourceMagnitudes</code>

## GetAllSourceModes

Obtains mode information of all defined sources.

<b>UI Access</b>	N/A
<b>Parameters</b>	None.
<b>Return Value</b>	Array of strings containing mode information.

<b>Python Syntax</b>	GetAllSourceModes()
<b>Python Example</b>	<code>oModule.GetAllSourceModes ()</code>

<b>VB Syntax</b>	GetAllSourceModes
<b>VB Example</b>	<code>oModule.GetAllSourceModes</code>

## GetAllSourcePhases

Obtains the phase values of all defined sources.

<b>UI Access</b>	N/A
<b>Parameters</b>	None.
<b>Return Value</b>	Array of strings containing phase values.

<b>Python Syntax</b>	GetAllSourcePhases()
<b>Python Example</b>	<code>oModule.GetAllSourcePhases ()</code>

<b>VB Syntax</b>	GetAllSourcePhases
<b>VB Example</b>	<code>oModule.GetAllSourcePhases</code>

## GetAllSources

Retrieves all sources defined in the current solution setup.

<b>UI Access</b>	N/A
<b>Parameters</b>	None.
<b>Return Value</b>	Array of strings containing source names.

<b>Python Syntax</b>	GetAllSources()
<b>Python Example</b>	<code>oModule.GetAllSources ()</code>

<b>VB Syntax</b>	GetAllSources
<b>VB Example</b>	<code>oModule.GetAllSources</code>

## GetAntennaParameters

Retrieves antenna parameters defined in a specified setup.

<b>UI Access</b>	N/A		
<b>Parameters</b>	Name <code>&lt;SetupName&gt;</code>	Type String	Description Name of specified setup.
<b>Return Value</b>	Array of strings containing antenna parameters.		

<b>Python Syntax</b>	GetAntennaParameters(<SetupName>)
<b>Python Example</b>	oModule.GetAntennaParameters ("Setup1")

<b>VB Syntax</b>	GetAntennaParameters <SetupName>
<b>VB Example</b>	oModule.GetAntennaParameters "Setup1"

## GetAvailableVariations

Returns the available variation for a solution.

<b>UI Access</b>	N/A						
<b>Parameters</b>	<table border="1"><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;SoluName&gt;</td><td>String</td><td>Name of specified solution.</td></tr></table>	Name	Type	Description	<SoluName>	String	Name of specified solution.
Name	Type	Description					
<SoluName>	String	Name of specified solution.					
<b>Return Value</b>	Array of strings containing available variations.						

<b>Python Syntax</b>	GetAvailableVariations(<SoluName>)
<b>Python Example</b>	oModule.GetAvailableVariations ("Setup1 : LastAdaptive")

<b>VB Syntax</b>	GetAvailableVariations <SoluName>
<b>VB Example</b>	oModule.GetAvailableVariations "Setup1 : LastAdaptive"

## GetExcitationScaling

Gets source scaling parameters.

<b>UI Access</b>	N/A		
<b>Parameters</b>	Name	Type	Description
	<portName>	String	Port ID.
<b>Return Value</b>	Array of strings containing source scaling parameters.		

<b>Python Syntax</b>	GetExcitationScaling(<portName>, <portIndex>)
<b>Python Example</b>	oDesign.GetExcitationScaling ("Port1", 2)

<b>VB Syntax</b>	GetExcitationScaling(<portName>, <portIndex>)
<b>VB Example</b>	oDesign.GetExcitationScaling ("Port1", 2)

## GetFieldType

Gets the field type of a driven modal design solution.

<b>UI Access</b>	N/A
<b>Parameters</b>	None.
<b>Return Value</b>	String containing field type.

<b>Python Syntax</b>	GetFieldType()
<b>Python Example</b>	<code>oModule.GetFieldType ()</code>

<b>VB Syntax</b>	GetFieldType
<b>VB Example</b>	<code>oModule.GetFieldType</code>

## GetIncludePortPostProcessing

Determines whether the Include Port Post Processing Effects option is enabled.

<b>UI Access</b>	N/A
<b>Parameters</b>	None.
<b>Return Value</b>	<p>Integer.</p> <ul style="list-style-type: none"><li>• 1 - Include Port Post Processing Effects is enabled.</li><li>• 0 - Include Port Post Processing Effects is disabled.</li></ul>

<b>Python Syntax</b>	GetIncludePortPostProcessing()
<b>Python Example</b>	<code>oModule.GetIncludePortPostProcessing ()</code>

<b>VB Syntax</b>	GetIncludePortPostProcessing
------------------	------------------------------

<b>VB Example</b>	<code>oModule.GetIncludePortPostProcessing</code>
-------------------	---------------------------------------------------

## GetMultipactionBreakdown

Gets multipaction breakdown from the specified setup.

<b>UI Access</b>	N/A		
<b>Parameters</b>	Name	Type	Description
	<code>&lt;SetupName&gt;</code>	String	Name of specified setup.
<b>Return Value</b>	Array of string containing the multipaction breakdown.		

<b>Python Syntax</b>	<code>GetMultipactionBreakdown(&lt;SetupName&gt;, &lt;Variation&gt;)</code>
<b>Python Example</b>	<code>oModule.GetMultipactionBreakdown("Setup1", "")</code>

<b>VB Syntax</b>	<code>GetMultipactionBreakdown &lt;SetupName&gt;, &lt;Variation&gt;</code>
<b>VB Example</b>	<code>oModule.GetMultipactionBreakdown "Setup1", ""</code>

## GetNetworkDataSolution

Gets matrix data solution from a specified setup.

<b>UI Access</b>	N/A		
<b>Parameters</b>	Name	Type	Description
	<code>&lt;SoluName&gt;</code>	String	Name of solution.

	<VariationKey>	String	Design variation key. Pass empty string for the current nominal variation.
<b>Return Value</b>	String containing matrix data.		

<b>Python Syntax</b>	GetNetworkDataSolution(<SoluName>, <VariationKey>)
<b>Python Example</b>	<code>oModule.GetNetworkDataSolution("Setup1:Sweep1", "")</code>

<b>VB Syntax</b>	GetNetworkDataSolution <SoluName>, <VariationKey>
<b>VB Example</b>	<code>oModule.GetNetworkDataSolution "Setup1:Sweep1", ""</code>

## GetNetworkDataSolutionDefinition

Gets definition of network data solution from a specified setup.

<b>UI Access</b>	N/A		
<b>Parameters</b>	Name	Type	Description
	<SoluName>	String	Name of specified solution.
<b>Return Value</b>	String containing network data definition.		

<b>Python Syntax</b>	GetNetworkDataSolutionDefinition(<SoluName>)
<b>Python Example</b>	<code>oModule.GetNetworkDataSolutionDefinition("Setup1:Sweep1")</code>

---

<b>VB Syntax</b>	GetNetworkDataSolutionDefinition < <i>SoluName</i> >
<b>VB Example</b>	<code>oModule.GetNetworkDataSolutionDefinition "Setup1:Sweep1"</code>

## GetNetworkPostprocSetup

Obtains network post-processing setup.

<b>UI Access</b>	N/A
<b>Parameters</b>	None.
<b>Return Value</b>	String containing post-processing setup.

<b>Python Syntax</b>	GetNetworkPostprocSetup()
<b>Python Example</b>	<code>oModule.GetNetworkPostprocSetup()</code>

<b>VB Syntax</b>	GetNetworkPostprocSetup
<b>VB Example</b>	<code>oModule.GetNetworkPostprocSetup</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.

<b>UI Access</b>	N/A		
<b>Parameters</b>	Name	Type	Description
	<SoluName>	String	Name of specified solution.
<b>Return Value</b>	Array of strings containing frequency values.		

<b>Python Syntax</b>	GetSolveRangeInfo(<SoluName>, [Optional <IncludeRangeFieldSaved>])
<b>Python Example</b>	oModule.GetSolveRangeInfo ("Setup1:Sweep1")

<b>VB Syntax</b>	GetSolveRangeInfo <SoluName>, [Optional <IncludeRangeFieldSaved>]
<b>VB Example</b>	oModule.GetSolveRangeInfo "Setup1:Sweep1"

## GetSourceContexts

Obtains sources currently enabled as context in the Edit Sources dialog Source Context tab.

<b>UI Access</b>	N/A
<b>Parameters</b>	None.
<b>Return Value</b>	Array of enabled source names

<b>Python Syntax</b>	GetSourceContexts()
<b>Python Example</b>	<code>oModule.GetSourceContexts ()</code>

<b>VB Syntax</b>	GetSourceContexts
<b>VB Example</b>	<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")
```

## GetTerminalExcitationType

Obtains the type for terminal excitation in a driven terminal design.

<b>UI Access</b>	N/A
<b>Parameters</b>	None.
<b>Return Value</b>	String containing excitation type.

<b>Python Syntax</b>	GetTerminalExcitationType()
<b>Python Example</b>	<code>oModule.GetTerminalExcitationType ()</code>

<b>VB Syntax</b>	GetTerminalExcitationType
<b>VB Example</b>	<code>oModule.GetTerminalExcitationType</code>

## GetTransientSolveTimes

Gets the transient solution solve time.

<b>UI Access</b>	N/A		
<b>Parameters</b>	Name <code>&lt;SolnName&gt;</code>	Type String	Description Name of specified solution.

	<code>&lt;Variation&gt;</code>	String	Design variation. Pass empty string for the current nominal variation.
<b>Return Value</b>	Array of strings containing solve time.		

<b>Python Syntax</b>	<code>GetTransientSolveTimes(&lt;SolName&gt;, &lt;Variation&gt;)</code>
<b>Python Example</b>	<code>oModule.GetTransientSolveTimes ("Setup1:Transient1", "")</code>

<b>VB Syntax</b>	<code>GetTransientSolveTimes &lt;SolName&gt;, &lt;Variation&gt;</code>
<b>VB Example</b>	<code>oModule.GetTransientSolveTimes "Setup1:Transient1", ""</code>

## GetValidISolutionList

Gets all available solution names that exist in a design.

<b>UI Access</b>	N/A		
<b>Parameters</b>	Name <code>&lt;IncludeImportedSolutions&gt;</code>	Type Boolean	Description Optional. Specifies whether to include imported solutions. Default is false.
<b>Return Value</b>	Array of strings containing solution names.		

<b>Python Syntax</b>	<code>GetValidISolutionList([Optional &lt;IncludeImportedSolutions&gt;])</code>
<b>Python Example</b>	<code>oModule.GetValidISolutionList()</code>

<b>VB Syntax</b>	GetValidISolutionList [Optional <IncludeImportedSolutions>]
<b>VB Example</b>	oModule.GetValidISolutionList

## HasFields

Determines if fields exist for a particular solution.

<b>UI Access</b>	N/A									
<b>Parameters</b>	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>&lt;SolName&gt;</td><td>String</td><td>Name of specified solution.</td></tr><tr><td>&lt;DesignVar&gt;</td><td>String</td><td>Design variation.</td></tr></tbody></table>	Name	Type	Description	<SolName>	String	Name of specified solution.	<DesignVar>	String	Design variation.
Name	Type	Description								
<SolName>	String	Name of specified solution.								
<DesignVar>	String	Design variation.								
<b>Return Value</b>	Integer (1 - True, 0 - False).									

<b>Python Syntax</b>	HasFields(<SolName>, <DesignVar>)
<b>Python Example</b>	oModule.HasFields("Setup1:Sweep1", "x_size=2mm")

<b>VB Syntax</b>	HasFields <SolName>, <DesignVar>
<b>VB Example</b>	oModule.HasFields "Setup1:Sweep1", "x_size=2mm"

## HasMatrixData

Determines if matrix data exists for a particular solution.

<b>UI Access</b>	N/A									
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;SoluName&gt;</td> <td>String</td> <td>Name of specified solution.</td> </tr> <tr> <td>&lt;DesignVar&gt;</td> <td>String</td> <td>Design variation.</td> </tr> </tbody> </table>	Name	Type	Description	<SoluName>	String	Name of specified solution.	<DesignVar>	String	Design variation.
Name	Type	Description								
<SoluName>	String	Name of specified solution.								
<DesignVar>	String	Design variation.								
<b>Return Value</b>	Integer (1 - True, 0 - False).									

<b>Python Syntax</b>	HasMatrixData(<SoluName>, <DesignVar>)
<b>Python Example</b>	oModule.HasMatrixData ("Setup1:Sweep1", "")

<b>VB Syntax</b>	HasMatrixData <SoluName>, <DesignVar>
<b>VB Example</b>	oModule.HasMatrixData "Setup1:Sweep1", ""

## HasMesh

Determines if a current mesh exists for a particular simulation setup, not including the initial mesh.

<b>UI Access</b>	N/A									
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;DesignName&gt;</td> <td>String</td> <td>Name of specified design.</td> </tr> <tr> <td>&lt;DesignVar&gt;</td> <td>String</td> <td>Design variation.</td> </tr> </tbody> </table>	Name	Type	Description	<DesignName>	String	Name of specified design.	<DesignVar>	String	Design variation.
Name	Type	Description								
<DesignName>	String	Name of specified design.								
<DesignVar>	String	Design variation.								
<b>Return Value</b>	Integer (1 - True, 0 - False).									

<b>Python Syntax</b>	HasMesh(<DesignName>, <DesignVar>)
<b>Python Example</b>	<code>oModule.HasMesh ("Setup1", "")</code>

<b>VB Syntax</b>	HasMesh <DesignName>, <DesignVar>
<b>VB Example</b>	<code>oModule.HasMesh "Setup1", ""</code>

## 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.

<b>UI Access</b>	Right-click on <b>Results &gt; Import Solutions...</b> , click on <b>Import Solution...</b> in the opened dialog window.		
<b>Parameters</b>	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.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	ImportSolution(<FileName>, <ImportName>, <SoluArray>)
----------------------	-------------------------------------------------------

<b>Python Example</b>	<pre>oModule.ImportSolution(     "c:\\mydir\\in.s2p", "MeasuredData", ["Sweep1"])</pre>
-----------------------	---------------------------------------------------------------------------------------------

<b>VB Syntax</b>	ImportSolution <FileName>, <ImportName>, <SoluArray>
<b>VB Example</b>	<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.

<b>UI Access</b>	Right-click on <b>Results &gt; Import Solutions...</b> , click on <b>Import Table...</b> in the opened dialog window.		
<b>Parameters</b>	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.
	<CollndepFlags>	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.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	ImportTable(<FileName>, <ImportName>, <TableName>, <IsReallmag>, <ContainsSoluData>, <ColNames>, <CollndepFlags>)
<b>Python Example</b>	<pre>oModule.ImportTable("c:\mydir\mytable.tab",     "ImportData", "Measurements", True, True,     [], [True, True, False, False, False])</pre>

<b>VB Syntax</b>	ImportTable <FileName>, <ImportName>, <TableName>, <IsReallmag>, <ContainsSoluData>, <ColNames>, <CollndepFlags>
<b>VB Example</b>	<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.

<b>UI Access</b>	N/A
------------------	-----

Parameters	Name	Type	Description
	<SoluName>	String	Name of specified solution.
	<DesignVar>	String	Design variable.
<b>Return Value</b>		Specified frequency.	
<b>Return Value</b>		Integer ( 1 - True, 0 - False).	

<b>Python Syntax</b>	IsFieldAvailableAt(<SoluName>, <DesignVar>, <Freq>)
<b>Python Example</b>	oModule.IsFieldAvailableAt ("Setup1:Sweep1", "", 100.0)

<b>VB Syntax</b>	IsFieldAvailableAt <SoluName>, <DesignVar>, <Freq>
<b>VB Example</b>	oModule.IsFieldAvailableAt "Setup1:Sweep1", "", 100.0

## ListMatchingVariations

Gets a list of solved variations that include the specified variable values.

<b>UI Access</b>	N/A
<b>Parameters</b>	Name
	<SoluName>
	<MatchVarNames>
<b>Return Value</b>	<MatchValStrings>
	String
	Array
Name of specified solutions.	
Array of matching variable names.	
Array of matching variable value strings including units.	
<b>Return Value</b>	
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.	

<b>Python Syntax</b>	ListMatchingVariations(<SolName>, <MatchVarNames>, <MatchValStrings>)
<b>Python Example</b>	<pre>oModule.ListMatchingVariations(     "Setup1:LastAdaptive",     ["x_size", "y_size"], ["2mm", "1mm"])</pre>

<b>VB Syntax</b>	ListMatchingVariations <SolName>, <MatchVarNames>, <MatchValStrings>
<b>VB Example</b>	<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.

<b>UI Access</b>	N/A									
<b>Parameters</b>	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>&lt;SolName&gt;</td><td>String</td><td>Name of specified solution.</td></tr><tr><td>&lt;VariableName&gt;</td><td>String</td><td>Name of specified variable.</td></tr></tbody></table>	Name	Type	Description	<SolName>	String	Name of specified solution.	<VariableName>	String	Name of specified variable.
Name	Type	Description								
<SolName>	String	Name of specified solution.								
<VariableName>	String	Name of specified variable.								
<b>Return Value</b>	Array of double precision values in SI units interpreted as the specified variable corresponding to the solved variations.									

---

<b>Python Syntax</b>	ListValuesOfVariable(<SoluName>, <VariableName>)
<b>Python Example</b>	<code>oModule.ListValuesOfVariable("Setup1:LastAdaptive", "x_size")</code>

<b>VB Syntax</b>	ListValuesOfVariable <SoluName>, <VariableName>
<b>VB Example</b>	<code>oModule.ListValuesOfVariable "Setup1:LastAdaptive", "x_size"</code>

## ListVariations

Gets a list of solved variations.

<b>UI Access</b>	N/A						
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;SoluName&gt;</td> <td>String</td> <td>Name of specified solution.</td> </tr> </tbody> </table>	Name	Type	Description	<SoluName>	String	Name of specified solution.
Name	Type	Description					
<SoluName>	String	Name of specified solution.					
<b>Return Value</b>	Array of strings corresponding to solved variations.						

<b>Python Syntax</b>	ListVariations(<SoluName>)
<b>Python Example</b>	<code>oModule.ListVariations ("Setup1 : LastAdaptive")</code>

<b>VB Syntax</b>	ListVariations <SoluName>
<b>VB Example</b>	<code>oModule.ListVariations "Setup1 : LastAdaptive"</code>

## SetExternalExcitations

Can be used instead of the Circuit push excitations workflow to import excitation datasets into Q3D Extractor.

UI Access	N/A.												
Parameters	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>&lt;folderName&gt;</td><td>Array</td><td>Structured array containing the folder path for import. Array ("NAME:filename", &lt;string path&gt;)</td></tr><tr><td>&lt;args&gt;</td><td>Array</td><td>Structured array containing excitation argument arrays. Array (&lt;argArray&gt;, &lt;argArray&gt;, &lt;argArray&gt;...)</td></tr><tr><td>&lt;argArray&gt;</td><td>Array</td><td>Structured array containing parameter arrays. Array (     Array ("NAME:Name", &lt;string object name&gt;)     Array ("NAME:Re", &lt;string path to *.tab file containing real current values&gt;)     Array ("NAME:Im", &lt;string path to *.tab file containing imaginary current values&gt;) )</td></tr></tbody></table>	Name	Type	Description	<folderName>	Array	Structured array containing the folder path for import. Array ("NAME:filename", <string path>)	<args>	Array	Structured array containing excitation argument arrays. Array (<argArray>, <argArray>, <argArray>...)	<argArray>	Array	Structured array containing parameter arrays. Array ( Array ("NAME:Name", <string object name>) Array ("NAME:Re", <string path to *.tab file containing real current values>) Array ("NAME:Im", <string path to *.tab file containing imaginary current values>) )
Name	Type	Description											
<folderName>	Array	Structured array containing the folder path for import. Array ("NAME:filename", <string path>)											
<args>	Array	Structured array containing excitation argument arrays. Array (<argArray>, <argArray>, <argArray>...)											
<argArray>	Array	Structured array containing parameter arrays. Array ( Array ("NAME:Name", <string object name>) Array ("NAME:Re", <string path to *.tab file containing real current values>) Array ("NAME:Im", <string path to *.tab file containing imaginary current values>) )											
Return Value	None.												

**Python Syntax**

```
SetExternalExcitations(<folderName>, <args>)
```

## Python Example

```
oModule.SetExternalExcitations(
[
    "NAME:foldername",
    "F:/Project/Push_Excitation/testPushExcitation"
],
[
[
    [
        [
            "NAME:Name",
            "Box1:PH1",
        ],
        [
            "NAME:Re",
            "F:/Project/Push_Excitation/testPushExcitation/PH1_real.tab"
        ],
        [
            "NAME:Im",
            "F:/Project/Push_Excitation/testPushExcitation/PH1_imag.tab"
        ]
    ],
    [
        [
            [
                "NAME:Name",
                "Box1:PH2",
            ],
            [
                "NAME:Re",
                "F:/Project/Push_Excitation/testPushExcitation/PH2_real.tab"
            ],
            [
                "NAME:Im",
                "F:/Project/Push_Excitation/testPushExcitation/PH2_imag.tab"
            ]
        ],
        [
            [
                "NAME:Name",
                "Box1:PH3"
            ],
            [
                "NAME:Re",
                "F:/Project/Push_Excitation/testPushExcitation/PH3_real.tab"
            ],
            [
                "NAME:Im",
                "F:/Project/Push_Excitation/testPushExcitation/PH3_imag.tab"
            ]
        ]
    ]
]
```

```
[  
  [  
    "NAME:Name",  
    "Box1:PH3",  
  ],  
  [  
    "NAME:Re",  
    "F:/Project/Push_Excitation/testPushExcitation/PH3_real.tab",  
  ],  
  [  
    "NAME:Im",  
    "F:/Project/Push_Excitation/testPushExcitation/PH3_imag.tab",  
  ]  
])
```

<b>VB Syntax</b>	SetExternalExcitations<folderName>, <args>
<b>VB Example</b>	<pre> oModule.SetExternalExcitations Array("NAME:filename", "F:/Project/Push_Excitation/testPushExcitation"), _ Array(Array(Array("NAME:Name", "Box1:PH1", ), Array("NAME:Re", _ "F:/Project/Push_Excitation/testPushExcitation/PH1_real.tab", ), _ Array("NAME:Im", "F:/Project/Push_Excitation/testPushExcitation/PH1_imag.tab", )), _ Array(Array("NAME:Name", "Box1:PH2"), Array("NAME:Re", _ "F:/Project/Push_Excitation/testPushExcitation/PH2_real.tab", ), Array("NAME:Im", _ "F:/Project/Push_Excitation/testPushExcitation/PH2_imag.tab")), Array(Array("NAME:Name", </pre>

```
"Box1:PH3",), _  
Array("NAME:Re", "F:/Project/Push_Excitation/testPushExcitation/PH3_real.tab"),Array  
("NAME:Im", _  
"F:/Project/Push_Excitation/testPushExcitation/PH3_imag.tab")))) _
```

## SetSourceContexts

For Near or Far Field projects for Driven Modal or Driven Terminal Network Analysis Solutions, specifies the port name and all modes/terminals of that port to be enabled as Source Context.

<b>UI Access</b>	<b>HFSS &gt; Fields &gt; Edit Sources.</b>						
<b>Parameters</b>	<table border="1"> <tr> <td>Name</td> <td>Type</td> <td>Description</td> </tr> <tr> <td>&lt;SourceId&gt;</td> <td>Array</td> <td>Name of modes/terminals to be set as source context.</td> </tr> </table>	Name	Type	Description	<SourceId>	Array	Name of modes/terminals to be set as source context.
Name	Type	Description					
<SourceId>	Array	Name of modes/terminals to be set as source context.					
<b>Return Value</b>	None.						

<b>Python Syntax</b>	SetSourceContexts(<SourceId>)
<b>Python Example</b>	<pre>oModule.SetSourceContexts ( [ "Box1_T1", "Box1_T2", "Box1_T3", "Current1", "IncPWave1" ])</pre>

<b>VB Syntax</b>	SetSourceContexts <SourceId>
<b>VB Example</b>	<pre>oModule.SetSourceContexts _</pre>

	Array("Box1_T1", "Box1_T2", "Box1_T3", "Current1", "IncPWave1")
--	-----------------------------------------------------------------

## TDROnReport

Performs a time-domain reflectometry (TDR) analysis on a specified report.

UI Access	Right-click on <b>Results</b> > <b>Perform TDR On Report...</b>		
<b>Parameters</b>	Name	Type	Description
	<ReportName>	String	Name of specified report.
	<IsStep>	Boolean	Specifies the input signal as Step or Impulse.
	<RiseTime>	Double	Specifies rise time with unit in 'ps'.
	<WindowName>	String	Name of window to apply. Possible values are "Rectangular", "Bartlett", "Blackman", "Hamming", "Hanning", "Kaiser", "Welch".
	<WindowWidth>	Integer	Width of window in percentage.
	<KaiserParams>	Double	Optional. If Kaiser window is specified, set corresponding parameters.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	TDROnReport(<ReportName>, <IsStep>, <RiseTime>, <WindowName>, <WindowWidth>, [Optional <Kaiser-Params>])
<b>Python Example</b>	oModule.TDROnReport ("Rept2DRectFreq", True, 1E-12, "Hanning", 100, 1)

<b>VB Syntax</b>	TDROnReport <ReportName>, <IsStep>, <RiseTime>, <WindowName>, <WindowWidth>, [Optional <Kaiser-Params>]
<b>VB Example</b>	oModule.TDROnReport "Rept2DRectFreq", true, 1E-12, "Hanning", 100, 1

# 16 - 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](#)

[AddNamedExpr](#)

[AddNamedExpression](#)

[CalcOp](#)

[CalcStack](#)

[CalcWrite](#)

[CalculatorRead](#)

[CalculatorWrite](#)

[ChangeGeomSettings](#)

[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](#)

## AddMarker[Fields Reporter]

Adds a marker to the current fields plot.

<b>UI Access</b>	Right-click <b>Field Overlays &gt; Plot Fields &gt; Marker &gt; Add Marker</b> .								
<b>Parameters</b>	<table border="1"><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;Position&gt;</td><td>Array</td><td>Array containing X, Y and Z coordinates.</td></tr></table>			Name	Type	Description	<Position>	Array	Array containing X, Y and Z coordinates.
Name	Type	Description							
<Position>	Array	Array containing X, Y and Z coordinates.							
<b>Return Value</b>	None.								

<b>Python Syntax</b>	AddMarker(<Position>)
<b>Python Example</b>	<pre>oModule.AddMarker([     "-267.007756778494mil",     "-640.898759461403mil",     "9.09494701772928e-13mil"])</pre>

<b>VB Syntax</b>	AddMarker <Position>
<b>VB Example</b>	<pre> oModule.AddMarker _     Array("-267.007756778494mil", _         "-640.898759461403mil", _         "9.09494701772928e-13mil") </pre>

## AddMarkerToPlot

Adds a marker to a trace on a named field plot.

<b>UI Access</b>	N/A									
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;Location&gt;</td> <td>Array</td> <td>Array of strings containing X,Y, and Z coordinates for the marker.</td> </tr> <tr> <td>&lt;PlotName&gt;</td> <td>String</td> <td>Name of the field plot.</td> </tr> </tbody> </table>	Name	Type	Description	<Location>	Array	Array of strings containing X,Y, and Z coordinates for the marker.	<PlotName>	String	Name of the field plot.
Name	Type	Description								
<Location>	Array	Array of strings containing X,Y, and Z coordinates for the marker.								
<PlotName>	String	Name of the field plot.								
<b>Return Value</b>	None.									

<b>Python Syntax</b>	AddMarkerToPlot( <Location>, <PlotName>)
<b>Python Example</b>	<pre> oModule.AddMarkerToPlot ([ "0.290455877780914in", "-0.616900205612183in",     "1.77635683940025e-015in"],     "Mag_H1")  oModule.AddMarkerToPlot ([ "-0.317279517650604in", </pre>

	"1.22481322288513in", "0in"], "Mag_H1")
--	--------------------------------------------

<b>VB Syntax</b>	AddMarkerToPlot <Location>, <PlotName>
<b>VB Example</b>	<pre>oModule.AddMarkerToPlot _     Array("0.290455877780914in", _         "-0.616900205612183in", "1.77635683940025e-015in"), _         "Mag_E1" oModule.AddMarkerToPlot _     Array("-0.317279517650604in", _         "1.22481322288513in", "0in"), _         "Mag_E1"</pre>

## AddNamedExpr

Creates a named expression using the expression at the top of the stack.

<b>UI Access</b>	Click <b>Add</b> in the <b>Fields Calculator</b> dialogue.		
<b>Parameters</b>	Name <ExprName>	Type String	Description Name for the new named expression.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	AddNamedExpr(<ExprName>)
<b>Python Example</b>	<code>oModule.AddNamedExpr ("Mag_JxE")</code>

<b>VB Syntax</b>	AddNamedExpr <ExprName>
<b>VB Example</b>	<code>oModule.AddNamedExpr "Mag_JxE"</code>

## AddNamedExpression

Creates a named expression using the expression at the top of the stack.

<b>UI Access</b>	Click <b>Add</b> in the <b>Fields Calculator</b> dialogue.									
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;ExprName&gt;</td> <td>String</td> <td>Name for the new named expression.</td> </tr> <tr> <td>&lt;FieldType&gt;</td> <td>String</td> <td>Type of field.</td> </tr> </tbody> </table>	Name	Type	Description	<ExprName>	String	Name for the new named expression.	<FieldType>	String	Type of field.
Name	Type	Description								
<ExprName>	String	Name for the new named expression.								
<FieldType>	String	Type of field.								
<b>Return Value</b>	None.									

<b>Python Syntax</b>	AddNamedExpression(<ExprName>, <FieldType>)
<b>Python Example</b>	<code>oModule.AddNamedExpression ("Mag_JxE", "Fields")</code>

<b>VB Syntax</b>	AddNamedExpression <ExprName>, <FieldType>
<b>VB Example</b>	<code>oModule.AddNamedExpression "Mag_JxE", "Fields"</code>

## CalcOp

Performs a calculator operation.

<b>UI Access</b>	Operation commands like <b>Mag</b> , <b>+</b> , etc.		
<b>Parameters</b>	Name <i>&lt;OpString&gt;</i>	Type String	Description The text on the corresponding calculator button.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	CalcOp( <i>&lt;OpString&gt;</i> )
<b>Python Example</b>	<code>oModule.CalcOp ("+")</code>

<b>VB Syntax</b>	CalcOp <i>&lt;OpString&gt;</i>
<b>VB Example</b>	<code>oModule.CalcOp "+"</code>

## CalcStack

Performs an operation on the stack.

<b>UI Access</b>	Stack operation buttons such as <b>Push</b> and <b>Pop</b> .		
<b>Parameters</b>	Name <i>&lt;OpString&gt;</i>	Type String	Description The text on the corresponding calculator button.

<b>Return Value</b>	None.
---------------------	-------

<b>Python Syntax</b>	CalcStack(<OpString>)
<b>Python Example</b>	oModule.CalcStack("push")

<b>VB Syntax</b>	CalcStack <OpString>
<b>VB Example</b>	oModule.CalcStack "push"

## CalcWrite

Writes contents of top register to file.

<b>UI Access</b>	N/A												
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;FilePath&gt;</td> <td>String</td> <td>Path to output and including name of output register file.</td> </tr> <tr> <td>&lt;SoluName&gt;</td> <td>String</td> <td>Name of solution.</td> </tr> <tr> <td>&lt;VariablesArray&gt;</td> <td>Array</td> <td>Array of variable names, value pairs.</td> </tr> </tbody> </table>	Name	Type	Description	<FilePath>	String	Path to output and including name of output register file.	<SoluName>	String	Name of solution.	<VariablesArray>	Array	Array of variable names, value pairs.
Name	Type	Description											
<FilePath>	String	Path to output and including name of output register file.											
<SoluName>	String	Name of solution.											
<VariablesArray>	Array	Array of variable names, value pairs.											
<b>Return Value</b>	None.												

<b>Python Syntax</b>	CalcWrite(<FilePath>, <SoluName>, <VariablesArray>)
<b>Python Example</b>	<pre>oModule.CalcWrite("C:\Ansoft\smoothedTemp.fld", "Setup1 : LastAdaptive", ["\$conductivity:=", "50000000"])</pre>

<b>VB Syntax</b>	CalcWrite <FilePath>, <SoluName>, <VariablesArray>
<b>VB Example</b>	<pre>oModule.CalcWrite "C:\Ansoft\smoothedTemp.fld", _ "Setup1 : LastAdaptive", _ Array("\$conductivity:=", "50000000")</pre>

## CalculatorRead

Gets a register file and applies it to the calculator stack.

<b>UI Access</b>	Click <b>Read...</b> in the <b>Fields Calculator</b> dialog.															
<b>Parameters</b>	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>&lt;FileName&gt;</td><td>String</td><td>Path to and including name of input register file.</td></tr><tr><td>&lt;SoluName&gt;</td><td>String</td><td>Specified solution to read in.</td></tr><tr><td>&lt;FieldType&gt;</td><td>String</td><td>Type of specified field.</td></tr><tr><td>&lt;VarArray&gt;</td><td>Array</td><td>Array of variable names, value pairs.</td></tr></tbody></table>	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.
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.														
<b>Return Value</b>	None.															

<b>Python Syntax</b>	CalculatorRead(<FileName>, <SoluName>, <FieldType>, <VarArray>)
<b>Python Example</b>	<pre>oModule.CalculatorRead( "c:\example.reg", "Setup1: LastAdaptive", "Fields",</pre>

	["Freq:=", "10GHz", "Phase:=", "0deg"])
--	-----------------------------------------

<b>VB Syntax</b>	CalculatorRead <FileName>, <SoluName>, <FieldType>, <VarArray>
<b>VB Example</b>	<pre> oModule.CalculatorRead _ "c:\example.reg", _ "Setup1: LastAdaptive", "Fields", _ Array("Freq:=", "10GHz", "Phase:=", "0deg") </pre>

## CalculatorWrite

Writes contents of top register to file.

<b>UI Access</b>	Click <b>Write...</b> in the <b>Fields Calculator</b> dialog.												
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;OutputFilePath&gt;</td> <td>String</td> <td>Path to and including name of output register file.</td> </tr> <tr> <td>&lt;SoluNameArray&gt;</td> <td>Array</td> <td>Array of strings containing solution names.</td> </tr> <tr> <td>&lt;VarArray&gt;</td> <td>Array</td> <td>Array of variable names, value pairs.</td> </tr> </tbody> </table>	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.
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.											
<b>Return Value</b>	None.												

<b>Python Syntax</b>	CalculatorWrite(<OutputFilePath>, <SoluNameArray>, <VarArray>)
<b>Python Example</b>	<pre> oModule.CalculatorWrite("c:\test.reg", ["Solution:=", "Setup1 : LastAdaptive"], ["Freq:=", "1GHz", "Phase:=", "0deg"]) </pre>

	)
--	---

<b>VB Syntax</b>	CalculatorWrite <OutputFilePath>, <SoluNameArray>, <VarArray>
<b>VB Example</b>	<pre>oModule.CalculatorWrite "c:\test.reg", _     Array("Solution:=", "Setup1 : LastAdaptive"),_     Array("Freq:=", "1GHz", "Phase:=", "0deg")</pre>

## ChangeGeomSettings

Changes the line discretization setting.

<b>UI Access</b>	In the <b>Fields Calculator</b> dialog box, click on <b>Geom Settings...</b>						
<b>Parameters</b>	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>&lt;LineDiscr&gt;</td><td>Integer</td><td>Line discretization value.</td></tr></tbody></table>	Name	Type	Description	<LineDiscr>	Integer	Line discretization value.
Name	Type	Description					
<LineDiscr>	Integer	Line discretization value.					
<b>Return Value</b>	None.						

<b>Python Syntax</b>	ChangeGeomSettings(<LineDiscr>)
<b>Python Example</b>	<pre>oModule.ChangeGeomSettings (500)</pre>

<b>VB Syntax</b>	ChangeGeomSettings <LineDiscr>
------------------	--------------------------------

<b>VB Example</b>	<code>oModule.ChangeGeomSettings 500</code>
-------------------	---------------------------------------------

## ClcEval

Evaluates the expression at the top of the stack using the provided solution name and variable values.

<b>UI Access</b>	Click <b>Eval</b> in the <b>Fields Calculator</b> dialog.												
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><code>&lt;SolName&gt;</code></td> <td>String</td> <td>Name of specified solution.</td> </tr> <tr> <td><code>&lt;VarArray&gt;</code></td> <td>Array</td> <td>Array of variable name, value pairs.</td> </tr> <tr> <td><code>&lt;FieldType&gt;</code></td> <td>String</td> <td>Optional. Type of specified field.</td> </tr> </tbody> </table>	Name	Type	Description	<code>&lt;SolName&gt;</code>	String	Name of specified solution.	<code>&lt;VarArray&gt;</code>	Array	Array of variable name, value pairs.	<code>&lt;FieldType&gt;</code>	String	Optional. Type of specified field.
Name	Type	Description											
<code>&lt;SolName&gt;</code>	String	Name of specified solution.											
<code>&lt;VarArray&gt;</code>	Array	Array of variable name, value pairs.											
<code>&lt;FieldType&gt;</code>	String	Optional. Type of specified field.											
<b>Return Value</b>	None.												

<b>Python Syntax</b>	<code>ClcEval(&lt;SolName&gt;, &lt;VarArray&gt;, [Optional &lt;FieldType&gt;])</code>
<b>Python Example</b>	<pre>oModule.ClcEval(     "Setup1: LastAdaptive",     ["Freq:=", "10GHz",      "Phase:=", "0deg"])</pre>

<b>VB Syntax</b>	<code>ClcEval &lt;SolName&gt;, &lt;VarArray&gt;, [Optional &lt;FieldType&gt;]</code>
<b>VB Example</b>	<pre>oModule.ClcEval "Setup1: LastAdaptive", _     Array ("Freq:=", "10GHz", _            "Phase:=", "0deg")</pre>

## ClcMaterial

Performs a material operation on the top stack element.

<b>UI Access</b>	Click <b>Matl...</b> in the <b>Fields Calculator</b> dialog.		
<b>Parameters</b>	Name	Type	Description
	<MatString>	String	The material property to apply.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	ClcMaterial(<MatString>, <OpString>)
<b>Python Example</b>	oModule.ClcMaterial("Permeability (mu)" "mult")

<b>VB Syntax</b>	ClcMaterial <MatString>, <OpString>
<b>VB Example</b>	oModule.ClcMaterial "Permeability (mu)" "mult"

## ClcMaterialValue

Shows the value of the material property without performing any operation.

<b>UI Access</b>	Select <b>None</b> in the <b>Material Operation</b> dialog.		
<b>Parameters</b>	Name	Type	Description
	<MaterialName>	String	Name of specified material property.

<b>Return Value</b>	None.
---------------------	-------

<b>Python Syntax</b>	ClcMaterialValue(<MaterialName>)
<b>Python Example</b>	oModule.ClcMaterialValue("Mass Density")

<b>VB Syntax</b>	ClcMaterialValue <MaterialName>
<b>VB Example</b>	oModule.ClcMaterialValue "Mass Density"

## ClearAllMarkers[Fields Reporter]

Clears all markers in the current fields overlay plot.

<b>UI Access</b>	Right-click <b>Field Overlays &gt; Plot Fields &gt; Marker &gt; Clear All.</b>
<b>Parameters</b>	None.
<b>Return Value</b>	None.

<b>Python Syntax</b>	ClearAllMarkers()
<b>Python Example</b>	oModule.ClearAllMarkers ()

<b>VB Syntax</b>	ClearAllMarkers
<b>VB Example</b>	oModule.ClearAllMarkers

## ClearAllNamedExpr

Clears all user-defined named expressions from the list.

<b>UI Access</b>	Click <b>ClearAll</b> in the <b>Fields Calculator</b> dialog.
<b>Parameters</b>	None.
<b>Return Value</b>	None.

<b>Python Syntax</b>	ClearAllNamedExpr()
<b>Python Example</b>	<code>oModule.ClearAllNamedExpr ()</code>

<b>VB Syntax</b>	ClearAllNamedExpr
<b>VB Example</b>	<code>oModule.ClearAllNamedExpr</code>

## CopyNamedExprToStack

Copies the named expression selected to the calculator stack.

<b>UI Access</b>	Select a named expression and then click <b>Copy to stack</b> .						
<b>Parameters</b>	<table border="1"><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><code>&lt;ExprName&gt;</code></td><td>String</td><td>Name of the expression to be copied to the top of the stack.</td></tr></table>	Name	Type	Description	<code>&lt;ExprName&gt;</code>	String	Name of the expression to be copied to the top of the stack.
Name	Type	Description					
<code>&lt;ExprName&gt;</code>	String	Name of the expression to be copied to the top of the stack.					
<b>Return Value</b>	None.						

<b>Python Syntax</b>	CopyNamedExprToStack(<ExprName>)
<b>Python Example</b>	<code>oModule.CopyNamedExprToStack ("Mag_JxE")</code>

<b>VB Syntax</b>	CopyNamedExprToStack <ExprName>
<b>VB Example</b>	<code>oModule.CopyNamedExprToStack "Mag_JxE"</code>

## CreateFieldPlot

**Note:** Use in conjunction with "["GetGeometryIdsForNetLayerCombinations" on page 9-86](#) and "["GetGeometryIdsForAllNetLayerCombinations" on page 9-87](#).

- GetGeometryIdsForNetLayerCombinations returns ID numbers of all faces and edges of the active design related to the current target combination of nets and layers.
- GetGeometryIdsForAllNetLayerCombinations returns ID numbers for all possible combinations of nets and layers it is possible to choose.

**Use:** Creates a field/mesh plot "Field" or a visual ray trace ("VRT") plot.

**Command:** **HFSS>Fields>Plot Fields><field\_quantity>**

**Command:** **Maxwell3D or Maxwell2D>Fields>Fields><field\_quantity>**

**Command:** **Maxwell3D>Fields>Fields>Field Line Trace** (for 3D Electrostatic designs - see [Examples](#) below)

**Command:** **Icepak>Fields>Plot Fields><field\_quantity>**

*Syntax:* CreateFieldPlot <PlotParameterArray> ["Field" | "VRT"]

*Return Value:* None

*Parameters:* <PlotParameterArray> "Field"

```
Array ("NAME:<PlotName>",
      "SolutionName:=", <string>,
      "QuantityName:=", <string>,
      "PlotFolder:=", <string>,
      "UserSpecifyName:=", <int>,
      "UserSpecifyFolder:=", <int>,
      "IntrinsicVar:=", <string>,
      "PlotGeomInfo:=", <PlotGeomArray>,
      "FilterBoxes:=", <FilterBoxArray>,
      <PlotOnPointsSettings>,
      <PlotOnLineSettings>,
      <PlotOnSurfaceSettings>,
      <PlotOnVolumeSettings>)
```

SolutionName

Name of the solution setup and solution formatted as:

```
"<SolveSetupName>: <WhichSolution>";
```

where <WhichSolution> can be "Adaptive\_<n>",  
"LastAdaptive", or "PortOnly".

For example: "Setup1 : Adaptive\_2"

HFSS and Maxwell require a space on either side of the ':' character. If it is missing, the plot will not be created.

QuantityName

*Type of plot to create. Possible values include:*

**Mesh plots:** "Mesh"

**Field plots (HFSS) include:**

```
"Mag_E", "Mag_H", "Mag_Jvol", "Mag_Jsurf",
"ComplexMag_E", "ComplexMag_H", "ComplexMag_Jvol",
"ComplexMag_Jsurf", "Vector_E", "Vector_H",
"Vector_Jvol", "Vector_Jsurf", "Vector_RealPoynting",
"Local_SAR", "Average_SAR"
```

**Field plots (Maxwell) include:**

```
"Mag_E", "Mag_H", "E_Vector", "H_Vector", "Mag_J", "J_Vector",
"ComplexMag_J", "Mag_Jc", "Jc_Vector", "ComplexMag_Jc",
"Energy", "coEnergy", "appEnergy", "Ohmic_Loss", "Total_Loss",
""Hysteresis_Loss", "Dielectric_Loss", "Temperature", "Volume_Force_Density", Average_
Surface_Loss_Density
"Surface_Force_Density", "Demag_Coef", "QuantityName_FieldLineTrace", "Surface_Loss_
Density", QSurf
```

**Field plots (Mechanical) include - (dependent on solution type):**

```
"Mag_Displacement", "Displacement_Vector", "Temperature", "HeatFlux",
"Mag_HeatFlux", "Surface Loss Density", "Volume Loss Density",
"Linked Heat Transfer Coefficient", "Equivalent Stress"
```

**PlotFolder**

*Name of the folder to which the plot should be added. (Values vary with the design type.) Some possible values include:*

```
"E Field", "H Field", "Jvol", "Jsurf", "SAR Field", "Jc", "Surface-Loss", QSurf, Temperature, Energy, Average-Surface-Loss-Density, "Dielectric_Loss", "MeshPlots", "Heat Flux", and "Displacement".
```

**UserSpecifyName**

0 if default name for plot is used, 1 otherwise.

Not needed. <PlotName> will be respected regardless of whether 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'"

## IntrinsicVar (*Maxwell*)

Formatted string that specifies the skew slice number at which to make the plot.

For example: "IntrinsicVar:=" , "Slice='2' Time='\0.00020000000000000001s'"

## PlotGeomInfo

Creating field plot on selected layer-net pairs:

For example, Maxwell: "PlotGeomInfo :=" , [1,"Volume","ObjList",2,"LC1\_1:Top","LC1\_1:Top#L1"]

HFSS example with "Plot on surface" option is not checked

```
"PlotGeomInfo:=" , [1,"Volume","LayerNets",2, "Top",2,"net1","net2",_
"Ground",1,"GND"]
```

The command creates field plot on 2 layer-nets combinations. It starts with type "Volume", and subtype "LayerNets", followed by the number of layer-nets combinations which is 2 in this example.

The two layer-nets combinations are (1) "Top" layer, 2 nets, net names are "net1" and "net2", i.e., [Top, net1] pair, and [Top, net2] pair. (2) "Ground" layer and 1 net, net name is "GND", i.e., [Ground, GND] pair.

HFSS example with "Plot on surface" option is checked

```
"PlotGeomInfo:=" , [1,"Surface","LayerNetsExtFace",2,"Top",2,"net1","net2",_
"Ground",1,"GND"],
```

The command creates field plot on 2 layer-nets combinations. It starts with type "Surface", and subtype "LayerNetsExtFace", followed by the number of layer-nets combinations which is 2 in this example.

The two layer-nets combinations are (1) "Top" layer, 2 nets, net names are "net1" and "net2", i.e., [Top, net1] pair, and [Top, net2] pair. (2) "Ground" layer and 1 net, net name is "GND", i.e., [Ground, GND] pair.

Limitation:

No Layer/Nets name is supported for field plot command recording

Should be able to add it based on this change.

No Layer/Nets name support for pure Layout design field plot command.

```
<PlotGeomArray>
  Array(<NumGeomTypes>, <GeomTypeData>,
    <GeomTypeData>, ...)
  For example: Array(4, "Volume", "ObjList", 1, "Box1",
    "Surface", "FacesList", 1, "12", "Line", 1,
    "Polyline1", "Point", 2, "Point1", "Point2")
```

```
<NumGeomTypes>
  Type: <int>
  Number of different geometry types (volume, surface, line, point)
  plotted on at the same time.
```

```
<GeomTypeData>
```

<GeomType>, <ListType>, <NumIDs>, <ID>, <ID>, ...)

<GeomType>

Type: <string>

Possible values are "Volume", "Surface", "Line", "Point".

<ListType>

Type: <string>

Possible values are "ObjList", or "FacesList".

These are used for the GeomType of "Line" or "Point".

<NumIDs>

Type: <int>

Number of IDs or object names that will follow.

<ID>

Type: <int> or <string>

*ID of a face or name of an object, line, or point on which to plot.*

<FilterBoxArray>

*Array of names of objects to use to restrict the plot range.*

Array(<NumFilters>, <ObjName>, <ObjName>, ...)

Example: Array(1, "Box1")

Example: Array(0) *no filtering*

<PlotOnPointSettings>

```
    Array("NAME:PlotOnPointSettings",
          "PlotMarker:=", <bool>,
          "PlotArrow:=", <bool>)
```

<PlotOnLineSettings>

```
    Array("NAME:PlotOnLineSettings",
          Array("NAME:LineSettingsID",
                "Width:=", <int>,
                "Style:=", <string>),
                "IsoValType:=", <string>,
                "ArrowUniform:=", <bool>,
                "NumofArrow:=", <int>)
```

Style

*Possible values are "Cylinder", "Solid", "Dashdash",*

"Dotdot", "Dotdash"

IsoValType

Possible values are "Tone", "Fringe", "Gourard"

<PlotOnSurfaceSettings>

```
    Array("NAME:PlotOnSurfaceSettings",
          "Filled:=", <bool>,
          "IsoValType:=", <string>,
          "SmoothShade:=", <bool>,
          "AddGrid:=", <bool>,
          "MapTransparency:=", <bool>,
          "Transparency:=", <doubl.e>,
          "ArrowUniform:=", <bool>
          "ArrowSpacing:=", <double>
          "GridColor:=", Array(<int>, <int>, <int>))
```

IsoValType

Possible values are: "Tone", "Line", "Fringe", "Gourard"

GridColor

Array containing the R, G, B components of the color. Components should be in the range 0 to 255.

```
<PlotOnVolumeSettings>
    Array("NAME:PlotOnVolumeSettings",
        "PlotIsoSurface:=", <bool>,
        "CloudDensity:=", <double>,
        "PointSize:=", <int>,
        "ArrowUniform:=", <bool>,
        "ArrowSpacing:=", <double>)
```

### **Example Field Plot:**

```
oModule.CreateFieldPlot Array ("NAME:Mag_E1", _
    "SolutionName:=", "Setup1 : LastAdaptive", _
    "QuantityName:=", "Mag_E", _
    "PlotFolder:=", "E Field1", _
    "UserSpecifyName:=", 0, _
    "UserSpecifyFolder:=", 0, _
    "IntrinsicVar:=", "Freq='1GHz' Phase='0deg'", _
    "PlotGeomInfo:=", Array( 1, "Surface", _
```

```
"FacesList", 1, "7"), _  
"FilterBoxes:=", Array(0),  
Array("NAME:PlotOnSurfaceSettings", _  
    "Filled:=", false, _  
    "IsoValType:=", "Fringe", _  
    "SmoothShade:=", true, _  
    "AddGrid:=", false, _  
    "MapTransparency:=", true, _  
    "Transparency:=", 0, _  
    "ArrowUniform:=", true, _  
    "ArrowSpacing:=", 0.100000001490116, _  
    "GridColor:=", Array(255, 255, 255)))
```

### Example Demag\_Coef Field Plot:

```
oModule.CreateFieldPlot Array("NAME:Demag_Coef2", "SolutionName:=", _  
    "Setup1 : Transient", "UserSpecifyName:=", 0, "UserSpecifyFolder:=", 0, _  
    "QuantityName:=", "Demag_Coef", "PlotFolder:=", "Demag-Coef", "StreamlinePlot:=", _  
    false, "AdjacentSidePlot:=", false, "FullModelPlot:=", false, "IntrinsicVar:=", _  
    "Time=" & Chr(39) & "0s" & Chr(39) & "", "PlotGeomInfo:=", Array( 1, _  
    "Volume", "ObjList", 1, "Mag2_0"), "FilterBoxes:=", Array(1, "Mag1_0"), _  
    Array("NAME:PlotOnVolumeSettings", "PlotIsoSurface:=", true, "PointSize:=", 1, _
```

```
"Refinement:=", 0, "CloudSpacing:=", 0.5, "CloudMinSpacing:=", -1, _  
"CloudMaxSpacing:=", -1, Array( "NAME:Arrow3DSpacingSettings", "ArrowUniform:=", _  
true, "ArrowSpacing:=", 0, "MinArrowSpacing:=", 0, "MaxArrowSpacing:=", 0)), _  
"EnableGaussianSmoothing:=", false), "Field"
```

*Parameters:* <PlotParameterArray> "VRT"

```
Array("NAME:<PlotName>",  
"SolutionName:=", <string>, _  
"QuantityName:=", <string>, _  
"PlotFolder:=", <string>, _  
"UserSpecifyName:=", <int>, _  
"UserSpecifyFolder:=", <int>, _  
"IntrinsicVar:=", <string>, _  
"MaxFrequency:=", "<Number><FreqUnits>", _  
"LaunchFrom:=", ["Launch from custom", | "LaunchFromPointID:=", ] _  
18007, "RayDensity:=", <int>, _  
"NumberBounces:=", <int>, _  
"Multi-Bounce Ray Density Control:=", <Boolean>, _  
"MBRD Max sub divisions:=", <int>, _  
"ShootFilterType:=", ["All Rays" | _
```

```
"Rays by index", "start index:=", <int>, "stop index:=", <int>, |_
"index step:=", <int> "Rays in box", "FilterBoxID:=", <objID> | _
"Single ray", "Ray elevation:=", "<real>deg", "Ray azimuth:=", "<real>deg"
"),
"VRT"
```

### Example VRT Plot:

```
Dim oAnsoftApp
Dim oDesktop
Dim oProject
Dim oDesign
Dim oEditor
Dim oModule
Set oAnsoftApp = CreateObject("Ansoft.ElectronicsDesktop")
Set oDesktop = oAnsoftApp.GetAppDesktop()
oDesktop.RestoreWindow
Set oProject = oDesktop.SetActiveProject("abrams_1p5")
Set oDesign = oProject.SetActiveDesign("UseSbr")
Set oModule = oDesign.GetModule("FieldsReporter")
oModule.CreateFieldPlot Array("NAME:VRT_Plot1", _
"SolutionName:=", "Setup1 : LastAdaptive", _
```

```
"QuantityName:=", "QuantityName_SBR", _
"PlotFolder:=", "Visual Ray Trace SBR", "UserSpecifyName:=", 0, _
"UserSpecifyFolder:=", 0, "IntrinsicVar:="", "", "MaxFrequency:=", "1GHz", _
"LaunchFrom:=", "Launch from custom", "LaunchFromPointID:=", 18007, _
"RayDensity:=", 2, "NumberBounces:=", 5, "Multi-Bounce Ray Density Control:=", _
false, "MBRD Max sub divisions:=", 2, "ShootFilterType:=", "All Rays"), "VRT"
```

**VB Example:**

```
oModule.CreateFieldPlot Array("NAME:Mag_E1", _
"SolutionName:=", "Setup1 : LastAdaptive", _
"QuantityName:=", "Mag_E", _
"PlotFolder:=", "E Field1", _
>UserSpecifyName:=", 0, _
>UserSpecifyFolder:=", 0, _
>IntrinsicVar:=", "Freq='1GHz' Phase='0deg'",_
"PlotGeomInfo:=", Array( 1, "Surface",_
    "FacesList", 1, "7"),_
"FilterBoxes:=", Array(0),
Array("NAME:PlotOnSurfaceSettings", _
    "Filled:=", false, _ "IsoValType:=", "Fringe", _
```

```
"SmoothShade:=", true, _  
"AddGrid:=", false, _  
"MapTransparency:=", true, _  
"Transparency:=", 0, _  
"ArrowUniform:=", true, _  
"ArrowSpacing:=", 0.100000001490116, _  
"GridColor:=", Array(255, 255, 255))), "Field"
```

## Example Electron Density Plot

```
oModule = oDesign.GetModule("FieldsReporter")  
oModule.CreateFieldPlot(  
[  
    "NAME:Electron_Density3",  
    "SolutionName:=" , "AIR : RFDischarge",  
    "UserSpecifyName:=" , 0,  
    "UserSpecifyFolder:=" , 0,  
    "QuantityName:=" , "Electron_Density",  
    "PlotFolder:=" , "RF Discharge Fields",  
    "StreamlinePlot:=" , False,  
    "AdjacentSidePlot:=" , False,  
    "FullModelPlot:=" , False,
```

```
"IntrinsicVar:=" , "Freq=\'0.2000000000000001GHz\' GasPressure=\'0.02kPascal\'",
"PlotGeomInfo:=" , [1,"Surface","FacesList",1,"48"],
"FilterBoxes:=" , [0],
[
    "NAME:PlotOnSurfaceSettings",
    "Filled:=" , False,
    "IsoValType:=" , "Tone",
    "AddGrid:=" , False,
    "MapTransparency:=" , True,
    "Refinement:=" , 0,
    "Transparency:=" , 0,
    "SmoothingLevel:=" , 0,
    "ShadingType:=" , 0,
    [
        "NAME:Arrow3DSpacingSettings",
        "ArrowUniform:=" , True,
        "ArrowSpacing:=" , 0,
        "MinArrowSpacing:=" , 0,
        "MaxArrowSpacing:=" , 0
    ],
]
```

```
    "GridColor:=", [255,255,255]
],
"EnableGaussianSmoothing:=", False,
"SurfaceOnly:=", False
], "Field")
```

## Example Mesh Plot

```
Dim oAnsoftApp
Dim oDesktop
Dim oProject
Dim oDesign
Dim oEditor
Dim oModule
Set oAnsoftApp = CreateObject("Ansoft.ElectronicsDesktop")
Set oDesktop = oAnsoftApp.GetAppDesktop()
oDesktop.RestoreWindow
Set oProject = oDesktop.SetActiveProject("abrams_1p5")
Set oDesign = oProject.SetActiveDesign("UseSbr")
Set oModule = oDesign.GetModule("FieldsReporter")
oModule.CreateFieldPlot Array("NAME:Mesh1", "SolutionName:=", "Setup1 : LastAdaptive", _
"QuantityName:=", "Mesh", "PlotFolder:=", "MeshPlots", "FieldType:=", "Fields", _
```

```
"UserSpecifyName:=", 0, "UserSpecifyFolder:=", 0, "StreamlinePlot:=", false, _  
"IntrinsicVar:=", "Freq=" & Chr(39) & "2GHz" & Chr(39) & " Phase=" _  
& Chr(39) & "0deg" & Chr(39) & "" & "", "PlotGeomInfo:=", Array(1, _  
"Surface", "FacesList", 1, "10568"), "FilterBoxes:=", Array(0), "Real time mode:=", _  
true, Array("NAME:MeshSettings", "Scale factor:=", 100, "Transparency:=", 0, _  
"Mesh type:=", "Shaded", "Surface only:=", true, "Add grid:=", true, "Refinement:=", _  
0, "Use geometry color:=", true, "Mesh line color:=", Array(0, 0, 255), _  
"Filled color:=", Array( 255, 255, 255))), "Field"
```

### VB Example: Maxwell Plot with Skew Slice

```
oModule.CreateFieldPlot Array("NAME:energy2", "SolutionName:=", "Setup1 : Transient", "User-  
SpecifyName:=", 0, "UserSpecifyFolder:=", 0, "QuantityName:=", "energy", "PlotFolder:=",  
"Energy", "StreamlinePlot:=", false, "AdjacentSidePlot:=", false, "FullModelPlot:=", false,  
"IntrinsicVar:=", "Slice=" & Chr(39) & "2" & Chr(39) & " Time=" & Chr(39) &  
"0.00020000000000000001s" & Chr(39) & "",...)
```

### Python Example: Maxwell Plot with Skew Slice

```
oModule.CreateFieldPlot(  
[  
    "NAME:Mesh1",  
    "SolutionName:=", "Setup1 : Transient",  
    "UserSpecifyName:=", 0,
```

```
"UserSpecifyFolder:=", 0,
"QuantityName:=" "Mesh",
"PlotFolder:=", "MeshPlots",
"FieldType:=", "Fields",
"StreamlinePlot:=" , False,
"AdjacentSidePlot:=" , False,
"FullModelPlot:=" , False,
"IntrinsicVar:=", "Slice=\'2\' Time=\'0.00020000000000000001s\'",
"PlotGeomInfo:=", [1,"Surface","FacesList",1,"2955"],
"FilterBoxes:=", [0],
...]
```

### Python Example: Maxwell Average\_Surface\_Loss\_Density Plot

```
oModule.CreateFieldPlot(
[
    "NAME:Average_Surface_Loss_Density1",
    "SolutionName:=" , "Setup1 : Transient",
    "UserSpecifyName:=" , 0,
    "UserSpecifyFolder:=" , 0,
    "QuantityName:=" , "Average_Surface_Loss_Density",
    "PlotFolder:=" , "Average-Surface-Loss-Density",
```

```
"StreamlinePlot:=", False,  
"AdjacentSidePlot:=", False,  
"FullModelPlot:=", False,  
"IntrinsicVar:=", "Time='4us'\ Time0='0s'",  
"PlotGeomInfo:=", [1,"Surface","FacesList",1,"6"],  
...  
])
```

### Python Examples: Maxwell ACConduction Design Plot

```
oModule.CreateFieldPlot(  
[  
    "NAME:Mag_Jc1",  
    "SolutionName:=", "Setup1 : LastAdaptive",  
    "UserSpecifyName:=", 0,  
    "UserSpecifyFolder:=", 0,  
    "QuantityName:=", "Mag_Jc",  
    "PlotFolder:=", "Jc",  
    "StreamlinePlot:=", False,  
    "AdjacentSidePlot:=", False,  
    "FullModelPlot:=", False,
```

```
"IntrinsicVar:=", "Freq=\'1000000Hz\' Phase=\'0deg\'",
"PlotGeomInfo:=", [1,"Volume","ObjList",1,"Box1"],
"FilterBoxes:=", [1,""],
[
    "NAME:PlotOnVolumeSettings",
    "PlotIsoSurface:=", True,
    "PointSize:=", 1,
    "Refinement:=", 0,
    "CloudSpacing:=", 0.5,
    "CloudMinSpacing:=", -1,
    "CloudMaxSpacing:=", -1,
    "ShadingType:=", 0,
    [
        "NAME:Arrow3DSpacingSettings",
        "ArrowUniform:=", True,
        "ArrowSpacing:=", 0,
        "MinArrowSpacing:=", 0,
        "MaxArrowSpacing:=", 0
    ]
],
"EnableGaussianSmoothing:=", False
```

```
], "Field")

oModule.CreateFieldPlot(
[
    "NAME:ComplexMag_Jc1",
    "SolutionName:=", "Setup1 : LastAdaptive",
    "UserSpecifyName:=", 0,
    "UserSpecifyFolder:=", 0,
    "QuantityName:=", "ComplexMag_Jc",
    "PlotFolder:=", "Jc",
    ...
)

oModule.CreateFieldPlot(
[
    "NAME:Jc_Vector2",
    "SolutionName:=", "Setup1 : LastAdaptive",
    "UserSpecifyName:=", 0,
    "UserSpecifyFolder:=", 0,
    "QuantityName:=", "Jc_Vector",
    "PlotFolder:=", "Jc",
```

```
]
...)

oModule.CreateFieldPlot(
[
    "NAME:Dielectric_Loss1",
    "SolutionName:=", "Setup1 : LastAdaptive",
    "UserSpecifyName:=", 0,
    "UserSpecifyFolder:=", 0,
    "QuantityName:=", "Dielectric_Loss",
    "PlotFolder:=", "Dielectric-Loss",
]
...)
```

## Maxwell Field Line Trace Plot Examples

### VB Example: Maxwell Field Line Trace Plot

```
oModule.CreateFieldPlot
Array("NAME:FieldLineTrace_Plot3",
"SsolutionName:=", "Setup1 : LastAdaptive",
"UserSpecifyName:=", 0,
"UserSpecifyFolder:=", 0,
"QuantityName:=", "QuantityName_FieldLineTrace",
"PlotFolder:=", "Field line trace plot",
"IntrinsicVar:=", "")
```

```
"Trace Step Length:=", "0.001mm",
"Use Adaptive Step:=", true,
"Seeding Faces:=", Array( 1, 15091),
"Seeding Markers:=", Array(0),
"Surface Tracing Objects:=", Array(1, 15),
"Volume Tracing Objects:=", Array(1, 36),
"Seeding Sampling Option:=", true,
"Seeding Points Number:=", 15,
"Fractional of Maximal:=", 0.8,
"Discrete Seeds Option:=", "Marker Point",
Array("NAME:InceptionEvaluationSettings",
"Gas Type:=", 0, "Gas Pressure:=", 1),
Array("NAME:FieldLineTracePlotSettings",
Array("NAME:LineSettingsID",
"Width:=", 1,
"Style:=", "Solid"),
"IsoValType:=", "Tone"))
"FieldLineTrace")

oModule.ModifyInceptionParameters "FieldLineTrace_Plot3",
Array("NAME:InceptionEvaluationSettings",
"Gas Type:=", 2, "Gas Pressure:=", 1,
"Use Inception:=", True,
"Potential U0:=", 0,
"Potential K:=", 1,
"Potential A:=", 1
"Critical Value:=", 5.9426,
"Streamer Constant:=", 6.5,
"Ionization Coefficient Dataset:=", Array( 0))
```

---

### Python Example: Maxwell Field Line Trace Plot

```
oModule.CreateFieldPlot(  
[  
    "NAME:FieldLineTrace_Plot3",  
    "SolutionName:=", "Setup1 : LastAdaptive",  
    "UserSpecifyName:=", 0,  
    "UserSpecifyFolder:=", 0,  
    "QuantityName:=", "QuantityName_FieldLineTrace",  
    "PlotFolder:=", "Field line trace plot",  
    "IntrinsicVar:=", "",  
    "Trace Step Length:=", "0.001mm",  
    "Use Adaptive Step:=", True,  
    "Seeding Faces:=", [1,15091],  
    "Seeding Markers:=", [0],  
    "Surface Tracing Objects:=", [1,15],  
    "Volume Tracing Objects:=", [1,36],  
    "Seeding Sampling Option:=", True,  
    "Seeding Points Number:=", 15,  
    "Fractional of Maximal:=", 0.8,  
    "Discrete Seeds Option:=", "Marker Point",  
    [  
        "NAME:InceptionEvaluationSettings",
```

```
"Gas Type:=", 0,
"Gas Pressure:=", 1
"Use Inception:=", True,
"Potential U0:=", 0,
"Potential K:=", 1,
"Potential A:=", 1
],
[
"NAME:FieldLineTracePlotSettings",
[
"NAME:LineSettingsID",
"Width:=", 1,
"Style:=", "Solid"
],
"IsoValType:=", "Tone"
]
], "FieldLineTrace")
```

**For Q3D Extractor and 2D Extractor, the command details are as follows:**

*Use:* Creates a field/mesh plot.

*Command:* **Q3D Extractor** or **2D Extractor>Fields**

*Syntax:* CreateFieldPlot <PlotParameterArray>

*Return Value:* None

*Parameters:* <PlotParameterArray>

```
Array ("NAME:<PlotName>",
      "SolutionName:=", <string>,
      "QuantityName:=", <string>,
      "PlotFolder:=", <string>,
      "UserSpecifyName:=", <int>,
      "UserSpecifyFolder:=", <int>,
      "IntrinsicVar:=", <string>,
      "PlotGeomInfo:=", <PlotGeomArray>,
      "FilterBoxes:=", <FilterBoxArray>,
      <PlotOnPointSettings>, <PlotOnLineSettings>,
      <PlotOnSurfaceSettings>, <PlotOnVolumeSettings>)
```

**SolutionName**

Name of the solution setup and solution formatted as:

```
"<SolveSetupName> : <WhichSolution>",
where <WhichSolution> can be "Adaptive_<n>",
"LastAdaptive", or "PortOnly".
```

For example: "Setup1 : Adaptive\_2"

HFSS requires a space on both sides of the ':' character. Otherwise, the plot is not be created.

QuantityName

*Type of plot to create. Possible values are:*

Mesh plots: "Mesh"

Q3D Field Plots:

Field type	Plot quantity names
AC R/L Fields	"SurfaceJac", "Mag_SurfaceJac"
DC R/L PEC Fields	"SurfaceJdc", "Mag_SurfaceJdc"
DC R/L Fields	"VolumeJdc", "Mag_VolumeJdc", "Phidc"
C Fields	"SmoothQ", "ABS_Q"

2D Extractor Field Plots:

Field type	Plot quantity names
CG Fields	"Mag_Phi", "PhiAtPhase", "Mag_E", "VectorE", "Mag_Jcg", "VectorJcg" and "energyCG"
RL Fields	"Flux Lines", "VectorA", "Mag_B", "VectorB", "Mag_H", "VectorH", "Jrl", "VectorJrl", "energyRL", "coenergy", "appenergy" and "emloss"

**PlotFolder**

*Name of the folder to which the plot should be added. Possible values are: "Q", "ABS\_Q", "JDC Vol", "Phi", "JDC Surf", and "JAC".*

**UserSpecifyName**

0 if default name for plot is used, 1 otherwise.

This parameter is not essential. <PlotName> is respected regardless of whether this flag is set.

**UserSpecifyFolder**

0 if the default folder for plot is used, 1 otherwise.

This parameter is not essential. The specified PlotFolder is respected regardless of whether this flag is set.

**IntrinsicVar**

Formatted string that specifies the frequency and phase at which to create the plot.

For example: "Freq='1GHz' Phase='30deg'"

**<PlotGeomArray>**

*Array(<NumGeomTypes>, <GeomTypeData>,*

*<GeomTypeData>, ...)*

For example: Array(4, "Volume", "ObjList", 1, "Box1",  
"Surface", "FacesList", 1, "12", "Line", 1,  
"Polyline1", "Point", 2, "Point1", "Point2")

**<NumGeomTypes>**

Type: <int>

Number of different geometry types (volume, surface, line, point) plotted at the same time.

```
<GeomTypeData>
  <GeomType>, <ListType>, <NumIDs>, <ID>, <ID>, ...
<GeomType>
  Type: <string>
  Possible values are "Volume", "Surface", "Line", "Point".
<ListType>
  Type: <string>
  Possible values are "ObjList" or "FacesList".
  These are used for GeomType values "Line" or "Point"
<NumIDs>
  Type: <int>
  Number of IDs or object names that will follow.
<ID>
  Type: <int> or <string>
  ID of a face or name of an object, line, or point on which to plot.
<FilterBoxArray>
  Array of object names used to restrict the plot range.
  Array(<NumFilters>, <ObjName>, <ObjName>, ...)
  Example: Array(1, "Box1")
  Example: Array(0) no filtering
```

```
<PlotOnPointSettings>
    Array("NAME:PlotOnPointSettings",
          "PlotMarker:=", <bool>,
          "PlotArrow:=", <bool>)

<PlotOnLineSettings>
    Array("NAME:PlotOnLineSettings",
          Array("NAME:LineSettingsID",
                "Width:=", <int>,
                "Style:=", <string>),
                "IsoValType:=", <string>,
                "ArrowUniform:=", <bool>,
                "NumofArrow:=", <int>)

    Style
        Possible values are "Cylinder", "Solid", "Dashdash",
        "Dotdot", "Dotdash".

    IsoValType
        Possible values are "Tone", "Fringe", "Gourard".

<PlotOnSurfaceSettings>
    Array("NAME:PlotOnSurfaceSettings",
          "Filled:=", <bool>,
          "IsoValType:=", <string>,
```

```
"SmoothShade:=", <bool>,
"AddGrid:=", <bool>,
"MapTransparency:=", <bool>,
"Transparency:=", <double>,
"ArrowUniform:=", <bool>
"ArrowSpacing:=", <double>
"GridColor:=", Array(<int>, <int>, <int>)

IsoValType
    Possible values are: "Tone", "Line", "Fringe", "Gourard".

GridColor
    Array containing the R, G, B components of the color. Components should be in the range 0 to
    255.

<PlotOnVolumeSettings>
    Array("NAME:PlotOnVolumeSettings",
        "PlotIsoSurface:=", <bool>,
        "CloudDensity:=", <double>,
        "PointSize:=", <int>,
        "ArrowUniform:=", <bool>,
        "ArrowSpacing:=", <double>)
```

## DeleteFieldPlot

Deletes one or more field plots.

<b>UI Access</b>	Right-click on one filed plot, select <b>Delete</b> .								
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;NameArray&gt;</td> <td>Array</td> <td>Array of strings containing the names of the plots to delete.</td> </tr> </tbody> </table>			Name	Type	Description	<NameArray>	Array	Array of strings containing the names of the plots to delete.
Name	Type	Description							
<NameArray>	Array	Array of strings containing the names of the plots to delete.							
<b>Return Value</b>	None.								

<b>Python Syntax</b>	DeleteFieldPlot(<NameArray>)
<b>Python Example</b>	oModule.DeleteFieldPlot(["Mag_E1", "Vector_E1"])

<b>VB Syntax</b>	DeleteFieldPlot <NameArray>
<b>VB Example</b>	oModule.DeleteFieldPlot Array("Mag_E1", "Vector_E1")

## DeleteMarker[Fields Reporter]

Deletes one or more marker in the current field plot.

<b>UI Access</b>	Right-click <b>Field Overlays &gt; Plot Fields &gt; Marker &gt; Delete Marker</b> .								
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;MarkerNames&gt;</td> <td>Array</td> <td>Array of strings containing marker names.</td> </tr> </tbody> </table>			Name	Type	Description	<MarkerNames>	Array	Array of strings containing marker names.
Name	Type	Description							
<MarkerNames>	Array	Array of strings containing marker names.							
<b>Return Value</b>	None.								

<b>Python Syntax</b>	DeleteMarker(<MarkerNames>)
<b>Python Example</b>	<code>oModule.DeleteMarker ( ["m1", "m2"] )</code>

<b>VB Syntax</b>	DeleteMarker <MarkerNames>
<b>VB Example</b>	<code>oModule.DeleteMarker Array("m1", "m2")</code>

## DeleteNamedExpr

Deletes the selected named expression from the list.

<b>UI Access</b>	Select a named expression and then click <b>Delete</b> .						
<b>Parameters</b>	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>&lt;ExprName&gt;</td><td>String</td><td>Name of specified named expression.</td></tr></tbody></table>	Name	Type	Description	<ExprName>	String	Name of specified named expression.
Name	Type	Description					
<ExprName>	String	Name of specified named expression.					
<b>Return Value</b>	None.						

<b>Python Syntax</b>	DeleteNamedExpr(<ExprName>)
<b>Python Example</b>	<code>oModule.DeleteNamedExpr ("Mag_JxE")</code>

<b>VB Syntax</b>	DeleteNamedExpr <ExprName>
------------------	----------------------------

<b>VB Example</b>	<code>oModule.DeleteNamedExpr "Mag_JxE"</code>
-------------------	------------------------------------------------

## DeleteUneditablePlot

Deletes one or more uneditable plots.

<b>UI Access</b>	N/A		
<b>Parameters</b>	Name <code>&lt;NameArray&gt;</code>	Type Array	Description Array of the plot names to delete.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	<code>DeleteUneditablePlot(&lt;NameArray&gt;)</code>
<b>Python Example</b>	<code>oModule.DeleteUneditablePlot(["Mag_E1", "Vector_E1"])</code>

<b>VB Syntax</b>	<code>DeleteUneditablePlot &lt;NameArray&gt;</code>
<b>VB Example</b>	<code>oModule.DeleteUneditablePlot Array("Mag_E1", "Vector_E1")</code>

## DoesNamedExpressionExists

Determines whether specified named expression exists.

<b>UI Access</b>	N/A		
<b>Parameters</b>	Name <code>&lt;ExpName&gt;</code>	Type String	Description Name of the specified named expression.

	Boolean: <ul style="list-style-type: none"><li>• <b>1</b> - named expression exists.</li><li>• <b>0</b> - named expression does not exist.</li></ul>
--	------------------------------------------------------------------------------------------------------------------------------------------------------

<b>Python Syntax</b>	DoesNamedExpressionExists(<ExpName>)
<b>Python Example</b>	<code>oModule.DoesNamedExpressionExists ("Mag_JxE")</code>

<b>VB Syntax</b>	DoesNamedExpressionExists <ExpName>
<b>VB Example</b>	<code>oModule.DoesNamedExpressionExists "Mag_JxE"</code>

## EditSurfaceMeshSummaryData

*Use:* Defines or Edits Surface Mesh Summary Data.

*Command:* Create Surface Mesh Summary, **Setup...**

*Syntax:* EditSurfaceMeshSummaryData Array(Array(<Parameters Array>))

*Return Value:* None

*Parameters:*

"NAME:SurfaceMeshSummary"

Type: <string>

"NAME:SurfaceMeshSummary"

```
"SolutionName:=" <string>
    Type: <string>
        Solution Name.

"Variation Name:=" <string>
    Type: <string>
        Variation name.

<RowItemsArray>
    Array<MeshRowItems>
        Row data.

<ItemPerRow>
    Array<EntityPerRow>
        Entity ID.

<MeshDataPer Item>
    Array<Mesh Data Category and Stats>
```

*VB Example:*

```
Set oModule = oDesign.GetModule("FieldsReporter")
oModule.EditSurfaceMeshSummaryData Array(Array("NAME:SurfaceMeshSummary", "SolutionName:=", _
"Setup1 : LastAdaptive", "Variation:=", "Nominal", Array("NAME:MeshRowItems", Array
("NAME:ItemPerRow", "EntityPerRow:=", _
"annular_rng", Array("NAME:MeshDataPerItem", "Min Edge Len:=", 0.000166961133900917, "Max Edge
Len:=", _
```

```

0.00145730991671888, "Mean Edge Len:=", 0.000524160923260581, "Std Devn Edge Len:=", _
0.000217758706109324, "Min Tri Area:=", 3.4982570283924E-09, "Max Tri Area:=", _
3.73116346661249E-07, "Mean Tri Area:=", 8.91526236529066E-08, "Std Devn Tri Area:=", _
6.89482807990471E-08, "DataState:=", 2)), Array("NAME:ItemPerRow", "EntityPerRow:=", _
"gap", Array("NAME:MeshDataPerItem", "Min Edge Len:=", 0.000105681286042456, "Max Edge Len:=", _
0.000362084671538928, "Mean Edge Len:=", 0.000233090954707978, "Std Devn Edge Len:=", _
7.55387369793154E-05, "Min Tri Area:=", 1.74640378345369E-09, "Max Tri Area:=", _
3.78166789228631E-08, "Mean Tri Area:=", 1.56497981245286E-08, "Std Devn Tri Area:=", _
1.04691637802914E-08, "DataState:=", 2)))) )

```

Python Syntax	RenameReport (<OldReportName>, <NewReportName>)
Python Example	<pre>oModule.EditSurfaceMeshSummaryData (     [         [             "NAME:SurfaceMeshSummary",             "SolutionName:="           , "Setup1 : LastAdaptive",             "Variation:="              , "Nominal",             [                 "NAME:MeshRowItems",                 [ </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",
            [
                "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
    ]
]
]
])
])
```

## EnterComplex

Enters a complex number onto the stack.

<b>UI Access</b>	Click <b>Number</b> , and then click <b>Scalar. Complex</b> option is selected.								
<b>Parameters</b>	<table border="1"><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;ComplexNum&gt;</td><td>String</td><td>String of complex value. Ex. "1 + 2j".</td></tr></table>			Name	Type	Description	<ComplexNum>	String	String of complex value. Ex. "1 + 2j".
Name	Type	Description							
<ComplexNum>	String	String of complex value. Ex. "1 + 2j".							
<b>Return Value</b>	None.								

---

<b>Python Syntax</b>	EnterComplex(<ComplexNum>)
<b>Python Example</b>	oModule.EnterComplex ("1 + 2 j")

<b>VB Syntax</b>	EnterComplex <ComplexNum>
<b>VB Example</b>	oModule.EnterComplex "1 + 2 j"

## EnterComplexVector

Enters a complex vector onto the stack.

<b>UI Access</b>	Click <b>Number</b> , and then click <b>Vector. Complex</b> option is selected.		
<b>Parameters</b>	Name <ComplexVector>	Type Array	Description Array of strings containing X, Y and Z complex values.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	EnterComplexVector(<ComplexVector>)
<b>Python Example</b>	<pre>oModule.EnterComplexVector ([      "1 + 2 j",      "1 + 2 j",      "1 + 2 j"]</pre>

	])
--	----

<b>VB Syntax</b>	EnterComplexVector <ComplexVector>
<b>VB Example</b>	<pre>oModule.EnterComplexVector _     Array("1 + 2 j",_         "1 + 2 j",_         "1 + 2 j")</pre>

## EnterCoord

Enters a coordinate system defined in the 3D Modeler editor.

<b>UI Access</b>	Click <b>Geometry</b> and then select <b>Coord</b> .						
<b>Parameters</b>	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>&lt;CoordName&gt;</td><td>String</td><td>Name of a coordinate system defined in the 3D Modeler editor.</td></tr></tbody></table>	Name	Type	Description	<CoordName>	String	Name of a coordinate system defined in the 3D Modeler editor.
Name	Type	Description					
<CoordName>	String	Name of a coordinate system defined in the 3D Modeler editor.					
<b>Return Value</b>	None.						

<b>Python Syntax</b>	EnterCoord(<CoordName>)
<b>Python Example</b>	<pre>oModule.EnterCoord("Global")</pre>

---

<b>VB Syntax</b>	EnterCoord <CoordName>
<b>VB Example</b>	oModule.EnterCoord "Global"

## EnterEdge

Enters an edge defined in the 3D Modeler editor.

<b>UI Access</b>	N/A		
<b>Parameters</b>	Name <EdgeName>	Type String	Description Name of an edge defined in the 3D Modeler editor.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	EnterEdge(<EdgeName>)
<b>Python Example</b>	oModule.EnterPoint ("Edge_1")

<b>VB Syntax</b>	EnterEdge <EdgeName>		
<b>VB Example</b>	oModule.EnterPoint "Edge_1"		

## EnterLine

Enters a line defined in the 3D Modeler editor.

<b>UI Access</b>	Click <b>Geometry</b> and then select <b>Line</b>		
<b>Parameters</b>	Name	Type	Description

	<LineName>	String	Name of a line defined in the 3D Modeler editor.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	EnterLine(<LineName>)
<b>Python Example</b>	<code>oModule.EnterLine("Line1")</code>

<b>VB Syntax</b>	EnterLine <LineName>
<b>VB Example</b>	<code>oModule.EnterLine "Line1"</code>

## EnterOutputVar

Enters Output Vars, only valid for Eigenmode problems.

<b>UI Access</b>	Click <b>Geometry</b> and then select <b>Output Vars</b> .		
<b>Parameters</b>	Name	Type	Description
	<VarName>	String	Name of the output vars. Only 'freq' is supported.
	<VarType>	String	Type of the output vars. 'Complex' type.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	EnterOutputVar(<VarName>, <VarType>)
----------------------	--------------------------------------

<b>Python Example</b>	<code>oModule.EnterOutputVar("Freq", "Complex")</code>
-----------------------	--------------------------------------------------------

<b>VB Syntax</b>	<code>EnterOutputVar &lt;VarName&gt;, &lt;VarType&gt;</code>
------------------	--------------------------------------------------------------

<b>VB Example</b>	<code>oModule.EnterOutputVar "Freq", "Complex"</code>
-------------------	-------------------------------------------------------

## EnterPoint

Enters a point defined in the 3D Modeler editor.

<b>UI Access</b>	Click <b>Geometry</b> and then select <b>Point</b> .								
<b>Parameters</b>	<table border="1"> <tr> <td>Name</td> <td>Type</td> <td>Description</td> </tr> <tr> <td><code>&lt;PointName&gt;</code></td> <td>String</td> <td>Name of a point defined in the 3D Modeler editor.</td> </tr> </table>			Name	Type	Description	<code>&lt;PointName&gt;</code>	String	Name of a point defined in the 3D Modeler editor.
Name	Type	Description							
<code>&lt;PointName&gt;</code>	String	Name of a point defined in the 3D Modeler editor.							
<b>Return Value</b>	None.								

<b>Python Syntax</b>	<code>EnterPoint(&lt;PointName&gt;)</code>
----------------------	--------------------------------------------

<b>Python Example</b>	<code>oModule.EnterPoint ("Point1")</code>
-----------------------	--------------------------------------------

<b>VB Syntax</b>	<code>EnterPoint &lt;PointName&gt;</code>
------------------	-------------------------------------------

<b>VB Example</b>	<code>oModule.EnterPoint "Point1"</code>
-------------------	------------------------------------------

## EnterQty

Enters a field quantity.

UI Access	Click <b>Quantity</b> , and then select from the list.		
Parameters	Name <i>&lt;FieldQty&gt;</i>	Type String	Description The field quantity to be entered onto the stack.
Return Value	None.		

Python Syntax	EnterQty( <i>&lt;FieldQty&gt;</i> )
Python Example	<code>oModule.EnterQty ("E")</code>

VB Syntax	EnterQty < <i>FieldQty</i> >
VB Example	<code>oModule.EnterQty "E"</code>

## EnterScalar

Enters a scalar onto the stack.

UI Access	Click <b>Number</b> and then click <b>Scalar</b> . <b>Complex</b> option not selected.		
Parameters	Name <i>&lt;Scalar&gt;</i>	Type Double	Description The real number to enter onto the stack.
Return Value	None.		

---

<b>Python Syntax</b>	EnterScalar(<Scalar>)
<b>Python Example</b>	<code>oModule.EnterScalar(3.14159265358979)</code>

<b>VB Syntax</b>	EnterScalar <Scalar>
<b>VB Example</b>	<code>oModule.EnterScalar 3.14159265358979</code>

## EnterScalarFunc

Enters a scalar function.

<b>UI Access</b>	Click <b>Function</b> and then select <b>Scalar</b> .						
<b>Parameters</b>	<table border="1"> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> <tr> <td>&lt;VarName&gt;</td> <td>String</td> <td>Name of a variable to enter as a scalar function onto the stack.</td> </tr> </table>	Name	Type	Description	<VarName>	String	Name of a variable to enter as a scalar function onto the stack.
Name	Type	Description					
<VarName>	String	Name of a variable to enter as a scalar function onto the stack.					
<b>Return Value</b>	None.						

<b>Python Syntax</b>	EnterScalarFunc(<VarName>)
<b>Python Example</b>	<code>oModule.EnterScalarFunc ("Phase")</code>

<b>VB Syntax</b>	EnterScalarFunc <VarName>
<b>VB Example</b>	<code>oModule.EnterScalarFunc "Phase"</code>

## EnterSurf

Enters a surface defined in the 3D Modeler editor.

<b>UI Access</b>	Click <b>Geometry</b> and then select <b>Surface</b> .		
<b>Parameters</b>	Name <i>&lt;SurfName&gt;</i>	Type String	Description Name of a surface defined in the 3D Modeler editor.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	EnterSurf( <i>&lt;SurfName&gt;</i> )
<b>Python Example</b>	oModule.EnterSurf("Rectangle1")

<b>VB Syntax</b>	EnterSurf <i>&lt;SurfName&gt;</i>
<b>VB Example</b>	oModule.EnterSurf "Rectangle1"

## EnterVector

Enters a vector onto the stack.

<b>UI Access</b>	Click <b>Number</b> , and then click <b>Vector. Complex</b> option not selected.		
<b>Parameters</b>	Name <i>&lt;VectorArray&gt;</i>	Type Array	Description Array of strings containing X, Y and Z components of the vector.

<b>Return Value</b>	None.
---------------------	-------

<b>Python Syntax</b>	EnterVector(<VectorArray>)
<b>Python Example</b>	<code>oModule.EnterVector([1.0, 1.0, 1.0])</code>

<b>VB Syntax</b>	EnterVector <VectorArray>
<b>VB Example</b>	<code>oModule.EnterVector Array(1.0, 1.0, 1.0)</code>

## EnterVectorFunc

Enters a vector function.

<b>UI Access</b>	Click <b>Function</b> and then select <b>Vector</b> .		
<b>Parameters</b>	Name <VarNames>	Type Array	Description Array of strings containing names of a variable for the X, Y, and Z coordinates, respectively, to enter as a vector function on the stack.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	EnterVectorFunc(<VarNames>)
<b>Python Example</b>	<code>oModuleEnterVectorFunc(["X", "Y", "Z"])</code>

<b>VB Syntax</b>	EnterVectorFunc <VarNames>
<b>VB Example</b>	oModuleEnterVectorFunc Array("X", "Y", "Z")

## EnterVol

Enters a volume defined in the 3D Modeler editor.

<b>UI Access</b>	Click <b>Geometry</b> and then select <b>Volume</b> .						
<b>Parameters</b>	<table border="1"><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;VolumeName&gt;</td><td>String</td><td>Name of a volume defined in the 3D Modeler editor.</td></tr></table>	Name	Type	Description	<VolumeName>	String	Name of a volume defined in the 3D Modeler editor.
Name	Type	Description					
<VolumeName>	String	Name of a volume defined in the 3D Modeler editor.					
<b>Return Value</b>	None.						

<b>Python Syntax</b>	EnterVol(<VolumeName>)
<b>Python Example</b>	oModule.EnterVol ("Box1")

<b>VB Syntax</b>	EnterVol <VolumeName>
<b>VB Example</b>	oModule.EnterVol "Box1"

## ExportFieldPlot

Exports a field plot to a file.

<b>UI Access</b>	N/A
------------------	-----

<b>Parameters</b>	Name	Type	Description
	<PlotName>	String	Name of the field plot to export.
	<ShowHeader>	Boolean	<b>True</b> - export field plot header to a file. <b>False</b> - export field plot.
	<FileName>	String	Name of file to save as, including the file path.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	ExportFieldPlot(<PlotName>, <ShowHeader>, <FileName>)
<b>Python Example</b>	<pre>oModule.ExportFieldPlot("Mag_E1", True, "C:/field_report.dsp")</pre>

<b>VB Syntax</b>	ExportFieldPlot <PlotName>, <ShowHeader>, <FileName>
<b>VB Example</b>	<pre>oModule.ExportFieldPlot "Mag_E1", true, "C:/field_report.dsp"</pre>

## ExportMarkerTable

Exports the marker table to a .csv or .tab file.

<b>UI Access</b>	[product] > Fields > Plot Fields > Marker > Export Marker Table.								
<b>Parameters</b>	<table border="1"> <tr> <td>Name</td> <td>Type</td> <td>Description</td> </tr> <tr> <td>&lt;FileName&gt;</td> <td>String</td> <td>Name of export file include path.</td> </tr> </table>			Name	Type	Description	<FileName>	String	Name of export file include path.
Name	Type	Description							
<FileName>	String	Name of export file include path.							
<b>Return Value</b>	None.								

<b>Python Syntax</b>	ExportMarkerTable(<FileName>)
<b>Python Example</b>	<code>oModule.ExportMarkerTable ("C:/work/FieldMarkerTable.csv")</code>

<b>VB Syntax</b>	ExportMarkerTable <FileName>
<b>VB Example</b>	<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.

<b>UI Access</b>	Click <b>Export</b> , and then click <b>On Grid</b> .		
<b>Parameters</b>	Name	Type	Description
	<OutputFile>	String	Name of the output file.
	<Min>	Array	Minimum values for the coordinate components of the grid system
	<Max>	Array	Maximum values for the coordinate components of the grid system
	<Spacing>	Array	Spacing values for the coordinate components of the grid system
	<SolnName>	String	Name of the simulation setup
	<VarVals>	Array	Array of strings containing setup definitions.
	<IncludePoints>	Boolean	Optional. Specifies whether include points in the output file.
	<CSType>	String	Optional. Type of coordinate system. "Cartesian" (default)   "Cylindrical"   "Spherical"
	<Offsets>	Array	Optional. Origin for the offset coordinate system. For Cartesian, x, y, z, for Cylindrical, R, Phi, Z, for Spherical, Rho, Theta, Phi.
	<ByCount>	Boolean	Optional.

<b>Return Value</b>	None.
---------------------	-------

**Note:** Regarding the **ExportOnGrid** legacy script which only has “IncludePtInOutput” argument (the last Boolean one), AEDT can still read it and assign other new arguments as default values. Those default values are RefCSName = “global”, PtInSI = “True”, FieldInRefCS = “False”

<b>Python Syntax</b>	ExportOnGrid(<OutputFile>, <Min>, <Max>, <Spacing>, <SolName>, <SolParameters>, [<ExportOption>, "IncludePointsInOutput:=", <boolean>], "RefCSName:=", <CSName>, "PtsInSI:=", <boolean>, "FieldRefCS:=", <boolean>], "<CSType>", [<Offsets>, <ByCount>])
<b>Python Example</b>	<pre> oModule.ExportOnGrid("C:\offset_grid_model_unit_ref.fld",                      [-1mm, "16mm", "0mm"],                      [1mm, "18mm", "1mm"],                      [2mm, "2mm", "1mm"],                      "4500MHz : LastAdaptive",                      [Freq:="4.5GHz", Phase:="0deg"],                      [                          "NAME:ExportOption",                          "IncludePtInOutput:=" , True,                          "RefCSName:=" , "offset",                          "PtInSI:=" , False,                          "FieldInRefCS:=" , True                      ],                      "Cartesian", ["0mm", "0mm", "0mm"], False ) </pre>

<b>VB Syntax</b>	ExportOnGrid(<OutputFile>, <Min>, <Max>, <Spacing>, <SolnName>, <SolnParameters>, [<ExportOption>, "IncludePointsInOutput:=", <boolean>], "RefCSName:=", <CSName>, "PtsInSI:=", <boolean>, "FieldRefCS:=", <boolean>], "<CSType>", [ <Offsets>, <ByCount>])
<b>VB Example</b>	<pre>oModule.ExportOnGrid "C:/Grid.fld", _     Array( "0mm", "0deg", "-25mm"), _     Array("20mm", "90deg", "125mm"), _     Array("10mm", "45deg", "50mm"), _     "Setup1 : LastAdaptive", _     Array("Freq:=", "10000Hz", "Phase:=", "0deg"), _     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>

```

        "NAME:ExportOption",
        "IncludePtInOutput:=" , True,
        "RefCSName:=" , "offset",
        "PtInSI:=" , False,
        "FieldInRefCS:=" , True
    ],
    "Cartesian", Array("0mm", "0mm", "0mm"), False
)

```

## ExportPlotImageToFile

Deprecated. Use [ExportPlotImageWithViewToFile](#).

Creates field plot exports of existing field plots from a given view points, and with the model being auto-sized automatically for each view.

<b>UI Access</b>	N/A		
<b>Parameters</b>	Name	Type	Description
	<FileName>	String	Full path plus file name.
	<FolderName>	String	Plot folder name.
	<ItemName>	String	Name of fields to plot.
	<SetViewTopDownDirectionByRCS>	String	Optional. Name of relative coordinate system to use for the field plot.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	ExportPlotImageToFile(<FileName>, <FolderName>, <ItemName>, <SetViewTopDownDirectionByRCS>)
<b>Python Example</b>	<pre>oModule.ExportPlotImageToFile(     "C:\TestEPITF2.jpg", "",      "Mag_E2", "RelativeCS1")</pre>

<b>VB Syntax</b>	ExportPlotImageToFile <FileName>, <FolderName>, <ItemName>, <SetViewTopDownDirectionByRCS>
<b>VB Example</b>	<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.

<b>UI Access</b>	N/A		
<b>Parameters</b>	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.

	<code>&lt;PixelSizeY&gt;</code>	Integer	Desired image height, in pixels.
	<code>&lt;ViewOrientation&gt;</code>	String	<i>Optional.</i> Name of orientation to use for plot.
<b>Return Value</b>	Image file is exported to the specified path.		

<b>Python Syntax</b>	<code>ExportPlotImageWithViewToFile(&lt;FileName&gt;, &lt;PlotQuantityName&gt;, &lt;PlotItemName&gt;, &lt;PixelSizeX&gt;, &lt;PixelSizeY&gt;, &lt;ViewOrientation&gt;)</code>
<b>Python Example</b>	<code>oModule.ExportPlotImageWithViewToFile ("E:/MyDir/OptimToutput/magE2.gif", "E Field", "Mag_E2", 1920, 1080, "newot2")</code>

<b>VB Syntax</b>	<code>ExportPlotImageWithViewToFile &lt;FileName&gt;, &lt;PlotQuantityName&gt;, &lt;PlotItemName&gt;, &lt;PixelSizeX&gt;, &lt;PixelSizeY&gt;, &lt;ViewOrientation&gt;</code>
<b>VB Example</b>	<code>oModule.ExportPlotImageWithViewToFile "E:/MyDir/OptimToutput/magE2.gif", "E Field", _ "Mag_E2", 1920, 1080, "newot2"</code>

## ExportSurfaceMeshSummary

*Use:* Exports a Surface Mesh Summary.

*Command:* Create Surface Mesh Summary.

*Syntax:* ExportSurfaceMeshSummary Array(<SolutionName>, <DesignVariationKey> <ExportFileName>)

*Return Value:* None

*Parameters:* <SolutionName>

Type: <string>

*Original name of the plot.*

<DesignVariationKey>

Type: <string>

New name of the plot.

*VB Example:*

```
oModule.ExportSurfaceMeshSummary Array("SolutionName:=", "Setup1 : LastAdaptive", "DesignVariationKey:=", _  
"Nominal", "ExportFileName:=", "D:\documents\surfaceMeshSummaryReport.csv")
```

Python Syntax	ExportSurfaceMeshSummary (<SolutionName>, <DesignVariationKey>, <ExportFileName>)
<b>Python Example</b>	<pre>oModule.ExportSurfaceMeshSummary(     [         "SolutionName:=" , "Setup1 : LastAdaptive",         "DesignVariationKey:=" , "Nominal",         "ExportFileName:=" , "D:\\Program Files\\Documents\\surfaceMeshSummaryReport.csv"     ])</pre>

## 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 <b>Export Data</b> .		
<b>Parameters</b>	Name	Type	Description
	<ReportName>	String	Name of report to be exported.
	<FileName>	String	Full path of the exported image file name; with extension of .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"
<b>Return Value</b>	None.		

**Python Syntax**

ExportToFile(<ReportName>, <FileName>)

**Python Example**

```
oModule.ExportToFile("rETotal", "C:/Users/Documents/rETotal.csv")
oModule.ExportToFile("S Parameter Table 1", "D:/Users/Documents/cftt.csv", False, " kV, MHz, yd ", True)
```

**VB Syntax**

```
ExportToFile <ReportName>, <FileName>
```

**VB Example**

```
oModule.ExportToFile "rETotal", "C:/Documents/rETotal.csv"
```

## 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**

```
oModule.GetFieldFolderNames()
```

**VB Syntax**

```
GetFieldFolderNames
```

**VB Example**

```
oModule.GetFieldFolderNames
```

## GetFieldPlotNames

Gets the names of field overlay plots defined in a design.

<b>UI Access</b>	N/A
<b>Parameters</b>	None.
<b>Return Value</b>	Array of strings containing field plots names.

<b>Python Syntax</b>	GetFieldPlotNames()
<b>Python Example</b>	<code>oModule.GetFieldPlotNames ()</code>

<b>VB Syntax</b>	GetFieldPlotNames
<b>VB Example</b>	<code>oModule.GetFieldPlotNames</code>

## GetFieldPlotQuantityName

Gets the quantity name of a specified field plot.

<b>UI Access</b>	N/A		
<b>Parameters</b>	Name <i>&lt;PlotName&gt;</i>	Type String	Description Name of specified plot.
<b>Return Value</b>	String plot quantity name.		

<b>Python Syntax</b>	GetFieldPlotQuantityName(<PlotName>)
<b>Python Example</b>	<code>oModule.GetFieldPlotQuantityName ("Mag_H1")</code>

<b>VB Syntax</b>	GetFieldPlotQuantityName <PlotName>
<b>VB Example</b>	<code>oModule.GetFieldPlotQuantityName "Mag_H1"</code>

## GetMeshPlotNames

Gets the names of mesh plots defined in a design.

<b>UI Access</b>	N/A
<b>Parameters</b>	None..
<b>Return Value</b>	Array of plot name.

<b>Python Syntax</b>	GetMeshPlotNames
<b>Python Example</b>	<code>oModule.GetMeshPlotNames ()</code>

<b>VB Syntax</b>	GetMeshPlotNames
<b>VB Example</b>	<code>oModule.GetMeshPlotNames</code>

## GetTopEntryValue

Gets the value of the top entry of the calculator stack.

<b>UI Access</b>	N/A		
<b>Parameters</b>	Name	Type	Description
	<SolnName>	String	Name of specified solution.
<b>Return Value</b>	Array of strings containing top entry values.		

<b>Python Syntax</b>	GetTopEntryValue(<SolnName>, <VarVals>)
<b>Python Example</b>	<pre> oModule.GetTopEntryValue(     "Setup1:LastAdaptive",     ["Freq:=", "1GHz", "Phase:=", "0deg", "x_size:=", "2mm"] ) </pre>

<b>VB Syntax</b>	GetTopEntryValue <SolnName>, <VarVals>
<b>VB Example</b>	<pre> oModule.GetTopEntryValue "Setup1:LastAdaptive", _     Array("Freq:=", "1GHz", "Phase:=", "0deg", _         "x_size:=", "2mm") </pre>

## HideAntennaParametersOverlay

Hides a currently visible antenna parameters overlay.

<b>UI Access</b>	<b>HFSS or HFSS-IE &gt; Fields &gt; Plot Fields &gt; Radiation Field</b>		
<b>Parameters</b>	Name <i>&lt;Name&gt;</i>	Type String	Description The name of the antenna parameters overlay.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	HideAntennaParametersOverlay( <i>&lt;Name&gt;</i> )
<b>Python Example</b>	<pre>oModule.HideAntennaParameterOverlay(     "Antenna Parameter Overlay1")</pre>

<b>VB Syntax</b>	HideAntennaParametersOverlay <i>&lt;Name&gt;</i>
<b>VB Example</b>	<pre>oModule.HideAntennaParameterOverlay _     "Antenna Parameter Overlay1"</pre>

## HidePolarPlot

Hides a currently visible polar plot.

<b>UI Access</b>	N/A
------------------	-----

<b>Parameters</b>	Name <i>&lt;PlotName&gt;</i>	Type String	Description Name of the polar plot.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	HidePolarPlot( <i>&lt;PlotName&gt;</i> )
<b>Python Example</b>	<code>oModule.HidePolarPlot("rE Plot 1")</code>

<b>VB Syntax</b>	HidePolarPlot <PlotName>
<b>VB Example</b>	<code>oModule.HidePolarPlot "rE Plot 1"</code>

## HideRadiatedPlotOverlay

Hides overlay radiation fields plot.

<b>UI Access</b>	N/A
<b>Parameters</b>	Name <i>&lt;PlotName&gt;</i>
<b>Return Value</b>	None.

<b>Python Syntax</b>	HideRadiatedPlotOverlay( <i>&lt;PlotName&gt;</i> )
<b>Python Example</b>	<code>oModule.HideRadiatedPlotOverlay("Gain Plot 1")</code>

<b>VB Syntax</b>	HideRadiatedPlotOverlay < <i>PlotName</i> >
<b>VB Example</b>	oModule.HideRadiatedPlotOverlay "Gain Plot 1"

## LoadNamedExpressions

Loads a named expression definition from a saved file.

<b>UI Access</b>	In the Fields Calculator, click <b>Load From...</b> in the Library area.												
<b>Parameters</b>	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>&lt;<i>FileName</i>&gt;</td><td>String</td><td>Filename and full path to the file to hold the named expression definition.</td></tr><tr><td>&lt;<i>FieldType</i>&gt;</td><td>String</td><td>For products with just one field type, it is set to "Fields".</td></tr><tr><td>&lt;<i>NamedExpr</i>&gt;</td><td>Array</td><td>rray of strings containing the names of expression definitions to load from the file.</td></tr></tbody></table>	Name	Type	Description	< <i>FileName</i> >	String	Filename and full path to the file to hold the named expression definition.	< <i>FieldType</i> >	String	For products with just one field type, it is set to "Fields".	< <i>NamedExpr</i> >	Array	rray of strings containing the names of expression definitions to load from the file.
Name	Type	Description											
< <i>FileName</i> >	String	Filename and full path to the file to hold the named expression definition.											
< <i>FieldType</i> >	String	For products with just one field type, it is set to "Fields".											
< <i>NamedExpr</i> >	Array	rray of strings containing the names of expression definitions to load from the file.											
<b>Return Value</b>	None.												

<b>Python Syntax</b>	LoadNamedExpressions(< <i>FileName</i> >, < <i>FieldType</i> >, < <i>NamedExpr</i> >)
<b>Python Example</b>	oModule.LoadNamedExpressions( "C:\\Ansoft\\PersonalLib\\smth.clc", "Fields", ["smoothedtemp"])

<b>VB Syntax</b>	LoadNamedExpressions < <i>FileName</i> >, < <i>FieldType</i> >, < <i>NamedExpr</i> >
<b>VB Example</b>	oModule.LoadNamedExpressions _

```
"C:\Ansoft\PersonalLib\smth.clc", _  
"Fields", Array("smoothedtemp")
```

## ModifyFieldPlot

Modifies a plot definition.

<b>UI Access</b>	[product] > <b>Fields</b> > <b>Modify Plot</b>									
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;OriginalName&gt;</td> <td>String</td> <td>Name of the plot to be modified.</td> </tr> <tr> <td>&lt;PlotParams&gt;</td> <td>Array</td> <td>Array containing modified settings.</td> </tr> </tbody> </table>	Name	Type	Description	<OriginalName>	String	Name of the plot to be modified.	<PlotParams>	Array	Array containing modified settings.
Name	Type	Description								
<OriginalName>	String	Name of the plot to be modified.								
<PlotParams>	Array	Array containing modified settings.								
<b>Return Value</b>	None.									

<b>Python Syntax</b>	ModifyFieldPlot(<OriginalName>, <PlotParams>)
<b>Python Example</b>	<pre>oModule.ModifyFieldPlot("Vector_E1" ,     ["NAME:Vector_E2" ,      "SolutionName:=", "Setup1 : LastAdaptive" ,      "QuantityName:=", "Vector_E" ,      "PlotFolder:=", "E Field1" ,      "UserSpecifyName:=", 0 ,      "UserSpecifyFolder:=", 0 ,      "IntrinsicVar:=", "Freq='1GHz' Phase='30deg'" ,</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:=", 0.10000001490116,
"GridColor:=", [255, 255, 255]]
])
```

<b>VB Syntax</b>	ModifyFieldPlot <OriginalName>, <PlotParams>
<b>VB Example</b>	<pre>oModule.ModifyFieldPlot "Vector_E1" ,     Array("NAME:Vector_E2", _         "SolutionName:=", "Setup1 : LastAdaptive", _         "QuantityName:=", "Vector_E", _</pre>

```

"PlotFolder:=", "E Field1", _
"UserSpecifyName:=", 0, _
"UserSpecifyFolder:=", 0, _
"IntrinsicVar:=", "Freq='1GHz' Phase='30deg'", _
"PlotGeomInfo:=", Array(1, "Surface", "FacesList", 1, "7"), _
"FilterBoxes:=", Array(0), _
Array("NAME:PlotOnSurfaceSettings", _
"Filled:=", false, _
"IsoValType:=", "Fringe", _
"SmoothShade:=", true, _
"AddGrid:=", false, _
"MapTransparency:=", true, _
"Transparency:=", 0, _
"ArrowUniform:=", true, _
"ArrowSpacing:=", 0.100000001490116, _
"GridColor:=", Array(255, 255, 255)))

```

## ReassignFieldPlot

Reassigns a field plot.

<b>UI Access</b>	Right-click on a field plot > <b>Reassign</b> .								
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;PlotName&gt;</td> <td>String</td> <td>Name of specified plot to reassign.</td> </tr> </tbody> </table>			Name	Type	Description	<PlotName>	String	Name of specified plot to reassign.
Name	Type	Description							
<PlotName>	String	Name of specified plot to reassign.							

	<PlotParameterArray>	Array	See <a href="#">CreateFieldPlot</a>
<b>Return Value</b>	None.		

<b>Python Syntax</b>	ReassignFieldPlot(<PlotName>, <PlotParameterArray>)
<b>Python Example</b>	<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,</pre>

```

    "IsoValType:=", "Fringe",
    "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
])

```

<b>VB Syntax</b>	ReassignFieldPlot <PlotName>, <PlotParameterArray>
<b>VB Example</b>	<pre> oModule.ReassignFieldPlot "Mag_E1", _     Array("NAME:Mag_E1", "SolutionName:=", _ </pre>

```
"Setup1 : LastAdaptive", "UserSpecifyName:=", 0, _
"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.

<b>UI Access</b>	Right-click the plot you want to rename in the project tree, and then click <b>Rename</b> on the shortcut menu.									
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;OldName&gt;</td> <td>String</td> <td>Original name of the plot.</td> </tr> <tr> <td>&lt;NewName&gt;</td> <td>String</td> <td>New name of the plot.</td> </tr> </tbody> </table>	Name	Type	Description	<OldName>	String	Original name of the plot.	<NewName>	String	New name of the plot.
Name	Type	Description								
<OldName>	String	Original name of the plot.								
<NewName>	String	New name of the plot.								
<b>Return Value</b>	None.									

<b>Python Syntax</b>	<code>RenameFieldPlot(&lt;OldName&gt;, &lt;NewName&gt;)</code>
<b>Python Example</b>	<code>oModule.RenameFieldPlot(     "Vector_E1", "Vector_E2")</code>

<b>VB Syntax</b>	<code>RenameFieldPlot &lt;OldName&gt;, &lt;NewName&gt;</code>
<b>VB Example</b>	<code>oModule.RenameFieldPlot _     "Vector_E1", "Vector_E2"</code>

## RenamePlotFolder

Renames a plot folder.

<b>UI Access</b>	Right-click a plot folder in the project tree, and then click <b>Rename</b> on the shortcut menu.									
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;OldName&gt;</td> <td>String</td> <td>Original name of the folder.</td> </tr> <tr> <td>&lt;NewName&gt;</td> <td>String</td> <td>New name of the folder.</td> </tr> </tbody> </table>	Name	Type	Description	<OldName>	String	Original name of the folder.	<NewName>	String	New name of the folder.
Name	Type	Description								
<OldName>	String	Original name of the folder.								
<NewName>	String	New name of the folder.								
<b>Return Value</b>	None.									

<b>Python Syntax</b>	RenamePlotFolder(<OldName>, <NewName>)
<b>Python Example</b>	<pre>oModule.RenamePlotFolder(     "E Field", "Surface Plots")</pre>

<b>VB Syntax</b>	RenamePlotFolder <OldName>, <NewName>
<b>VB Example</b>	<pre>oModule.RenamePlotFolder _     "E Field", "Surface Plots"</pre>

## SaveFieldsPlots

Saves field plot(s) to a file.

<b>UI Access</b>	<b>HFSS &gt; Fields &gt; Save As...</b>									
<b>Parameters</b>	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>&lt;PlotNames&gt;</td><td>Array</td><td>Array of plot names.</td></tr><tr><td>&lt;FileName&gt;</td><td>String</td><td>Name of the file to save as, including the file path.</td></tr></tbody></table>	Name	Type	Description	<PlotNames>	Array	Array of plot names.	<FileName>	String	Name of the file to save as, including the file path.
Name	Type	Description								
<PlotNames>	Array	Array of plot names.								
<FileName>	String	Name of the file to save as, including the file path.								
<b>Return Value</b>	None.									

<b>Python Syntax</b>	SaveFieldsPlots(<PlotNames>, <FileName>)
<b>Python Example</b>	<pre>oModule.SaveFieldsPlots(["Mag_E1", "Mag_E2"],</pre>

	"C:/field_report.dsp")
--	------------------------

<b>VB Syntax</b>	SaveFieldsPlots <PlotNames>, <FileName>
<b>VB Example</b>	<pre>oModule.SaveFieldsPlots Array("Mag_E1", "Mag_E2"), _ "C:/field_report.dsp"</pre>

## SaveNamedExpressions

Saves a named expression definition to a file.

<b>UI Access</b>	In the Fields Calculator, click <b>Save To...</b> in the Library area.												
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;FileName&gt;</td> <td>String</td> <td>Filename and full path to the file to hold the named expression definition.</td> </tr> <tr> <td>&lt;NamedExprs&gt;</td> <td>Array</td> <td>Array of strings containing the names of expression definitions to load from the file.</td> </tr> <tr> <td>&lt;Overwrite&gt;</td> <td>Boolean</td> <td>Specifies whether to overwrite the file.</td> </tr> </tbody> </table>	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.
Name	Type	Description											
<FileName>	String	Filename and full path to the file to hold the named expression definition.											
<NamedExprs>	Array	Array of strings containing the names of expression definitions to load from the file.											
<Overwrite>	Boolean	Specifies whether to overwrite the file.											
<b>Return Value</b>	None.												

<b>Python Syntax</b>	SaveNamedExpressions(<FileName>, <NamedExprs>, <Overwrite>)
<b>Python Example</b>	<pre>oModule.SaveNamedExpressions( "C:\Ansoft\PersonalLib\smth.clc", ["smoothedtemp"], True)</pre>

<b>VB Syntax</b>	SaveNamedExpressions <FileName>, <NamedExprs>, <Overwrite>
<b>VB Example</b>	<pre>oModule.SaveNamedExpressions _     "C:\Ansoft\PersonalLib\smth.clc", _     Array("smoothedtemp"), true</pre>

## SetFieldPlotSettings

Sets plot attributes.

<b>UI Access</b>	[product] > Fields > Modify Plot Attributes									
<b>Parameters</b>	<table border="1"><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;PlotName&gt;</td><td>String</td><td>Name of the plot to modify.</td></tr><tr><td>&lt;PlotItemAttributes&gt;</td><td>Array</td><td>Structured array.  Array ("NAME:FieldsPlotItemSettings",         &lt;PlotOnPointsSettings&gt;,         &lt;PlotOnLineSettings&gt;,         &lt;PlotOnSurfaceSettings&gt;,         &lt;PlotOnVolumeSettings&gt;)  <i>See description of CreateFieldPlot command for details.</i></td></tr></table>	Name	Type	Description	<PlotName>	String	Name of the plot to modify.	<PlotItemAttributes>	Array	Structured array.  Array ("NAME:FieldsPlotItemSettings",  <PlotOnPointsSettings>,  <PlotOnLineSettings>,  <PlotOnSurfaceSettings>,  <PlotOnVolumeSettings>)  <i>See description of CreateFieldPlot command for details.</i>
Name	Type	Description								
<PlotName>	String	Name of the plot to modify.								
<PlotItemAttributes>	Array	Structured array.  Array ("NAME:FieldsPlotItemSettings",  <PlotOnPointsSettings>,  <PlotOnLineSettings>,  <PlotOnSurfaceSettings>,  <PlotOnVolumeSettings>)  <i>See description of CreateFieldPlot command for details.</i>								
<b>Return Value</b>	None.									

<b>Python Syntax</b>	<code>SetFieldPlotSettings(&lt;PlotName&gt; &lt;PlotItemAttributes&gt;)</code>
<b>Python Example</b>	<pre> oModule.SetFieldPlotSettings ("Mag_E2",     [         ["NAME:FieldsPlotItemSettings",             [                 ["NAME:PlotOnLineSettings",                     [                         ["NAME:LineSettingsID",                             "Width:=", 4,                             "Style:=", "Cylinder"),                             "IsoValType:=", "Tone",                             "ArrowUniform:=", True,                             "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]]]                 )             ]         ]     ) ) </pre>

<b>VB Syntax</b>	<b>SetFieldPlotSettings &lt;PlotName&gt; &lt;PlotItemAttributes&gt;</b>
<b>VB Example</b>	<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, _</pre>

	"GridColor:=", Array(255, 255, 255) )
--	---------------------------------------

## SetPlotFolderSettings

Sets the attributes of all plots in the specified folder.

UI Access	[product] > Fields > Modify Plot Attributes		
Parameters	Name <i>&lt;PlotFolderName&gt;</i>	Type String	Description Name of the folder with the attributes to modify.
	<i>&lt;PlotFolderAttributes&gt;</i>	Array	Structured array.  Array ("NAME:FieldsPlotSettings", "Real time mode:=", <bool>, <ColorMapSettings>, <Scale3DSettings>, <Marker3DSettings>, <Arrow3DSettings>)
	<i>&lt;ColorMapSettings&gt;</i>	Array	Structured array.  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-

		ponents of the color. Components should be in the range 0 to 255.)
<Scale3DSettings>	Array	<p>Structured array.</p> <pre>Array("NAME:Scale3DSettings",       "m_nLevels:=", &lt;int&gt;,       "m_autoScale:=", &lt;bool&gt;,       "minvalue:=", &lt;double&gt;,       "maxvalue:=", &lt;double&gt;,       "log:=", &lt;bool&gt;,       "IntrinsicMin:=", &lt;double&gt;,       "IntrinsicMax:=", &lt;double&gt;)</pre>
<Marker3DSettings>	Array	<p>Structured array.</p> <pre>Array("NAME:Marker3DSettings",       "MarkerType:=", &lt;integer&gt; 9:Sphere 10: Box 11: Tetrahedron 12: Octahedron default: Sphere&gt;,       "MarkerMapSize:=", &lt;bool&gt;,       "MarkerMapColor:=", &lt;bool&gt;,</pre>

		<pre>"MarkerSize:=", &lt;double&gt;) &lt;Arrow3DSettings&gt;          Array Structured array.  Array("NAME:Arrow3DSettings",       "ArrowType:=", &lt;integer                     0: Line                     1: Cylinder                     2: Umbrella                     default: Line&gt;,       "ArrowMapSize:=", &lt;bool&gt;,       "ArrowMapColor:=", &lt;bool&gt;,       "ShowArrowTail:=", &lt;bool&gt;,       "ArrowSize:=", &lt;double&gt;)</pre>
<b>Return Value</b>	None.	

<b>Python Syntax</b>	SetPlotFolderSettings(<PlotFolderName>, <PlotFolderAttributes>)
<b>Python Example</b>	<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,
```

	<pre>"ArrowSize:=", 0.25]] )</pre>
--	----------------------------------------

<b>VB Syntax</b>	SetPlotFolderSettings <PlotFolderName>, <PlotFolderAttributes>
<b>VB Example</b>	<pre>oModule.SetPlotFolderSettings "E Field1", _     Array("NAME:FieldsPlotSettings", _         "Real time mode:=", true, _         Array("NAME:ColorMapSettings", _             "ColorMapType:=", "Spectrum", _             "SpectrumType:=", "Rainbow", _             "UniformColor:=", Array(127, 255, 255), _             "RampColor:=", Array(255, 127, 127)), _             Array("NAME:Scale3DSettings", _                 "m_nLevels:=", 27, _                 "m_autoScale:=", true, _                 "minvalue:=", 9.34379863739014, _                 "maxvalue:=", 13683.755859375, _                 "log:=", false, _                 "IntrinsicMin:=", 9.34379863739014, _                 "IntrinsicMax:=", 13683.755859375), _                 Array("NAME:Marker3DSettings", _</pre>

```
"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.

<b>UI Access</b>	<b>HFSS or HFSS-IE &gt; Fields &gt; Plot &gt; Radiation Fields</b>								
<b>Parameters</b>	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>&lt;Name&gt;</td><td>String</td><td>The name of the antenna parameters overlay.</td></tr></tbody></table>			Name	Type	Description	<Name>	String	The name of the antenna parameters overlay.
Name	Type	Description							
<Name>	String	The name of the antenna parameters overlay.							
<b>Return Value</b>	None.								

<b>Python Syntax</b>	ShowAntennaParamedterOverlay(<Name>)
<b>Python Example</b>	oModule.ShowAntennaParameterOverlay(

	"Antenna Parameter Overlay1")
--	-------------------------------

<b>VB Syntax</b>	ShowAntennaParamedterOverlay <Name>
<b>VB Example</b>	<pre>oModule.ShowAntennaParameterOverlay _ "Antenna Parameter Overlay1"</pre>

## UpdateAllFieldsPlots

Updates the All Fields Plots.

<b>UI Access</b>	N/A
<b>Parameters</b>	None.
<b>Return Value</b>	None.

<b>Python Syntax</b>	UpdateAllFieldsPlots()
<b>Python Example</b>	<pre>oModule.UpdateAllFieldsPlots()</pre>

<b>VB Syntax</b>	UpdateAllFieldsPlots
<b>VB Example</b>	<pre>oModule.UpdateAllFieldsPlots</pre>

## UpdateQuantityFieldsPlots

Updates the Quantity Fields Plots.

<b>UI Access</b>	N/A								
<b>Parameters</b>	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td><i>&lt;FolderName&gt;</i></td><td>String</td><td>Folder containing the plotted quantities.</td></tr></tbody></table>			Name	Type	Description	<i>&lt;FolderName&gt;</i>	String	Folder containing the plotted quantities.
Name	Type	Description							
<i>&lt;FolderName&gt;</i>	String	Folder containing the plotted quantities.							
<b>Return Value</b>	None.								

<b>Python Syntax</b>	UpdateQuantityFieldsPlots(<FolderName>)
<b>Python Example</b>	<code>oModule.UpdateQuantityFieldsPlots ("fldplt")</code>

<b>VB Syntax</b>	UpdateQuantityFieldsPlots <FolderName>
<b>VB Example</b>	<code>oModule.UpdateQuantityFieldsPlots "fldplt"</code>

# 17 - 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](#)

[AddNamedExpr](#)

[CalcOp](#)

[CalculatorRead](#)

[CalcStack](#)

[CalculatorWrite](#)

[ChangeGeomSettings](#)

[ClcEval](#)

[ClcMaterial](#)

[ClearAllNamedExpr](#)

[CopyNamedExprToStack](#)

[DeleteNamedExpr](#)

[EnterComplex](#)

[EnterComplexVector](#)

[EnterLine](#)

[EnterPoint](#)[EnterQty](#)[EnterScalar](#)[EnterScalarFunc](#)[EnterSurf](#)[EnterVector](#)[EnterVectorFunc](#)[EnterVol](#)[ExportOnGrid \[Fields Calculator\]](#)[ExportToFile \[Fields Calculator\]](#)[GetTopEntryValue](#)[LoadNamedExpressions](#)[SaveNamedExpressions](#)

## AddNamedExpression

Creates a named expression using the expression at the top of the stack.

<b>UI Access</b>	Click <b>Add</b> in the <b>Fields Calculator</b> dialogue.		
<b>Parameters</b>	Name	Type	Description
	<ExprName>	String	Name for the new named expression.
<b>Return Value</b>	None.		

---

<b>Python Syntax</b>	AddNamedExpression(< <i>ExprName</i> >, < <i>FieldType</i> >)
<b>Python Example</b>	<code>oModule.AddNamedExpression ("Mag_JxE", "Fields")</code>

<b>VB Syntax</b>	AddNamedExpression < <i>ExprName</i> >, < <i>FieldType</i> >
<b>VB Example</b>	<code>oModule.AddNamedExpression "Mag_JxE", "Fields"</code>

## AddNamedExpr

Creates a named expression using the expression at the top of the stack.

<b>UI Access</b>	Click <b>Add</b> in the <b>Fields Calculator</b> dialogue.						
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;<i>ExprName</i>&gt;</td> <td>String</td> <td>Name for the new named expression.</td> </tr> </tbody> </table>	Name	Type	Description	< <i>ExprName</i> >	String	Name for the new named expression.
Name	Type	Description					
< <i>ExprName</i> >	String	Name for the new named expression.					
<b>Return Value</b>	None.						

<b>Python Syntax</b>	AddNamedExpr(< <i>ExprName</i> >)
<b>Python Example</b>	<code>oModule.AddNamedExpr ("Mag_JxE")</code>

<b>VB Syntax</b>	AddNamedExpr < <i>ExprName</i> >
<b>VB Example</b>	<code>oModule.AddNamedExpr "Mag_JxE"</code>

## CalcOp

Performs a calculator operation.

<b>UI Access</b>	Operation commands like <b>Mag</b> , <b>+</b> , etc.		
<b>Parameters</b>	Name <i>&lt;OpString&gt;</i>	Type String	Description The text on the corresponding calculator button.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	CalcOp( <i>&lt;OpString&gt;</i> )
<b>Python Example</b>	<code>oModule.CalcOp ("+")</code>

<b>VB Syntax</b>	CalcOp <i>&lt;OpString&gt;</i>
<b>VB Example</b>	<code>oModule.CalcOp "+"</code>

## CalcRead(deprecated)

Reads a file that is written out by the CalcWrite command, and puts the result into a calculator numeric.

<b>UI Access</b>	Click <b>Read...</b> in the <b>Fields Calculator</b> dialog.		
<b>Parameters</b>	Name <i>&lt;FileName&gt;</i>	Type String	Description Name of the file to be read.
	<i>&lt;SoluName&gt;</i>	Type String	Description Specified solution to read in.

	<code>&lt;VarArray&gt;</code>	Array	Array of variables to read.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	<code>CalcRead(&lt;FileName&gt;, &lt;SoluName&gt;, &lt;VarArray&gt;)</code>
<b>Python Example</b>	<pre>oModule.CalcRead(     "c:\example.reg",     "Setup1: LastAdaptive",     ["Freq:=", "10GHz", "Phase:=", "0deg"])</pre>

<b>VB Syntax</b>	<code>CalcRead &lt;FileName&gt;, &lt;SoluName&gt;, &lt;VarArray&gt;</code>
<b>VB Example</b>	<pre>oModule.CalcRead _     "c:\example.reg", _     "Setup1: LastAdaptive", _     Array("Freq:=", "10GHz", "Phase:=", "0deg")</pre>

## CalculatorRead

Gets a register file and applies it to the calculator stack.

<b>UI Access</b>	Click <b>Read...</b> in the <b>Fields Calculator</b> dialog.								
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><code>&lt;FileName&gt;</code></td> <td>String</td> <td>Path to and including name of input register file.</td> </tr> </tbody> </table>			Name	Type	Description	<code>&lt;FileName&gt;</code>	String	Path to and including name of input register file.
Name	Type	Description							
<code>&lt;FileName&gt;</code>	String	Path to and including name of input register file.							

	<i>&lt;SoluName&gt;</i>	String	Specified solution to read in.
	<i>&lt;FieldType&gt;</i>	String	Type of specified field.
	<i>&lt;VarArray&gt;</i>	Array	Array of variable names, value pairs.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	CalculatorRead(<FileName>, <SoluName>, <FieldType>, <VarArray>)
<b>Python Example</b>	<pre>oModule.CalculatorRead(     "c:\\example.reg",     "Setup1: LastAdaptive", "Fields",     ["Freq:=", "10GHz", "Phase:=", "0deg"])</pre>

<b>VB Syntax</b>	CalculatorRead <FileName>, <SoluName>, <FieldType>, <VarArray>
<b>VB Example</b>	<pre>oModule.CalculatorRead _     "c:\\example.reg", _     "Setup1: LastAdaptive", "Fields", _     Array("Freq:=", "10GHz", "Phase:=", "0deg")</pre>

## CalcStack

Performs an operation on the stack.

<b>UI Access</b>	Stack operation buttons such as <b>Push</b> and <b>Pop</b> .		
<b>Parameters</b>	Name	Type	Description
	<code>&lt;OpString&gt;</code>	String	The text on the corresponding calculator button.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	<code>CalcStack(&lt;OpString&gt;)</code>
<b>Python Example</b>	<code>oModule.CalcStack("push")</code>

<b>VB Syntax</b>	<code>CalcStack &lt;OpString&gt;</code>
<b>VB Example</b>	<code>oModule.CalcStack "push"</code>

## CalculatorWrite

Writes contents of top register to file.

<b>UI Access</b>	Click <b>Write...</b> in the <b>Fields Calculator</b> dialog.		
<b>Parameters</b>	Name	Type	Description
	<code>&lt;OutputFilePath&gt;</code>	String	Path to and including name of output register file.
	<code>&lt;SoluNameArray&gt;</code>	Array	Array of strings containing solution names.
	<code>&lt;VarArray&gt;</code>	Array	Array of variable names, value pairs.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	CalculatorWrite(<OutputFilePath>, <SoluNameArray>, <VarArray>)
<b>Python Example</b>	<pre> oModule.CalculatorWrite("c:\test.reg",     ["Solution:=", "Setup1 : LastAdaptive"],     ["Freq:=", "1GHz", "Phase:=", "0deg"] ) </pre>

<b>VB Syntax</b>	CalculatorWrite <OutputFilePath>, <SoluNameArray>, <VarArray>
<b>VB Example</b>	<pre> oModule.CalculatorWrite "c:\test.reg", _     Array("Solution:=", "Setup1 : LastAdaptive"), _     Array("Freq:=", "1GHz", "Phase:=", "0deg") </pre>

## CalcWrite

Writes contents of top register to file.

<b>UI Access</b>	N/A		
<b>Parameters</b>	Name	Type	Description
	<FilePath>	String	Path to output and including name of output register file.
	<SoluName>	String	Name of solution.
	<VariablesArray>	Array	Array of variable names, value pairs.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	CalcWrite(<FilePath>, <SoluName>, <VariablesArray>)
<b>Python Example</b>	<pre>oModule.CalcWrite("C:\Ansoft\smoothedTemp.fld", "Setup1 : LastAdaptive", ["\$conductivity:=", "50000000"])</pre>

<b>VB Syntax</b>	CalcWrite <FilePath>, <SoluName>, <VariablesArray>
<b>VB Example</b>	<pre>oModule.CalcWrite "C:\Ansoft\smoothedTemp.fld", "Setup1 : LastAdaptive", _ Array("\$conductivity:=", "50000000")</pre>

## ChangeGeomSettings

Changes the line discretization setting.

<b>UI Access</b>	In the <b>Fields Calculator</b> dialog box, click on <b>Geom Settings...</b>						
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;LineDiscr&gt;</td> <td>Integer</td> <td>Line discretization value.</td> </tr> </tbody> </table>	Name	Type	Description	<LineDiscr>	Integer	Line discretization value.
Name	Type	Description					
<LineDiscr>	Integer	Line discretization value.					
<b>Return Value</b>	None.						

<b>Python Syntax</b>	ChangeGeomSettings(<LineDiscr>)
<b>Python Example</b>	<pre>oModule.ChangeGeomSettings(500)</pre>

<b>VB Syntax</b>	ChangeGeomSettings <LineDiscr>
<b>VB Example</b>	oModule.ChangeGeomSettings 500

## ClcEval

Evaluates the expression at the top of the stack using the provided solution name and variable values.

<b>UI Access</b>	Click <b>Eval</b> in the <b>Fields Calculator</b> dialog.												
<b>Parameters</b>	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>&lt;SoluName&gt;</td><td>String</td><td>Name of specified solution.</td></tr><tr><td>&lt;VarArray&gt;</td><td>Array</td><td>Array of variable name, value pairs.</td></tr><tr><td>&lt;FieldType&gt;</td><td>String</td><td>Optional. Type of specified field.</td></tr></tbody></table>	Name	Type	Description	<SoluName>	String	Name of specified solution.	<VarArray>	Array	Array of variable name, value pairs.	<FieldType>	String	Optional. Type of specified field.
Name	Type	Description											
<SoluName>	String	Name of specified solution.											
<VarArray>	Array	Array of variable name, value pairs.											
<FieldType>	String	Optional. Type of specified field.											
<b>Return Value</b>	None.												

<b>Python Syntax</b>	ClcEval(<SoluName>, <VarArray>, [Optional <FieldType>])
<b>Python Example</b>	oModule.ClcEval( "Setup1: LastAdaptive", [ "Freq:=", "10GHz", "Phase:=", "0deg" ] )

<b>VB Syntax</b>	ClcEval <SoluName>, <VarArray>, [Optional <FieldType>]
------------------	--------------------------------------------------------

<b>VB Example</b> <pre>oModule.ClcEval "Setup1: LastAdaptive", _     Array ("Freq:=", "10GHz", _         "Phase:=", "0deg")</pre>
--------------------------------------------------------------------------------------------------------------------------------------

## ClcMaterial

Performs a material operation on the top stack element.

<b>UI Access</b>	Click <b>Matl...</b> in the <b>Fields Calculator</b> dialog.									
<b>Parameters</b>	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;MatString&gt;</td> <td>String</td> <td>The material property to apply.</td> </tr> <tr> <td>&lt;OpString&gt;</td> <td>String</td> <td>Name of operation. Possible values are "mult", or "div".</td> </tr> </tbody> </table>	Name	Type	Description	<MatString>	String	The material property to apply.	<OpString>	String	Name of operation. Possible values are "mult", or "div".
Name	Type	Description								
<MatString>	String	The material property to apply.								
<OpString>	String	Name of operation. Possible values are "mult", or "div".								
<b>Return Value</b>	None.									

<b>Python Syntax</b>	ClcMaterial(<MatString>, <OpString>)
<b>Python Example</b>	oModule.ClcMaterial("Permeability (mu)" "mult")

<b>VB Syntax</b>	ClcMaterial <MatString>, <OpString>
<b>VB Example</b>	oModule.ClcMaterial "Permeability (mu)" "mult"

## ClcMaterialValue

Shows the value of the material property without performing any operation.

UI Access	Select <b>None</b> in the <b>Material Operation</b> dialog.						
Parameters	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>&lt;MaterialName&gt;</td><td>String</td><td>Name of specified material property.</td></tr></tbody></table>	Name	Type	Description	<MaterialName>	String	Name of specified material property.
Name	Type	Description					
<MaterialName>	String	Name of specified material property.					
Return Value	None.						

Python Syntax	ClcMaterialValue(<MaterialName>)
Python Example	<pre>oModule.ClcMaterialValue("Mass Density")</pre>

VB Syntax	ClcMaterialValue <MaterialName>
VB Example	<pre>oModule.ClcMaterialValue "Mass Density"</pre>

## ClearAllNamedExpr

Clears all user-defined named expressions from the list.

UI Access	Click <b>ClearAll</b> in the <b>Fields Calculator</b> dialog.
Parameters	None.
Return Value	None.

Python Syntax	ClearAllNamedExpr()
---------------	---------------------

<b>Python Example</b>	<code>oModule.ClearAllNamedExpr()</code>
-----------------------	------------------------------------------

<b>VB Syntax</b>	<code>ClearAllNamedExpr</code>
------------------	--------------------------------

<b>VB Example</b>	<code>oModule.ClearAllNamedExpr</code>
-------------------	----------------------------------------

## CopyNamedExprToStack

Copies the named expression selected to the calculator stack.

<b>UI Access</b>	Select a named expression and then click <b>Copy to stack</b> .		
<b>Parameters</b>	Name <code>&lt;ExprName&gt;</code>	Type String	Description Name of the expression to be copied to the top of the stack.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	<code>CopyNamedExprToStack(&lt;ExprName&gt;)</code>
----------------------	-----------------------------------------------------

<b>Python Example</b>	<code>oModule.CopyNamedExprToStack ("Mag_JxE")</code>
-----------------------	-------------------------------------------------------

<b>VB Syntax</b>	<code>CopyNamedExprToStack &lt;ExprName&gt;</code>
------------------	----------------------------------------------------

<b>VB Example</b>	<code>oModule.CopyNamedExprToStack "Mag_JxE"</code>
-------------------	-----------------------------------------------------

## DeleteNamedExpr

Deletes the selected named expression from the list.

<b>UI Access</b>	Select a named expression and then click <b>Delete</b> .						
<b>Parameters</b>	<table border="1"><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;ExprName&gt;</td><td>String</td><td>Name of specified named expression.</td></tr></table>	Name	Type	Description	<ExprName>	String	Name of specified named expression.
Name	Type	Description					
<ExprName>	String	Name of specified named expression.					
<b>Return Value</b>	None.						

<b>Python Syntax</b>	<code>DeleteNamedExpr(&lt;ExprName&gt;)</code>
<b>Python Example</b>	<code>oModule.DeleteNamedExpr ("Mag_JxE")</code>

<b>VB Syntax</b>	<code>DeleteNamedExpr &lt;ExprName&gt;</code>
<b>VB Example</b>	<code>oModule.DeleteNamedExpr "Mag_JxE"</code>

## DoesNamedExpressionExists

Determines whether specified named expression exists.

<b>UI Access</b>	N/A						
<b>Parameters</b>	<table border="1"><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;ExpName&gt;</td><td>String</td><td>Name of the specified named expression.</td></tr></table>	Name	Type	Description	<ExpName>	String	Name of the specified named expression.
Name	Type	Description					
<ExpName>	String	Name of the specified named expression.					
<b>Return Value</b>	Boolean: <ul style="list-style-type: none"><li>• <b>1</b> - named expression exists.</li><li>• <b>0</b> - named expression does not exist.</li></ul>						

---

<b>Python Syntax</b>	DoesNamedExpressionExists(<ExpName>)
<b>Python Example</b>	<code>oModule.DoesNamedExpressionExists ("Mag_JxE")</code>

<b>VB Syntax</b>	DoesNamedExpressionExists <ExpName>
<b>VB Example</b>	<code>oModule.DoesNamedExpressionExists "Mag_JxE"</code>

## EnterComplex

Enters a complex number onto the stack.

<b>UI Access</b>	Click <b>Number</b> , and then click <b>Scalar. Complex</b> option is selected.						
<b>Parameters</b>	<table border="1"> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> <tr> <td>&lt;ComplexNum&gt;</td> <td>String</td> <td>String of complex value. Ex. "1 + 2j".</td> </tr> </table>	Name	Type	Description	<ComplexNum>	String	String of complex value. Ex. "1 + 2j".
Name	Type	Description					
<ComplexNum>	String	String of complex value. Ex. "1 + 2j".					
<b>Return Value</b>	None.						

<b>Python Syntax</b>	EnterComplex(<ComplexNum>)
<b>Python Example</b>	<code>oModule.EnterComplex ("1 + 2 j")</code>

<b>VB Syntax</b>	EnterComplex <ComplexNum>
<b>VB Example</b>	<code>oModule.EnterComplex "1 + 2 j"</code>

## EnterComplexVector

Enters a complex vector onto the stack.

<b>UI Access</b>	Click <b>Number</b> , and then click <b>Vector. Complex</b> option is selected.								
<b>Parameters</b>	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>&lt;ComplexVector&gt;</td><td>Array</td><td>Array of strings containing X, Y and Z complex values.</td></tr></tbody></table>			Name	Type	Description	<ComplexVector>	Array	Array of strings containing X, Y and Z complex values.
Name	Type	Description							
<ComplexVector>	Array	Array of strings containing X, Y and Z complex values.							
<b>Return Value</b>	None.								

<b>Python Syntax</b>	EnterComplexVector(<ComplexVector>)
<b>Python Example</b>	<pre>oModule.EnterComplexVector ([     "1 + 2 j",     "1 + 2 j",     "1 + 2 j" ])</pre>

<b>VB Syntax</b>	EnterComplexVector <ComplexVector>
<b>VB Example</b>	<pre>oModule.EnterComplexVector _     Array("1 + 2 j", _</pre>

	"1 + 2 j", "1 + 2 j")
--	--------------------------

## EnterCoord

Enters a coordinate system defined in the 3D Modeler editor.

<b>UI Access</b>	Click <b>Geometry</b> and then select <b>Coord</b> .		
<b>Parameters</b>	Name <CoordName>	Type String	Description Name of a coordinate system defined in the 3D Modeler editor.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	EnterCoord(<CoordName>)
<b>Python Example</b>	oModule.EnterCoord ("Global")

<b>VB Syntax</b>	EnterCoord <CoordName>
<b>VB Example</b>	oModule.EnterCoord "Global"

## EnterEdge

Enters an edge defined in the 3D Modeler editor.

<b>UI Access</b>	N/A		
<b>Parameters</b>	Name	Type	Description

	<i>&lt;EdgeName&gt;</i>	String	Name of an edge defined in the 3D Modeler editor.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	EnterEdge(<EdgeName>)
<b>Python Example</b>	<code>oModule.EnterPoint ("Edge_1")</code>

<b>VB Syntax</b>	EnterEdge <EdgeName>
<b>VB Example</b>	<code>oModule.EnterPoint "Edge_1"</code>

## EnterLine

Enters a line defined in the 3D Modeler editor.

<b>UI Access</b>	Click <b>Geometry</b> and then select <b>Line</b>		
<b>Parameters</b>	Name	Type	Description
	<i>&lt;LineName&gt;</i>	String	Name of a line defined in the 3D Modeler editor.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	EnterLine(<LineName>)
<b>Python Example</b>	<code>oModule.EnterLine ("Line1")</code>

---

<b>VB Syntax</b>	EnterLine <LineName>
<b>VB Example</b>	oModule.EnterLine "Line1"

## EnterOutputVar

Enters Output Vars, only valid for Eigenmode problems.

<b>UI Access</b>	Click <b>Geometry</b> and then select <b>Output Vars</b> .									
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;VarName&gt;</td> <td>String</td> <td>Name of the output vars. Only 'freq' is supported.</td> </tr> <tr> <td>&lt;VarType&gt;</td> <td>String</td> <td>Type of the output vars. 'Complex' type.</td> </tr> </tbody> </table>	Name	Type	Description	<VarName>	String	Name of the output vars. Only 'freq' is supported.	<VarType>	String	Type of the output vars. 'Complex' type.
Name	Type	Description								
<VarName>	String	Name of the output vars. Only 'freq' is supported.								
<VarType>	String	Type of the output vars. 'Complex' type.								
<b>Return Value</b>	None.									

<b>Python Syntax</b>	EnterOutputVar(<VarName>, <VarType>)
<b>Python Example</b>	oModule.EnterOutputVar ("Freq", "Complex")

<b>VB Syntax</b>	EnterOutputVar <VarName>, <VarType>
<b>VB Example</b>	oModule.EnterOutputVar "Freq", "Complex"

## EnterPoint

Enters a point defined in the 3D Modeler editor.

<b>UI Access</b>	Click <b>Geometry</b> and then select <b>Point</b> .								
<b>Parameters</b>	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>&lt;PointName&gt;</td><td>String</td><td>Name of a point defined in the 3D Modeler editor.</td></tr></tbody></table>			Name	Type	Description	<PointName>	String	Name of a point defined in the 3D Modeler editor.
Name	Type	Description							
<PointName>	String	Name of a point defined in the 3D Modeler editor.							
<b>Return Value</b>	None.								

<b>Python Syntax</b>	EnterPoint(<PointName>)
<b>Python Example</b>	oModule.EnterPoint ("Point1")

<b>VB Syntax</b>	EnterPoint <PointName>
<b>VB Example</b>	oModule.EnterPoint "Point1"

## EnterQty

Enters a field quantity.

<b>UI Access</b>	Click <b>Quantity</b> , and then select from the list.								
<b>Parameters</b>	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>&lt;FieldQty&gt;</td><td>String</td><td>The field quantity to be entered onto the stack.</td></tr></tbody></table>			Name	Type	Description	<FieldQty>	String	The field quantity to be entered onto the stack.
Name	Type	Description							
<FieldQty>	String	The field quantity to be entered onto the stack.							
<b>Return Value</b>	None.								

---

<b>Python Syntax</b>	EnterQty(<FieldQty>)
<b>Python Example</b>	oModule.EnterQty("E")

<b>VB Syntax</b>	EnterQty <FieldQty>
<b>VB Example</b>	oModule.EnterQty "E"

## EnterScalar

Enters a scalar onto the stack.

<b>UI Access</b>	Click <b>Number</b> and then click <b>Scalar. Complex</b> option not selected.						
<b>Parameters</b>	<table border="1"> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> <tr> <td>&lt;Scalar&gt;</td> <td>Double</td> <td>The real number to enter onto the stack.</td> </tr> </table>	Name	Type	Description	<Scalar>	Double	The real number to enter onto the stack.
Name	Type	Description					
<Scalar>	Double	The real number to enter onto the stack.					
<b>Return Value</b>	None.						

<b>Python Syntax</b>	EnterScalar(<Scalar>)
<b>Python Example</b>	oModule.EnterScalar(3.14159265358979)

<b>VB Syntax</b>	EnterScalar <Scalar>
<b>VB Example</b>	oModule.EnterScalar 3.14159265358979

## EnterScalarFunc

Enters a scalar function.

<b>UI Access</b>	Click <b>Function</b> and then select <b>Scalar</b> .		
<b>Parameters</b>	Name <VarName>	Type String	Description Name of a variable to enter as a scalar function onto the stack.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	EnterScalarFunc(<VarName>)
<b>Python Example</b>	oModule.EnterScalarFunc ("Phase")

<b>VB Syntax</b>	EnterScalarFunc <VarName>
<b>VB Example</b>	oModule.EnterScalarFunc "Phase"

## EnterSurf

Enters a surface defined in the 3D Modeler editor.

<b>UI Access</b>	Click <b>Geometry</b> and then select <b>Surface</b> .		
<b>Parameters</b>	Name <SurfName>	Type String	Description Name of a surface defined in the 3D Modeler editor.

<b>Return Value</b>	None.
---------------------	-------

<b>Python Syntax</b>	EnterSurf(< <i>SurfName</i> >)
<b>Python Example</b>	<code>oModule.EnterSurf ("Rectangle1")</code>

<b>VB Syntax</b>	EnterSurf < <i>SurfName</i> >
<b>VB Example</b>	<code>oModule.EnterSurf "Rectangle1"</code>

## EnterVector

Enters a vector onto the stack.

<b>UI Access</b>	Click <b>Number</b> , and then click <b>Vector</b> . <b>Complex</b> option not selected.								
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;<i>VectorArray</i>&gt;</td> <td>Array</td> <td>Array of strings containing X, Y and Z components of the vector.</td> </tr> </tbody> </table>			Name	Type	Description	< <i>VectorArray</i> >	Array	Array of strings containing X, Y and Z components of the vector.
Name	Type	Description							
< <i>VectorArray</i> >	Array	Array of strings containing X, Y and Z components of the vector.							
<b>Return Value</b>	None.								

<b>Python Syntax</b>	EnterVector(< <i>VectorArray</i> >)
<b>Python Example</b>	<code>oModule.EnterVector([1.0, 1.0, 1.0])</code>

<b>VB Syntax</b>	EnterVector < <i>VectorArray</i> >
------------------	------------------------------------

**VB Example**

```
oModule.EnterVector Array(1.0, 1.0, 1.0)
```

## EnterVectorFunc

Enters a vector function.

<b>UI Access</b>	Click <b>Function</b> and then select <b>Vector</b> .		
<b>Parameters</b>	Name <VarNames>	Type Array	Description Array of strings containing names of a variable for the X, Y, and Z coordinates, respectively, to enter as a vector function on the stack.
<b>Return Value</b>	None.		

**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**.

<b>Parameters</b>	<table border="1"> <tr> <th>Name</th><th>Type</th><th>Description</th></tr> <tr> <td>&lt;VolumeName&gt;</td><td>String</td><td>Name of a volume defined in the 3D Modeler editor.</td></tr> </table>	Name	Type	Description	<VolumeName>	String	Name of a volume defined in the 3D Modeler editor.
Name	Type	Description					
<VolumeName>	String	Name of a volume defined in the 3D Modeler editor.					
<b>Return Value</b>	None.						

<b>Python Syntax</b>	EnterVol(<VolumeName>)
<b>Python Example</b>	oModule.EnterVol ("Box1")

<b>VB Syntax</b>	EnterVol <VolumeName>
<b>VB Example</b>	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.

<b>UI Access</b>	Click <b>Export</b> , and then click <b>On Grid</b> .																											
<b>Parameters</b>	<table border="1"> <tr> <td>Name</td> <td>Type</td> <td>Description</td> </tr> <tr> <td>&lt;OutputFile&gt;</td> <td>String</td> <td>Name of the output file.</td> </tr> <tr> <td>&lt;Min&gt;</td> <td>Array</td> <td>Minimum values for the coordinate components of the grid system</td> </tr> <tr> <td>&lt;Max&gt;</td> <td>Array</td> <td>Maximum values for the coordinate components of the grid system</td> </tr> <tr> <td>&lt;Spacing&gt;</td> <td>Array</td> <td>Spacing values for the coordinate components of the grid system</td> </tr> <tr> <td>&lt;SolnName&gt;</td> <td>String</td> <td>Name of the simulation setup</td> </tr> <tr> <td>&lt;VarVals&gt;</td> <td>Array</td> <td>Array of strings containing setup definitions.</td> </tr> <tr> <td>&lt;IncludePoints&gt;</td> <td>Boolean</td> <td>Optional. Specifies whether include points in the output file.</td> </tr> <tr> <td>&lt;CSType&gt;</td> <td>String</td> <td>Optional. Type of coordinate system. "Cartesian" (default)   "Cylindrical"   "Spherical"</td> </tr> </table>	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"
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"																										

	<b>&lt;Offsets&gt;</b>	Array	Optional. Origin for the offset coordinate system. For Cartesian, x, y, z, for Cylindrical, R, Phi, Z, for Spherical, Rho, Theta, Phi.
	<b>&lt;ByCount&gt;</b>	Boolean	Optional.
<b>Return Value</b>	None.		

**Note:** Regarding the **ExportOnGrid** legacy script which only has “IncludePtInOutput” argument (the last Boolean one), AEDT can still read it and assign other new arguments as default values. Those default values are RefCSName = “global”, PtInSI = “True”, FieldInRefCS = “False”

<b>Python Syntax</b>	ExportOnGrid(<OutputFile>, <Min>, <Max>, <Spacing>, <SolnName>, <SolnParameters>, [<ExportOption>, "IncludePointsInOutput:=", <boolean>], "RefCSName:=", <CSName>, "PtsInSI:=", <boolean>, "FieldRefCS:=", <boolean>], "<CSType>", [<Offsets>, <ByCount>])
<b>Python Example</b>	<pre> oModule.ExportOnGrid("C:\offset_grid_model_unit_ref.fld",                      ["-1mm", "16mm", "0mm"],                      ["1mm", "18mm", "1mm"],                      ["2mm", "2mm", "1mm"],                      "4500MHz : LastAdaptive",                      ["Freq:=", "4.5GHz", "Phase:=" , "0deg"],                      [                          "NAME:ExportOption",                          "IncludePtInOutput:=" , True,                          "RefCSName:=" , "offset",                          "PtInSI:=" , False,                      ] ) </pre>

	<pre>         "FieldInRefCS:="           , True     ] ,     "Cartesian", ["0mm", "0mm", "0mm"], False ) </pre>
--	----------------------------------------------------------------------------------------------------------------

<b>VB Syntax</b>	ExportOnGrid(<OutputFile>, <Min>, <Max>, <Spacing>, <SolnName>, <SolnParameters>, [<ExportOption>, "IncludePointsInOutput:=", <boolean>], "RefCSName:=", <CSName>, "PtsInSI:=", <boolean>, "FieldRefCS:=", <boolean>], "<CSType>", [<Offsets>, <ByCount>])
<b>VB Example</b>	<pre> oModule.ExportOnGrid "C:/Grid.fld", _ Array( "0mm", "0deg", "-25mm"), _ Array("20mm", "90deg", "125mm"), _ Array("10mm", "45deg", "50mm"), _ "Setup1 : LastAdaptive", _ Array("Freq:=", "10000Hz", "Phase:=", "0deg"), _ 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>

```
"4500MHz : LastAdaptive",
    Array("Freq:=", "4.5GHz", "Phase:=" , "0deg"),
    [
        "NAME:ExportOption",
        "IncludePtInOutput:=" , True,
        "RefCSName:=" , "offset",
        "PtInSI:=" , False,
        "FieldInRefCS:=" , True
    ],
    "Cartesian", Array("0mm", "0mm", "0mm"), False
)
```

## ExportToFile [Fields Calculator]

Evaluates the top stack element at a set of points specified in an external file and exports the data to a file.

### Note:

Regarding to legacy **ExportToFile** script which only has “IncludePtInOutput” argument (the last Boolean one), AEDT can still read it and assign other new arguments as default values. Those default values are RefCSName = “global”, PtInSI = “True”, FieldInRefCS = “False”

UI Access

Click **Export**, and then click **To File**.

	Name	Type	Description
<b>Parameters</b>	<ReportName>	String	Name of report to be exported.
	<FileName>	String	Full path of the exported image file name; with extension of .txt - Post processor format file .csv - Comma-delimited data file .tab - Tab-separated file .dat - Ansys plot data file
	<solution>		Solution Name
	<freq>		frequency
	<phase>		phase
	<Export Options>		Whether to include points in output, RefCSName, Pts in SI units <boolean>, Field in RefCS, <boolean>.
	<b>Return Value</b>	None.	

<b>Python Syntax</b>	ExportToFile(<ReportName>, <FileName>, <solution>, ["Freq:=", <freq>, "Phase:=", "<phase>"], <ExportOptions>)
<b>Python Example</b>	<pre> oModule.ExportToFile("C:\\\\offset_file_model_unit_ref.fld", "C:\\\\offset_SI.pts", "4500MHz : LastAdaptive", [     "Freq:="      , "4.5GHz",     "Phase:="     , "0deg" ],</pre>

```
[  
    "NAME:ExportOption",  
    "IncludePtInOutput:=" , True,  
    "RefCSName:=" , "offset",  
    "PtInSI:=" , False,  
    "FieldInRefCS:=" , True  
])
```

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",</pre>

```

    "PtInSI:="           , False,
    "FieldInRefCS:="     , True
]
)

```

## GetTopEntryValue

Gets the value of the top entry of the calculator stack.

<b>UI Access</b>	N/A		
<b>Parameters</b>	Name	Type	Description
	<SolnName>	String	Name of specified solution.
<b>Return Value</b>	Array of strings containing top entry values.		

<b>Python Syntax</b>	GetTopEntryValue(<SolnName>, <VarVals>)
<b>Python Example</b>	<pre> oModule.GetTopEntryValue(     "Setup1:LastAdaptive",     ["Freq:=", "1GHz", "Phase:=", "0deg", "x_size:=", "2mm"] ) </pre>

<b>VB Syntax</b>	GetTopEntryValue <SolnName>, <VarVals>
------------------	----------------------------------------

**VB Example**

```
oModule.GetTopEntryValue "Setup1:LastAdaptive", _  
    Array("Freq:=", "1GHz", "Phase:=", "0deg", _  
        "x_size:=", "2mm")
```

## LoadNamedExpressions

Loads a named expression definition from a saved file.

<b>UI Access</b>	In the Fields Calculator, click <b>Load From...</b> in the Library area.												
<b>Parameters</b>	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>&lt;FileName&gt;</td><td>String</td><td>Filename and full path to the file to hold the named expression definition.</td></tr><tr><td>&lt;FieldType&gt;</td><td>String</td><td>For products with just one filed type, it is set to "Fields".</td></tr><tr><td>&lt;NamedExpr&gt;</td><td>Array</td><td>rray of strings containing the names of expression definitions to load from the file.</td></tr></tbody></table>	Name	Type	Description	<FileName>	String	Filename and full path to the file to hold the named expression definition.	<FieldType>	String	For products with just one filed type, it is set to "Fields".	<NamedExpr>	Array	rray of strings containing the names of expression definitions to load from the file.
Name	Type	Description											
<FileName>	String	Filename and full path to the file to hold the named expression definition.											
<FieldType>	String	For products with just one filed type, it is set to "Fields".											
<NamedExpr>	Array	rray of strings containing the names of expression definitions to load from the file.											
<b>Return Value</b>	None.												

**Python Syntax**

```
LoadNamedExpressions(<FileName>, <FieldType>, <NamedExpr>)
```

**Python Example**

```
oModule.LoadNamedExpressions (  
    "C:\Ansoft\PersonalLib\smth.clc",  
    "Fields", ["smoothedtemp"])
```

**VB Syntax**

```
LoadNamedExpressions <FileName>, <FieldType>, <NamedExpr>
```

<b>VB Example</b>	<pre> oModule.LoadNamedExpressions _ "C:\Ansoft\PersonalLib\smth.clc", _ "Fields", Array("smoothedtemp") </pre>
-------------------	-----------------------------------------------------------------------------------------------------------------

## SaveNamedExpressions

Saves a named expression definition to a file.

<b>UI Access</b>	In the Fields Calculator, click <b>Save To...</b> in the Library area.												
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;FileName&gt;</td> <td>String</td> <td>Filename and full path to the file to hold the named expression definition.</td> </tr> <tr> <td>&lt;NamedExprs&gt;</td> <td>Array</td> <td>Array of strings containing the names of expression definitions to load from the file.</td> </tr> <tr> <td>&lt;Overwrite&gt;</td> <td>Boolean</td> <td>Specifies whether to overwrite the file.</td> </tr> </tbody> </table>	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.
Name	Type	Description											
<FileName>	String	Filename and full path to the file to hold the named expression definition.											
<NamedExprs>	Array	Array of strings containing the names of expression definitions to load from the file.											
<Overwrite>	Boolean	Specifies whether to overwrite the file.											
<b>Return Value</b>	None.												

<b>Python Syntax</b>	SaveNamedExpressions(<FileName>, <NamedExprs>, <Overwrite>)
<b>Python Example</b>	<pre> oModule.SaveNamedExpressions( "C:\Ansoft\PersonalLib\smth.clc", ["smoothedtemp"], True) </pre>

<b>VB Syntax</b>	SaveNamedExpressions <FileName>, <NamedExprs>, <Overwrite>
<b>VB Example</b>	<pre> oModule.SaveNamedExpressions _ </pre>

```
"C:\Ansoft\PersonalLib\smth.clc", _  
    Array("smoothedtemp"), true
```

# 18 - 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.

<b>UI Access</b>	N/A						
<b>Parameters</b>	<table border="1"><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;SetupName&gt;</td><td>String</td><td>Name of radiation field setup to be copied.</td></tr></table>	Name	Type	Description	<SetupName>	String	Name of radiation field setup to be copied.
Name	Type	Description					
<SetupName>	String	Name of radiation field setup to be copied.					
<b>Return Value</b>	None.						

<b>Python Syntax</b>	CopyRadFieldSetup(<SetupName>)
<b>Python Example</b>	oModule.CopyRadFieldSetup ("RadField1")

<b>VB Syntax</b>	CopyRadFieldSetup <SetupName>
<b>VB Example</b>	oModule.CopyRadFieldSetup "RadField1"

## DeleteFarFieldSetup

Deletes an existing far-field setup.

<b>UI Access</b>	Right-click on a far field setup under <b>Radiation</b> in the project tree, select <b>Delete</b> .								
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;SetupName&gt;</td> <td>Array</td> <td>Name of setup(s) to be deleted.</td> </tr> </tbody> </table>			Name	Type	Description	<SetupName>	Array	Name of setup(s) to be deleted.
Name	Type	Description							
<SetupName>	Array	Name of setup(s) to be deleted.							
<b>Return Value</b>	None.								

<b>Python Syntax</b>	DeleteFarFieldSetup(<SetupName>)
<b>Python Example</b>	<code>oModule.DeleteFarFieldSetup(["Infinite Sphere1"])</code>

<b>VB Syntax</b>	
<b>VB Example</b>	<code>oModule.DeleteFarFieldSetup _</code> <code>Array("Infinite Sphere1")</code>

## DeleteNearFieldSetup

Deletes an existing near-field line radiation setup.

<b>UI Access</b>	<b>Project Manager &gt; Radiation.</b> Right-click a setup and click <b>Delete</b> .								
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;NameArray&gt;</td> <td>Array</td> <td>List of fields for deletion.</td> </tr> </tbody> </table>			Name	Type	Description	<NameArray>	Array	List of fields for deletion.
Name	Type	Description							
<NameArray>	Array	List of fields for deletion.							
<b>Return Value</b>	None.								

<b>Python Syntax</b>	DeleteNearFieldSetup( <i>NameArray</i> )
<b>Python Example</b>	oModule.DeleteNearFieldSetup(['Line1', 'Sphere1', 'Box2', 'Rectangle1'])

<b>VB Syntax</b>	DeleteNearFieldSetup( <i>NameArray</i> )
<b>VB Example</b>	oModule.DeleteNearFieldSetup Array("Line1", "Sphere1", "Box2", "Rectangle1")

## DeleteSetup

Deletes specified radiation field setup(s).

<b>UI Access</b>	Right-click on a setup under <b>Radiation</b> in the project, select <b>Delete</b> .						
<b>Parameters</b>	<table border="1"><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;SetupName&gt;</td><td>Array</td><td>Array of setup names to be deleted.</td></tr></table>	Name	Type	Description	<SetupName>	Array	Array of setup names to be deleted.
Name	Type	Description					
<SetupName>	Array	Array of setup names to be deleted.					
<b>Return Value</b>	None.						

<b>Python Syntax</b>	DeleteSetup(<SetupName>)
<b>Python Example</b>	oModule.DeleteSetup ( ["Rad Field1"] )

<b>VB Syntax</b>	DeleteSetup <SetupName>
------------------	-------------------------

<b>VB Example</b>	<code>oModule.DeleteSetup Array("Rad Field1")</code>
-------------------	------------------------------------------------------

## EditRadiatedPowerCalculationMethod

Selects the method used to calculate radiated power.

<b>UI Access</b>	Right-click Radiation in the project tree, select <b>Radiated Power Calculation Method</b> .		
<b>Parameters</b>	Name <i>&lt;PowerCalculationMethod&gt;</i>	Type String	Description One of: "Auto", "Radiation Surface Integral", or "Far Field Integral".
<b>Return Value</b>	None.		

<b>Python Syntax</b>	<code>EditRadiatedPowerCalculationMethod(&lt;PowerCalculationMethod&gt;)</code>
<b>Python Example</b>	<code>oModule.EditRadiatedPowerCalculationMethod("Auto")</code> <code>oModule.EditRadiatedPowerCalculationMethod("Radiation Surface Integral")</code> <code>oModule.EditRadiatedPowerCalculationMethod("Far Field Integral")</code>

<b>VB Syntax</b>	<code>EditRadiatedPowerCalculationMethod &lt;PowerCalculationMethod&gt;</code>
<b>VB Example</b>	<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.

<b>Python Syntax</b>	GetChildNames ()
<b>Python Example</b>	<pre>GetChildNames() //return all setup names.</pre>

### **GetChildObject [Radiation]**

Gets a Setup Object of the Radiation module.

<b>Python Syntax</b>	GetChildObject()
<b>Python Example</b>	<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.

<b>Python Syntax</b>	GetChildTypes ()
----------------------	------------------

<b>Python Example</b> <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.

<b>Python Syntax</b> <pre>GetPropNames ()</pre>	
<b>Python Example</b> <pre>oRadModule.GetPropNames () oRadModule.GetPropNames (True) oRadModule.GetPropNames (False)</pre>	

## GetPropValue [Radiation]

Get the property value for a setup object.

<b>UI Access</b> NA			
<b>Parameters</b>	Name PropPath Value	Type text string	Description A Property path begin with a setup name. See <a href="#">property path discussion here</a> .
<b>Return Value</b>	The property value.		

<b>Python Syntax</b>	GetPropValue(propPath)
<b>Python Example</b>	<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.

<b>UI Access</b>	N/A						
<b>Parameters</b>	<table border="1"><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;RadiationType&gt;</td><td>String</td><td>Type of radiation.</td></tr></table>	Name	Type	Description	<RadiationType>	String	Type of radiation.
Name	Type	Description					
<RadiationType>	String	Type of radiation.					
<b>Return Value</b>	Array of strings containing setup names.						

<b>Python Syntax</b>	GetSetupNames(<RadiationType>)
<b>Python Example</b>	<pre>oModule.GetSetupNames ("Infinite Sphere")</pre>

<b>VB Syntax</b>	GetSetupNames <RadiationType>
<b>VB Example</b>	<pre>oModule.GetSetupNames "Infinite Sphere"</pre>

## PasteRadFieldSetup

Pastes a copied radiation field setup.

<b>UI Access</b>	N/A
<b>Parameters</b>	None.
<b>Return Value</b>	String name of pasted setup.

<b>Python Syntax</b>	PasteRadFieldSetup()
<b>Python Example</b>	<code>oModule.PasteRadFieldSetup ()</code>

<b>VB Syntax</b>	PasteRadFieldSetup
<b>VB Example</b>	<code>oModule.PasteRadFieldSetup</code>

## RenameSetup [Radiation]

Renames an existing radiation setup.

<b>UI Access</b>	Right-click a radiation setup in the project tree, and then click <b>Rename</b> on the shortcut menu.	
<b>Parameters</b>	Name	Type
	<i>&lt;OldName&gt;</i>	String
	Name of specified setup to be renamed.	
	<i>&lt;NewName&gt;</i>	String
New name for the setup.		
<b>Return Value</b>	None.	

<b>Python Syntax</b>	RenameSetup(<OldName>, <NewName>)
<b>Python Example</b>	<code>oModule.RenameSetup ("Sphere1", "MyNearSphere")</code>

<b>VB Syntax</b>	RenameSetup <OldName>, <NewName>
<b>VB Example</b>	<code>oModule.RenameSetup "Sphere1", "MyNearSphere"</code>

## SetPropValue [Radiation]

Sets the property value for the active Radiation property object.

<b>UI Access</b>	Edit Properties on ProjectTree objects		
<b>Parameters</b>	Name	Type	Description
	propPath	text string	A Property path begin with a setup name. See <a href="#">property path discussion here</a> .
<b>Return Value</b>	True if the property is found and the new value is valid. Otherwise return False.		

<b>Python Syntax</b>	SetPropValue(propPath, value)
<b>Python Example</b>	<code>oRadModule.SetPropValue('Line1/Num Points', 100) ; Set the Line1 setups' "Num Points" property to 100.</code>

<b>VB Syntax</b>	SetPropValue(propPath, value)
------------------	-------------------------------

<b>VB Example</b>	<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 an overlay on far field plots.

<b>UI Access</b>	Click on <b>Save for Overlay</b> in the <b>Antenna Parameters</b> dialog.		
<b>Parameters</b>	Name <code>&lt;AntennaParams&gt;</code>	Type Array	Description Structured array.  <code>Array ("NAME:&lt;AntennaOverlayName&gt;","</code> <code>"Setup:=", &lt;string far field radiation setup name&gt;,</code> <code>"Solution:=", &lt;string for the Solution Setup and Pass&gt;,</code>

		"IntrinsicVariation:=", <string for variation definition>, "UseNominalDesign:=", <boolean>, "DesignVariation:=", <string>, "Calculations:=" , <list of each antenna parameter to display>)
<b>Return Value</b>	None.	

<b>Python Syntax</b>	AddAntennaOverlay(<AntennaParams>)
<b>Python Example</b>	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" ])

<b>VB Syntax</b>	AddAntennaOverlay <AntennaParams>
<b>VB Example</b>	<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.

<b>UI Access</b>	Right-click on <b>Radiation</b> in the project tree, select <b>Insert Far Field Setup &gt; Source Group...</b>								
<b>Parameters</b>	<table border="1"> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> <tr> <td>&lt;SourceGroupParams&gt;</td> <td>Array</td> <td>           Structured array            Array ("NAME:&lt;SourceGroupName&gt;",            Array ("NAME:Sources",            &lt;SourceName&gt;, &lt;SourceName&gt;...))         </td> </tr> </table>	Name	Type	Description	<SourceGroupParams>	Array	Structured array Array ("NAME:<SourceGroupName>", Array ("NAME:Sources", <SourceName>, <SourceName>...))		
Name	Type	Description							
<SourceGroupParams>	Array	Structured array Array ("NAME:<SourceGroupName>", Array ("NAME:Sources", <SourceName>, <SourceName>...))							
<b>Return Value</b>	None.								

<b>Python Syntax</b>	AddRadFieldSourceGroup(<SourceGroupParams>)
<b>Python Example</b>	<pre>oModule.AddRadFieldSourceGroup(     [         "NAME:Source Group1",         ["NAME:Sources",          "feedpin1_T1",          "feedpin1_T2"]     ] )</pre>

<b>VB Syntax</b>	AddRadFieldSourceGroup <SourceGroupParams>
<b>VB Example</b>	<pre>oModule.AddRadFieldSourceGroup _     Array("NAME:Source Group1", _         Array("NAME:Sources", _             "feedpin1_T1", "feedpin1_T2"))</pre>

## EditAntennaOverlay

Edits parameters in an existing antenna overlay.

<b>UI Access</b>	N/A		
<b>Parameters</b>	Name <AntennaOverlayName>	Type String	Description Name of the antenna overlay to be edited.

	<code>&lt;AntennaParams&gt;</code>	Array	<p>Structured array.</p> <pre>Array ("NAME:&lt;AntennaOverlayName&gt;",       "Setup:=", &lt;string far field radiation setup name&gt;,       "Solution:=", &lt;string for the Solution Setup and Pass&gt;,       "IntrinsicVariation:=", &lt;string for variation definition&gt;,       "UseNominalDesign:=", &lt;boolean&gt;,       "DesignVariation:=", &lt;string&gt;,       "Calculations:=" , &lt;list of each antenna parameter to display&gt;)</pre>
<b>Return Value</b>	None.		

<b>Python Syntax</b>	<code>EditAntennaOverlay(&lt;AntennaOverlayName&gt;, &lt;AntennaParams&gt;)</code>
<b>Python Example</b>	<pre>oModule.EditAntennaOverlay("Antenna Parameter Overlay1", [      "NAME:Antenna Parameter Overlay1",     "Setup:=", "Infinite Sphere1",     "Solution:=", "4500MHz : LastAdaptive",     "IntrinsicVariation:=", "Freq=4.5GHz ",     "UseNominalDesign:=", False,     "DesignVariation:=", "",</pre>

```

    "Calculations:=", "Max U,Peak Directivity,Peak Gain,Radiation Efficiency,
    Front To Back Ratio,Total Efficiency,System Efficiency"
  ])
)

```

<b>VB Syntax</b>	EditAntennaOverlay <AntennaOverlayName>, <AntennaParams>
<b>VB Example</b>	<pre> oModule.AddAntennaOverlay "Antenna Parameter Overlay1", _ Array("NAME:Antenna Parameter Overlay1", _ "Setup:=", "Infinite Sphere1", _ "Solutions:=", "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.

<b>UI Access</b>	<b>Project Manager &gt; Radiation &gt; Box.</b> Double-click and edit information.		
<b>Parameters</b>	Name <BoxParams>	Type Array	Description Structured array.  Array("NAME:<SetupName>",

			<pre>"UseCustomRadiationSurface:=", &lt;bool&gt;, "CustomRadiationSurface:=", &lt;FaceListName&gt;, "Length:=", &lt;string&gt;, "Width:=", &lt;string&gt;, "LengthSamples:=", &lt;int&gt;, "WidthSamples:=", &lt;int&gt;, "CoordSystem:=", &lt;CSname&gt; "Height:=", &lt;string&gt;, "HeightSamples:=", &lt;int&gt;)</pre> <p>For UseCustomRadiationSurface parameter, provide value if True. If False, radiation boundary/PML boundaries will be used as radiation surfaces.</p>
<b>Return Value</b>	None.		

<b>Python Syntax</b>	EditBoxSetup(<BoxParams>)
<b>Python Example</b>	<pre>oModule.EditBoxSetup (     ['NAME:MyBox',      'UseCustomRadiationSurface:=', False,      'Length:=' , '20mm',      'Width:=' , '20mm',      'LengthSamples:=' , 21,</pre>

```
'WidthSamples:=' , 21,  
'CoordSystem:=' , 'Global',  
'Height:=' , '20mm',  
'HeightSamples:=' , 21  
])
```

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.

<b>UI Access</b>	Double-click a radiation setup in the project tree to modify its settings.		
	Name	Type	Description
<b>Parameters</b>	<p>&lt;<i>FarFieldSphereName</i>&gt;</p> <p>&lt;<i>FarFieldSphereParams</i>&gt;</p>	<p>String</p> <p>Array</p>	<p>Name of the Far Field Sphere Radiation Setup to be edited.</p> <p>Structured array.</p> <pre> Array("NAME:&lt;InfSphereName&gt;",  "UseCustomRadiationSurface:=", &lt;boolean&gt;,  "CSDefinition:=", &lt;string, "Theta-Phi"   "Az Over EI"   "EI Over AS"&gt;,  "Polarization:=", &lt;string, "Linear"   "Slant"&gt;,  "ThetaStart:=", "&lt;value&gt;&lt;unit&gt;",  "ThetaStop:=", "&lt;value&gt;&lt;unit&gt;",  "ThetaStep:=", "&lt;value&gt;&lt;unit&gt;",  "PhiStart:=", "&lt;value&gt;&lt;unit&gt;",  "PhiStop:=", "&lt;value&gt;&lt;unit&gt;",  "PhiStep:=", "&lt;value&gt;&lt;unit&gt;",  "UseLocalCS:=", boolean) </pre>
<b>Return Value</b>	None		

<b>Python Syntax</b>	EditFarFieldSphereSetup(< <i>FarFieldSphereName</i> >, < <i>FarFieldSphereParams</i> >)
<b>Python Example</b>	<pre> oModule.EditFarFieldSphereSetup("Infinite Sphere1", [</pre>

```
"NAME:Infinite Sphere1",
"UseCustomRadiationSurface:=", False,
"ThetaStart:=" , "0deg",
"ThetaStop:=" , "180deg",
"ThetaStep:=" , "3deg",
"PhiStart:=" , "-180deg",
"PhiStop:=" , "180deg",
"PhiStep:=" , "3deg",
"UseLocalCS:=" , False
])
```

<b>VB Syntax</b>	EditFarFieldSphereSetup <FarFieldSphereName>, <FarFieldSphereParams>
<b>VB Example</b>	<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.

<b>UI Access</b>	Double-click on a setup under <b>Radiation</b> in the project tree.											
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><code>&lt;InfSphereName&gt;</code></td> <td>String</td> <td>Name of the infinite sphere setup to be edited.</td> </tr> <tr> <td><code>&lt;InfSphereParams&gt;</code></td> <td>Array</td> <td> <p>Structured array.</p> <pre>Array("NAME:&lt;InfSphereName&gt;",       "UseCustomRadiationSurface:=", &lt;boolean&gt;,       "CSDefinition:=", &lt;string, "Theta-Phi"   "Az Over EI"         "EI Over AS"&gt;,       "Polarization:=", &lt;string, "Linear"   "Slant"&gt;,       "ThetaStart:=", "&lt;value&gt;&lt;unit&gt;",       "ThetaStop:=", "&lt;value&gt;&lt;unit&gt;",       "ThetaStep:=", "&lt;value&gt;&lt;unit&gt;",       "PhiStart:=", "&lt;value&gt;&lt;unit&gt;",       "PhiStop:=", "&lt;value&gt;&lt;unit&gt;",       "PhiStep:=", "&lt;value&gt;&lt;unit&gt;",       "UseLocalCS:=", boolean)</pre> </td> </tr> </tbody> </table>	Name	Type	Description	<code>&lt;InfSphereName&gt;</code>	String	Name of the infinite sphere setup to be edited.	<code>&lt;InfSphereParams&gt;</code>	Array	<p>Structured array.</p> <pre>Array("NAME:&lt;InfSphereName&gt;",       "UseCustomRadiationSurface:=", &lt;boolean&gt;,       "CSDefinition:=", &lt;string, "Theta-Phi"   "Az Over EI"         "EI Over AS"&gt;,       "Polarization:=", &lt;string, "Linear"   "Slant"&gt;,       "ThetaStart:=", "&lt;value&gt;&lt;unit&gt;",       "ThetaStop:=", "&lt;value&gt;&lt;unit&gt;",       "ThetaStep:=", "&lt;value&gt;&lt;unit&gt;",       "PhiStart:=", "&lt;value&gt;&lt;unit&gt;",       "PhiStop:=", "&lt;value&gt;&lt;unit&gt;",       "PhiStep:=", "&lt;value&gt;&lt;unit&gt;",       "UseLocalCS:=", boolean)</pre>		
Name	Type	Description										
<code>&lt;InfSphereName&gt;</code>	String	Name of the infinite sphere setup to be edited.										
<code>&lt;InfSphereParams&gt;</code>	Array	<p>Structured array.</p> <pre>Array("NAME:&lt;InfSphereName&gt;",       "UseCustomRadiationSurface:=", &lt;boolean&gt;,       "CSDefinition:=", &lt;string, "Theta-Phi"   "Az Over EI"         "EI Over AS"&gt;,       "Polarization:=", &lt;string, "Linear"   "Slant"&gt;,       "ThetaStart:=", "&lt;value&gt;&lt;unit&gt;",       "ThetaStop:=", "&lt;value&gt;&lt;unit&gt;",       "ThetaStep:=", "&lt;value&gt;&lt;unit&gt;",       "PhiStart:=", "&lt;value&gt;&lt;unit&gt;",       "PhiStop:=", "&lt;value&gt;&lt;unit&gt;",       "PhiStep:=", "&lt;value&gt;&lt;unit&gt;",       "UseLocalCS:=", boolean)</pre>										
<b>Return Value</b>	None.											

<b>Python Syntax</b>	EditInfiniteSphereSetup(<InfSphereName>, <InfSphereParams>)
<b>Python Example</b>	<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>

<b>VB Syntax</b>	EditInfiniteSphereSetup <InfSphereName>, <InfSphereParams>
<b>VB Example</b>	<pre> oModule.EditInfiniteSphereSetup "Infinite Sphere1", _ Array("NAME:Infinite Sphere1", _ </pre>

```

    "UseCustomRadiationSurface:=", false, _
    "CSDefinition:=", "Theta-Phi", _
    "Polarization:=", "Linear", _
    "ThetaStart:=", "0deg", "ThetaStop:=", "180deg", _
    "ThetaStep:=", "2deg", "PhiStart:=", "-180deg", _
    "PhiStop:=", "180deg", "PhiStep:=", "2deg", _
    "UseLocalCS:=", false)

```

## EditLineSetup

Modifies a near field line setup.

<b>UI Access</b>	Double-click on a setup under <b>Radiation</b> in the project tree.			
<b>Parameters</b>	Name <i>&lt;SetupName&gt;</i>	Type String	Description Name of the setup to be edited.	
	<i>&lt;LineParams&gt;</i>	Array	Structured array. Array ("NAME:<LineName>", "UseCustomRadiationSurface:=", <boolean>, "Line:=", <string, name of specified line>, "NumPts:=", "<integer, number of points>")	
<b>Return Value</b>	None.			

<b>Python Syntax</b>	EditLineSetup(<SetupName>, <LineParams>)
----------------------	------------------------------------------

<b>Python Example</b>	<pre>oModule.EditLineSetup("Line1", [     "NAME:Line1",     "UseCustomRadiationSurface:=", False,     "Line:=" , "Polyline1",     "NumPts:=" , "1000" ])</pre>
-----------------------	----------------------------------------------------------------------------------------------------------------------------------------------------------------

<b>VB Syntax</b>	EditLineSetup <SetupName>, <LineParams>
<b>VB Example</b>	<pre>oModule&gt;EditLineSetup "Line1", _ Array("NAME:Line1", _ "UseCustomRadiationSurface:=", false, _ "Line:=", "Polyline1", "NumPts:=", "1000")</pre>

## EditNearFieldBoxSetup

Modifies a near field box radiation field setup.

<b>UI Access</b>	N/A									
<b>Parameters</b>	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>&lt;BoxSetupName&gt;</td><td>String</td><td>Name of the near field box setup to be edited.</td></tr><tr><td>&lt;BoxSetupParams&gt;</td><td>Array</td><td>Structured array.</td></tr></tbody></table>	Name	Type	Description	<BoxSetupName>	String	Name of the near field box setup to be edited.	<BoxSetupParams>	Array	Structured array.
Name	Type	Description								
<BoxSetupName>	String	Name of the near field box setup to be edited.								
<BoxSetupParams>	Array	Structured array.								

			<pre>Array ("NAME:&lt;BoxSetupName&gt;",       "UseCustomRadiationSurface:=", &lt;boolean&gt;,       "Length:=", "&lt;value&gt;&lt;unit&gt;",       "Width:=", "&lt;value&gt;&lt;unit&gt;",       "LengthSamples:=", &lt;value&gt;,       "WidthSamples:=", &lt;value&gt;,       "CoordSystem:=", &lt;string name of coordinate system&gt;,       "Height:=", "&lt;value&gt;&lt;unit&gt;",       "HeightSamples:=", &lt;value&gt;)</pre>
<b>Return Value</b>	None.		

<b>Python Syntax</b>	EditNearFieldBoxSetup(<BoxSetupName>, <BoxSetupParams>)
<b>Python Example</b>	<pre>oModule.EditNearFieldBoxSetup("Box1", [     "NAME:Box1",     "UseCustomRadiationSurface:=", False,     "Length:" , "20meter",     "Width:" , "20meter",     "LengthSamples:" , 21,     "WidthSamples:" , 21,</pre>

```

    "CoordSystem:="           , "Global",
    "Height:="                , "20meter",
    "HeightSamples:="          , 21
]
)

```

<b>VB Syntax</b>	EditNearFieldBoxSetup <BoxSetupName>, <BoxSetupParams>
<b>VB Example</b>	<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.

<b>UI Access</b>	N/A		
<b>Parameters</b>	Name	Type	Description
	<SetupName>	String	Name of the setup to be edited.
	<LineParams>	Array	Structured array.

			Array("NAME:<LineName>",       "UseCustomRadiationSurface:=", <boolean>,       "Line:=", <string, name of specified line>,       "NumPts:=", "<integer, number of points>")
<b>Return Value</b>	None.		

<b>Python Syntax</b>	EditNearFieldLineSetup(<SetupName>, <LineParams>)
<b>Python Example</b>	<pre>oModule.EditNearFieldLineSetup("Line1", [     "NAME:Line1",     "UseCustomRadiationSurface:=", False,     "Line:=" , "Polyline1",     "NumPts:=" , "1000" ])</pre>

<b>VB Syntax</b>	EditNearFieldLineSetup <SetupName>, <LineParams>
<b>VB Example</b>	<pre>oModule.EditNearFieldLineSetup "Line1", _ Array("NAME:Line1", _ "UseCustomRadiationSurface:=", false, _ "Line:=", "Polyline1", "NumPts:=", "1000")</pre>

## EditNearFieldRectangleSetup

Modifies a near field rectangle radiation setup.

<b>UI Access</b>	N/A.		
<b>Parameters</b>	<b>Name</b> <i>&lt;RectangleSetupName&gt;</i>	Type String	Description Name of rectangle radiation setup to be edited.
	<b>&lt;RectangleSetupParams&gt;</b>	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>)
<b>Return Value</b>	None.		

<b>Python Syntax</b>	EditNearFieldRectangleSetup(<RectangleSetupName>, <RectangleSetupParams>)
<b>Python Example</b>	<pre>oModule.EditNearFieldRectangleSetup("Rectangle1", [     "NAME:Rectangle1",     ...]</pre>

```

    "UseCustomRadiationSurface:=", False,
    "Length:="                  , "20meter",
    "Width:="                   , "20meter",
    "LengthSamples:="           , 41,
    "WidthSamples:="            , 41,
    "CoordSystem:="             , "Global"
  )
)

```

<b>VB Syntax</b>	EditNearFieldRectangleSetup <RectangleSetupName>, <RectangleSetupParams>
<b>VB Example</b>	<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.

<b>UI Access</b>	N/A		
<b>Parameters</b>	Name <SphereSetupName>	Type String	Description Name of sphere radiation field setup to be edited.

	<code>&lt;SphereSetupParams&gt;</code>	Array	<p>Structured array.</p> <pre>Array("NAME:&lt;SphereSetupName&gt;",       "UseCustomRadiationSurface:=", &lt;boolean&gt;,       "Radius:=", "&lt;value&gt;&lt;unit&gt;",       "ThetaStart:=", "&lt;value&gt;&lt;unit&gt;",       "ThetaStop:=", "&lt;value&gt;&lt;unit&gt;",       "ThetaStep:=", "&lt;value&gt;&lt;unit&gt;",       "PhiStart:=", "&lt;value&gt;&lt;unit&gt;",       "PhiStop:=", "&lt;value&gt;&lt;unit&gt;",       "PhiStep:=", "&lt;value&gt;&lt;unit&gt;",       "UseLocalCS:=", &lt;boolean&gt;)</pre>
<b>Return Value</b>	None.		

<b>Python Syntax</b>	<code>EditNearFieldSphereSetup(&lt;SphereSetupName&gt;, &lt;SphereSetupParams&gt;)</code>
<b>Python Example</b>	<pre>oModule.EditNearFieldSphereSetup ("Sphere1", [     "NAME:Sphere2",     "UseCustomRadiationSurface:=", False,     "Radius:=" , "20meter",</pre>

```

    "ThetaStart:="           , "0deg",
    "ThetaStop:="            , "180deg",
    "ThetaStep:="            , "2deg",
    "PhiStart:="             , "-180deg",
    "PhiStop:="              , "180deg",
    "PhiStep:="              , "2deg",
    "UseLocalCS:="           , False
  )
)

```

<b>VB Syntax</b>	EditNearFieldSphereSetup <SphereSetupName>, <SphereSetupParams>
<b>VB Example</b>	<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.

<b>UI Access</b>	Double-click on a source group setup under <b>Radiation</b> in the project tree.
------------------	----------------------------------------------------------------------------------

	Name	Type	Description
<b>Parameters</b>	<code>&lt;SourceGroupName&gt;</code>	String	Name of the source group setup to be edited.
	<code>&lt;SourceGroupParams&gt;</code>	Array	Structured array  Array ("NAME:<SourceGroupName>"," Array ("NAME:Sources", <SourceName>, <SourceName>...))
<b>Return Value</b>	None.		

<b>Python Syntax</b>	<code>EditRadFieldSourceGroup(&lt;SourceGroupName&gt;, &lt;SourceGroupParams&gt;)</code>
<b>Python Example</b>	<pre> oModule.EditRadFieldSourceGroup("Source Group1", [     "NAME:Source Group1",     ["NAME:Sources",      "IncPWave1",      "IncPWave2"] ]) </pre>

<b>VB Syntax</b>	<code>EditRadFieldSourceGroup &lt;SourceGroupName&gt;, &lt;SourceGroupParams&gt;</code>
<b>VB Example</b>	<code>oModule.EditRadFieldSourceGroup "Source Group1", _</code>

```
Array("NAME:Source Group1", _  
      Array("NAME:Sources", _  
            "IncPWave1", "IncPWave2"))
```

## EditRectangleSetup

Modifies a near field rectangle radiation setup.

<b>UI Access</b>	Double-click on a rectangle setup under <b>Radiation</b> in the project tree.											
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;RectangleSetupName&gt;</td> <td>String</td> <td>Name of rectangle radiation setup to be edited.</td> </tr> <tr> <td>&lt;RectangleSetupParams&gt;</td> <td>Array</td> <td> <p>Structured array.</p> <pre>Array ("NAME:&lt;RectangleSetupName&gt;",       "UseCustomRadiationSurface:=", &lt;boolean&gt;,       "Length:=", "&lt;value&gt;&lt;unit&gt;",       "Width:=", "&lt;value&gt;&lt;unit&gt;",       "LengthSamples:=", &lt;value&gt;,       "WidthSamples:=", &lt;value&gt;,       "CoordSystem:=", &lt;string name of coordinate system&gt;)</pre> </td> </tr> </tbody> </table>	Name	Type	Description	<RectangleSetupName>	String	Name of rectangle radiation setup to be edited.	<RectangleSetupParams>	Array	<p>Structured array.</p> <pre>Array ("NAME:&lt;RectangleSetupName&gt;",       "UseCustomRadiationSurface:=", &lt;boolean&gt;,       "Length:=", "&lt;value&gt;&lt;unit&gt;",       "Width:=", "&lt;value&gt;&lt;unit&gt;",       "LengthSamples:=", &lt;value&gt;,       "WidthSamples:=", &lt;value&gt;,       "CoordSystem:=", &lt;string name of coordinate system&gt;)</pre>		
Name	Type	Description										
<RectangleSetupName>	String	Name of rectangle radiation setup to be edited.										
<RectangleSetupParams>	Array	<p>Structured array.</p> <pre>Array ("NAME:&lt;RectangleSetupName&gt;",       "UseCustomRadiationSurface:=", &lt;boolean&gt;,       "Length:=", "&lt;value&gt;&lt;unit&gt;",       "Width:=", "&lt;value&gt;&lt;unit&gt;",       "LengthSamples:=", &lt;value&gt;,       "WidthSamples:=", &lt;value&gt;,       "CoordSystem:=", &lt;string name of coordinate system&gt;)</pre>										
<b>Return Value</b>	None.											

<b>Python Syntax</b>	EditRectangleSetup(<RectangleSetupName>, <RectangleSetupParams>)
<b>Python Example</b>	oModule.EditRectangleSetup ("Rectangle1",

```
[  
    "NAME:Rectangle1",  
    "UseCustomRadiationSurface:=", False,  
    "Length:=" , "20meter",  
    "Width:=" , "20meter",  
    "LengthSamples:=" , 41,  
    "WidthSamples:=" , 41,  
    "CoordSystem:=" , "Global"  
)
```

<b>VB Syntax</b>	EditRectangleSetup <RectangleSetupName>, <RectangleSetupParams>
<b>VB Example</b>	<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.

<b>UI Access</b>	Double-click on a sphere setup under <b>Radiation</b> in the project tree.											
<b>Parameters</b>	<table border="1"> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> <tr> <td><code>&lt;SphereSetupName&gt;</code></td> <td>String</td> <td>Name of sphere radiation field setup to be edited.</td> </tr> <tr> <td><code>&lt;SphereSetupParams&gt;</code></td> <td>Array</td> <td> <p>Structured array.</p> <pre>Array ("NAME:&lt;SphereSetupName&gt;",       "UseCustomRadiationSurface:=", &lt;boolean&gt;,       "Radius:=", "&lt;value&gt;&lt;unit&gt;",       "ThetaStart:=", "&lt;value&gt;&lt;unit&gt;",       "ThetaStop:=", "&lt;value&gt;&lt;unit&gt;",       "ThetaStep:=", "&lt;value&gt;&lt;unit&gt;",       "PhiStart:=", "&lt;value&gt;&lt;unit&gt;",       "PhiStop:=", "&lt;value&gt;&lt;unit&gt;",       "PhiStep:=", "&lt;value&gt;&lt;unit&gt;",       "UseLocalCS:=", &lt;boolean&gt;)</pre> </td> </tr> </table>	Name	Type	Description	<code>&lt;SphereSetupName&gt;</code>	String	Name of sphere radiation field setup to be edited.	<code>&lt;SphereSetupParams&gt;</code>	Array	<p>Structured array.</p> <pre>Array ("NAME:&lt;SphereSetupName&gt;",       "UseCustomRadiationSurface:=", &lt;boolean&gt;,       "Radius:=", "&lt;value&gt;&lt;unit&gt;",       "ThetaStart:=", "&lt;value&gt;&lt;unit&gt;",       "ThetaStop:=", "&lt;value&gt;&lt;unit&gt;",       "ThetaStep:=", "&lt;value&gt;&lt;unit&gt;",       "PhiStart:=", "&lt;value&gt;&lt;unit&gt;",       "PhiStop:=", "&lt;value&gt;&lt;unit&gt;",       "PhiStep:=", "&lt;value&gt;&lt;unit&gt;",       "UseLocalCS:=", &lt;boolean&gt;)</pre>		
Name	Type	Description										
<code>&lt;SphereSetupName&gt;</code>	String	Name of sphere radiation field setup to be edited.										
<code>&lt;SphereSetupParams&gt;</code>	Array	<p>Structured array.</p> <pre>Array ("NAME:&lt;SphereSetupName&gt;",       "UseCustomRadiationSurface:=", &lt;boolean&gt;,       "Radius:=", "&lt;value&gt;&lt;unit&gt;",       "ThetaStart:=", "&lt;value&gt;&lt;unit&gt;",       "ThetaStop:=", "&lt;value&gt;&lt;unit&gt;",       "ThetaStep:=", "&lt;value&gt;&lt;unit&gt;",       "PhiStart:=", "&lt;value&gt;&lt;unit&gt;",       "PhiStop:=", "&lt;value&gt;&lt;unit&gt;",       "PhiStep:=", "&lt;value&gt;&lt;unit&gt;",       "UseLocalCS:=", &lt;boolean&gt;)</pre>										
<b>Return Value</b>	None.											

<b>Python Syntax</b>	<code>EditSphereSetup(&lt;SphereSetupName&gt;, &lt;SphereSetupParams&gt;)</code>
<b>Python Example</b>	<pre>oModule.EditSphereSetup("Sphere1", [     "NAME:Sphere2",     "UseCustomRadiationSurface:=", False,</pre>

```
"Radius:=" , "20meter",
"ThetaStart:=" , "0deg",
"ThetaStop:=" , "180deg",
"ThetaStep:=" , "2deg",
"PhiStart:=" , "-180deg",
"PhiStop:=" , "180deg",
"PhiStep:=" , "2deg",
"UseLocalCS:=" , False
])
```

<b>VB Syntax</b>	EditSphereSetup <SphereSetupName>, <SphereSetupParams>
<b>VB Example</b>	<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 <i>&lt;BoxParams&gt;</i>	Type Array	Description Structured array.  <pre>Array ("NAME:&lt;SetupName&gt;",       "UseCustomRadiationSurface:=", &lt;bool&gt;,       "CustomRadiationSurface:=", &lt;FaceListName&gt;,       "Length:=", &lt;string&gt;,       "Width:=", &lt;string&gt;,       "LengthSamples:=", &lt;int&gt;,       "WidthSamples:=", &lt;int&gt;,       "CoordSystem:=", &lt;CSname&gt;       "Height:=", &lt;string&gt;,       "HeightSamples:=", &lt;int&gt;)</pre> <p>For UseCustomRadiationSurface parameter, provide value if True. If False, radiation boundary/PML boundaries will be used as radiation surfaces.</p>
Return Value	None.		

### Python Syntax

```
InsertBoxSetup(<BoxParams>)
```

<b>Python Example</b>	<pre>oModule.InsertBoxSetup(     [         'NAME:MyBox',         'UseCustomRadiationSurface:=', False,         'Length:=', '20mm', 'Width:=', '20mm',         'LengthSamples:=', 21,         'WidthSamples:=', 21,         'CoordSystem:=', 'Global',         'Height:=', '20mm',         'HeightSamples:=', 21     ])</pre>
-----------------------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

<b>VB Syntax</b>	<b>InsertBoxSetup &lt;BoxParams&gt;</b>
<b>VB Example</b>	<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	<p>Structured array.</p> <pre>Array ("NAME:&lt;InfSphereName&gt;",       "UseCustomRadiationSurface:=", &lt;boolean&gt;,       "CSDefinition:=", &lt;string, "Theta-Phi"   "Az Over EI"         "EI Over AS"&gt;,       "Polarization:=", &lt;string, "Linear"   "Slant"&gt;,       "ThetaStart:=", "&lt;value&gt;&lt;unit&gt;",       "ThetaStop:=", "&lt;value&gt;&lt;unit&gt;",       "ThetaStep:=", "&lt;value&gt;&lt;unit&gt;",       "PhiStart:=", "&lt;value&gt;&lt;unit&gt;",       "PhiStop:=", "&lt;value&gt;&lt;unit&gt;",       "PhiStep:=", "&lt;value&gt;&lt;unit&gt;",       "UseLocalCS:=", boolean)</pre>
Return Value	None.		

<b>Python Syntax</b>	InsertFarFieldSphereSetup(<SphereParams>)
<b>Python Example</b>	<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>

<b>VB Syntax</b>	InsertFarFieldSphereSetup<SphereParams>
<b>VB Example</b>	oModule.InsertFarFieldSphereSetup _

```
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 <b>Radiation</b> in the project tree, select <b>Insert Far Field Setup &gt; Infinite Sphere...</b>		
Parameters	Name <i>&lt;InfSphereParams&gt;</i>	Type Array	Description Structured array.  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>"PhiStart:=", "&lt;value&gt;&lt;unit&gt;", "PhiStop:=", "&lt;value&gt;&lt;unit&gt;", "PhiStep:=", "&lt;value&gt;&lt;unit&gt;", "UseLocalCS:=", boolean)</pre>
<b>Return Value</b>	None.		

Python Syntax	InsertInfiniteSphereSetup(<InfSphereParams>)
<b>Python Example</b>	<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>

```
"UseLocalCS:=" , False
])
```

<b>VB Syntax</b>	<code>InsertInfiniteSphereSetup &lt;InfSphereParams&gt;</code>
<b>VB Example</b>	<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.

<b>UI Access</b>	Right-click on <b>Radiation</b> in the project tree, select <b>Insert Near Field Setup &gt; Line...</b>								
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><code>&lt;LineParams&gt;</code></td> <td>Array</td> <td> <p>Structured array.</p> <pre> Array("NAME:&lt;LineName&gt;", "UseCustomRadiationSurface:=", &lt;boolean&gt;, </pre> </td> </tr> </tbody> </table>	Name	Type	Description	<code>&lt;LineParams&gt;</code>	Array	<p>Structured array.</p> <pre> Array("NAME:&lt;LineName&gt;", "UseCustomRadiationSurface:=", &lt;boolean&gt;, </pre>		
Name	Type	Description							
<code>&lt;LineParams&gt;</code>	Array	<p>Structured array.</p> <pre> Array("NAME:&lt;LineName&gt;", "UseCustomRadiationSurface:=", &lt;boolean&gt;, </pre>							

		"Line:=", <string, name of specified line>, "NumPts:=", "<integer, number of points>")
<b>Return Value</b>	None.	

<b>Python Syntax</b>	InsertLineSetup(<LineParams>)
<b>Python Example</b>	<pre>oModule.InsertLineSetup( [     "NAME:Line1",     "UseCustomRadiationSurface:=", False,     "Line:=" , "Polyline1",     "NumPts:=" , "1000" ])</pre>

<b>VB Syntax</b>	InsertLineSetup <LineParams>
<b>VB Example</b>	<pre>oModule.InsertLineSetup _ Array("NAME:Line1", _ "UseCustomRadiationSurface:=", false, _ "Line:=", "Polyline1", "NumPts:=", "1000")</pre>

## InsertNearFieldBoxSetup

Inserts a near field box radiation field setup.

<b>UI Access</b>	N/A								
<b>Parameters</b>	<table border="1"> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> <tr> <td><code>&lt;BoxSetupParams&gt;</code></td> <td>Array</td> <td> <p>Structured array.</p> <pre>Array ("NAME:&lt;BoxSetupName&gt;",       "UseCustomRadiationSurface:=", &lt;boolean&gt;,       "Length:=", "&lt;value&gt;&lt;unit&gt;",       "Width:=", "&lt;value&gt;&lt;unit&gt;",       "LengthSamples:=", &lt;value&gt;,       "WidthSamples:=", &lt;value&gt;,       "CoordSystem:=", &lt;string name of coordinate system&gt;,       "Height:=", "&lt;value&gt;&lt;unit&gt;",       "HeightSamples:=", &lt;value&gt;)</pre> </td> </tr> </table>	Name	Type	Description	<code>&lt;BoxSetupParams&gt;</code>	Array	<p>Structured array.</p> <pre>Array ("NAME:&lt;BoxSetupName&gt;",       "UseCustomRadiationSurface:=", &lt;boolean&gt;,       "Length:=", "&lt;value&gt;&lt;unit&gt;",       "Width:=", "&lt;value&gt;&lt;unit&gt;",       "LengthSamples:=", &lt;value&gt;,       "WidthSamples:=", &lt;value&gt;,       "CoordSystem:=", &lt;string name of coordinate system&gt;,       "Height:=", "&lt;value&gt;&lt;unit&gt;",       "HeightSamples:=", &lt;value&gt;)</pre>		
Name	Type	Description							
<code>&lt;BoxSetupParams&gt;</code>	Array	<p>Structured array.</p> <pre>Array ("NAME:&lt;BoxSetupName&gt;",       "UseCustomRadiationSurface:=", &lt;boolean&gt;,       "Length:=", "&lt;value&gt;&lt;unit&gt;",       "Width:=", "&lt;value&gt;&lt;unit&gt;",       "LengthSamples:=", &lt;value&gt;,       "WidthSamples:=", &lt;value&gt;,       "CoordSystem:=", &lt;string name of coordinate system&gt;,       "Height:=", "&lt;value&gt;&lt;unit&gt;",       "HeightSamples:=", &lt;value&gt;)</pre>							
<b>Return Value</b>	None.								

<b>Python Syntax</b>	<code>InsertNearFieldBoxSetup(&lt;BoxSetupParams&gt;)</code>
<b>Python Example</b>	<pre>oModule.InsertNearFieldBoxSetup (     [         "NAME:Box1",</pre>

```
"UseCustomRadiationSurface:=", False,  
"Length:=" , "20meter",  
"Width:=" , "20meter",  
"LengthSamples:=" , 21,  
"WidthSamples:=" , 21,  
"CoordSystem:=" , "Global",  
"Height:=" , "20meter",  
"HeightSamples:=" , 21  
])
```

VB Syntax	InsertNearFieldBoxSetup <BoxSetupParams>
<b>VB Example</b>	<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.

<b>UI Access</b>	N/A								
<b>Parameters</b>	<table border="1"> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> <tr> <td>&lt;LineParams&gt;</td> <td>Array</td> <td> <p>Structured array.</p> <pre>Array ("NAME:&lt;LineName&gt;",       "UseCustomRadiationSurface:=", &lt;boolean&gt;,       "Line:=", &lt;string, name of specified line&gt;,       "NumPts:=", "&lt;integer, number of points&gt;")</pre> </td> </tr> </table>	Name	Type	Description	<LineParams>	Array	<p>Structured array.</p> <pre>Array ("NAME:&lt;LineName&gt;",       "UseCustomRadiationSurface:=", &lt;boolean&gt;,       "Line:=", &lt;string, name of specified line&gt;,       "NumPts:=", "&lt;integer, number of points&gt;")</pre>		
Name	Type	Description							
<LineParams>	Array	<p>Structured array.</p> <pre>Array ("NAME:&lt;LineName&gt;",       "UseCustomRadiationSurface:=", &lt;boolean&gt;,       "Line:=", &lt;string, name of specified line&gt;,       "NumPts:=", "&lt;integer, number of points&gt;")</pre>							
<b>Return Value</b>	None.								

<b>Python Syntax</b>	InsertNearFieldLineSetup(<LineParams>)
<b>Python Example</b>	<pre>oModule.InsertNearFieldLineSetup (     [         "NAME:Line1",         "UseCustomRadiationSurface:=", False,         "Line:=" , "Polyline1",         "NumPts:=" , "1000"     ])</pre>

<b>VB Syntax</b>	<code>InsertNearFieldLineSetup&lt;LineParams&gt;</code>
<b>VB Example</b>	<pre> oModule.InsertNearFieldLineSetup _     Array("NAME:Line1", _         "UseCustomRadiationSurface:=", false, _         "Line:=", "Polyline1", "NumPts:=", "1000")     </pre>

## InsertNearFieldRectangleSetup

Inserts a near field rectangle radiation setup.

<b>UI Access</b>	N/A								
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><code>&lt;RectangleSetupParams&gt;</code></td> <td>Array</td> <td> <p>Structured array.</p> <pre> Array ("NAME:&lt;RectangleSetupName&gt;",     "UseCustomRadiationSurface:=", &lt;boolean&gt;,     "Length:=", "&lt;value&gt;&lt;unit&gt;",     "Width:=", "&lt;value&gt;&lt;unit&gt;",     "LengthSamples:=", &lt;value&gt;,     "WidthSamples:=", &lt;value&gt;,     "CoordSystem:=", &lt;string name of coordinate system&gt; )     </pre> </td></tr> </tbody> </table>			Name	Type	Description	<code>&lt;RectangleSetupParams&gt;</code>	Array	<p>Structured array.</p> <pre> Array ("NAME:&lt;RectangleSetupName&gt;",     "UseCustomRadiationSurface:=", &lt;boolean&gt;,     "Length:=", "&lt;value&gt;&lt;unit&gt;",     "Width:=", "&lt;value&gt;&lt;unit&gt;",     "LengthSamples:=", &lt;value&gt;,     "WidthSamples:=", &lt;value&gt;,     "CoordSystem:=", &lt;string name of coordinate system&gt; )     </pre>
Name	Type	Description							
<code>&lt;RectangleSetupParams&gt;</code>	Array	<p>Structured array.</p> <pre> Array ("NAME:&lt;RectangleSetupName&gt;",     "UseCustomRadiationSurface:=", &lt;boolean&gt;,     "Length:=", "&lt;value&gt;&lt;unit&gt;",     "Width:=", "&lt;value&gt;&lt;unit&gt;",     "LengthSamples:=", &lt;value&gt;,     "WidthSamples:=", &lt;value&gt;,     "CoordSystem:=", &lt;string name of coordinate system&gt; )     </pre>							
<b>Return Value</b>	None.								

<b>Python Syntax</b>	InsertNearFieldRectangleSetup(<RectangleSetupParams>)
<b>Python Example</b>	<pre> oModule.InsertNearFieldRectangleSetup( [     "NAME:Rectangle1",     "UseCustomRadiationSurface:=", False,     "Length:=", "20meter",     "Width:=", "20meter",     "LengthSamples:=", 41,     "WidthSamples:=", 41,     "CoordSystem:=", "Global" ]) </pre>

<b>VB Syntax</b>	InsertNearFieldRectangleSetup <RectangleSetupParams>
<b>VB Example</b>	<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 <i>&lt;SphereSetupParams&gt;</i>	Type Array	Description 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( <i>&lt;SphereSetupParams&gt;</i> )
Python Example	oModule.InsertNearFieldSphereSetup (

```
[  

    "NAME:Sphere2",  

    "UseCustomRadiationSurface:=", False,  

    "Radius:=" , "20meter",  

    "ThetaStart:=" , "0deg",  

    "ThetaStop:=" , "180deg",  

    "ThetaStep:=" , "2deg",  

    "PhiStart:=" , "-180deg",  

    "PhiStop:=" , "180deg",  

    "PhiStep:=" , "2deg",  

    "UseLocalCS:=" , False  

])
```

<b>VB Syntax</b>	InsertNearFieldSphereSetup <SphereSetupParams>
<b>VB Example</b>	<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 <b>Radiation</b> in the project tree ,select <b>Insert Near Field Setup &gt; Rectangle...</b>		
Parameters	Name <code>&lt;RectangleSetupParams&gt;</code>	Type Array	Description Structured array.  <code>Array ("NAME:&lt;RectangleSetupName&gt;,"       "UseCustomRadiationSurface:=", &lt;boolean&gt;,       "Length:=", "&lt;value&gt;&lt;unit&gt;,"       "Width:=", "&lt;value&gt;&lt;unit&gt;,"       "LengthSamples:=", &lt;value&gt;,       "WidthSamples:=", &lt;value&gt;,       "CoordSystem:=", &lt;string name of coordinate sys-                 tem&gt;)</code>
Return Value	None.		

Python Syntax	<code>InsertRectangleSetup(&lt;RectangleSetupParams&gt;)</code>
Python Example	<pre>oModule.InsertRectangleSetup(     [         "NAME:Rectangle1",</pre>

```

    "UseCustomRadiationSurface:=", False,
    "Length:="                  , "20meter",
    "Width:="                   , "20meter",
    "LengthSamples:="           , 41,
    "WidthSamples:="            , 41,
    "CoordSystem:="             , "Global"
  ]
)

```

<b>VB Syntax</b>	InsertRectangleSetup <RectangleSetupParams>
<b>VB Example</b>	<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.

<b>UI Access</b>	Right-click on <b>Radiation</b> in the project tree ,select <b>Insert Near Field Setup &gt; Sphere...</b>		
<b>Parameters</b>	Name <SphereSetupParams>	Type Array	Description Structured array.

		<pre>Array("NAME:&lt;SphereSetupName&gt;",       "UseCustomRadiationSurface:=", &lt;boolean&gt;,       "Radius:=", "&lt;value&gt;&lt;unit&gt;",       "ThetaStart:=", "&lt;value&gt;&lt;unit&gt;",       "ThetaStop:=", "&lt;value&gt;&lt;unit&gt;",       "ThetaStep:=", "&lt;value&gt;&lt;unit&gt;",       "PhiStart:=", "&lt;value&gt;&lt;unit&gt;",       "PhiStop:=", "&lt;value&gt;&lt;unit&gt;",       "PhiStep:=", "&lt;value&gt;&lt;unit&gt;",       "UseLocalCS:=", &lt;boolean&gt;)</pre>
<b>Return Value</b>	None.	

<b>Python Syntax</b>	<code>InsertSphereSetup(&lt;SphereSetupParams&gt;)</code>
<b>Python Example</b>	<pre>oModule.InsertSphereSetup( [     "NAME:Sphere2",     "UseCustomRadiationSurface:=", False,     "Radius:", "20meter",     "ThetaStart:", "0deg",</pre>

```

    "ThetaStop:" , "180deg",
    "ThetaStep:" , "2deg",
    "PhiStart:" , "-180deg",
    "PhiStop:" , "180deg",
    "PhiStep:" , "2deg",
    "UseLocalCS:" , False
)

```

<b>VB Syntax</b>	<code>InsertSphereSetup &lt;SphereSetupParams&gt;</code>
<b>VB Example</b>	<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 <b>Radiation</b> in the project tree, select <b>Antenna Array Setup...</b>		
<b>Parameters</b>	Name <code>&lt;AntennaArrayParams&gt;</code>	Type Array	Description Structured array.  <code>Array("NAME:ArraySetupInfo", "UseOption:=", &lt;string "No Array"   "Parametric Array"   "Custom Array"&gt;, &lt;ParametricArrayParams&gt;   &lt;CustomArrayParams&gt;)</code>
	<code>&lt; ParametricArrayParams&gt;</code>	Array	Structured array.  <code>Array("NAME:ParametricArray", "DesignFrequency:=", "&lt;value&gt;&lt;unit&gt;", "LayoutType:=", &lt;integer&gt;, "CenterCellX:=", "&lt;value&gt;&lt;unit&gt;", "CenterCellY:=", "&lt;value&gt;&lt;unit&gt;", "CenterCellZ:=", "&lt;value&gt;&lt;unit&gt;", "SpecifyDesignInWavelength:=", &lt;boolean&gt;, "WidthSpacing:=", "&lt;value&gt;&lt;unit&gt;", "WidthSpacingInWavelength:=", "&lt;value&gt;", "Width:=", "&lt;value&gt;&lt;unit&gt;",</code>

```
"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>"UDirnPhaseShift:=", "&lt;value&gt;&lt;unit&gt;", "VDirnPhaseShift:=", "&lt;value&gt;&lt;unit&gt;")</pre>
<CustomArrayParams>	Array	<p>Structured array.</p> <pre>Array("NAME:CustomArray",       "NumCells:=", &lt;int&gt;,       &lt;CellsParamsArray&gt;)</pre>
<CellsParamsArray>	Array	<p>Structured array.</p> <pre>Array("NAME:Cell",       &lt;CellParams&gt;, &lt;CellParams&gt;, ...)</pre>
<CellParams>	Array	<p>Structured array.</p> <pre>Array("Name:&lt;CellName&gt;",       "XCoord:=", &lt;double&gt;,       "YCoord:=", &lt;double&gt;,       "ZCoord:=", &lt;double&gt;,       "Amplitude:=", &lt;double&gt;,       "Phase:=", &lt;double&gt;)</pre>
<b>Return Value</b>	None.	

<b>Python Syntax</b>	EditAntennaArraySetup(<AntennaArrayParams>)
<b>Python Example</b>	oModule.EditAntennaArraySetup(

```
[  
  "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,
```

```
"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"]
])
```

<b>VB Syntax</b>	EditAntennaArraySetup <AntennaArrayParams>
<b>VB Example</b>	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 <b>Antenna Parameters</b> dialog box.		
Parameters	Name <i>&lt;ExportElementPatternFileParams&gt;</i>	Type Array	Description 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.		

<b>Python Syntax</b>	ExportElementPatternToFile(<ExportElementPatternFileParams>)
<b>Python Example</b>	<pre> oModule.ExportElementPatternToFile( [     "ExportFileName:=",      "C:\Documents\\exportelement2.ffd",     "SetupName:=",           "3D",     "IntrinsicVariationKey:=", "Freq='28GHz'",     "DesignVariationKey:=",   "antipad_rad='0.1499999999999999mm'",     "SolutionName:=",         "Setup1 : LastAdaptive" ]) </pre>

<b>VB Syntax</b>	ExportElementPatternToFile <ExportElementPatternFileParams>
<b>VB Example</b>	<pre> oModule.ExportElementPatternToFile Array("ExportFileName:=", "C:\Documents\exportelement1.ffd",                                        _                                        "SetupName:=", "3D", _                                        "IntrinsicVariationKey:=", "Freq=" &amp; Chr(39) &amp; "28GHz" &amp; Chr(39) &amp; "", _                                        "DesignVariationKey:=", "antipad_rad=" &amp; Chr(39) &amp; "0.1499999999999999mm" &amp; Chr(39) &amp;                                        ", _                                        "SolutionName:=", "Setup1 : LastAdaptive") </pre>

## ExportFieldsToFile

Exports fields to a file.

<b>UI Access</b>	From the <b>Antenna Parameters</b> dialog box, <b>Export Fields</b> to .ffd file.						
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><code>&lt;ExportParams&gt;</code></td> <td>Array</td> <td> <p>Structured array.</p> <pre>Array("ExportFileName:=", &lt;string file name include path&gt;,       "SetupName:=", &lt;string name of the Radiation Far Field or Near Field setup&gt;,       "IntrinsicVariationKey:=", &lt;string from the Antenna Parameters dialog.&gt;,       "DesignVariationKey:=", &lt;string&gt;,       "SolutionName:=", &lt;string name of solution to use&gt;,       "Quantity:=", &lt;string antenna Parameter name. If empty, all parameters are exported.&gt;)</pre> </td> </tr> </tbody> </table>	Name	Type	Description	<code>&lt;ExportParams&gt;</code>	Array	<p>Structured array.</p> <pre>Array("ExportFileName:=", &lt;string file name include path&gt;,       "SetupName:=", &lt;string name of the Radiation Far Field or Near Field setup&gt;,       "IntrinsicVariationKey:=", &lt;string from the Antenna Parameters dialog.&gt;,       "DesignVariationKey:=", &lt;string&gt;,       "SolutionName:=", &lt;string name of solution to use&gt;,       "Quantity:=", &lt;string antenna Parameter name. If empty, all parameters are exported.&gt;)</pre>
Name	Type	Description					
<code>&lt;ExportParams&gt;</code>	Array	<p>Structured array.</p> <pre>Array("ExportFileName:=", &lt;string file name include path&gt;,       "SetupName:=", &lt;string name of the Radiation Far Field or Near Field setup&gt;,       "IntrinsicVariationKey:=", &lt;string from the Antenna Parameters dialog.&gt;,       "DesignVariationKey:=", &lt;string&gt;,       "SolutionName:=", &lt;string name of solution to use&gt;,       "Quantity:=", &lt;string antenna Parameter name. If empty, all parameters are exported.&gt;)</pre>					
<b>Return Value</b>	None.						

<b>Python Syntax</b>	<code>ExportFieldsToFile (&lt;ExportParams&gt;)</code>
<b>Python Example</b>	<pre>oModule.ExportFieldsToFile (     [         "ExportFileName:=", "C:/exportfields.ffd",         "SetupName:=", "Infinite Sphere1",         "IntrinsicVariationKey:=", "Freq='4.5GHz'",     ] )</pre>

```

    "DesignVariationKey:=", "",  

    "SolutionName:=", "4500MHz : LastAdaptive",  

    "Quantity:=", ""  

])

```

<b>VB Syntax</b>	ExportFieldsToFile < <i>ExportParams</i> >
<b>VB Example</b>	<pre> oModule.ExportFieldsToFile _      Array("ExportFileName:=", "C:/exportfields.ffd", _          "SetupName:=", "Infinite Sphere1", _          "IntrinsicVariationKey:=", "Freq=" &amp; Chr(39) &amp; "6GHz" &amp; Chr(39) &amp; "", _          "DesignVariationKey:=", "",          "SolutionName:=", "Setup1 : LastAdaptive", _          "SourceGroupName:=", "Source Group1", _          "Quantity:=", "")</pre>

## ExportParametersToFile

Exports antenna parameters to a file.

<b>UI Access</b>	Click <b>Export</b> in the <b>Antenna Parameters</b> dialog.		
<b>Parameters</b>	Name < <i>DefParams</i> >	Type Array	Description Structured array. Array(

			<pre>"ExportFileName:=", &lt;string file name include path&gt;, "SetupName:=", &lt;string name of setup to export&gt;, "IntrinsicVariationKey:=", &lt;string&gt;, "DesignVariationKey:=" , &lt;string&gt;, "SolutionName:=", &lt;string name of solution to export&gt;, "Quantity:=" , &lt;string name of quantity, empty to export all&gt;)</pre>
<b>Return Value</b>	None.		

<b>Python Syntax</b>	ExportParametersToFile(< <i>DefParams</i> >)
<b>Python Example</b>	<pre>oModule.ExportParametersToFile( [     "ExportFileName:=", "C:/exportparams.txt",     "SetupName:=", "Infinite Sphere1",     "IntrinsicVariationKey:=", "Freq=\'4.5GHz\'",     "DesignVariationKey:=", "",     "SolutionName:=", "4500MHz : LastAdaptive",     "Quantity:=", "" ])</pre>

<b>VB Syntax</b>	ExportParametersToFile(<DefParams>
<b>VB Example</b>	<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.

<b>UI Access</b>	N/A								
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;ExportParams&gt;</td> <td>Array</td> <td> <p>Structured array.</p> <p>Array("ExportFileName:=", &lt;string file name include path&gt;,</p> <p>"SetupName:=", &lt;string name of the Radiation Far Field or Near Field setup&gt;,</p> <p>"IntrinsicVariationKey:=", &lt;string from the Antenna Parameters dialog.&gt;,</p> <p>"DesignVariationKey:=", &lt;string&gt;,</p> <p>"SolutionName:=", &lt;string name of solution to use&gt;,</p> <p>"Quantity:=", &lt;string antenna Parameter name. If</p> </td></tr> </tbody> </table>	Name	Type	Description	<ExportParams>	Array	<p>Structured array.</p> <p>Array("ExportFileName:=", &lt;string file name include path&gt;,</p> <p>"SetupName:=", &lt;string name of the Radiation Far Field or Near Field setup&gt;,</p> <p>"IntrinsicVariationKey:=", &lt;string from the Antenna Parameters dialog.&gt;,</p> <p>"DesignVariationKey:=", &lt;string&gt;,</p> <p>"SolutionName:=", &lt;string name of solution to use&gt;,</p> <p>"Quantity:=", &lt;string antenna Parameter name. If</p>		
Name	Type	Description							
<ExportParams>	Array	<p>Structured array.</p> <p>Array("ExportFileName:=", &lt;string file name include path&gt;,</p> <p>"SetupName:=", &lt;string name of the Radiation Far Field or Near Field setup&gt;,</p> <p>"IntrinsicVariationKey:=", &lt;string from the Antenna Parameters dialog.&gt;,</p> <p>"DesignVariationKey:=", &lt;string&gt;,</p> <p>"SolutionName:=", &lt;string name of solution to use&gt;,</p> <p>"Quantity:=", &lt;string antenna Parameter name. If</p>							

			empty, all parameters are exported.>)
<b>Return Value</b>	None.		

<b>Python Syntax</b>	ExportRadiationFieldsToFile (< <i>ExportParams</i> >)
<b>Python Example</b>	<pre> oModule.ExportRadiationFieldsToFile( [     "ExportFileName:=", "C:/exportfields.ffd",     "SetupName:=", "Infinite Sphere1",     "IntrinsicVariationKey:=", "Freq='4.5GHz'",     "DesignVariationKey:=", "",     "SolutionName:=", "4500MHz : LastAdaptive",     "Quantity:=", "" ]) </pre>

<b>VB Syntax</b>	ExportRadiationFieldsToFile < <i>ExportParams</i> >
<b>VB Example</b>	<pre> oModule.ExportRadiationFieldsToFile _ Array("ExportFileName:=", "C:/exportfields.ffd", _       "SetupName:=", "Infinite Sphere1", _ </pre>

```

    "IntrinsicVariationKey:=", "Freq=" & Chr(39) & "6GHz" & Chr(39) & "", _
    "DesignVariationKey:=", "",_
    "SolutionName:=", "Setup1 : LastAdaptive",_
    "SourceGroupName:=", "Source Group1",_
    "Quantity:=", "")

```

## 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 <i>&lt;ExportToFileParams&gt;</i>	Type Array	Description Structured array.  Array("ExportFileName:=", <string file name include path>  "SetupName:=", <string>  "IntrinsicVariationKey:=", <string>,  "DesignVariationKey:=", <string>,  "SolutionName:=", <string>)
Return Value	None.		

Python Syntax	ExportRadiationParametersToFile( <i>&lt;ExportToFileParams&gt;</i> )
Python Example	oModule.ExportRadiationParametersToFile (

```
[  
    "ExportFileName:=", "C:\projects\exportantparams.txt",  
    "SetupName:=", "Infinite Sphere1",  
    "IntrinsicVariationKey:=", "Freq='10GHz'",  
    "DesignVariationKey:=", "",  
    "SolutionName:=", "LastAdaptive"  
)
```

<b>VB Syntax</b>	ExportRadiationParametersToFile < <i>ExportToFileParams</i> >
<b>VB Example</b>	<pre>oModule.ExportRadiationParametersToFile _     Array("ExportFileName:=", _         "C:\projects\exportantparams.txt", _         "SetupName:=", "Infinite Sphere1", _         "IntrinsicVariationKey:=", "Freq='10GHz'", _         "DesignVariationKey:=", "",         "SolutionName:=", "LastAdaptive")</pre>

# 19 - 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 <b>Radiation</b> and in the right-click menu, click <b>Insert Far Field Setup&gt;Infinite Sphere</b> .		
Parameters	Name	Type	Description
	<Name>	String	A user-defined name for the infinite sphere.

	<code>&lt;VersionID&gt;</code>	Integer	Optional. Set to <b>0</b> .
	<code>&lt;ID&gt;</code>	Integer	Optional. Identifier that is created differently for every setup.
	<code>&lt;StartTheta &gt;</code>	String	Optional. Starting point for theta in the spherical coordinate system for the sphere being defined. The default value is " <b>0deg</b> ". Use the form " <code>&lt;num&gt;deg</code> ".
	<code>&lt;StartPhi&gt;</code>	String	Optional. Starting point for phi in the spherical coordinate system for the sphere being defined. The default value is " <b>-180deg</b> ". Use the form " <code>&lt;num&gt;deg</code> ".
	<code>&lt;StopTheta&gt;</code>	String	Optional. Stopping point for theta in the spherical coordinate system for the sphere being defined. The default value is " <b>180deg</b> ". Use the form " <code>&lt;num&gt;deg</code> ".
	<code>&lt;StopPhi&gt;</code>	String	Optional. Stopping point for phi in the spherical coordinate system for the sphere being defined. The default value is " <b>1800deg</b> ". Use the form " <code>&lt;num&gt;deg</code> ".
	<code>&lt;StepTheta&gt;</code>	String	Optional. Step size between the start and the stop theta values at which the far field is calculated. The default value is " <b>2deg</b> ". Use the form " <code>&lt;num&gt;deg</code> ".
	<code>&lt;StepPhi&gt;</code>	String	Optional. Step size between the start and the stop phi values at which the far field is calculated. The default value is " <b>2deg</b> ". Use the form " <code>&lt;num&gt;deg</code> ".
	<code>&lt;CS&gt;</code>	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 <b>-1</b> for a global coordinate system. The default value is <b>-1</b> .
	<code>&lt;UseLocalCS&gt;</code>	Boolean	Optional. <b>True</b> to select a local coordinate system, and <b>False</b> to use the global coordinate system. The default value is <b>False</b> .
<b>Return Value</b>	None.		

<b>Python Syntax</b>	<pre>oModule.Add( [     "NAME:&lt;Name&gt;",     "Type:=", "InfiniteSphereFarField",     "VersionID:=", 0,     "ID:=", &lt;ID&gt;,     "StartTheta:=", "&lt;StartTheta&gt;deg",     "StartPhi:=", "&lt;StartPhi&gt;deg",     "StopTheta:=", "&lt;StopTheta&gt;deg",     "StopPhi:=", "&lt;StopPhi&gt;deg",     "StepTheta:=", "&lt;StepTheta&gt;deg",     "StepPhi:=", "&lt;StepPhi&gt;deg",     "CS:=", &lt;CS&gt;,     "UseLocalCS:=", &lt;UseLocalCS&gt; ])</pre>
<b>Python Example</b>	<pre>oDesign = oDesign.GetModule("RadiationSetupMgr")  oModule.Add( [     "NAME:Infinite Sphere 1",     "Type:=", "InfiniteSphereFarField",     "VersionID:=", 0,</pre>

```

    "ID:=", 5,
    "StartTheta:=", "0deg",
    "StartPhi:=", "-180deg",
    "StopTheta:=", "180deg",
    "StopPhi:=", "180deg",
    "StepTheta:=", "2deg",
    "StepPhi:=", "2deg",
    "CS:=", 1,
    "UseLocalCS:=", False
)

```

<b>VB Syntax</b>	<pre> oModule.Add Array("NAME:&lt;Name&gt;", "Type:=", "InfiniteSphereFarField", "VersionID:=", 0, "ID:=", &lt;/ID&gt;, "StartTheta:=", "&lt;StartTheta&gt;deg", "StartPhi:=", "&lt;StartPhi&gt;deg", "StopTheta:=", "&lt;StopPhi&gt;deg", "StopPhi:=", "&lt;StopPhi&gt;deg", "StepTheta:=", "&lt;StepTheta&gt;deg", "StepPhi:=", "&lt;StepPhi&gt;deg", "CS:=", &lt;CS&gt;, "UseLocalCS:=", &lt;UseLocalCS&gt;) </pre>
<b>VB Example</b>	<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 > <b>Radiation</b> , 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 <b>InfiniteSphereFarField</b> .
	<VersionID>	Integer	Optional. Set to <b>0</b> .
	<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> <b>deg</b> ".
	<StartPhi>	String	Optional. Starting point for phi in the spherical coordinate system for the sphere being defined. Use the form "<num> <b>deg</b> ".
	<StopTheta>	String	Optional. Stopping point for theta in the spherical coordinate system for the sphere being defined. Use the form "<num> <b>deg</b> ".
	<StopPhi>	String	Optional. Stopping point for phi in the spherical coordinate system for the sphere being defined. Use the form "<num> <b>deg</b> ".
	<StepTheta>	String	Optional. Step size between the start and the stop theta values at which the far field is calculated. Use the form "<num> <b>deg</b> ".
	<StepPhi>	String	Optional. Step size between the start and the stop phi values at which the far field is calculated. Use the form "<num> <b>deg</b> ".
	<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 <b>-1</b> for a global coordinate system.
	<UseLocalCS>	Boolean	Optional. <b>True</b> to select a local coordinate system, and <b>False</b> to use the global coordinate system.

<b>Return Value</b>	None.
---------------------	-------

<b>Python Syntax</b>	<pre>oModule.Edit(     [         "NAME:&lt;Name&gt;",         "Type:=", "InfiniteSphereFarField",         "VersionID:=", 0,         "ID:=", &lt;ID&gt;,         "StartTheta:=", "&lt;StartTheta&gt;deg",         "StartPhi:=", "&lt;StartPhi&gt;deg",         "StopTheta:=", "&lt;StopTheta&gt;deg",         "StopPhi:=", "&lt;StopPhi&gt;deg",         "StepTheta:=", "&lt;StepTheta&gt;deg",         "StepPhi:=", "&lt;StepPhi&gt;deg",         "CS:=", &lt;CS&gt;,         "UseLocalCS:=", &lt;UseLocalCS&gt;<br ])<="" pre=""/></pre>
<b>Python Example</b>	<pre>oDesign = oDesign.GetModule("RadiationSetupMgr") oModule.Edit (</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  

])
```

<b>VB Syntax</b>	<code>oModule.Edit Array("NAME:&lt;Name&gt;", "Type:=", "InfiniteSphereFarField", "VersionID:=", 0, "ID:=", &lt;ID&gt;, "StartTheta:=", "&lt;StartTheta&gt;deg", "StartPhi:=", "&lt;StartPhi&gt;deg", "StopTheta:=", "&lt;StopPhi&gt;deg", "StopPhi:=", "&lt;StopPhi&gt;deg", "StepTheta:=", "&lt;StepTheta&gt;deg", "StepPhi:=", "&lt;StepPhi&gt;deg", "CS:=", &lt;CS&gt;, "UseLocalCS:=", &lt;UseLocalCS&gt;)</code>
<b>VB Example</b>	<code>Set oModule = oDesign.GetModule("RadiationSetupMgr") oModule.Edit Array("NAME:Infinite Sphere 1", "Type:=", "InfiniteSphereFarField", "VersionID:=", _ 0, "ID:=", 0, "StartTheta:=", "0deg", "StartPhi:=", "-180deg", "StopTheta:=", _</code>

```
"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 <b>Radiation</b> and in the right-click menu, click <b>Insert Near Field Setup&gt;Line</b> .																					
Parameters	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>&lt;Name&gt;</td><td>String</td><td>A user-defined name for the near field line.</td></tr><tr><td>&lt;Type&gt;</td><td>String</td><td>Type of radiation setup. It should be <b>NearFieldLine</b>.</td></tr><tr><td>&lt;VersionID&gt;</td><td>Integer</td><td>Optional. Set to <b>0</b>.</td></tr><tr><td>&lt;ID&gt;</td><td>Integer</td><td>Optional. Identifier that is created differently for every setup.</td></tr><tr><td>&lt;NFLineID &gt;</td><td>Integer</td><td>Optional. Index of the 3D line in the design area across which the near fields are calculated.</td></tr><tr><td>&lt;NumSamples&gt;</td><td>String</td><td>Optional. Number of points (samples) needed to calculate the near fields across the line. Use the form "&lt;num&gt;". The default value is <b>"1000"</b>.</td></tr></tbody></table>	Name	Type	Description	<Name>	String	A user-defined name for the near field line.	<Type>	String	Type of radiation setup. It should be <b>NearFieldLine</b> .	<VersionID>	Integer	Optional. Set to <b>0</b> .	<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>". The default value is <b>"1000"</b> .
Name	Type	Description																				
<Name>	String	A user-defined name for the near field line.																				
<Type>	String	Type of radiation setup. It should be <b>NearFieldLine</b> .																				
<VersionID>	Integer	Optional. Set to <b>0</b> .																				
<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>". The default value is <b>"1000"</b> .																				
Return Value	None.																					

Python Syntax	<pre>oModule.Add(     [         "NAME:&lt;Name&gt;","</pre>
---------------	---------------------------------------------------------------------

	<pre>         "Type:=", "NearFieldLine",         "VersionID:=", 0,         "ID:=", &lt;ID&gt;,         "NFLineID:=", "&lt;NFLineID&gt;",         "NumSamples:=", "&lt;NumSamples&gt;"     ]) </pre>
<b>Python Example</b>	<pre> oDesign = oDesign.GetModule("RadiationSetupMgr") oModule.Add( [     "NAME:Line 1",     "Type:=", "NearFieldLine",     "VersionID:=", 0,     "ID:=", 7,     "NFLineID:=", 312,     "NumSamples:=", "1000" ]) </pre>

<b>VB Syntax</b>	<pre> oModule.Add Array("NAME:&lt;Name&gt;", "Type:=", "NearFieldLine", "VersionID:=", 0, "ID:=", &lt;ID&gt;, "NFLineID:=", "&lt;NFLineID&gt;","NumSamples:=", " &lt;NumSamples&gt;") </pre>
<b>VB Example</b>	<pre> Set oModule = oDesign.GetModule("RadiationSetupMgr") oModule.Add Array("NAME:Line 1", "Type:=", "NearFieldLine", "VersionID:=", 0, "ID:=", </pre>

	<pre>— 4, "NFLLineID:=", 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.

<b>UI Access</b>	In the Project Manager under the design name > <b>Radiation</b> , double click on the name of the line.		
<b>Parameters</b>	Name	Type	Description
	<Name>	String	A user-defined name for the near field line.
	<Type>	String	Type of radiation setup. It should be <b>NearFieldLine</b> .
	<VersionID>	Integer	Optional. Set to <b>0</b> .
	<ID>	Integer	Optional. Identifier that is created differently for every setup.
	<NFLLineID>	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>".
<b>Return Value</b>	None.		

<b>Python Syntax</b>	<pre>oModule.Edit(     [         "NAME:&lt;Name&gt;",         "Type:=", "NearFieldLine",</pre>
----------------------	------------------------------------------------------------------------------------------------------------

	<pre>     "VersionID:=", 0,     "ID:=", &lt;ID&gt;,     "NFLineID:=", "&lt;NFLineID&gt;",     "NumSamples:=", "&lt;NumSamples&gt;"    ]) </pre>
<b>Python Example</b>	<pre> oDesign = oDesign.GetModule("RadiationSetupMgr") oModule.Edit( [     "NAME:Line 1",     "Type:=", "NearFieldLine",     "VersionID:=", 0,     "ID:=", 7,     "NFLineID:=", 312,     "NumSamples:=", "1000" ]) </pre>

<b>VB Syn-tax</b>	<pre> oModule.Edit Array("NAME:&lt;Name&gt;", "Type:=", "NearFieldLine", "VersionID:=", 0, "ID:=", &lt;ID&gt;, "NFLineID:=", &lt;NFLineID&gt;,"NumSamples:=", "&lt;NumSamples&gt;") </pre>
<b>VB Example</b>	<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 <b>Radiation</b> and in the right-click menu, click <b>Insert Near Field Setup&gt;Plane</b> .		
Parameters	Name	Type	Description
	<Name>	String	A user-defined name for the near field sphere.
	<Type>	String	Type of radiation setup. It should be <b>NearFieldSphere</b> .
	<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 " <b>-41.5mm</b> ". 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 " <b>8.5mm</b> ". 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 " <b>16mm</b> ". 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 " <b>16mm</b> ". Use the form "<num><unit>".
	<Height>	String	Optional. Height of the reference rectangle. The default value is " <b>1.652145mm</b> ". 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 <b>1</b> to use the default calculated dimensions of the plane or set to <b>2</b> to double each dimension of the plane. The default value is " <b>1</b> ". Use the form "<num>".
	<UseExpFactor>	Boolean	Optional. Set to <b>True</b> to add the exponent factor to the plane dimensions or <b>False</b> to use the automatically calculated plane dimensions or user-defined plane dimensions. The default value is <b>False</b> .
	<CSID>	Integer	Optional. An identifier for the coordinate system.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	<pre> oModule.Add(     [         "NAME:&lt;Name&gt;",         "Type:=", "NearFieldPlane",         "VersionID:=", 0,         "ID:=", &lt;ID&gt;,         "MinX:=", "&lt;MinX&gt;",         "MaxX:=", "&lt;MaxX&gt;",         "MinY:=", "&lt;MinY&gt;",         "MaxY:=", "&lt;MaxY&gt;",         "Height:=", "&lt;Height&gt;",         "ExpFactor:=", "&lt;ExpFactor&gt;",         "UseExpFactor:=", &lt;UseExpFactor&gt;,         "CSID:=", &lt;CSID&gt;     ] ) </pre>
----------------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

	])
Python Example	<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 ])</pre>

VB Syntax	<pre>oModule.Add Array("NAME:&lt;Name&gt;", "Type:=", "NearFieldPlane", "VersionID:=", 0, "ID:=", &lt;/ID&gt;, "MinX:=", "&lt;MinX&gt;",     "MaxX:=", "&lt;MaxX&gt;", "MinY:=", "&lt;MinY&gt;", "MaxY:=", "&lt;MaxY&gt;", "Height:=", "&lt;Height&gt;", "ExpFactor:=", "&lt;ExpFactor&gt;")</pre>
-----------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

	"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 > <b>Radiation</b> , 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 <b>NearFieldSphere</b> .
	<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 <b>1</b> to use the default calculated dimensions of the plane or set to <b>2</b> to double each dimension of the plane. Use the form "<num>".
	<UseExpFactor>	Boolean	Optional. Set to <b>True</b> to add the exponent factor to the plane dimensions or <b>False</b> to use the automatically calculated plane dimensions or user-defined plane dimensions.
	<CSID>	Integer	Optional. An identifier for the coordinate system.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	<pre> oModule.Edit(     [         "NAME:&lt;Name&gt;",         "Type:=", "NearFieldPlane",         "VersionID:=", 0,         "ID:=", &lt;ID&gt;,         "MinX:=", "&lt;MinX&gt;",         "MaxX:=", "&lt;MaxX&gt;",         "MinY:=", "&lt;MinY&gt;",         "MaxY:=", "&lt;MaxY&gt;",         "Height:=", "&lt;Height&gt;",         "ExpFactor:=", "&lt;ExpFactor&gt;",     ] ) </pre>
----------------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

	<pre>     "UseExpFactor:=", &lt;UseExpFactor&gt;,     "CSID:=", &lt;CSID&gt;   ]) </pre>
<b>Python Example</b>	<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>

<b>VB Syn-</b>	<code>oModule.Edit Array("NAME:&lt;Name&gt;", "Type:=", "NearFieldPlane", "VersionID:=", 0, "ID:=", &lt;ID&gt;, "MinX:=", "&lt;MinX&gt;","</code>
----------------	---------------------------------------------------------------------------------------------------------------------------------------------------

<b>tax</b>	"MaxX:=", "<MaxX>", "MinY:=", "<MinY>", "MaxY:=", "<MaxY>", "Height:=", "<Height>", "ExpFactor:=", "<ExpFactor>", "CSID:=", "<CSID>")
<b>VB Example</b>	<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 *UseLocal/CS* parameters.

<b>UI Access</b>	In the Project Manager under the design name, right click <b>Radiation</b> and in the right-click menu, click <b>Insert Near Field Setup&gt;Sphere</b> .		
<b>Parameters</b>	Name	Type	Description
	<Name>	String	A user-defined name for the near field sphere.
	<Type>	String	Type of radiation setup. It should be <b>NearFieldSphere</b> .
	<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".

	<code>&lt;StartPhi&gt;</code>	String	Optional. Starting point for phi in the spherical coordinate system for the sphere being defined. The default value is " <b>-180deg</b> ". Use the form " <code>&lt;num&gt;deg</code> ".
	<code>&lt;StopTheta&gt;</code>	String	Optional. Stopping point for theta in the spherical coordinate system for the sphere being defined. The default value is " <b>180deg</b> ". Use the form " <code>&lt;num&gt;deg</code> ".
	<code>&lt;StopPhi&gt;</code>	String	Optional. Stopping point for phi in the spherical coordinate system for the sphere being defined. The default value is " <b>1800deg</b> ". Use the form " <code>&lt;num&gt;deg</code> ".
	<code>&lt;StepTheta&gt;</code>	String	Optional. Step size between the start and the stop theta values at which the far field is calculated. The default value is " <b>2deg</b> ". Use the form " <code>&lt;num&gt;deg</code> ".
	<code>&lt;StepPhi&gt;</code>	String	Optional. Step size between the start and the stop phi values at which the far field is calculated. The default value is " <b>2deg</b> ". Use the form " <code>&lt;num&gt;deg</code> ".
	<code>&lt;CS&gt;</code>	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 <b>-1</b> for a global coordinate system. The default value is <b>-1</b> .
	<code>&lt;UseLocalCS&gt;</code>	Boolean	Optional. <b>True</b> to select a local coordinate system, and <b>False</b> to use the global coordinate system. The default value is <b>False</b> .
	<code>&lt;NFRadius&gt;</code>	String	Optional. Radius of the sphere we want to calculate near fields on. The default value is " <b>20mm</b> ". Use the form " <code>&lt;num&gt;&lt;unit&gt;</code> ".
<b>Return Value</b>	None.		

<b>Python Syntax</b>	<pre>oModule.Add(     [         "NAME:&lt;Name&gt;",         "Type:=" , "InfiniteSphereFarField",</pre>
----------------------	---------------------------------------------------------------------------------------------------------

	<pre>"VersionID:=" , 0, "ID:=" , &lt;ID&gt;, "StartTheta:=" , "&lt;StartTheta&gt;deg", "StartPhi:=" , "&lt;StartPhi&gt;deg", "StopTheta:=" , "&lt;StopTheta&gt;deg", "StopPhi:=" , "&lt;StopPhi&gt;deg", "StepTheta:=" , "&lt;StepTheta&gt;deg", "StepPhi:=" , "&lt;StepPhi&gt;deg", "CS:=" , &lt;CS&gt;, "UseLocalCS:=" , &lt;UseLocalCS&gt;, "NFRadius:=" , "&lt;NFRadius&gt;"</pre> <p>)</p>
<b>Python Example</b>	<pre>oDesign = oDesign.GetModule("RadiationSetupMgr") oModule.Add( [     "NAME:Sphere 1",     "Type:=" , "NearFieldSphere",     "VersionID:=" , 0,     "ID:=" , 7,     "StartTheta:=" , "0deg",</pre>

```

    "StartPhi:=", "-180deg",
    "StopTheta:=", "180deg",
    "StopPhi:=", "180deg",
    "StepTheta:=", "2deg",
    "StepPhi:=", "2deg",
    "CS:=", -1,
    "UseLocalCS:=", False,
    "NFRadius:=", "20mm"
]
)

```

<b>VB Syntax</b>	<pre> oModule.Add Array("NAME:&lt;Name&gt;", "Type:=", "NearFieldSphere", "VersionID:=", 0, "ID:=", &lt;ID&gt;, "StartTheta:=", "&lt;StartTheta&gt;deg", "StartPhi:=", "&lt;StartPhi&gt;deg", "StopTheta:=", "&lt;StopPhi&gt;deg", "StopPhi:=", "&lt;StopPhi&gt;deg", "StepTheta:=", "&lt;StepTheta&gt;deg", "StepPhi:=", "&lt;StepPhi&gt;deg", "CS:="",&lt;CS&gt;, "UseLocalCS:=", &lt;UseLocalCS&gt;,"NFRadius:=", "&lt;NFRadius&gt;") </pre>
<b>VB Example</b>	<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, "NFRadius:=", "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 *UseLocal/CS* parameters.

UI Access	In the Project Manager under the design name > <b>Radiation</b> , 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 <b>NearFieldSphere</b> .
	<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> <b>deg</b> ".
	<StartPhi>	String	Optional. Starting point for phi in the spherical coordinate system for the sphere being defined. Use the form "<num> <b>deg</b> ".
	<StopTheta>	String	Optional. Stopping point for theta in the spherical coordinate system for the sphere being defined. Use the form "<num> <b>deg</b> ".
	<StopPhi>	String	Optional. Stopping point for phi in the spherical coordinate system for the sphere being defined. Use the form "<num> <b>deg</b> ".
	<StepTheta>	String	Optional. Step size between the start and the stop theta values at which the far field is calculated. Use the form "<num> <b>deg</b> ".
	<StepPhi>	String	Optional. Step size between the start and the stop phi values at which the far field is calculated. Use the form "<num> <b>deg</b> ".
	<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.

	<code>&lt;UseLocalCS&gt;</code>	Boolean	Optional. <b>True</b> to select a local coordinate system, and <b>False</b> to use the global coordinate system.
	<code>&lt;NRadius&gt;</code>	String	Optional. Radius of the sphere we want to calculate near fields on. Use the form "<num><unit>".
<b>Return Value</b>	None.		

<b>Python Syntax</b>	<pre> oModule.Edit(     [         "NAME:&lt;Name&gt;",         "Type:=" , "InfiniteSphereFarField",         "VersionID:=" , 0,         "ID:=" , &lt;ID&gt;,         "StartTheta:=" , "&lt;StartTheta&gt;deg",         "StartPhi:=" , "&lt;StartPhi&gt;deg",         "StopTheta:=" , "&lt;StopTheta&gt;deg",         "StopPhi:=" , "&lt;StopPhi&gt;deg",         "StepTheta:=" , "&lt;StepTheta&gt;deg",         "StepPhi:=" , "&lt;StepPhi&gt;deg",         "CS:=" , &lt;CS&gt;,         "UseLocalCS:=" , &lt;UseLocalCS&gt;,         "NRadius:=" , "&lt;NRadius&gt;"     ] ) </pre>
----------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**Python Example**

```
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"
])
```

**VB Syn-  
tax**

```
oModule.Edit 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>, "NFRadius:=", "<NFRadius>")
<b>VB Example</b>	<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.

# 20 - User Defined Document Script Commands

The product has to implement the [GetModule](#) call to create the UserDefinedDocument scripting object (e.g., Check AltraSimDesign.cpp (function GetMgrIDispatch())). To access the UserDefinedDocuments scripting object, use:

```
Set oModule = oDesign.GetModule("UserDefinedDocuments")
```

Once you have the scripting object, you can use the following methods:

- [AddDocument](#)
- [EditDocument](#)
- [RenameDocument](#)
- [DeleteDocument](#)
- [UpdateDocument](#)
- [ViewHtmlDocument](#)
- [ViewPdfDocument](#)
- [SaveHtmlDocumentAs](#)
- [SavePdfDocumentAs](#)
- [GetDocumentDefinitionNames](#)
- [DeleteAllDocuments](#)
- [UpdateAllDocuments](#)

You can find examples of how to use these methods on each of the methods' pages. A complete [example of a Python script](#) is also available, as is an [example with a line by line explanation](#).

## AddDocument

Creates a new document based on provided data and traces. The document names come from UserDefinedDocument folder under syslib, userlib, and personallib. This creates a document and places it in the Project Manager under **Results > Documents**.

---

UI Access	Right click on <b>Results &gt; Create Document</b> . Choose a document name.		
Parameters	Name	Type	Description
	<data>	Array	Data that defines the document.
Return Value	None		

Python Syntax	AddDocument (<data>, <traces>)
Python Example	<pre>oModule.AddDocument (     [         "NAME:Design Summary",         "",         "SysLib",         "DesignSummary",         [ "NAME:Inputs" ]     ],     [ "NAME:DocTraces" ] )</pre>

VB 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("Nominal"), _  
"Pulse_Width:=", Array("Nominal"), "Data_Rate:=", Array("Nominal"), _  
"Length:=", Array("Nominal")), Array("Probe Component:=", Array("DQ0")), Array())))
```

## DeleteAllDocuments

Deletes all documents in the object.

<b>UI Access</b>	Right click on <b>Documents</b> in the Project Manager and click <b>Delete All Documents</b> .
<b>Parameters</b>	None.
<b>Return Value</b>	None.

<b>Python Syntax</b>	DeleteAllDocuments()
<b>Python Example</b>	oModule.DeleteAllDocuments ()

<b>VB Syntax</b>	DeleteAllDocuments
------------------	--------------------

<b>VB Example</b>	<code>oModule.DeleteAllDocuments</code>
-------------------	-----------------------------------------

## DeleteDocument

Deletes a specified document.

<b>UI Access</b>	Right click on the created document in the Project Manager under <b>Results &gt; Documents</b> and click <b>Delete</b> .						
<b>Parameters</b>	<table border="1"> <tr> <td>Name</td> <td>Type</td> <td>Description</td> </tr> <tr> <td><code>&lt;name&gt;</code></td> <td>String</td> <td>Name of the document to be deleted.</td> </tr> </table>	Name	Type	Description	<code>&lt;name&gt;</code>	String	Name of the document to be deleted.
Name	Type	Description					
<code>&lt;name&gt;</code>	String	Name of the document to be deleted.					
<b>Return Value</b>	None.						

<b>Python Syntax</b>	<code>DeleteDocument(&lt;names&gt;)</code>
<b>Python Example</b>	<code>oModule.DeleteDocument ("Project Design Summary")</code>

<b>VB Syntax</b>	<code>DeleteDocument &lt;names&gt;</code>
<b>VB Example</b>	<code>oModule.DeleteDocument "Project Design Summary"</code>

## EditDocument

Edits specified documents. If the document pops up a dialog box, the user can make a change the inputs for the document. The document is regenerated and updated. A new one is *not* created.

<b>UI Access</b>	Right click on the created document in the Project Manager under <b>Results &gt; Documents</b> and click <b>Modify document</b> .
------------------	-----------------------------------------------------------------------------------------------------------------------------------

Parameters	Name	Type	Description
	<originalName>	String	Name of the original document
	<modifiedData>	Array	New data to add to or modify the document
	<modifiedTraces>	Array	Trace data for the inputs of the document
Return Value	None.		

<b>Python Syntax</b>	EditDocument( <name>,<data>,<traces> )
<b>Python Example</b>	<pre>oModule.EditDocument("Design Summary", [     "NAME:Design Summary",     "",     "SysLib",     "DesignSummary",     ["NAME:Inputs"] ], ["NAME:DocTraces"] )</pre>

<b>VB Syntax</b>	EditDocument <name> , <data> , <traces>
------------------	-----------------------------------------

<b>VB Example</b>	<pre> oModule.EditDocument "Design Summary", _ Array("NAME:Design Summary", "", "SysLib", _ "DesignSummary", Array("NAME:Inputs")), Array("NAME:DocTraces") </pre>
-------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------

## GetDocumentDefinitionNames

Document definition names are the list of names that can be used to create a document. They appear when you click on **Create document**. This method returns the filenames of document definitions according to the files in the installation directories:

- syslib/UserDefinedDocuments
- userlib/UserDefinedDocuments
- personallib/UserDefinedDocuments

<b>UI Access</b>	NA						
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;separator&gt;</td> <td>String</td> <td>Separator used to convey the directory "level"</td> </tr> </tbody> </table>	Name	Type	Description	<separator>	String	Separator used to convey the directory "level"
Name	Type	Description					
<separator>	String	Separator used to convey the directory "level"					
<b>Return Value</b>	None.						

<b>Python Syntax</b>	GetDocumentDefinitionNames(<separator>)
<b>Python Example</b>	<code>oModule.GetDocumentDefinitionNames ("")</code>

<b>VB Syntax</b>	GetDocumentDefinitionNames <separator>
<b>VB Example</b>	<code>oModule.GetDocumentDefinitionNames "</code>

## GetDocumentNames

Retrieves the names for all documents.

<b>UI Access</b>	N/A
<b>Parameters</b>	None.
<b>Return Value</b>	Array of strings containing document names.

<b>Python Syntax</b>	GetDocumentNames()
<b>Python Example</b>	<code>oModule.GetDocumentNames ()</code>

<b>VB Syntax</b>	GetDocumentNames
<b>VB Example</b>	<code>oModule.GetDocumentNames</code>

## RenameDocument

Changes the name of a document.

<b>UI Access</b>	Right click on the created document in the Project Manager under <b>Results&gt; Documents</b> and click <b>Rename</b> .											
<b>Parameters</b>	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td><code>&lt;oldName&gt;</code></td><td>String</td><td>Current name of the document</td></tr><tr><td><code>&lt;newName&gt;</code></td><td>String</td><td>New name of the document</td></tr></tbody></table>			Name	Type	Description	<code>&lt;oldName&gt;</code>	String	Current name of the document	<code>&lt;newName&gt;</code>	String	New name of the document
Name	Type	Description										
<code>&lt;oldName&gt;</code>	String	Current name of the document										
<code>&lt;newName&gt;</code>	String	New name of the document										

<b>Return Value</b>	None.
---------------------	-------

<b>Python Syntax</b>	RenameDocument(<oldName>, <newName>)
<b>Python Example</b>	<code>oModule.RenameDocument ("Design Summary", "Project Design Summary")</code>

<b>VB Syntax</b>	RenameDocument <oldName>, <newName>
<b>VB Example</b>	<code>oModule.RenameDocument "Design Summary", "Project Design Summary"</code>

## SaveHtmlDocumentAs

Saves a pre-existing HTML file to a different name and/or location.

<b>UI Access</b>	Right click on the created document in the Project Manager under <b>Results &gt; Documents</b> and click <b>Save As &gt; HTML</b> .									
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;name&gt;</td> <td>String</td> <td>Name of the document to be saved.</td> </tr> <tr> <td>&lt;saveTo&gt;</td> <td>String</td> <td>File path to save the document to.</td> </tr> </tbody> </table>	Name	Type	Description	<name>	String	Name of the document to be saved.	<saveTo>	String	File path to save the document to.
Name	Type	Description								
<name>	String	Name of the document to be saved.								
<saveTo>	String	File path to save the document to.								
<b>Return Value</b>	none									

<b>Python Syntax</b>	SaveHtmlDocumentAs(<name>, <saveTo>)
<b>Python Example</b>	<code>oModule.SaveHtmlDocumentAs ("Design Summary 1", "DS1.html")</code>

<b>VB Syntax</b>	SaveHtmlDocumentAs <name> <saveTo>
<b>VB Example</b>	oModule.SaveHtmlDocumentAs "Design Summary 1" "DS1.html"

## SavePdfDocumentAs

Saves a pre-existing PDF file to a different name and/or location.

<b>UI Access</b>	Right click on the created document in the Project Manager under <b>Results &gt; Documents</b> and click <b>Save As &gt; PDF</b> .									
<b>Parameters</b>	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>&lt;name&gt;</td><td>String</td><td>Name of the document to be saved.</td></tr><tr><td>&lt;saveTo&gt;</td><td>String</td><td>File path to save the document at.</td></tr></tbody></table>	Name	Type	Description	<name>	String	Name of the document to be saved.	<saveTo>	String	File path to save the document at.
Name	Type	Description								
<name>	String	Name of the document to be saved.								
<saveTo>	String	File path to save the document at.								
<b>Return Value</b>	None.									

<b>Python Syntax</b>	SavePdfDocumentAs(<name>, <saveTo>)
<b>Python Example</b>	oModule.SavePdfDocumentAs("Design Summary 1", "DS1.pdf")

<b>VB Syntax</b>	SavePdfDocumentAs <name>, <saveTo>
<b>VB Example</b>	oModule.SavePdfDocumentAs "Design Summary 1", "DS1.pdf"

## UpdateAllDocuments

Refreshes the contents of all created documents. This action is made on the folder rather than the individual document.

---

<b>UI Access</b>	Right click on <b>Results &gt; Documents</b> in the Project Manager and click <b>Update All Documents</b> .
<b>Parameters</b>	None.
<b>Return Value</b>	None.

<b>Python Syntax</b>	<code>UpdateAllDocuments()</code>
<b>Python Example</b>	<code>oModule.UpdateAllDocuments ()</code>

<b>VB Syntax</b>	<code>UpdateAllDocuments</code>
<b>VB Example</b>	<code>oModule.UpdateAllDocuments</code>

## UpdateDocument

Refreshes the contents of the selected document.

<b>UI Access</b>	Right click on the created document in the Project Manager under <b>Results &gt; Documents</b> and click <b>Update Document</b> .		
<b>Parameters</b>	Name <code>&lt;name&gt;</code>	Type <code>String</code>	Description Name of the document to be updated
<b>Return Value</b>	None.		

<b>Python Syntax</b>	UpdateDocument(<name>)
<b>Python Example</b>	<code>oModule.UpdateDocument ("Test UDD Report")</code>

<b>VB Syntax</b>	UpdateDocument <name>
<b>VB Example</b>	<code>oModule.UpdateDocument "Test UDD Report"</code>

## ViewHtmlDocument

Displays a pre-existing document as HTML.

<b>UI Access</b>	Right click on the created document in the Project Manager under <b>Results &gt; Documents</b> and click <b>View Xml Document</b> .								
<b>Parameters</b>	<table border="1"><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;name&gt;</td><td>String</td><td>Name of the document to be viewed as a HTML</td></tr></table>			Name	Type	Description	<name>	String	Name of the document to be viewed as a HTML
Name	Type	Description							
<name>	String	Name of the document to be viewed as a HTML							
<b>Return Value</b>	None								

<b>Python Syntax</b>	ViewHtmlDocument(<name>)
<b>Python Example</b>	<code>oModule.ViewHtmlDocument ("Design Summary 1")</code>

<b>VB Syntax</b>	ViewHtmlDocument <name>
<b>VB Example</b>	<code>oModule.ViewHtmlDocument "Design Summary 1"</code>

## ViewPdfDocument

Displays a pre-existing document as a PDF file.

<b>UI Access</b>	Right click on the created document in the Project Manager under <b>Results &gt; Documents</b> and click <b>View PDF Document</b> .								
<b>Parameters</b>	<table border="1"> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> <tr> <td>&lt;name&gt;</td> <td>String</td> <td>Name of the document to be viewed as a PDF</td> </tr> </table>			Name	Type	Description	<name>	String	Name of the document to be viewed as a PDF
Name	Type	Description							
<name>	String	Name of the document to be viewed as a PDF							
<b>Return Value</b>	None.								

<b>Python Syntax</b>	ViewPdfDocument(<name>)
<b>Python Example</b>	oModule.ViewPdfDocument ("Design Summary 1")

<b>VB Syntax</b>	ViewPdfDocument <name>
<b>VB Example</b>	oModule.ViewPdfDocument "Design Summary 1"

## Explication of a Sample UDD Script

This VB script defines a document. It is a portion of one of the UDD example VB scripts.

```
Array("NAME:Test Report",           ' Name of the document
      "Test Report",                 ' Description of the document
      "SysLib",                      ' Location of the python script: Syslib, Userlib, PeronalLib, etc.
      "TestUDDReport",                ' Relative path of the script in the UserDefinedDocuments folder
```

' This array is the start of the input definition.

```
Array("NAME:Inputs",  
      ' Document Inputs keyword  
  
'This array contains the Solution input.  
Array("NAME:DLMetrics",  
      "Solution",  
      "Data Line Metrics",  
      -1,  
      -1),  
      ' Input name  
      ' Solution Input Type  
      ' Input Description  
      ' Solution ID  
      ' Report ID  
  
'This array contains the trace input.  
Array("NAME:DQ0",  
      "Trace",  
      "DQ0",  
      -1,  
      -1),  
      ' Input name  
      ' Trace Input Type  
      ' Input Description  
      ' Solution ID  
      ' Report ID  
  
'This array contains the text input.  
Array("NAME:Name",  
      "Text",  
      "User Name",  
      Array("Sita Ramesh")),  
      ' Input name  
      ' Text Input Type  
      ' Input Description  
      ' Default Value  
  
'This array contains the Bool input.  
Array("NAME:Summary",  
      "Bool",  
      "Display Summary",  
      Array(true)),  
      ' Input name  
      ' Boolean Input Type  
      ' Input Description  
      ' Default Value  
  
'This array contains the number input.
```

```

Array("NAME:Version",
      ' Input name
      "Number",
      ' Number Input Type
      "Script Version",
      ' Input Description
      Array(1021))),           ' Default Value

'This array contains trace selection for the solution and trace inputs.
Array("NAME:DocTraces",           ' Document traces keyword

'This array has input for "DLMetrics".
Array("NAME:DLMetrics",           ' Input name

'This array defines a trace similar to the UDO. This trace definition is a User defined solution
Array("User Defined", "", "DDR3 AC-Timing 4-DQ1", Array("Context:=", ""), Array("Index:=", Array
("All"), "Trise:=", Array("Nominal"), "Tfall:=", Array("Nominal"), "Pulse_Width:=", Array("Nom-
inal"), "Data_Rate:=", Array("Nominal"), "Length:=", Array("Nominal")), Array("Probe Com-
ponent:=", Array(""))), Array()))

'This array is for input "DQ0".
Array("NAME:DQ0",

'This array defines a trace similar to the UDO. This trace definition is a Standard solution.
Array("Standard", "DQ0", "NexximTransient", Array("NAME:Context", "SimValueContext:=", Array(1,
0, 2, 0, false, false, -1, 1, 0, 1, 1, "", 0, 0, "DE", false, "0", "DP",
false, "20000000", "DT", false, "0.001", "WE", false, "100ns", "WM", false,
"100ns", "WN", false, "0ps", "WS", false, "0ps")), Array("Time:=", Array("All"), "Trise:=",
Array(
"Nominal"), "Tfall:=", Array("Nominal"), "Pulse_Width:=", Array("Nominal"), "Data_Rate:=", Array
("Nominal"), "Length:=", Array("Nominal")), Array("Probe Component:=", Array(
"DQ0")), Array())))

```

## Example Python Script: Defining a Document

This script creates a user-defined solution and a document.

```
import ScriptEnv
ScriptEnv.Initialize("Ansoft.ElectronicsDesktop")
oDesktop.RestoreWindow()
oProject = oDesktop.SetActiveProject("BJT_Inverter")
oDesign = oProject.SetActiveDesign("Nexxim1")
oModule = oDesign.GetModule("UserDefinedSolutionModule")

oModule.CreateUserDefinedSolution("UDS Distance Trace Arithmetic Result1", "SysLib", "TraceArith-
metic/Distance Sweep Trace Arithmetic",
[
    "Offset 1:=", "0",
    "Scale 1:=", "1",
    "Offset 2:=", "0",
    "Scale 2:=", "1",
    "Operation:=", "Add"
],
[
[
    [
        "Standard",
        "probe1",
        "Transient",
        [
            style="font-family: monospace;">"NAME:Context",
            style="font-family: monospace;">"SimValueContext:=", [1,0,2,0,False,False,-
1,1,0,1,1,"",0,0,"DE",False,"0","DP",False,"500000000","DT",False,"0.001","NUMLEVELS",False,"0","-
WE",False,"10us","WM",False,"10us","WN",False,"0ns","WS",False,"0ns"]
        ],
        [
            "Time:=", ["All"]
        ],
        [
            "Probe Component:=", ["V(Port1)"]
        ]
    ]
]
```

```
        ],
        []
    ],
    [
        "Standard",
        "probe2",
        "Transient",
        [
            "NAME:Context",
            "SimValueContext:=", [1,0,2,0,False,False,-
1,1,0,1,1,"",0,0,"DE",False,"0","DP",False,"500000000","DT",False,"0.001","NUMLEVELS",False,"0",-"
"WE",False,"10us","WM",False,"10us","WN",False,"0ns","WS",False,"0ns"]
        ],
        [
            "Time:=", ["All"]
        ],
        [
            "Probe Component:=", ["V(Port1)"]
        ],
        []
    ]
],
[])
oModule = oDesign.GetModule("UserDefinedDocuments")
oModule.AddDocument(
[
    "NAME:Test Report",
    "Test Report",
    "SysLib",
    "TestUDDInputs",
    [
        "NAME:Inputs",
        [
            "NAME:UDS1",

```

```
        "Solution",
        "UDS Distance Trace Arithmetic Result1",
        1000000,
        0
    ],
    [
        "NAME:UDS2",
        "Solution",
        "UDS Distance Trace Arithmetic Result2",
        1000000,
        2
    ]
],
[
    "NAME:DocTraces",
    [
        "NAME:UDS1",
        [
            "User Defined",
            "",
            "UDS Distance Trace Arithmetic Result1",
            [
                "Context:=", ""
            ],
            [
                "Distance:=", ["All"]
            ],
            [
                "Probe Component:=", []
            ],
            []
        ]
    ]
]
```

```
        ],
    ],
    [
        "NAME:UDS2",
        [
            "User Defined",
            "",
            "UDS Distance Trace Arithmetic Result1",
            [
                "Context:=", ""
            ],
            [
                "Distance:=", ["All"]
            ],
            [
                "Probe Component:=", []
            ],
            []
        ]
    ],
    []
)
])
```

This page intentionally  
left blank.

# 21 - 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 <b>Results &gt; Create User Defined Solution</b> .		
Parameters	Name	Type	Description
	<SolName>	String	Name of user defined solution.
	<LibType>	String	Indicates the library where the UDS plugin file is located. This parameter must be one of the following values: "SysLib", "UserLib", "PersonalLib".
	<RelativePath>	String	The path of the UDS plugin file relative to the "UserDefinedOutputs" sub-

		directory of the library specified by <LibType>.
	<PropList>	Array Strings specify name-value pairs corresponding to the UDS properties specified in the plugin file.  For example:  Array("multiply_factor:=", "2.0", "component_name:=", "resistor1")
	<ProbeSelections>	Array Name of the probe being specified. Note: this must match a probe name specified in the UDS plugin file.
	<DynamicProbes>	Array Array of <ProbeSelection>'s, representing the probes that are used by dynamic-probes.
<b>Return Value</b>	String name of created user defined solution.	

<b>Python Syntax</b>	CreateUserDefinedSolution(<SoluName>, <LibType>, <RelativePath>, <PropList>, <ProbeSelections>, <DynamicProbes>)
<b>Python Example</b>	<pre>oModule.CreateUserDefinedSolution(     "ConstantTimestep1", "SysLib",     "ConstantTimestep", [], [], [])</pre>

<b>VB Syntax</b>	CreateUserDefinedSolution <SoluName>, <LibType>, <RelativePath>, <PropList>, <ProbeSelections>, <DynamicProbes>
<b>VB Example</b>	<pre>oModule.CreateUserDefinedSolution "User Defined Solution 1", _     "SysLib", "Example", _</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()))

```

## DeleteUserDefinedSolutions

Deletes one or more user defined solutions.

<b>UI Access</b>	Delete button from the <b>User Defined Solutions</b> dialog.		
<b>Parameters</b>	Name	Type	Description
	<SolNames>	Array	Array of strings containing names of User Defined Solutions to be deleted.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	DeleteUserDefinedSolutions(<SolNames>)
<b>Python Example</b>	<code>oModule.DeleteUserDefinedSolutions(["Solution1", "Solution2"])</code>

<b>VB Syntax</b>	DeleteUserDefinedSolutions <SolNames>
<b>VB Example</b>	<code>oModule.DeleteUserDefinedSolutions _</code> <code>Array("Solution1", "Solution2")</code>

## EditUserDefinedSolution

Modifies an existing user defined solution.

<b>UI Access</b>	Edit button from the <b>User Defined Solutions</b> dialog box.		
<b>Parameters</b>	Name	Type	Description
	<SolName>	String	Name of user defined solution to be edited.
	<NewSolName>	String	New name for the specified user defined solution.
	<LibType>	String	Indicates the library where the UDS plugin file is located. This parameter must be one of the following values: "SysLib", "UserLib", "PersonalLib".
	<RelativePath>	String	The path of the UDS plugin file relative to the "UserDefinedOutputs" sub-directory of the library specified by <LibType>.
	<PropList>	Array	Strings specify name-value pairs corresponding to the UDS properties specified in the plugin file.  For example:  <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.
<b>Return Value</b>	String name of update user defined solution.	

<b>Python Syntax</b>	EditUserDefinedSolution(<SoluName>, <NewSoluName>, <LibType>, <RelativePath>, <PropList>, <ProbeSelections>, <DynamicProbes>)
<b>Python Example</b>	<pre>oModule.EditUserDefinedSolution("ConstantTimestep1" "ConstantTimestep1After", "SysLib", "ConstantTimestep", [], [], [])</pre>

<b>VB Syntax</b>	EditUserDefinedSolution <SoluName>, <NewSoluName>, <LibType>, <RelativePath>, <PropList>, <ProbeSelections>, <DynamicProbes>
<b>VB Example</b>	<pre>oModule.CreateUserDefinedSolution "User Defined Solution 1", _ "User Defined Solution 1", "SysLib", "Example", _ Array("multiplication_factor:=", "1.2"), _ Array(Array("Modal Solution Data", "Probe 1", "Setup1 : LastAdaptive", _ Array(), Array("Freq:=", Array( "All"))), _ Array("Probe Component:=", Array("dB(S(1,1))))), Array()), _ Array( "Modal Solution Data", "Probe 2", "Setup1 : LastAdaptive", Array(), _</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.

<b>UI Access</b>	N/A
<b>Parameters</b>	None.
<b>Return Value</b>	Array of strings containing solution names.

<b>Python Syntax</b>	GetUserDefinedSolutionNames()
<b>Python Example</b>	oModule.GetUserDefinedSolutionNames()

<b>VB Syntax</b>	GetUserDefinedSolutionNames
<b>VB Example</b>	oModule.GetUserDefinedSolutionNames

## GetUserDefinedSolutionProperties

Obtains properties for a specified user defined solution.

<b>UI Access</b>	N/A								
<b>Parameters</b>	<table border="1"><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;SoluName&gt;</td><td>String</td><td>Name of a specified user defined solution.</td></tr></table>			Name	Type	Description	<SoluName>	String	Name of a specified user defined solution.
Name	Type	Description							
<SoluName>	String	Name of a specified user defined solution.							
<b>Return Value</b>	Array of strings containing properties values.								

<b>Python Syntax</b>	GetUserDefinedSolutionProperties(<SoluName>)
<b>Python Example</b>	oModule.GetUserDefinedSolutionProperties ("ConstantTimestep1")

<b>VB Syntax</b>	GetUserDefinedSolutionProperties <SoluName>
<b>VB Example</b>	oModule.GetUserDefinedSolutionProperties "ConstantTimestep1"

This page intentionally  
left blank.

## 22 - Network Data Explorer Script Commands

Network Data Explorer (NDE) scripting uses three objects, each with unique script commands:

- **Network Data Explorer** – top-level object obtained by calling `oNDE=oDesktop.GetTool("ndExplorer")`. Network Data Explorer commands are called using `oNDE`.
- **Network Data** – single set of S-parameters, corresponding to a single entry in the UI tree. Network Data commands are called using `oData`.
- **Post Process Settings** – settings that can be applied and removed from network data without making permanent changes to the underlying data. Post Process Settings commands are called using `oPostProc`.

Examples using the above objects:

```
oNDE = oDesktop.GetTool("ndExplorer")
oData = oNDE.Open("D:\folder\test.s2p")
success = oPostProc.AddDiffPair(2, 1, "Diff1", "Comm1", 100, 25)
```

The following commands are described in this section:

**Warning: The following commands are preliminary. Their syntax may change. Be prepared to make changes to legacy scripts.**

[AddDiffPair](#)

[Cascade \(SPISim\)](#)

[ClearDiffPairs](#)

[Clone](#)

[Close](#)

[Combine \(SPISim\)](#)

[Deembed \(SPISim\)](#)

[DeembedBack \(SPISim\)](#)

[DeembedFront \(SPISim\)](#)

[DisableDiffPairs](#)

[EnableDiffPairs](#)

[ExportCitiFile](#)

[ExportFullWaveSpice](#)

[ExportMatlab](#)

[ExportNMFDATA](#)

[ExportNetworkData](#)

[ExportSpreadsheet](#)

[ExportTouchstone](#)

[ExportTouchstone2](#)

[Extract \(SPISim\)](#)

[GetFrequencies](#)

[GetFrequencyCount](#)

[GetName](#)

[GetPortCount](#)

[GetPortNumber](#)

[GetPostProcSettings](#)[GetSolutionVariation](#)[GetVariation](#)[HasSameData](#)[LoadSolution](#)[Open](#)[Rename \(SPISim\)](#)[Renormalize \(SPISim\)](#)[Reorder](#)[Reorder \(SPISim\)](#)[Reset](#)[SetAllPortImpedances](#)[SetAllPortImpedances](#)[SetPortDeembedDistance](#)[SetPortImpedance](#)[SetPostProcSettings](#)[Smooth](#)[Stretch \(SPISim\)](#)[Terminate](#)

**Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.**

## AddDiffPair

Specifies a differential pair from two terminal ports.

UI Access	From the <b>Network Data Explorer</b> tab, select <b>Differential Pairs</b> in the <b>NDE</b> ribbon to open the <b>Differential Pairs</b> window. Then select differential pairs from the <b>Pairs</b> group box and click <b>Add Pairs &gt;&gt;</b> .																										
Parameters	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><code>&lt;positiveTerminal&gt;</code></td> <td>Integer</td> <td>The portNumber of the positive terminal.</td> </tr> <tr> <td><code>&lt;negativeTerminal&gt;</code></td> <td>Integer</td> <td>The portNumber of the negative terminal.</td> </tr> <tr> <td><code>&lt;diffName&gt;</code></td> <td>String</td> <td>The new name of the differential pin (e.g., Diff1).</td> </tr> <tr> <td><code>&lt;commName&gt;</code></td> <td>String</td> <td>The new name of the common pin (e.g., Comm1).</td> </tr> <tr> <td><code>&lt;diffImpedance&gt;</code></td> <td>Double</td> <td>The differential pin characteristic impedance.</td> </tr> <tr> <td><code>&lt;commImpedance&gt;</code></td> <td>Double</td> <td>The common pin characteristic impedance.</td> </tr> <tr> <td><code>&lt;matched&gt;</code></td> <td>Boolean</td> <td>Specify whether to use matched pair.</td> </tr> </tbody> </table> <p><b>Note:</b> The default value is <b>False</b>.</p>			Name	Type	Description	<code>&lt;positiveTerminal&gt;</code>	Integer	The portNumber of the positive terminal.	<code>&lt;negativeTerminal&gt;</code>	Integer	The portNumber of the negative terminal.	<code>&lt;diffName&gt;</code>	String	The new name of the differential pin (e.g., Diff1).	<code>&lt;commName&gt;</code>	String	The new name of the common pin (e.g., Comm1).	<code>&lt;diffImpedance&gt;</code>	Double	The differential pin characteristic impedance.	<code>&lt;commImpedance&gt;</code>	Double	The common pin characteristic impedance.	<code>&lt;matched&gt;</code>	Boolean	Specify whether to use matched pair.
Name	Type	Description																									
<code>&lt;positiveTerminal&gt;</code>	Integer	The portNumber of the positive terminal.																									
<code>&lt;negativeTerminal&gt;</code>	Integer	The portNumber of the negative terminal.																									
<code>&lt;diffName&gt;</code>	String	The new name of the differential pin (e.g., Diff1).																									
<code>&lt;commName&gt;</code>	String	The new name of the common pin (e.g., Comm1).																									
<code>&lt;diffImpedance&gt;</code>	Double	The differential pin characteristic impedance.																									
<code>&lt;commImpedance&gt;</code>	Double	The common pin characteristic impedance.																									
<code>&lt;matched&gt;</code>	Boolean	Specify whether to use matched pair.																									
Return Value	Boolean.																										

Python Syntax	<code>AddDiffPair()</code>
Python Example	<pre>success = oPostProc.AddDiffPair(2, 1, "Diff1", "Comm1", 100, 25)</pre>

<b>VB Syntax</b>	AddDiffPair
<b>VB Example</b>	Set success = oPostProc.AddDiffPair 2, 1, "Diff1", "Comm1", 100, 25

**Warning:** The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.

## Cascade (SPISim)

Creates new network data from a group of network data objects cascaded together. The network data must have an even number of terminal ports and must have the same number of ports.

**Note:** Cascade opens and interacts with **SPISim** to complete.

<b>UI Access</b>	From the <b>Network Data Explorer</b> tab, select <b>Transforms</b> in the <b>NDE</b> ribbon. Then select <b>Cascade</b> from the drop-down menu.		
<b>Parameters</b>	<b>Name</b> <code>&lt;dataArray&gt;</code>	<b>Type</b> Array	<b>Description</b> These network data objects are cascaded together to create the new network data.
<b>Return Value</b>	Network IDispatch. <b>Note:</b> Cascade returns <b>None</b> if the specified network data does not have compatible terminal ports defined.		

<b>Python Syntax</b>	Cascade()
<b>Python Example</b>	<pre>oNDE = oDesktop.GetTool("ndExplorer") oData2 = oNDE.Cascade([oData1, oData2, oData3])</pre>

<b>VB Syntax</b>	Cascade
<b>VB Example</b>	<pre>Set oNDE = oDesktop.GetTool("ndExplorer") Set oData2 = oNDE.Cascade Array(oData1, oData2, oData3)</pre>

**Warning:** The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.

## ClearDiffPairs

Clears all differential pair definitions. All other post-processing settings remain the same.

<b>UI Access</b>	From the <b>Network Data Explorer</b> tab, select <b>Differential Pairs</b> in the <b>NDE</b> ribbon to open the <b>Differential Pairs</b> window. Then select differential pairs from the rightmost box and click <b>&lt;&lt; Remove Pairs</b> .
<b>Parameters</b>	None.
<b>Return Value</b>	None.

<b>Python Syntax</b>	ClearDiffPairs()
----------------------	------------------

<b>Python Example</b>	<pre>oNDE = oDesktop.GetTool("ndExplorer") oPostProc.ClearDiffPairs()</pre>
-----------------------	-----------------------------------------------------------------------------

<b>VB Syntax</b>	ClearDiffPairs
<b>VB Example</b>	<pre>Set oNDE = oDesktop.GetTool("ndExplorer") oPostProc.ClearDiffPairs</pre>

**Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.**

## Clone

Creates a copy of the network data object.

<b>UI Access</b>	From the <b>Network Data Explorer</b> tab, select the network data object to clone. Then select <b>Clone</b> from the <b>NDE</b> ribbon.
<b>Parameters</b>	None.
<b>Return Value</b>	Network IDispatch.

<b>Python Syntax</b>	Clone()
<b>Python Example</b>	<pre>oNDE = oDesktop.GetTool("ndExplorer") oData1 = oData.Clone()</pre>

<b>VB Syntax</b>	Clone
<b>VB Example</b>	<pre>Set oNDE = oDesktop.GetTool("ndExplorer") Set oData1 = oData.Clone</pre>

**Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.**

## Close

Closes the network data object. The object will no longer be accessible.

<b>UI Access</b>	From the <b>Network Data Explorer</b> tab, click <b>Close</b> in the <b>NDE</b> ribbon, or select <b>File &gt; Close</b> .
<b>Parameters</b>	None.
<b>Return Value</b>	None.

<b>Python Syntax</b>	Close()
<b>Python Example</b>	<pre>oNDE = oDesktop.GetTool("ndExplorer") oData.Close()</pre>

<b>VB Syntax</b>	Close
<b>VB Example</b>	<pre>Set oNDE = oDesktop.GetTool("ndExplorer")</pre>

	<code>oData.Close</code>
--	--------------------------

**Warning:** The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.

## Combine (SPISim)

Combines frequencies from multiple network data objects to create a new network data with all of the frequencies.

**Note:** Combine opens and interacts with **SPISim** to complete.

UI Access	From the <b>Network Data Explorer</b> tab, select <b>Transforms</b> in the <b>NDE</b> ribbon. Then select <b>Combine</b> from the drop-down menu.		
Parameters	<b>Name</b> <code>&lt;dataArray&gt;</code>	<b>Type</b> Array	<b>Description</b> These network data objects are combined to create the new network data.
Return Value	<b>Name</b> <code>&lt;freqTol&gt;</code> <b>Type</b> Double <p><b>Note:</b> The default value is <b>100</b>.</p>		

<b>Python Syntax</b>	Combine()
<b>Python Example</b>	<pre>oNDE = oDesktop.GetTool("ndExplorer") oData4 = oNDE.Combine([oData1, oData2, oData3], 1000)</pre>

<b>VB Syntax</b>	Combine
<b>VB Example</b>	<pre>Set oNDE = oDesktop.GetTool("ndExplorer") Set oData4 = oNDE.Combine Array(oData1, oData2, oData3), 1000</pre>

**Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.**

## Deembed (SPISim)

Creates a new network data, deembedding the frontData and the backData from the front and back of the originalData.

**Note:** Deembed opens and interacts with **SPISim** to complete.

<b>UI Access</b>	From the <b>Network Data Explorer</b> tab, select <b>Transforms</b> in the <b>NDE</b> ribbon. Then select <b>Deembed</b> from the drop-down menu to open the <b>Deembed...</b> window, and choose <b>Given 3 S-params in order of: Total, Front, Back</b> from the <b>Settings</b> group box.											
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;originalData&gt;</td> <td>Object</td> <td>Original network data that is deembedded.</td> </tr> <tr> <td>&lt;frontData&gt;</td> <td>Object</td> <td>Network data that is deembedded from the front of the original.</td> </tr> </tbody> </table>			Name	Type	Description	<originalData>	Object	Original network data that is deembedded.	<frontData>	Object	Network data that is deembedded from the front of the original.
Name	Type	Description										
<originalData>	Object	Original network data that is deembedded.										
<frontData>	Object	Network data that is deembedded from the front of the original.										

	<code>&lt;backData&gt;</code>	Object	Network data that is deembeded from the back of the original.
	Network IDispatch.		
<b>Return Value</b>	<p><b>Note:</b> Deembed returns <b>None</b> if the specified network data does not have compatible terminal ports defined.</p>		

<b>Python Syntax</b>	<code>Deembed()</code>
<b>Python Example</b>	<pre>oNDE = oDesktop.GetTool("ndExplorer") oData2 = oNDE.Deembed(oData1, oDataFront, oDataBack)</pre>

<b>VB Syntax</b>	<code>Deembed</code>
<b>VB Example</b>	<pre>Set oNDE = oDesktop.GetTool("ndExplorer") Set oData2 = oNDE.Deembed oData1, oDataFront, oDataBack</pre>

**Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.**

## DeembedBack (SPISim)

Creates a new network data, dembedding the backData from the back of the originalData.

**Note:** DeembedBack opens and interacts with **SPISim** to complete.

UI Access	From the <b>Network Data Explorer</b> tab, select <b>Transforms</b> in the <b>NDE</b> ribbon. Then select <b>Deembed</b> from the drop-down menu to open the <b>Deembed...</b> window, and choose <b>Given 3 S-params in order of: Total, Back</b> from the <b>Settings</b> group box.											
Parameters	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;originalData&gt;</td> <td>Object</td> <td>Original network data that is deembeded.</td> </tr> <tr> <td>&lt;backData&gt;</td> <td>Object</td> <td>Network data that is deembeded from the back of the original.</td> </tr> </tbody> </table>			Name	Type	Description	<originalData>	Object	Original network data that is deembeded.	<backData>	Object	Network data that is deembeded from the back of the original.
Name	Type	Description										
<originalData>	Object	Original network data that is deembeded.										
<backData>	Object	Network data that is deembeded from the back of the original.										
Return Value	<p>Network IDispatch.</p> <p><b>Note:</b> DeembedBack returns <b>None</b> if the specified network data does not have compatible terminal ports defined.</p>											

Python Syntax	DeembedBack()
Python Example	<pre>oNDE = oDesktop.GetTool("ndExplorer") oData2 = oNDE.DeembedBack(oData1, oDataBack)</pre>

VB Syntax	DeembedBack
VB Example	<pre>Set oNDE = oDesktop.GetTool("ndExplorer") Set oData2 = oNDE.DeembedBack oData1, oDataBack</pre>

**Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.**

## DeembedFront (SPISim)

Creates a new network data, deembedding the frontData from the front of the originalData.

**Note:** DeembedFront opens and interacts with **SPISim** to complete.

UI Access	From the <b>Network Data Explorer</b> tab, select <b>Transforms</b> in the <b>NDE</b> ribbon. Then select <b>Deembed</b> from the drop-down menu to open the <b>Deembed...</b> window, and choose <b>Given 3 S-params in order of: Total, Front</b> from the <b>Settings</b> group box.											
Parameters	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;originalData&gt;</td> <td>Object</td> <td>Original network data that is deembedded.</td> </tr> <tr> <td>&lt;frontData&gt;</td> <td>Object</td> <td>Network Data that is deembedded from the front of the original.</td> </tr> </tbody> </table>			Name	Type	Description	<originalData>	Object	Original network data that is deembedded.	<frontData>	Object	Network Data that is deembedded from the front of the original.
Name	Type	Description										
<originalData>	Object	Original network data that is deembedded.										
<frontData>	Object	Network Data that is deembedded from the front of the original.										
Return Value	<p>Network IDispatch.</p> <p><b>Note:</b> DeembedFront returns <b>None</b> if the specified network data does not have compatible terminal ports defined.</p>											

Python Syntax	DeembedFront()
Python Example	<pre>oNDE = oDesktop.GetTool("ndExplorer") oData2 = oNDE.DeembedFront(oData1, oDataFront)</pre>

<b>VB Syntax</b>	DeembedFront
<b>VB Example</b>	<pre>Set oNDE = oDesktop.GetTool("ndExplorer") Set oData2 = oNDE.DeembedFront oData1, oDataFront</pre>

**Warning:** The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.

## DisableDiffPairs

Deactivates all differential pair definitions. This script does not remove them, so they [can be reactivated](#).

<b>UI Access</b>	From the <b>Network Data Explorer</b> tab, select <b>Differential Pairs</b> in the <b>NDE</b> ribbon to open the <b>Differential Pairs</b> window. Once differential pairs have been added to the rightmost box (i.e., select differential pairs from the Pairs group box and click <b>Add Pairs &gt;&gt;</b> ), remove check marks from the <b>Enabled</b> column to deactivate the corresponding differential pairs.
<b>Parameters</b>	None.
<b>Return Value</b>	None.

<b>Python Syntax</b>	DisableDiffPairs()
<b>Python Example</b>	<pre>oNDE = oDesktop.GetTool("ndExplorer") oPostProc.DisableDiffPairs()</pre>

<b>VB Syntax</b>	DisableDiffPairs
<b>VB Example</b>	<pre>Set oNDE = oDesktop.GetTool("ndExplorer") oPostProc.DisableDiffPairs</pre>

**Warning:** The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.

## EnableDiffPairs

Enables all differential pair definitions.

<b>UI Access</b>	From the <b>Network Data Explorer</b> tab, select <b>Differential Pairs</b> in the <b>NDE</b> ribbon to open the <b>Differential Pairs</b> window. Once differential pairs have been added to the rightmost box (i.e., select differential pairs from the Pairs group box and click <b>Add Pairs &gt;&gt;</b> ), add check marks to the <b>Enabled</b> column to activate the corresponding differential pairs.
<b>Parameters</b>	None.
<b>Return Value</b>	None.

<b>Python Syntax</b>	EnableDiffPairs()
<b>Python Example</b>	<pre>oNDE = oDesktop.GetTool("ndExplorer") success = oPostProc.EnableDiffPairs()</pre>

<b>VB Syntax</b>	EnableDiffPairs
------------------	-----------------

**VB Example**

```
Set oNDE = oDesktop.GetTool("ndExplorer")
Set success = oPostProc.EnableDiffPairs
```

**Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.**

## ExportCitiFile

Writes the S-parameters to disk in Citifile format (\*.cit text file).

<b>UI Access</b>	From the <b>Network Data Explorer</b> tab, select <b>File &gt; Save As</b> to open a <b>Save As</b> explorer window. Then select <b>Citifile (*.cit)</b> from the <b>Save as type:</b> drop-down menu.		
<b>Parameters</b>	<b>Name</b>	<b>Type</b>	<b>Description</b>
	<filePath>	String	The full path to the file.
	<matrixType>	Integer	<p>One of the following:</p> <ul style="list-style-type: none"> <li>• 0 – S-Parameters</li> <li>• 1 – Y-Parameters</li> <li>• 2 – Z-Parameters</li> <li>• 3 – Gamma</li> <li>• 4 – Impedance</li> </ul> <p><b>Note:</b> The default value is <b>0</b>.</p>
<formalType>		Integer	<p>One of the following:</p> <ul style="list-style-type: none"> <li>• 0 – Magnitude/Angle (degrees)</li> </ul>

			<ul style="list-style-type: none"> <li>• 1 – Real/Imaginary</li> <li>• 2 – dB/Angle (degrees)</li> </ul> <p><b>Note:</b> The default value is <b>0</b>.</p>
	<code>&lt;precision&gt;</code>	Integer	The number of significant figures.  <b>Note:</b> The default value is <b>10</b> .
	<code>&lt;frequencies&gt;</code>	Array of Doubles	A subset of the calculated frequencies. Any array values that don't have calculated data will be ignored. An empty array will cause all frequencies to be written.  <b>Note:</b> The default value is an empty array.
<b>Return Value</b>	Boolean True if file was successfully written; else False.		

<b>Python Syntax</b>	<code>ExportCitiFile()</code>
<b>Python Example</b>	<pre>oNDE = oDesktop.GetTool("ndExplorer") success = oData.ExportCitiFile("D:\folder\ne133.cit", 0, 1, 12, [])</pre>

<b>VB Syntax</b>	<code>ExportCitiFile</code>
<b>VB Example</b>	<pre>Set oNDE = oDesktop.GetTool("ndExplorer") Set success = oData.ExportCitiFile "D:\folder\ne133.cit", 0, 1, 12, Array()</pre>

## 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"
"RelativeErrorHandler:=", 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 <b>Network Data Explorer</b> tab, select <b>File &gt; Save As</b> to open a <b>Save As</b> explorer window. Then select <b>MATLAB (*.m)</b> from the <b>Save as type:</b> drop-down menu.											
Parameters	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;filePath&gt;</td> <td>String</td> <td>The full path to the file.</td> </tr> <tr> <td>&lt;matrixType&gt;</td> <td>Integer</td> <td>One of the following: <ul style="list-style-type: none"> <li>• 0 – S-Parameters</li> </ul> </td> </tr> </tbody> </table>	Name	Type	Description	<filePath>	String	The full path to the file.	<matrixType>	Integer	One of the following: <ul style="list-style-type: none"> <li>• 0 – S-Parameters</li> </ul>		
Name	Type	Description										
<filePath>	String	The full path to the file.										
<matrixType>	Integer	One of the following: <ul style="list-style-type: none"> <li>• 0 – S-Parameters</li> </ul>										

			<ul style="list-style-type: none"> <li>• 1 – Y-Parameters</li> <li>• 2 – Z-Parameters</li> <li>• 3 – Gamma</li> <li>• 4 – Impedance</li> </ul> <p><b>Note:</b> The default value is <b>0</b>.</p>
	<code>&lt;formalType&gt;</code>	Integer	<p>One of the following:</p> <ul style="list-style-type: none"> <li>• 0 – Magnitude/Angle (degrees)</li> <li>• 1 – Real/Imaginary</li> <li>• 2 – dB/Angle (degrees)</li> </ul> <p><b>Note:</b> The default value is <b>0</b>.</p>
	<code>&lt;precision&gt;</code>	Integer	<p>The number of significant figures.</p> <p><b>Note:</b> The default value is <b>10</b>.</p>
	<code>&lt;frequencies&gt;</code>	Array of Doubles	<p>A subset of the calculated frequencies. Any array values that don't have calculated data will be ignored. An empty array will cause all frequencies to be written.</p> <p><b>Note:</b> The default value is an empty array.</p>
<b>Return Value</b>	Boolean True if file was successfully written; else False.		

<b>Python Syntax</b>	ExportMatlab()
<b>Python Example</b>	<pre>oNDE = oDesktop.GetTool("ndExplorer") success = oData.ExportMatlab("D:\folder\ne133.m", 0, 1, 12, [])</pre>

<b>VB Syntax</b>	ExportMatlab
<b>VB Example</b>	<pre>Set oNDE = oDesktop.GetTool "ndExplorer" Set success = oData.ExportMatlab "D:\folder\ne133.m", 0, 1, 12, Array()</pre>

## ExportNMFDATA

*Use:* Exports s-parameters in neutral file format.

*Command:* In the **matrix** tab of the **Solution** dialog box, click **Export->S-Parameter**. Then select the **Neutral Model** file format.

*Syntax:* ExportNetworkData <SolutionName> <FileName> <ReduceMatrix><Reference Impedance> <FrequencyArray> <DesignVariation> <Format> <Length> <PassNumber>

*Return Value:* None

*Parameters:* <SolutionName>

Type: <String>

Format: <SetupName>:<SolutionName>

Solution that is exported.

<FileName>

Type:<String>

The name of the file. The file extension will determine the file format.

```
<ReduceMatrix>
  Type: <string>
    Either "Original" or one of the reduce matrix setup name

<Reference Impedance>
  Type: <Double>
  Reference impedance

<FrequencyArray>
  Type: <Array of doubles>
  Value: Array(<double>,<double>....)
  Frequency points in the sweep that is exported.

<DesignVariation>
  Type: <String>
  Design variation at which the solution is exported.

<Format>
  Type: <String>
  Values: "MagPhase", "RealImag", "DbPhase".
  The format in which the sparameters will be exported.

<Length>
  Type:<String>
  Length for exporting sparameters.
```

```
<PassNumber>
Type:<integer>
    Pass number.
```

*VB Example:*

```
oDesign.ExportNMFDATA "Setup7 : LastAdaptive", "C:/temp/neu.nmf", "Original", _ 50, Array
(10000000, 20000000, 30000000, 40000000, 50000000, 60000000, 70000000, _ 80000000, 90000000,
100000000), "", "MagPhase", "7meter"
```

#### Python Syntax

```
ExportNMFDATA(
    "",
    # Empty string.

    1,
    #The number 1.

    "Name",
    # This is the full path of the file from which the solution is loaded
    "ExportFile",

    # full path of file to export to
    ["NAME:Frequencies"],

    #optional, if none defined all frequencies are used
    ["NAME:NMFOptions"],

    #Export NMF options object
    "DataTypes:=", ["S"],
```

```
#DataTypes can be "S", "Y", "Z", "G", and "Z0", for S , Y, Z matrix, Gamma and Z0 (zero)
    "DisplayFormat:=", "MA",
#DisplayFormat "MA", "RI", "DB"
    "FileType:=", "",

# Export File Type
2 - Spreadsheet(*.tab)
3 - Touchstone(*.sNp)
4 - Citifile(*.cit)
6 - Neutral format(*.nmf)
7 - Matlab format(*.m)

"Renormalize:=", false,
#Renormalize true/false
"RefImpedance:=", 50,
# Reference Impedance
"Precision:=", 8,
# Number of digits Precision
"Variables:=", [ "FF", "cap", "Rs"]
# Array of variables
"Variations:=", [ "", "", ""]
# Array of variations to export solutions for
```

```
[ "NAME:ConstantVars"]

#Array of variables that are constant, can be empty

[ "NAME: DependentVars"]

#Array of variables that are dependent, can be empty

"MatrixSize:=", 2,
#Matrix size, optional (used in nmf file header)

"CreateNPortModel:=", true
#Create a model based on the exported file true/false

])
```

```
oTool = oDesktop.GetTool("NdExplorer")
oTool.ExportNetworkData("", True,
"C:/.../1000HM.S2P", "$SYSLIB/Test.s2p", "",

[
    "NAME:Frequencies",
    500000000,
    1000000000,
    10000000000,
    25000000000,
    50000000000,
    75000000000,
```

**Python Example**

```
100000000000  
],  
[  
    "NAME:TSOptions",  
    "DataTypes:=" , ["S"],  
    "DisplayFormat:=" , "DB",  
    "FileType:=" , 3,  
    "Renormalize:=" , True,  
    "RefImpedance:=" , 50,  
    "Precision:=" , 6,  
    "UseMultipleCores:=" , False,  
    "NumberOfCores:=" , 1,  
    "Comments:=" , True,  
    "Noise:=" , False  
])
```

## ExportNetworkData

Exports matrix solution data to a file.

UI Access	N/A
-----------	-----

Parameters	Name	Type	Description
	<DesignVariationKey>	String	Design variation key. Pass empty string for the current nominal variation.
	<SolnSelectionArray>	Array	Array of selected solutions.  Array(<SolnSelector>, <SolnSelector>, ...)  If more than one array entry, this indicates a combined Interpolating sweep.
	<SolnSelector>	String	Solution setup name and solution name, separated by a colon.
	<FileFormat>	Integer	File format value.  2 : Tab delimited spreadsheet format (.tab)  3 : Touchstone (.sNp)  4 : CitiFile (.cit)  7 : Matlab (.m)  8 : Terminal Z0 spreadsheet
	<OutFile>	String	Full path to the file to write out.
	<FreqsArray>	Array	The frequencies to export. The <FreqsArray> argument contains a vector (e.g. "1GHz", "2GHz", ...) to use, or "all". To export all frequencies, use Array("all"). If no frequencies are specified, all frequencies are used.
	<DoRenorm>	Boolean	Specifies whether to renormalize the data before export.
	<RenormImped>	Double	Real impedance value in ohms, for renormalization. Required in 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>&lt;Precision&gt;</i>	Integer	Optional. Touchstone number of digits precision. Default if not specified is 15.
	<i>&lt;UseExportFreqs&gt;</i>	Boolean	Specifies whether to use export frequencies.
	<i>&lt;IncludeGammaComments&gt;</i>	Boolean	Specifies whether to include Gamma and Impedance comments.
	<i>&lt;SupportNonStdExport&gt;</i>	Boolean	Specifies whether to support non-standard Touchstone extensions for mixed reference impedance.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	ExportNetworkData(<DesignVariationKey>, <SolnSelectionArray>, <SolnSelector>, <FileFormat>, <OutFile>, <FreqsArray>, <DoRenorm>, <RenormImped>, [Optional <DataType>], [Optional <pass>], [Optional <ComplexFormat>], [Optional <Precision>], [Optional <UseExportFreqs>], [Optional <IncludeGammaComments>], [Optional <SupportNonStdExport>])
<b>Python Example</b>	<pre>oModule.ExportNetworkData("", ["Setup1:Sweep1"], 3, "C://Documents/package_HFSSDesign1.s4p", ['all'], True, 50, "S", -1, 0, 15, True, True, True</pre>

<b>VB Syntax</b>	ExportNetworkData <DesignVariationKey>, <SolnSelectionArray>, <SolnSelector>, <FileFormat>, <OutFile>, <FreqsArray>, <DoRenorm>, <RenormImped>, [Optional <DataType>], [Optional <pass>], [Optional <ComplexFormat>], [Optional <Precision>], [Optional <UseExportFreqs>], [Optional <IncludeGammaComments>], [Optional <SupportNonStdExport>]
------------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

<b>VB Example</b> <pre> oModule.ExportNetworkData "width='2in'", _     Array("Setup1:Sweep1"), 2, "c:\mydir\out.tab", _     Array("all"), false, 0  oModule.ExportNetworkData "width='2in'", _     Array("Setup1:Sweep1", "Setup1:Sweep2"), 3, _     "c:\mydir\out.s2p", Array(1.0e9, 1.5e9, 2.0e9), _     true, 50.0 </pre>
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**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).

<b>UI Access</b>	From the <b>Network Data Explorer</b> tab, select <b>File &gt; Save As</b> to open a <b>Save As</b> explorer window. Then select <b>Data Table (spreadsheet) (*.tab)</b> from the <b>Save as type:</b> drop-down menu.											
<b>Parameters</b>	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: left; padding: 2px;"><b>Name</b></th> <th style="text-align: left; padding: 2px;"><b>Type</b></th> <th style="text-align: left; padding: 2px;"><b>Description</b></th> </tr> </thead> <tbody> <tr> <td style="padding: 2px;"><code>&lt;filePath&gt;</code></td> <td style="padding: 2px;">String</td> <td style="padding: 2px;">The full path to the file.</td> </tr> <tr> <td style="padding: 2px;"><code>&lt;matrixType&gt;</code></td> <td style="padding: 2px;">Integer</td> <td style="padding: 2px;"> One of the following: <ul style="list-style-type: none"> <li>• 0 – S-Parameters</li> <li>• 1 – Y-Parameters</li> <li>• 2 – Z-Parameters</li> <li>• 3 – Gamma</li> <li>• 4 – Impedance</li> </ul> </td> </tr> </tbody> </table>			<b>Name</b>	<b>Type</b>	<b>Description</b>	<code>&lt;filePath&gt;</code>	String	The full path to the file.	<code>&lt;matrixType&gt;</code>	Integer	One of the following: <ul style="list-style-type: none"> <li>• 0 – S-Parameters</li> <li>• 1 – Y-Parameters</li> <li>• 2 – Z-Parameters</li> <li>• 3 – Gamma</li> <li>• 4 – Impedance</li> </ul>
<b>Name</b>	<b>Type</b>	<b>Description</b>										
<code>&lt;filePath&gt;</code>	String	The full path to the file.										
<code>&lt;matrixType&gt;</code>	Integer	One of the following: <ul style="list-style-type: none"> <li>• 0 – S-Parameters</li> <li>• 1 – Y-Parameters</li> <li>• 2 – Z-Parameters</li> <li>• 3 – Gamma</li> <li>• 4 – Impedance</li> </ul>										

			<b>Note:</b> The default value is <b>0</b> .
<code>&lt;formalType&gt;</code>	Integer	One of the following:	<ul style="list-style-type: none"> <li>• 0 – Magnitude/Angle (degrees)</li> <li>• 1 – Real/Imaginary</li> <li>• 2 – dB/Angle (degrees)</li> </ul> <b>Note:</b> The default value is <b>0</b> .
<code>&lt;precision&gt;</code>	Integer	The number of significant figures.	<b>Note:</b> The default value is <b>10</b> .
<code>&lt;frequencies&gt;</code>	Array of Doubles	A subset of the calculated frequencies. Any array values that don't have calculated data will be ignored. An empty array will cause all frequencies to be written.	<b>Note:</b> The default value is an empty array.
<code>&lt;renormalize&gt;</code>	Boolean	If <b>True</b> , will renormalize the data to the value of <code>renormImpedance</code> before writing.	<b>Note:</b> The default value is <b>False</b> .
<code>&lt;renormImpedance&gt;</code>	Double	Data will be renormalized to this impedance before writing only if <code>renormalize</code> is <b>True</b> .	

			<b>Note:</b> The default value is <b>50 ohms</b> .
<b>Return Value</b>	Boolean True if file was successfully written; else False.		

<b>Python Syntax</b>	ExportSpreadsheet()
<b>Python Example</b>	<pre>oNDE = oDesktop.GetTool("ndExplorer") success = oData.ExportSpreadsheet("D:\folder\ne133.tab", 0, 1, 12, [], True, 23)</pre>

<b>VB Syntax</b>	ExportSpreadsheet
<b>VB Example</b>	<pre>Set oData=oDesktop.GetTool("ndExplorer") success = oData.ExportSpreadsheet "D:\folder\ne133.tab", 0, 1, 12, Array(), True, 23</pre>

**Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.**

## ExportTouchstone

Writes the S-parameters to disk in Touchstone 1.0 format (\*.snp text file).

<b>UI Access</b>	From the <b>Network Data Explorer</b> tab, select <b>File &gt; Save As</b> to open a <b>Save As</b> explorer window. Then select <b>Touchstone Format 1.0 (*.snp)</b> from the <b>Save as type:</b> drop-down menu.
------------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

	<b>Name</b>	<b>Type</b>	<b>Description</b>
<b>Parameters</b>	<code>&lt;filePath&gt;</code>	String	The full path to the file.
	<code>&lt;matrixType&gt;</code>	Integer	<p>One of the following:</p> <ul style="list-style-type: none"> <li>• 0 – S-Parameters</li> <li>• 1 – Y-Parameters</li> <li>• 2 – Z-Parameters</li> <li>• 3 – Gamma</li> <li>• 4 – Impedance</li> </ul> <p><b>Note:</b> The default value is <b>0</b>.</p>
	<code>&lt;formalType&gt;</code>	Integer	<p>One of the following:</p> <ul style="list-style-type: none"> <li>• 0 – Magnitude/Angle (degrees)</li> <li>• 1 – Real/Imaginary</li> <li>• 2 – dB/Angle (degrees)</li> </ul> <p><b>Note:</b> The default value is <b>0</b>.</p>
	<code>&lt;precision&gt;</code>	Integer	The number of significant figures.
	<code>&lt;frequencies&gt;</code>	Array of Doubles	A subset of the calculated frequencies. Any array values that don't have calculated data will be ignored. An empty array will cause all frequencies to be written.

			<b>Note:</b> The default value is an empty array.
	<renormalize>	Boolean	If <b>True</b> , will renormalize the data to the value of renormImpedance before writing.  <b>Note:</b> The default value is <b>False</b> .
	<renormImpedance>	Double	Data will be renormalized to this impedance before writing only if renormalize is <b>True</b> .  <b>Note:</b> The default value is <b>50 ohms</b> .
	<freqUnits>	String	One of the following: "Hz", "KHz", "MHz", "GHz".  <b>Note:</b> The default value is " <b>Hz</b> ".
<b>Return Value</b>	Boolean True if file was successfully written; else False.		

<b>Python Syntax</b>	ExportTouchstone()
<b>Python Example</b>	<pre>oNDE = oDesktop.GetTool("ndExplorer") success = oData.ExportTouchstone("D:\folder\ne133.s2p", 0, 1, 12, [], True, 23, "GHz")</pre>

<b>VB Syntax</b>	ExportTouchstone
<b>VB</b>	<pre>Set oNDE = oDesktop.GetTool("ndExplorer")</pre>

<b>Example</b>	<pre>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).

<b>UI Access</b>	From the <b>Network Data Explorer</b> tab, select <b>File &gt; Save As</b> to open a <b>Save As</b> explorer window. Then select <b>Touchstone 2 Format (*.ts)</b> from the <b>Save as type:</b> drop-down menu.		
<b>Parameters</b>	<b>Name</b>	<b>Type</b>	<b>Description</b>
	<code>&lt;filePath&gt;</code>	String	The full path to the file.

<code>&lt;matrixType&gt;</code>	Integer	One of the following: <ul style="list-style-type: none"><li>• 0 – S-Parameters</li><li>• 1 – Y-Parameters</li><li>• 2 – Z-Parameters</li><li>• 3 – Gamma</li><li>• 4 – Impedance</li></ul>
---------------------------------	---------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**Note:** The default value is **0**.

<code>&lt;formatType&gt;</code>	Integer	One of the following: <ul style="list-style-type: none"><li>• 0 – Magnitude/Angle (degrees)</li></ul>
---------------------------------	---------	-------------------------------------------------------------------------------------------------------

			<ul style="list-style-type: none"> <li>• 1 – Real/Imaginary</li> <li>• 2 – dB/Angle (degrees)</li> </ul> <p><b>Note:</b> The default value is <b>0</b>.</p>
<code>&lt;precision&gt;</code>	Integer	The number of significant figures.	<p><b>Note:</b> The default value is <b>10</b>.</p>
<code>&lt;frequencies&gt;</code>	Array of Doubles	A subset of the calculated frequencies. Any array values that don't have calculated data will be ignored. An empty array will cause all frequencies to be written.	<p><b>Note:</b> The default value is an empty array.</p>
<code>&lt;renormalize&gt;</code>	Boolean	If <b>True</b> , will renormalize the data to the value of <code>renormImpedance</code> before writing.	<p><b>Note:</b> The default value is <b>False</b>.</p>
<code>&lt;renormImpedance&gt;</code>	Double	Data will be renormalized to this impedance before writing only if <code>renormalize</code> is <b>True</b> .	<p><b>Note:</b> The default value is <b>50 ohms</b>.</p>
<code>&lt;freqUnits&gt;</code>	String	One of the following: "Hz", "KHz", "MHz", "GHz".	<p><b>Note:</b> The default value is "<b>Hz</b>".</p>
<b>Return Value</b>	Boolean True if file was successfully written; else False.		

<b>Python Syntax</b>	ExportTouchstone2()
<b>Python Example</b>	<pre>oNDE = oDesktop.GetTool("ndExplorer") success = oData.ExportTouchstone2("D:\folder\ne133.ts", 0, 1, 12, [], True, 23, "GHz")</pre>

<b>VB Syntax</b>	ExportTouchstone2
<b>VB Example</b>	<pre>Set oNDE = oDesktop.GetTool("ndExplorer") Set success = oData.ExportTouchstone2 "D:\folder\ne133.ts", 0, 1, 12, Array(), True, 23, "GHz"</pre>

**Warning:** The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.

## Extract (SPISim)

Creates a new smaller network data after removing data for the specified terminal port numbers.

**Note:** Extract opens and interacts with **SPISim** to complete.

<b>UI Access</b>	From the <b>Network Data Explorer</b> tab, select <b>Transforms</b> in the <b>NDE</b> ribbon. Then select <b>Extract</b> from the drop-down menu.
------------------	---------------------------------------------------------------------------------------------------------------------------------------------------

<b>Parameters</b>	<b>Name</b>	<b>Type</b>	<b>Description</b>
	<oData>	Object	Data from this object is copied to a new network data and the copy is transformed.
	<portNumbers>	Array of Integers	The port numbers that will be retained (integers between 1 and <i>n</i> ).
<b>Return Value</b>	Network IDispatch.		

<b>Python Syntax</b>	Extract()
<b>Python Example</b>	<pre>oNDE = oDesktop.GetTool("ndExplorer") oData2 = oNDE.Extract(oData1, [3, 2])</pre>

<b>VB Syntax</b>	Extract
<b>VB Example</b>	<pre>Set oNDE = oDesktop.GetTool "ndExplorer" Set oData2 = oNDE.Extract oData1, Array(3, 2)</pre>

**Warning:** The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.

## GetFrequencies

Returns the frequency values with calculated data in the network data.

<b>UI Access</b>	None.
<b>Parameters</b>	None.

<b>Return Value</b>	Array of doubles.
---------------------	-------------------

<b>Python Syntax</b>	GetFrequencies()
<b>Python Example</b>	<pre>oNDE = oDesktop.GetTool("ndExplorer") Freqs = oData.GetFrequencies()</pre>

<b>VB Syntax</b>	GetFrequencies
<b>VB Example</b>	<pre>Set oNDE = oDesktop.GetTool("ndExplorer") Set Freqs = oData.GetFrequencies</pre>

**Warning:** The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.

## GetFrequencyCount

Returns the number of frequencies with calculated data in the network data.

<b>UI Access</b>	None.
<b>Parameters</b>	None.
<b>Return Value</b>	Integer number of frequencies.

<b>Python Syntax</b>	GetFrequencyCount()
<b>Python Example</b>	<pre>oData = oDesktop.GetTool("ndExplorer") numFreqs = oData.GetFrequencyCount()</pre>

<b>VB Syntax</b>	GetFrequencyCount
<b>VB Example</b>	<pre>Set oData = oDesktop.GetTool("ndExplorer") Set numFreqs = oData.GetFrequencyCount</pre>

**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.

<b>UI Access</b>	None.
<b>Parameters</b>	None.
<b>Return Value</b>	String name.

<b>Python Syntax</b>	GetName()
<b>Python Example</b>	<pre>oNDE = oDesktop.GetTool("ndExplorer") strName = oData.GetName()</pre>

<b>VB Syntax</b>	GetName
<b>VB Example</b>	<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.

<b>UI Access</b>	From the <b>Network Data Explorer</b> tab, select <b>Edit Ports</b> in the <b>NDE</b> ribbon to open the <b>Port properties</b> window.
<b>Parameters</b>	None.
<b>Return Value</b>	Integer number of ports.

<b>Python Syntax</b>	GetPortCount()
<b>Python Example</b>	<pre>oNDE = oDesktop.GetTool("ndExplorer") num = oData.GetPortCount()</pre>

<b>VB Syntax</b>	GetPortCount
<b>VB Example</b>	<pre>Set oNDE = oDesktop.GetTool("ndExplorer") num = oData.GetPortCount</pre>

**Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.**

## GetPortNumber

Returns the terminal port number for a given terminal port name.

**Note:** GetPortNumber can be used in other commands that require a port number, if a port name is preferable.

<b>UI Access</b>	From the <b>Network Data Explorer</b> tab, ensure the <b>Full Port Names</b> check box is activated. Port numbers will be represented in the object portNames.								
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;portName&gt;</td> <td>String</td> <td>The designation of the port number.</td> </tr> </tbody> </table>			Name	Type	Description	<portName>	String	The designation of the port number.
Name	Type	Description							
<portName>	String	The designation of the port number.							
<b>Return Value</b>	Integer.								

<b>Python Syntax</b>	GetPortNumber()
<b>Python Example</b>	<pre>oNDE = oDesktop.GetTool("ndExplorer") index = oPostProc.GetPortNumber("Input35")</pre>

<b>VB Syntax</b>	GetPortNumber
<b>VB Example</b>	<pre>Set oNDE = oDesktop.GetTool("ndExplorer") index = oPostProc.GetPortNumber "Input35"</pre>

**Warning:** The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.

## GetPostProcSettings

Returns a copy of the IDispatch to the postprocessing settings for the network data (oPostProc).

**Note:** Making changes to the PostProcSettings will not change the network data. To make changes, call oData.SetPostProcSettings to change the network data.

<b>UI Access</b>	None.
<b>Parameters</b>	None.
<b>Return Value</b>	PostProcSettings IDispatch

<b>Python Syntax</b>	GetPostProcSettings()
<b>Python Example</b>	<pre>oNDE = oDesktop.GetTool("ndExplorer") oPostProc = oData.GetPostProcSettings()</pre>

<b>VB Syntax</b>	GetPostProcSettings
<b>VB Example</b>	<pre>Set oNDE = oDesktop.GetTool("ndExplorer") Set oPostProc = oData.GetPostProcSettings</pre>

**Warning:** The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.

## GetSolutionVariation

Returns the network data object corresponding to a variation of the solution.

<b>UI Access</b>	None.		
<b>Parameters</b>	<b>Name</b>	<b>Type</b>	<b>Description</b>
	<solutionID>	Integer	ID of the solution (obtained with LoadSolution).
<b>Return Value</b>	Network IDispatch.		

<b>Python Syntax</b>	GetSolutionVariation()
<b>Python Example</b>	<pre>oNDE = oDesktop.GetTool("ndExplorer") oData = oNDE.GetSolutionVariation("O, "offset='0.1'")</pre>

<b>VB Syntax</b>	GetSolutionVariation
<b>VB Example</b>	<pre>Set oNDE = oDesktop.GetTool("ndExplorer")</pre>

```
oData = oNDE.GetSolutionVariation "O, "offset='0.1'"
```

**Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.**

## GetVariation

Returns the variation key for the network data.

<b>UI Access</b>	None.
<b>Parameters</b>	None.
<b>Return Value</b>	String variation key.

<b>Python Syntax</b>	GetVariation()
<b>Python Example</b>	<pre>oNDE = oDesktop.GetTool("ndExplorer") variation = oData.GetVariation()</pre>

<b>VB Syntax</b>	GetVariation
<b>VB Example</b>	<pre>Set oNDE = oDesktop.GetTool("ndExplorer") Set variation = oData.GetVariation</pre>

**Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.**

## HasSameData

Compares one network data object with another network data. Actual calculated values will be compared and no interpolation will be done.

UI Access	None.		
Parameters	<b>Name</b>	<b>Type</b>	<b>Description</b>
	<NetworkData>	Object	The second network data to compare against.
	<matrixType>	Integer	<p>One of the following:</p> <ul style="list-style-type: none"> <li>• 0 – S-Parameters</li> <li>• 1 – Y-Parameters</li> <li>• 2 – Z-Parameters</li> <li>• 3 – Gamma</li> <li>• 4 – Impedance</li> </ul> <p><b>Note:</b> The default value is <b>0</b>.</p>
	<comparePortNames>	Boolean	If <b>True</b> , port names must be identical.
			<b>Note:</b> The default value is <b>True</b> .
			<compareNoise>
			If <b>True</b> , and both network data have noise data, the comparison will fail unless the noise values also match.

		<b>Note:</b> The default value is <b>10</b> .
	<relativeTolerance>	Double If the absoluteTolerance test fails, then $\text{abs}(\text{value1} - \text{value2})/\max(\text{abs}(\text{value1}), \text{abs}(\text{value2}))$ must be less than this to be considered equal.
	<absoluteTolerance>	Double If <b>True</b> , then $\text{abs}(\text{value1} - \text{value2}) < \text{absoluteTolerance}$ .
<b>Return Value</b>	Boolean True if the compared network data have the same frequency and matrix values; otherwise, False.	

<b>Python Syntax</b>	HasSameData()
<b>Python Example</b>	<pre>oNDE = oDesktop.GetTool("ndExplorer") success = oData.HasSameData(oData2, 0, True, False, 1e-10, 0)</pre>

<b>VB Syntax</b>	HasSameData
<b>VB Example</b>	<pre>Set oNDE = oDesktop.GetTool("ndExplorer") Set success = oData.HasSameData oData2, 0, True, False, 1e-10, 0</pre>

**Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.**

## LoadSolution

Loads the specified design (all variations) and returns an integer ID.

<b>UI Access</b>	After analyzing a design, from the <b>Project Manager</b> window, expand the <b>Project Tree</b> and Analysis folders. Then right-click on the completed analysis icon and select <b>Network Data Explorer</b> to open the solution in the <b>Network Data Explorer</b> tab.												
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt; projectName &gt;</td> <td>String</td> <td>Full path to the Electronics Desktop file.</td> </tr> <tr> <td>&lt; designName &gt;</td> <td>String</td> <td>Name of the design.</td> </tr> <tr> <td>&lt; solutionName &gt;</td> <td>String</td> <td>Name of the solution.</td> </tr> </tbody> </table>	Name	Type	Description	< projectName >	String	Full path to the Electronics Desktop file.	< designName >	String	Name of the design.	< solutionName >	String	Name of the solution.
Name	Type	Description											
< projectName >	String	Full path to the Electronics Desktop file.											
< designName >	String	Name of the design.											
< solutionName >	String	Name of the solution.											
<b>Return Value</b>	Integer ID (identifying the solution); < 0 if the solution is not loaded.												

<b>Python Syntax</b>	LoadSolution()
<b>Python Example</b>	<pre>oNDE = oDesktop.GetTool("ndExplorer") id = oNDE.LoadSolution("D:\folder\Tee.aedt", "HFSS_Test", "Setup1:Sweep1")</pre>

<b>VB Syntax</b>	LoadSolution
<b>VB Example</b>	<pre>Set oNDE = oDesktop.GetTool("ndExplorer")</pre>

```
id = oNDE.LoadSolution "D:\folder\Tee.aedt", "HFSS_Test", "Setup1:Sweep1"
```

**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 <b>Network Data Explorer</b> tab, select <b>Open</b> in the <b>NDE</b> ribbon, or select <b>File &gt; Open</b> to open an explorer window. Then navigate to the required S-parameter file.								
Parameters	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;fileName&gt;</td> <td>String</td> <td>Full path to the file containing S-parameters.</td> </tr> </tbody> </table>			Name	Type	Description	<fileName>	String	Full path to the file containing S-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 <b>Network Data Explorer</b> tab, select <b>Transforms</b> in the <b>NDE</b> ribbon. Then select <b>Rename</b> from the drop-down menu.		
Parameters	<b>Name</b>	<b>Type</b>	<b>Description</b>
	< <i>oData</i> >	Object	Data from this object is copied to a new network data and the copy is transformed.
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**

```
Set oNDE = oDesktop.GetTool "ndExplorer"
Set oData2 = oNDE.Rename oData1, Array("input", "control", "output")
```

**Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.**

## Renormalize (SPISim)

Creates a new network data of the same size after renormalizing the terminal ports using the specified impedance values.

**Note:** Rename opens and interacts with **SPISim** to complete.

<b>UI Access</b>	From the <b>Network Data Explorer</b> tab, select <b>Transforms</b> in the <b>NDE</b> ribbon. Then select <b>Renormalize</b> from the drop-down menu.		
<b>Parameters</b>	<b>Name</b>	<b>Type</b>	<b>Description</b>
	<oData>	Object	Data from this object is copied to a new network data and the copy is transformed.
	<portImpedances>	Array of Doubles	The new impedance values to be applied to each port.
<b>Return Value</b>	Network IDispatch.  <b>Note:</b> Renormalize returns <b>None</b> if there is not an impedance value for each port.		

<b>Python Syntax</b>	Renormalize()
<b>Python Example</b>	<pre>oNDE = oDesktop.GetTool("ndExplorer") oData2 = oNDE.Renormalize(oData1, [3, 2])</pre>

<b>VB Syntax</b>	Renormalize
<b>VB Example</b>	<pre>Set oNDE = oDesktop.GetTool("ndExplorer") oData2 = oNDE.Renormalize oData1, Array(3, 2)</pre>

**Warning:** The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.

## Reorder

Rearranges the terminal port entries but does not change the port names (i.e., switching first and third terminal port positions will give switched values for S(1,1) and S(3,3) but S(Port1,Port1) and S(Port3,Port3) do not change).

<b>UI Access</b>	From the <b>Network Data Explorer</b> tab, select <b>Edit Ports</b> in the <b>NDE</b> ribbon to open the <b>Port properties</b> window. Click+drag the terminal port rows to reorder them. Make changes, as necessary, then click <b>OK</b> .								
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;newOrder&gt;</td> <td>Array of Integers</td> <td>Must have one entry for each port; all numbers 1 to <i>n</i> must be present.</td> </tr> </tbody> </table>			Name	Type	Description	<newOrder>	Array of Integers	Must have one entry for each port; all numbers 1 to <i>n</i> must be present.
Name	Type	Description							
<newOrder>	Array of Integers	Must have one entry for each port; all numbers 1 to <i>n</i> must be present.							
<b>Return Value</b>	None.								

<b>Python Syntax</b>	Reorder()
<b>Python Example</b>	<pre>oNDE = oDesktop.GetTool("ndExplorer") oPostProc.Reorder([3, 2, 1])</pre>

<b>VB Syntax</b>	Reorder
<b>VB Example</b>	<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.

<b>UI Access</b>	From the <b>Network Data Explorer</b> tab, select <b>Transforms</b> in the <b>NDE</b> ribbon. Then select <b>Reorder</b> from the drop-down menu.								
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;oData&gt;</td> <td>Object</td> <td>Data from this object is copied to a new network data and the copy is transformed.</td> </tr> </tbody> </table>			Name	Type	Description	<oData>	Object	Data from this object is copied to a new network data and the copy is transformed.
Name	Type	Description							
<oData>	Object	Data from this object is copied to a new network data and the copy is transformed.							

	<code>&lt;portNumbers&gt;</code>	Array of Integers	The port numbers in new order (integers between 1 and $n$ ).
<b>Return Value</b>	Network IDispatch.  <b>Note:</b> Reorder returns <b>None</b> if the array argument does not contain all the terminal port numbers, in any order.		

<b>Python Syntax</b>	<code>Reorder()</code>
<b>Python Example</b>	<pre>oNDE = oDesktop.GetTool("ndExplorer") oData2 = oNDE.Reorder([3, 2, 1])</pre>

<b>VB Syntax</b>	<code>Reorder</code>
<b>VB Example</b>	<pre>Set oNDE = oDesktop.GetTool("ndExplorer") Set oData2 = oNDE.Reorder Array(3, 2, 1)</pre>

**Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.**

## Reset

Resets the post-processing to what it was when the network data was created.

**Note:** This is not equivalent to setting post-processing to an empty state. The network data may have been read from a file or created from solution data that already had post-processing settings.

<b>UI Access</b>	From the <b>Network Data Explorer</b> tab, select <b>Reset postprocessing</b> in the <b>NDE</b> ribbon.
<b>Parameters</b>	None.
<b>Return Value</b>	None.

<b>Python Syntax</b>	Reset()
<b>Python Example</b>	<pre>oNDE = oDesktop.GetTool("ndExplorer") oPostProc.Reset()</pre>

<b>VB Syntax</b>	Reset
<b>VB Example</b>	<pre>Set oNDE = oDesktop.GetTool("ndExplorer") oPostProc.Reset</pre>

**Warning:** The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.

## SetAllPortImpedances

Sets all terminal port impedances in a single call. The impedances are in an array of real or complex values.

---

<b>UI Access</b>	Through the UI, the terminal ports can <b>only</b> be edited individually. From the <b>Network Data Explorer</b> tab, select <b>Edit Ports</b> in the <b>NDE</b> ribbon to open the <b>Port properties</b> window. Make changes, then click <b>OK</b> .		
<b>Parameters</b>	<b>Name</b> <i>&lt;impedances&gt;</i>	<b>Type</b> Array of Doubles or String	<b>Description</b> Must have one entry for each port or a single complex (or real) value which will be applied to all ports.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	SetAllPortImpedances()
<b>Python Example</b>	<pre>oNDE = oDesktop.GetTool("ndExplorer") oPostProc.SetAllPortImpedances([23, "2+3i", 50]) <b>-or-</b> oData.SetAllPortImpedances(50)</pre>

<b>VB Syntax</b>	SetAllPortImpedances
<b>VB Example</b>	<pre>Set oData=oDesktop.GetTool("ndExplorer") oPostProc.SetAllPortImpedances Array(23, "2+3i", 50) <b>-or-</b> oData.SetAllPortImpedances 50</pre>

**Warning:** The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.

## SetPortDeembedDistance

Sets terminal port impedance for a single port.

<b>UI Access</b>	From the <b>Network Data Explorer</b> tab, select <b>Edit Ports</b> in the <b>NDE</b> ribbon to open the <b>Port properties</b> window. Make changes, then click <b>OK</b> .											
<b>Parameters</b>	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td><code>&lt;portNumber&gt;</code></td><td>Integer</td><td>The port number of the changed terminal port (integers between 1 and <b>n</b>).</td></tr><tr><td><code>&lt;distance&gt;</code></td><td>String</td><td>Impedance with units (e.g., "20mm"); if no units, meters are assumed</td></tr></tbody></table>			Name	Type	Description	<code>&lt;portNumber&gt;</code>	Integer	The port number of the changed terminal port (integers between 1 and <b>n</b> ).	<code>&lt;distance&gt;</code>	String	Impedance with units (e.g., "20mm"); if no units, meters are assumed
Name	Type	Description										
<code>&lt;portNumber&gt;</code>	Integer	The port number of the changed terminal port (integers between 1 and <b>n</b> ).										
<code>&lt;distance&gt;</code>	String	Impedance with units (e.g., "20mm"); if no units, meters are assumed										
<b>Return Value</b>	None.											

<b>Python Syntax</b>	<code>SetPortDeembedDistance()</code>
<b>Python Example</b>	<pre>oNDE = oDesktop.GetTool("ndExplorer") oPostProc.SetPortDeembedDistance(2, "20mm")</pre>

<b>VB Syntax</b>	<code>SetPortDeembedDistance</code>
<b>VB Example</b>	<pre>Set oNDE = oDesktop.GetTool("ndExplorer") oPostProc.SetPortDeembedDistance 2, "20mm"</pre>

**Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.**

## SetPortImpedance

Sets port impedance for a single port.

<b>UI Access</b>	From the <b>Network Data Explorer</b> tab, select <b>Edit Ports</b> in the <b>NDE</b> ribbon to open the <b>Port properties</b> window. Make changes, then click <b>OK</b> .											
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><code>&lt;portNumber&gt;</code></td> <td>Integer</td> <td>The port number of the changed terminal port (integers between 1 and <i>n</i>).</td> </tr> <tr> <td><code>&lt;impedance&gt;</code></td> <td>Double or String</td> <td>Double if impedance value is real; string format (e.g., 24+10i) for complex impedance.</td> </tr> </tbody> </table>			Name	Type	Description	<code>&lt;portNumber&gt;</code>	Integer	The port number of the changed terminal port (integers between 1 and <i>n</i> ).	<code>&lt;impedance&gt;</code>	Double or String	Double if impedance value is real; string format (e.g., 24+10i) for complex impedance.
Name	Type	Description										
<code>&lt;portNumber&gt;</code>	Integer	The port number of the changed terminal port (integers between 1 and <i>n</i> ).										
<code>&lt;impedance&gt;</code>	Double or String	Double if impedance value is real; string format (e.g., 24+10i) for complex impedance.										
<b>Return Value</b>	None.											

<b>Python Syntax</b>	<code>SetPortImpedance()</code>
<b>Python Example</b>	<pre>oNDE = oDesktop.GetTool("ndExplorer") oPostProc.SetPortImpedance(2, "25+3i")</pre>

<b>VB Syntax</b>	<code>SetPortImpedance</code>
<b>VB Example</b>	<pre>Set oNDE = oDesktop.GetTool("ndExplorer")</pre>

	<code>oPostProc.SetPortImpedance 2, "25+3i"</code>
--	----------------------------------------------------

**Warning:** The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.

## SetPostProcSettings

Applies postprocessing settings (oPostProc) to the specified network data.

**Note:** Making changes to the PostProcSettings will not change the network data. To make changes, call oData.SetPostProcSettings to change the network data.

<b>UI Access</b>	None.		
<b>Parameters</b>	<b>Name</b> <code>&lt;PostProcSettings&gt;</code>	<b>Type</b> Object	<b>Description</b> The settings that will be applied to the data.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	<code>SetPostProcSettings()</code>
<b>Python Example</b>	<pre>oNDE = oDesktop.GetTool("ndExplorer") oData.SetPostProcSettings(oPostProc)</pre>

<b>VB Syntax</b>	<code>SetPostProcSettings</code>
------------------	----------------------------------

<b>VB Example</b>	<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.

<b>UI Access</b>	From the <b>Network Data Explorer</b> tab, select <b>Transforms</b> in the <b>NDE</b> ribbon. Then select <b>Smooth</b> from the drop-down menu.														
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;oData&gt;</td> <td>Object</td> <td>Data from this object is copied to a new network data and the copy is transformed.</td> </tr> <tr> <td>&lt;order&gt;</td> <td>Integer</td> <td>The number of adjacent points that are smoothed.   <b>Note:</b> The default value is <b>10</b>.</td> </tr> <tr> <td>&lt;enforceCausality&gt;</td> <td>Boolean</td> <td>Causal data is calculated before smoothing.   <b>Note:</b> The default value is <b>False</b>.</td> </tr> </tbody> </table>	Name	Type	Description	<oData>	Object	Data from this object is copied to a new network data and the copy is transformed.	<order>	Integer	The number of adjacent points that are smoothed.  <b>Note:</b> The default value is <b>10</b> .	<enforceCausality>	Boolean	Causal data is calculated before smoothing.  <b>Note:</b> The default value is <b>False</b> .		
Name	Type	Description													
<oData>	Object	Data from this object is copied to a new network data and the copy is transformed.													
<order>	Integer	The number of adjacent points that are smoothed.  <b>Note:</b> The default value is <b>10</b> .													
<enforceCausality>	Boolean	Causal data is calculated before smoothing.  <b>Note:</b> The default value is <b>False</b> .													
<b>Return Value</b>	Network IDispatch.														

<b>Python Syntax</b>	Smooth()
<b>Python Example</b>	<pre>oNDE = oDesktop.GetTool("ndExplorer")</pre>

	<code>oData2 = oNDE.Smooth(oData1, 3, True)</code>
--	----------------------------------------------------

<b>VB Syntax</b>	Smooth
<b>VB Example</b>	<pre>Set oNDE = oDesktop.GetTool("ndExplorer") Set oData2 = oNDE.Smooth oData1, 3, True</pre>

**Warning:** The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.

## Stretch (SPISim)

Creates a new network data representing a matrix of transmission lines with the length of those transmission lines multiplied by a factor of  $n$ .

**Note:** Stretch opens and interacts with **SPISim** to complete.

<b>UI Access</b>	From the <b>Network Data Explorer</b> tab, select <b>Transforms</b> in the <b>NDE</b> ribbon. Then select <b>Stretch</b> from the drop-down menu.											
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><code>&lt;oData&gt;</code></td> <td>Object</td> <td>Data from this object is copied to a new network data and the copy is transformed.</td> </tr> <tr> <td><code>&lt;factor&gt;</code></td> <td>Double</td> <td>The transmission line lengths are multiplied by this number.</td> </tr> </tbody> </table>			Name	Type	Description	<code>&lt;oData&gt;</code>	Object	Data from this object is copied to a new network data and the copy is transformed.	<code>&lt;factor&gt;</code>	Double	The transmission line lengths are multiplied by this number.
Name	Type	Description										
<code>&lt;oData&gt;</code>	Object	Data from this object is copied to a new network data and the copy is transformed.										
<code>&lt;factor&gt;</code>	Double	The transmission line lengths are multiplied by this number.										
<b>Return Value</b>	Network IDispatch.											

**Note:** Stretch returns **None** if the network data argument does not represent a matrix of transmission lines.

<b>Python Syntax</b>	Stretch()
<b>Python Example</b>	<pre>oNDE = oDesktop.GetTool("ndExplorer") oData2 = oNDE.Stretch(oData1, 3.1)</pre>

<b>VB Syntax</b>	Stretch
<b>VB Example</b>	<pre>Set oNDE = oDesktop.GetTool("ndExplorer") Set oData2 = oNDE.Stretch oData1, 3.1</pre>

**Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.**

## Terminate

Creates a new network data with specified ports terminated.

<b>UI Access</b>	From the <b>Network Data Explorer</b> tab, select <b>Transforms</b> in the <b>NDE</b> ribbon. Then select <b>Terminate</b> from the drop-down menu.								
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;oData&gt;</td> <td>Object</td> <td>Data from this object is copied to a new network data and the copy is transformed.</td> </tr> </tbody> </table>	Name	Type	Description	<oData>	Object	Data from this object is copied to a new network data and the copy is transformed.		
Name	Type	Description							
<oData>	Object	Data from this object is copied to a new network data and the copy is transformed.							

	<code>&lt;portNumbers&gt;</code>	Array of Integers	The port numbers that are terminated (integers between 1 and <i>n</i> ).
	<code>&lt;termImpedances&gt;</code>	Array of Complex Numbers	The impedance values used to terminate the specified ports.
<b>Return Value</b>	Network IDispatch.  <b>Note:</b> Terminate returns <b>None</b> if the port numbers are not valid or there are no impedance value for the port numbers.		

<b>Python Syntax</b>	<code>Terminate()</code>
<b>Python Example</b>	<pre>oNDE = oDesktop.GetTool("ndExplorer") oData2 = oNDE.Terminate(oData1, [2, 3], [23, "10+2i"])</pre>

<b>VB Syntax</b>	<code>Terminate</code>
<b>VB Example</b>	<pre>Set oNDE = oDesktop.GetTool("ndExplorer") Set oData2 = oNDE.Terminate oData1, Array(2, 3), Array(23, "10+2i")</pre>

## 23 - ComplInstance Script Commands

ComplInstance commands should be executed by the **oDesign2** or **oPropHost2** object.

```
Set oDesign2 = ComplInstance.GetParentDesign
```

```
Set oPropHost2 = ComplInstance.GetPropHost
```

### Callback Scripting Using ComplInstance Object

Callback scripts are scripts that can be set in the Property Dialog for individual properties by clicking the button in the Callback column and choosing a script that is saved with the project. Callback scripts can contain any legal script commands including general Ansys script function calls ( e.g., GetApplicationName() ). In addition, Callback scripts can also call functions on a special object named ComplInstance.

You can obtain an interface to a ComplInstance in a schematic or layout by calling `oEditor.GetComplInstanceFromRefDes(refDes)`. For more information see [Layout Scripting](#) and Schematic Scripting. This interface is also available as a ComplInstance object in [ComplInstance event callbacks](#), such as placing a component in a layout or schematic.

#### Definitions

**<propName>** = text string

**<value>** = double

**<valueText>** = text string

**<fileName>** = full path file name

**<choices>** = string containing menu choices separated by commas

**<initialChoice>** = string containing initial choice for menu; must be one of the <choices>

**<scriptName>** = string containing name of script stored in project

**<bool>** is 1 for true or 0 for false

<editorName> is either "Layout" or "SchematicEditor"

The topics for this section include:

[ComInstance Functions](#)

## ComInstance Functions

Following are commands that can be used to manipulate properties from a ComInstance script.

The topics for this section include:

[GetComponentName](#)

GetEditor

[GetInstanceID](#)

[GetInstanceName](#)

[GetParentDesign](#)

[GetPropHost](#)

[GetPropServerName](#)

### GetComponentName

Returns the name of the component corresponding to this ComInstance.

UI Access	NA			
Parameters	<table border="1"><tr><td>Name</td><td>Type</td><td>Description</td></tr></table>	Name	Type	Description
Name	Type	Description		

	None		
<b>Return Value</b>	String name of component (e.g. MS_TRL) and stores it in "name"		

<b>Python Syntax</b>	GetComponentName()
<b>Python Example</b>	name = CompInstance.GetComponentName()

<b>VB Syntax</b>	GetComponentName()
<b>VB Example</b>	name = CompInstance.GetComponentName()

## GetInstanceID [Component Instance]

*Use:* Returns the instanceID of the ComplInstance.

*Command:* None

*Syntax:* GetInstanceID()

*Return Value:* String

*VB Example:* id = CompInstance.GetInstanceID();

Returns id of compInstance (e.g. 7) and stores it in "id".

Note that this is not the same number as the one used in the RefDes.

<b>Python Syntax</b>	GetInstanceID()
----------------------	-----------------

**Python Example**

```
id = CompInstance.GetInstanceID()
```

## **GetInstanceName [Component Instance]**

*Use:* Returns the instance name of the component corresponding to this ComplInstance.

*Command:* None

*Syntax:* GetInstanceName()

*Return Value:* String

*VB Example:* name = CompInstance.GetInstanceName();

Returns instanceName (e.g. A7) of complInstance and stores it in "name".

Note that the Instance Name is not the same as the RefDes.

## **GetParentDesign**

*Use:* Returns an interface to the complInstance's parent design.

*Command:* None

*Syntax:* GetParentDesign()

*Return Value:* Returns interface to design.

*Example:* Set oDesign2 = CompInstance.GetParentDesign();

Returns the interface to the design containing the complInstance.

This interface can be used to call Design functions. See: [Design Object Script Commands](#).

<b>Python Syntax</b>	GetParentDesign()
<b>Python Example</b>	<code>oDesign2 = CompInstance.GetParentDesign()</code>

## GetPropHost

*Use:* Returns an interface to the PropHost of the Complnstance, which gives access to its properties.

*Command:* None

*Syntax:* GetPropHost()

*VB Example:* Set oPropHost2 = CompInstance.GetPropHost();

Returns the interface to the properties of the compInstance.

This interface can be used to call PropHost functions; for more information see [Callback Scripting Using PropHost Object](#).

<b>Python Syntax</b>	GetPropHost()
<b>Python Example</b>	<code>oPropHost2 = CompInstance.GetPropHost()</code>

## GetPropServerName

*Use:* Returns the PropServerName of the Component corresponding to this Complnstance.

*Command:* None

*Syntax:* GetPropServerName()

*Return Value:* String

*VB Example:*

```
name = CompInstance.GetPropServerName();
```

Returns propserver name of compInstance (e.g., CompInst@MS\_TRL;7) and stores it in "name".

<b>Python Syntax</b>	GetPropServerName()
<b>Python Example</b>	name = CompInstance.GetPropServerName()

## 24 - 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

[AddMeshOperation](#)

[AddSweep](#)

[Analyze](#)

[AnalyzeSweep](#)

[Delete](#)

[DeleteSweep](#)

[DynamicMeshOverlays](#)

[Edit](#)

[EditSweep](#)

[ExportToHFSS](#)

[ExportToQ3D](#)

[GenerateMesh](#)

[GetAllSolutionNames](#)

[GetMeshOperations](#)

[GetSetupData](#)

[GetSetups](#)

[GetSweeps](#)

[LayoutMeshOverlay](#)

[Layout3DMeshOverlay](#)

[ListVariations](#)

[RefreshMeshOverlays](#)

Rename

[RenameSweep](#)

[Validate](#)

## AddSetup (Layout Editor)

*Use:* Adds a new setup

*Syntax:* AddSetup(STRING newsetup) // new setup name

*VB Example:* oModule. Add <newsetup>

## AddMeshOperation

Adds a mesh operation to a specified solution.

<b>UI Access</b>	From the <b>Project Manager</b> , right-click a <b>solution</b> and select <b>Assign Mesh Operation &gt; [Any Submenu Item]</b> .		
<b>Parameters</b>	Name	Type	Description

	<code>&lt;name&gt;</code>	String	The name of the solution for which you want to apply a mesh operation.
	<code>&lt;MeshArray&gt;</code>	Array	<p>Structured array:</p> <pre>Array("NAME:&lt;mesh operation name&gt;",       "RefineInside:=" , &lt;boolean True for inside; False for on surface&gt;,       "Type:=" , &lt;string "LengthBased" or "SkinDepthLengthBased"&gt;,       "Enabled:=" , &lt;boolean True to enable operation; False to disable&gt;,       Array("NAME:Assignment",             &lt;MeshEntityInfo&gt;, &lt;MeshEntityInfo&gt;,             &lt;MeshEntityInfo&gt;,...)       "Region:=" , &lt;string name of region, if selecting one&gt;,       "RestrictElem:=" , &lt;boolean True to restrict number of mesh elements&gt;,       "NumMaxElem:=" , &lt;string containing maximum number of mesh elements&gt;,       "RestrictLength:=" , &lt;boolean True to restrict length of mesh elements&gt;,       "MaxLength:=" , &lt;string containing maximum length of mesh element&gt;)</pre>
	<code>&lt;MeshEntityInfo&gt;</code>	Array	<p>Structured array. Add one MeshEntityInfo array for each entity selected.</p> <pre>Array("NAME:MeshEntityInfo",       "IsFcSel:=" , &lt;boolean&gt;,</pre>

		<pre>"EntID:=" , &lt;integer&gt;, "FcIDs:=" , &lt;array&gt;, [   "NAME:MeshBody",   "Id:=" , &lt;integer&gt;,   "Nam:=" , &lt;string&gt;,   "Layer:=" , &lt;string layer name&gt;,   "Sheet:=" , &lt;boolean&gt;,   "Net:=" , &lt;string net name or "&lt;no-net&gt;"&gt;,   "OrigNet:=" , &lt;string net name or "&lt;no-net&gt;"&gt; ], "BBox:=" , &lt;array&gt; ]</pre>
<b>Return Value</b>	None.	

<b>Python Syntax</b>	AddMeshOperation(<SolutionName>, <MeshArray>)
<b>Python Example</b>	<pre>oModule.AddMeshOperation('HFSS Solution 1', [   "NAME:Length 2",</pre>

```
"RefineInside:="           , False,
"Type:="                  , "LengthBased",
"Enabled:="                , True,
[

"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,  

```

```

    "RestrictLength:=", True,
    "MaxLength:="      , "0.2mm"
]
)

```

<b>VB Syn- tax</b>	AddMeshOperation <SolutionName> <MeshArray>
<b>VB Exam- ple</b>	<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)

**Use:** Adds a sweep to a setup

*Use:* AddSweep(STRING setup) // setup name

```
STRING newsweep) // new sweep name
```

*VB Example:*

```
oModule.AddSweep "Setup 1", _  
    Array("NAME:Sweep 1", _  
        Array("NAME:Properties", "Enable:=", "true"), _  
        "GenerateSurfaceCurrent:=", false, _  
        "FastSweep:=", false, _  
        "ZoSelected:=", false, _  
        "SAbsError:=", 0.005, _  
        "ZoPercentError:=", 1, _  
        Array("NAME:Sweeps", _  
            "Variable:=", "F", _  
            "Data:=", "LINC 1GHz 10GHz 10 LINC 10GHz 12GHz 5"))
```

## 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)

*Use:* Removes a simulation setup

*Syntax:* Delete(STRING name) // The specified setup to delete

*VB Example:* oModule. Delete <name>

## DeleteSweep (Layout Editor)

*Use:* Removes a sweep from a setup

*Syntax:* DeleteSweep(STRING setup1) // The specified setup

    STRING sweep1) // The specified sweep

*VB Example:* oModule. DeleteSweep <setup1> <sweep1>

=

## DynamicMeshOverlays (Layout Editor)

*Use:* Turns on dynamic mesh overlays

*Syntax:* DynamicMeshOverlays(STRING setup) // setup name

*VB Example:* oModule. DynamicMeshOverlay <setup>

## Edit (Layout Editor)

*Use:* Edits an excitation

*Syntax:* Edit( STRING excitation\_name ) // excitation name

ARRAY excitation\_data) // excitation data

*VB Example:* oModule. Edit <excitation\_name> <excitation\_data>

## EditSweep (Layout Editor)

*Use:* Edit a sweep

*Syntax:* EditSweep(STRING setup) // setup name

ARRAY solvesetup\_data) // setup data

*VB Example:* oModule. EditSweep <setup> <solvesetup\_data>

## 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
```

## GetMeshOperations

Get the names of all mesh operations for a given setup.

<b>UI Access</b>	N/A
<b>Parameters</b>	None.
<b>Return Value</b>	Array of names of all mesh operations.

**Python Syntax**

GetMeshOperations

<b>Python Example</b>	<pre>oModule.GetMeshOperations() res = oModule.GetMeshOperations("Setup1") ['Length 1', 'Length 2', 'SkinDepth 1']</pre>
-----------------------	--------------------------------------------------------------------------------------------------------------------------

<b>VB Syntax</b>	<b>GetMeshOperations</b>
<b>VB Example</b>	<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")

## **RefreshMeshOverlays (Layout Editor)**

*Use:* Refreshes mesh display for a setup

*Syntax:* RefreshMeshOverlays(STRING setup) // setup name

*Example:* oModule.RefreshMeshOverlays <setup>

## **RenameSweep (Layout Editor)**

*Use:* Renames a simulation setup

*Syntax:* RenameSweep(STRING oldname) // old setup name

**STRING newname) // new setup name**

*Example:* oModule. RenameSweep <oldname> <newname>

## 25 - 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")  
oModule.CommandName <args>
```

The following commands are described in this section:

[Add](#)

[AddRefPort](#)

[AddRefPortUsingEdges](#)

[AlphaNumericMatrix](#)

[ConvertPlaneToTrace](#)

[CoupleEdgePorts](#)

[DecoupleEdgePorts](#)

[Delete](#)

[DeleteExcitation](#)

[DeleteSource](#)

[DeleteProbePortAndVia](#)

[Edit](#)

[EditCircuitPort](#)

[EditExcitations](#)

[GetAllBoundariesList](#)

[GetAllPortsList](#)

[RemoveRefPort](#)

[Rename](#)

[RenameSource](#)

ReorderMatrix

[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)

*Use:* Edits an excitation

*Syntax:* Edit( STRING excitation\_name) // excitation name

```
ARRAY excitation_data) // excitation data  
VB Example: oModule.Edit <excitation_name> <excitation_data>
```

## 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.

<b>UI Access</b>	None.		
<b>Parameters</b>	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.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	EditExcitations (<exciteArray>, <ppArray>, <gapSource>, <pushExParams>))
<b>Python Example</b>	<pre>oModule.EditExcitations(     [ "NAME:Excitations",</pre>

```

        ["Port1", "0deg", "0.099V"],
        ["Port2", "0deg", "0.099V"]

    ],
    ["NAME:PostProcess",
        ["Port1", True, True, "0mm", "50ohm + 0i ohm"],
        ["Port2", True, True, "0mm", "50ohm + 0i ohm"]
    ],
    "GapSource, IncidentVoltage",
    []
)

```

<b>VB Syntax</b>	EditExcitations <exciteArray>, <ppArray>, <gapSource>, <pushExParams>
<b>VB Example</b>	<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.

## 26 - 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.

## 27 - 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

# 28 - 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**).

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](#)

[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.

[CreateCircle](#)

[CreateLine](#)

[CreatePolygon](#)

[CreateRectangle](#)

[CreateText](#)**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") )
```

## 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") )
```

## 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"))
```

## 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.

[CreateLineVoid](#)

[CreatePolygonVoid](#)

[CreateCircleVoid](#)

[CreateRectangleVoid](#)**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"))
```

## Other Creation Methods

The following creation methods are available.

[CreateComponent <...>](#)

[CreateHole <...>](#)

[CreateMeasure <...>](#)

[CreatePin <...>](#)

[CreatePinGroup <...>](#)

[CreatePinGroupPort <...>](#)

[CreateTrace <...>](#)

[CreateVia <...>](#)

[DeletePinGroup <...>](#)

## 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("<fooprint stackup layer>:<design stackup layer>", ...),  
"DrawLayers:=", Array("<fooprint 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.

## 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" ] )

## 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>, "10:=",
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")
```

## 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>, // 1<sup>st</sup> 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.Disconnect 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.

		<ul style="list-style-type: none"> <li>• "CORNER"</li> </ul>
<replace>	Boolean	True to replace original object; False to create new object.
<items>	Array	Structured array containing object to expand:  Array ("NAME:elements", " <i>&lt;Name of Object to Expand&gt;</i> ")
<b>Return Value</b>	None.	

<b>Python Syntax</b>	Expand (<expansion>, <join>, <replace>, <items>)
<b>Python Example</b>	oEditor.Expand ("2mm", "ROUND", True, ["NAME:elements", "circle_0"])

<b>VB Syntax</b>	Expand <expansion>, <join>, <replace>, <items>
<b>VB Example</b>	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](#).

<b>UI Access</b>	Draw > Expand.		
<b>Parameters</b>	Name	Type	Description

<expansion> String Expansion amount, including unit. Value can be negative to shrink an object.

<join> String Corner style. One of:

- "ROUND"

		<ul style="list-style-type: none"> <li>• "MITER"</li> <li>• "CORNER"</li> </ul>
<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>")
<b>Return Value</b>	None.	

<b>Python Syntax</b>	ExpandWithPorts (<expansion>, <join>, <replace>, <items>)
<b>Python Example</b>	oEditor.ExpandWithPorts("2mm", "ROUND", True, ["NAME:elements", "circle_0"])

<b>VB Syntax</b>	ExpandWithPorts <expansion>, <join>, <replace>, <items>
<b>VB Example</b>	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 <object\\_names>](#)

[DeactivateOpen <object\\_names>](#)

[DeactivateShort <object\\_names>](#)

[Delete \(Layout Editior\)](#)

## Activate (Layout Editor)

*Use:* Causes activation of one or more components.

*Syntax:* Activate Array("NAME:components",  
<object\_name>, // 1<sup>st</sup> component name  
<object\_name>, // 2<sup>nd</sup> 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.

[Point \(Layout Editor\)](#)

[Polygon \(Layout Editor\)](#)

[FindObjects \(Layout Editor\)](#)

[FilterObjectList \(Layout Editor\)](#)

[GetCSObjects \(Layout Editor\)](#)

[GetPolygon \(Layout Editor\)](#)

[GetPolygonDef \(Layout Editor\)](#)

[GetBBox \(Layout Editor\)](#)

[FindObjectsByPolygon \(Layout Editor\)](#)

[FindObjectsByPoint \(Layout Editor\)](#)

[CreateObjectFromPolygon \(Layout Editor\)](#)

[CreateLineFromPolygon \(Layout Editor\)](#)

### **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()

## FindObjects (Layout Editor)

Returns a list of object names using a key/value pair.

UI Access	N/A.		
Parameters	Name <code>&lt;key&gt;</code>	Type String	<p>Description</p> <p>One of:</p> <ul style="list-style-type: none"> <li>• "<b>Name</b>" to search by object name.</li> <li>• "<b>Type</b>" to search by object type.</li> </ul> <p>Valid <code>&lt;value&gt;</code> 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'</p> <ul style="list-style-type: none"> <li>• "<b>Layer</b>" to search by layer.</li> </ul> <p>An asterisk ('*') in the <code>&lt;value&gt;</code> string matches all layers. Vias and pins match using their defined layer range. 'Multi' matches (non via/pin) multi-layer objects.</p> <ul style="list-style-type: none"> <li>• "<b>Net</b>" to search by net assignment.</li> </ul>
	<code>&lt;value&gt;</code>	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](#).

<b>UI Access</b>	N/A.							
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;key&gt;</td> <td>String</td> <td> <p>One of:</p> <ul style="list-style-type: none"> <li>• "<b>Name</b>" to search by object name.</li> <li>• "<b>Type</b>" to search by object type.</li> </ul> <p>Valid &lt;value&gt; 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'</p> <ul style="list-style-type: none"> <li>• "<b>Layer</b>" to search by layer.</li> </ul> <p>An asterisk (*) in the &lt;value&gt; string matches all layers. Vias and pins match using their defined layer range. 'Multi' matches (non via/pin) multi-layer objects.</p> </td> </tr> </tbody> </table>	Name	Type	Description	<key>	String	<p>One of:</p> <ul style="list-style-type: none"> <li>• "<b>Name</b>" to search by object name.</li> <li>• "<b>Type</b>" to search by object type.</li> </ul> <p>Valid &lt;value&gt; 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'</p> <ul style="list-style-type: none"> <li>• "<b>Layer</b>" to search by layer.</li> </ul> <p>An asterisk (*) in the &lt;value&gt; string matches all layers. Vias and pins match using their defined layer range. 'Multi' matches (non via/pin) multi-layer objects.</p>	
Name	Type	Description						
<key>	String	<p>One of:</p> <ul style="list-style-type: none"> <li>• "<b>Name</b>" to search by object name.</li> <li>• "<b>Type</b>" to search by object type.</li> </ul> <p>Valid &lt;value&gt; 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'</p> <ul style="list-style-type: none"> <li>• "<b>Layer</b>" to search by layer.</li> </ul> <p>An asterisk (*) in the &lt;value&gt; string matches all layers. Vias and pins match using their defined layer range. 'Multi' matches (non via/pin) multi-layer objects.</p>						

		<ul style="list-style-type: none"> <li>• "Net" to search by net assignment.</li> </ul>
<value>	String	Specific query. Queries use a basic string match. For instance, an asterisk (*) matches anything. All matching is case insensitive.
<b>Return Value</b>	Filtered list of object names.	

<b>Python Syntax</b>	FilterObjectList (<key>, <value>)
<b>Python Example</b>	RFIN_pins = oLayout.FilterObjectList('Type', 'Pin', oLayout.FindObjects('Net', 'RFIN'))

<b>VB Syntax</b>	FilterObjectList <key>, <value>
<b>VB Example</b>	<pre>Set objectlist = oEditor.FindObjects("Net", "RFIN") Set RFIN_pins = oEditor.FilterObjectList("Type", "Pin", objectlist)</pre>

### 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')

### **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')

### **FindObjectByPolygon (Layout Editor)**

*Use:* Get a list of all objects intersecting the specified Polygon object on the given layer.

*Syntax:* oLayout.FindObjectsByPolygon(<oPolygogn>, <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, '*')
```

### **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')
```

### **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')
```

### 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')
```

### 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.AreEqual(oPointOther)  
isEqual2 = oPointThis.AreEqual(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.

<b>UI Access</b>	N/A.		
<b>Parameters</b>	Name	Type	Description
	<factor>	Integer	Scaling factor.
	<centerPoint>	Object	Any oPoint object. See: <a href="#">Point Object</a> .
<b>Return Value</b>	Polygon object.		

<b>Python Syntax</b>	Scale(<factor>, <centerPoint>)
<b>Python Example</b>	<code>oPolygon.Scale(2, oPoint)</code>

<b>VB Syntax</b>	Scale <factor>, <centerPoint>
<b>VB Example</b>	<code>oPolygon.Scale 2, oPoint</code>

### 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 bouding 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.

[Unite](#)

[Intersect](#)

[SplitRegion](#)

[Subtract](#)

## 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

Split geometry against a polygon region.

<b>UI Access</b>	After the polygon is created and the design and polygon are selected, right-click and select <b>Draw &gt; Split Polygon Region</b> .
<b>Parameters</b>	None.
<b>Return Value</b>	"Name : Poly_#"

<b>Python Syntax</b>	SplitRegion
<b>Python Example</b>	<pre>oEditor.SplitRegion (     [         "NAME:ExampleName",         "Poly_1"     ] )</pre>

<b>VB Syntax</b>	SplitRegion
<b>VB Example</b>	<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")
```

## Coordinate System Methods

The following scripts are available for use with the Designer Coordinate System.

[CreateCS \(Layout Editor\)](#)

[ClearRelative \(Layout Editor\)](#)

[Create3DStructure \(Layout Editor\)](#)

[Group \(Layout Editor\)](#)

[CreateGroupSelected \(Layout Editor\)](#)

[PositionRelative \(Layout Editor\)](#)

[GetCSObjects \(Layout Editor\)](#)

[SetCS \(Layout Editor\)](#)

### **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">, "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))
```

## 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")

## **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")
```

## 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")

## 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 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 **IC Editor Script Commands** section is divided into two subsections.

### IC Mode Script Commands

---

The following commands can only be executed from the **IC** mode environment.

[ChangeActiveLayout \(IC Editor\)](#)

[CreateViaGroupsOnLayer \(IC Editor\)](#)

[ReconstructArcs \(IC Editor\)](#)

[ReconstructArcsOnLayer \(IC Editor\)](#)

[RemoveSmall Holes \(IC Editor\)](#)

[RemoveSmallLines \(IC Editor\)](#)

[RemoveSmallMetallIslands \(IC Editor\)](#)

[SnapPrimitives \(IC Editor\)](#)

[SnapPrimitivesOnLayer \(IC Editor\)](#)

[SnapVias \(IC Editor\)](#)

[SnapViasOnLayer \(IC Editor\)](#)

[UnitePrimitivesOnLayer \(IC Editor\)](#)

[WrapGeometries \(IC Editor\)](#)

[WrapGeometry \(IC Editor\)](#)

### **IC Mode Commands 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 \(Layout Editor\)](#)  
[DuplicateAcrossLyrs \(Layout Editor\)](#)  
[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 \(Layout Editor\)](#)  
[GetPropertyValues \(Layout Editor\)](#)

---

[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 \(Layout Editor\)](#)

## IC Mode Script Commands

The following commands can only be executed from the **IC** mode environment.

[ChangeActiveLayout \(IC Editor\)](#)  
[CreateViaGroupsOnLayer \(IC Editor\)](#)  
[ReconstructArcs \(IC Editor\)](#)  
[ReconstructArcsOnLayer \(IC Editor\)](#)  
[RemoveSmall Holes \(IC Editor\)](#)  
[RemoveSmallLines \(IC Editor\)](#)

---

[RemoveSmallMetallIslands \(IC Editor\)](#)

[SnapPrimitives \(IC Editor\)](#)

[SnapPrimitivesOnLayer \(IC Editor\)](#)

[SnapVias \(IC Editor\)](#)

[SnapViasOnLayer \(IC Editor\)](#)

[UnitePrimitivesOnLayer \(IC Editor\)](#)

[WrapGeometries \(IC Editor\)](#)

[WrapGeometry \(IC Editor\)](#)

### **ChangeActiveLayout (IC Editor)**

Switches the active layout.

<b>UI Access</b>	Navigate to the <b>Cells</b> window > <b>Hierarchical</b> tab. Then double-click the appropriate instance in the hierarchical tree.		
<b>Parameters</b>	Name	Type	Description
	<Parameters>	Array	Structured array containing these parameters in order: <ul style="list-style-type: none"><li>• &lt;NAME:CellInstances&gt;</li><li>• &lt;SelectedLayout&gt;</li></ul>
	<NAME:CellInstances>	String	Required first parameter which must have the specified value (i.e., "CellInstances").
<b>Return Value</b>	None		

<b>Python Syntax</b>	ChangeActiveLayout([<NAME:CellInstances>, <SelectedLayout>])
<b>Python Example</b>	<pre> oEditor.ChangeActiveLayout (     [         "NAME:CellInstances",         "0",         "1"     ] ) </pre>

<b>VB Syntax</b>	ChangeActiveLayout <NAME:CellInstances>, <SelectedLayout>
<b>VB Example</b>	<pre> oEditor.ChangeActiveLayout Array(     "NAME:CellInstances",     "0",     "1" ) </pre>

### CreateViaGroupsOnLayer (IC Editor)

Creates a via group on the target layer.

<b>UI Access</b>	From the <b>Layers</b> window, right-click the target layer and select <b>Group Vias &gt; Persistent or Non-Persistent</b> .								
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;NAME&gt;</td> <td>String</td> <td>Name of the target layer.</td> </tr> </tbody> </table>			Name	Type	Description	<NAME>	String	Name of the target layer.
Name	Type	Description							
<NAME>	String	Name of the target layer.							

	<i>&lt;Parameters&gt;</i>	Array	Structured array containing these parameters in order: <ul style="list-style-type: none"> <li>• &lt;NAME:Setting Contents&gt;</li> <li>• &lt;Group_By_Proximity&gt;</li> <li>• &lt;Via_Grouping_Tolerance&gt;</li> <li>• &lt;Check_Contentment&gt;</li> <li>• &lt;Via_Persistence&gt;</li> </ul>
	<i>&lt;NAME:Setting Contents&gt;</i>	String	Required first parameter which must have the specified value (i.e., "Setting Contents").
	<i>&lt;Group_By_Proximity&gt;</i>	Boolean	<b>True</b> groups via by proximity. <b>False</b> groups by range.
	<i>&lt;Via_Grouping_Tolerance&gt;</i>	Integer	When <i>&lt;Group_By_Proximity&gt;</i> = <b>False</b> , vias at or within this tolerance are grouped.
	<i>&lt;Check_Contentment&gt;</i>	Boolean	<b>True</b> analyzes the placement of via primitives. When the possibility of an electrical short is detected, <i>&lt;Check_Contentment&gt;</i> attempts to prevent the short. <b>False</b> deactivates the check.
	<i>&lt;Via_Persistence&gt;</i>	Boolean	<b>True</b> maintains the new via group. <b>False</b> selects the group of vias but does not maintain a new via group.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	CreateViaGroupsOnLayer(<NAME>, [NAME:Setting Contents, <Group_By_Proximity>, <Via_Grouping_Tolerance>, <Check_Contentment>, <Via_Persistence>])
<b>Python Example</b>	oEditor.CreateViaGroupsOnLayer (

```
[ "Trace",
    "NAME:Setting Contents",
    "group_by_proximity:=", "True",
    "via_grouping_tolerance:=", "100um",
    "checkContainment:=", "True",
    "via_persistence:=", "False"
] )
```

<b>VB Syntax</b>	CreateViaGroupsOnLayer <NAME>, (<NAME:Setting Contents>, <Group_By_Proximity>, <Via_Grouping_Tolerance>, <Check_Contentment>, <Via_Persistence>)
<b>VB Example</b>	oEditor.CreateViaGroupsOnLayer "Trace", Array("NAME:Setting Contents", "group_by_proximity:=", true, "via_grouping_tolerance:=", "100um", "checkContainment:=", true, "via_persistence:=", false)

### ReconstructArcs (IC Editor)

Fits arcs and circles to the selected polygons.

<b>UI Access</b>	After choosing polygons, select <b>Reconstruct Arcs</b> from the <b>Layout</b> ribbon.		
<b>Parameters</b>	Name <Parameters>	Type Array	Description Structured array containing these parameters in order: <ul style="list-style-type: none"><li>• &lt;NAME:polygons&gt;</li><li>• &lt;Objects&gt;</li></ul>

		<ul style="list-style-type: none"> <li>• &lt;Arcs_Tolerance&gt;</li> </ul>
<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)
<Arcs_Tolerance>	Integer	Arcs and circles at or within this tolerance are fitted.
<b>Return Value</b>	None.	

<b>Python Syntax</b>	ReconstructArcs([<NAME:polygons>, <Objects>], <Arcs_Tolerance>)
<b>Python Example</b>	<pre> oEditor.ReconstructArcs( [     "NAME:polygons",     "poly_30", "poly_31", "poly_32" ], 0.00254) </pre>

<b>VB Syntax</b>	ReconstructArcs (<NAME:polygons>, <Objects>), <Arcs_Tolerance>
<b>VB Example</b>	<pre> oEditor.ReconstructArcs Array("NAME:polygons", "poly_30", "poly_31", "poly_32"), 0.00254 </pre>

**ReconstructArcsOnLayer (IC Editor)**

Fits arcs and circles to the selected polygons on a target layer.

<b>UI Access</b>	After choosing polygons, navigate to the <b>Layers</b> window. Then right-click the target layer and select <b>Reconstruct Arcs</b> .		
<b>Parameters</b>	Name	Type	Description
	<NAME>	String	Name of the target layer.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	ReconstructArcsOnLayer(<NAME>, <Arcs_Tolerance>)
<b>Python Example</b>	<pre> oEditor.ReconstructArcsOnLayer(     "Trace",     0.00254) </pre>

<b>VB Syntax</b>	ReconstructArcsOnLayer <NAME>, <Arcs_Tolerance>
<b>VB Example</b>	<pre> oEditor.ReconstructArcs "Trace", 0.00254 </pre>

**RemoveSmallHoles (IC Editor)**

Removes small holes at or within tolerance from the target layer.

<b>UI Access</b>	Select <b>Remove Holes</b> from the <b>Layout</b> ribbon.
------------------	-----------------------------------------------------------

	Name	Type	Description
<b>Parameters</b>	<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: <ul style="list-style-type: none"> <li>• &lt;NAME:Nets&gt;</li> <li>• &lt;SelectedLayout&gt;</li> </ul>
	<NAME:Nets>	String	Required first parameter which must have the specified value (i.e., "Nets").
	<NetName>	String	Name of the target net.
<b>Return Value</b>	None		

<b>Python Syntax</b>	AddObject(<ObjectType>, <LayerName>)
<b>Python Example</b>	<pre> oEditor.RemoveSmallHoles("1000um2", "1um", [     "NAME:Nets",     "net1" ]) </pre>

<b>VB Syntax</b>	AddObject <ObjectType>, <LayerName>
<b>VB Example</b>	<pre> oEditor.RemoveSmallHoles "1000um2", "1um", Array("NAME:Nets", "net1") </pre>

**RemoveSmallLines (IC Editor)**

Removes small lines at or within tolerance from the target layer.

<b>UI Access</b>	From the <b>Layers</b> window, right-click the target layer and select <b>Remove Small Lines</b> .		
<b>Parameters</b>	Name <NAME>	Type String	Description Name of the target layer.
	<Max_Length_Tolerance>	Integer	Lines at or within this tolerance are removed.
<b>Return Value</b>	None		

<b>Python Syntax</b>	RemoveSmallLines(<NAME>, <Max_Length_Tolerance>)
<b>Python Example</b>	<code>oEditor.RemoveSmallLines ("Trace", "1mm")</code>

<b>VB Syntax</b>	RemoveSmallLines <NAME>, <Max_Length_Tolerance>
<b>VB Example</b>	<code>oEditor.RemoveSmallLines "Trace", "1mm"</code>

**RemoveSmallMetalsIslands (IC Editor)**

Removes primitives and vias at or within tolerance from the target layer.

<b>UI Access</b>	After choosing primitives, select <b>Remove Islands</b> from the <b>Layout Ribbon</b> .
------------------	-----------------------------------------------------------------------------------------

<b>Parameters</b>	Name	Type	Description
	<NAME>	String	Name of the target layer.
<Max_Area_Tolerance>		Integer	Primitives and vias at or within this tolerance are removed.
<b>Return Value</b>	None		

<b>Python Syntax</b>	RemoveSmallMetallIslands(<NAME>, <Max_Area_Tolerance>)
<b>Python Example</b>	<code>oEditor.RemoveSmallMetalIslands("Trace", "1mm")</code>

<b>VB Syntax</b>	RemoveSmallMetallIslands <NAME>, <Max_Area_Tolerance>
<b>VB Example</b>	<code>oEditor.RemoveSmallMetalIslands "Trace", "1mm"</code>

### SnapPrimitives (IC Editor)

Aligns the edge of the selected primitives.

<b>UI Access</b>	After choosing primitives, select <b>Snap Primitives</b> from the <b>Layout</b> ribbon.											
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;LAYER_NAME&gt;</td> <td>String</td> <td>Name of the target layer.</td> </tr> <tr> <td>&lt;Parameters&gt;</td> <td>Array</td> <td>Structured array containing these parameters in order:           <ul style="list-style-type: none"> <li>• &lt;NAME:primitives&gt;</li> <li>• &lt;Objects&gt;</li> </ul> </td> </tr> </tbody> </table>	Name	Type	Description	<LAYER_NAME>	String	Name of the target layer.	<Parameters>	Array	Structured array containing these parameters in order: <ul style="list-style-type: none"> <li>• &lt;NAME:primitives&gt;</li> <li>• &lt;Objects&gt;</li> </ul>		
Name	Type	Description										
<LAYER_NAME>	String	Name of the target layer.										
<Parameters>	Array	Structured array containing these parameters in order: <ul style="list-style-type: none"> <li>• &lt;NAME:primitives&gt;</li> <li>• &lt;Objects&gt;</li> </ul>										

			<ul style="list-style-type: none"> <li>• &lt;Snap_Dis_Tolerance&gt;</li> <li>• &lt;CheckConnectivity&gt;</li> </ul>
<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	<b>True</b> analyzes the placement of primitives to ensure snapping does not violate the connectivity of the original objects. <b>False</b> deactivates the check.	
<b>Return Value</b>	None.		

<b>Python Syntax</b>	SnapPrimitives(<LAYER_NAME> [<NAME:primitives>, <Objects>], <Snap_Dis_Tolerance>, <CheckConnectivity>)
<b>Python Example</b>	<pre>oEditor.SnapPrimitives(     ["Trace",         "NAME:primitives",         "rect_0:=",         "rect_1",         "rect_2",     ], "5mm", True)</pre>

<b>VB Syn-</b>	SnapPrimitives <LAYER_NAME> (<NAME:primitives>, <Objects>), <Snap_Dis_Tolerance>, <CheckConnectivity>
----------------	-------------------------------------------------------------------------------------------------------

<b>tax</b>	
<b>VB Example</b>	<pre>oEditor.SnapPrimitives "Trace", Array("NAME:primitives", "rect_0", "rect_1", "rect_2"), "5mm", True</pre>

**SnapPrimitivesOnLayer (IC Editor)**

Aligns the edge of the selected primitives on the target layer.

<b>UI Access</b>	From the <b>Layers</b> window, right-click the target layer and select <b>Snap Primitives</b> .												
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;LAYER_NAME&gt;</td> <td>String</td> <td>Name of the target layer.</td> </tr> <tr> <td>&lt;Snap_Dis_Tolerance&gt;</td> <td>Integer</td> <td>Primitives at or within this tolerance are snapped.</td> </tr> <tr> <td>&lt;CheckConnectivity&gt;</td> <td>Boolean</td> <td><b>True</b> analyzes the placement of primitives to ensure snapping does not violate the connectivity of the original objects. <b>False</b> deactivates the check.</td> </tr> </tbody> </table>	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	<b>True</b> analyzes the placement of primitives to ensure snapping does not violate the connectivity of the original objects. <b>False</b> deactivates the check.
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	<b>True</b> analyzes the placement of primitives to ensure snapping does not violate the connectivity of the original objects. <b>False</b> deactivates the check.											
<b>Return Value</b>	None.												

<b>Python Syntax</b>	<code>SnapPrimitivesOnLayer(&lt;LAYER_NAME&gt;, &lt;Snap_Dis_Tolerance&gt;, &lt;CheckConnectivity&gt;)</code>
<b>Python Example</b>	<pre>oEditor.SnapPrimitivesOnLayer("Trace", "5mm", True)</pre>

<b>VB Syntax</b>	<code>SnapPrimitivesOnLayer &lt;LAYER_NAME&gt;, &lt;Snap_Dis_Tolerance&gt;, &lt;CheckConnectivity&gt;</code>
<b>VB Example</b>	<pre>oEditor.SnapPrimitivesOnLayer "Trace", "1mm", True</pre>

## SnapVias (IC Editor)

Forces the edges of selected vias to snap to alignment.

UI Access	After choosing vias, select <b>Snap Vias</b> from the <b>Layout</b> ribbon.		
Parameters	<b>Name</b>	<b>Type</b>	<b>Description</b>
	<Parameters>	Array	Structured array containing these parameters in order: <ul style="list-style-type: none"> <li>• &lt;NAME:vias&gt;</li> <li>• &lt;Objects&gt;</li> </ul>
	<NAME:vias>	String	Required first parameter which must have the specified value (i.e., "vias").
	<Objects>	List	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: <ul style="list-style-type: none"> <li>• &lt;NAME:Setting Contents&gt;</li> <li>• &lt;IsFactorBased&gt;</li> <li>• &lt;SnapTol&gt;</li> <li>• &lt;AlignTol&gt;</li> <li>• &lt;RemoveDanglingVias&gt;</li> </ul>
	<NAME:Setting Contents>	String	Required first parameter which must have the specified value (i.e., "Setting Contents").
	<IsFactorBased>	Boolean	<b>True</b> 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. <b>False</b> 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.

	<code>&lt;RemoveDanglingVias&gt;</code>	Boolean	<b>True</b> removes vias from selection that have no connectivity or have connectivity only to a single layer. <b>False</b> deactivates this check.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	<code>SnapVias([&lt;NAME:vias&gt;, &lt;Objects&gt;], [&lt;NAME:Setting Contents&gt;, &lt;IsFactorBased&gt;, &lt;SnapTol&gt;, &lt;AlignTol&gt;, &lt;RemoveDanglingVias&gt;])</code>
<b>Python Example</b>	<pre>oEditor.SnapVias(     [         "NAME:vias",         "via_0",         "via_1",         "via_2"     ],     [         "NAME:Setting Contents",         "IsFactorBased:=", True,         "snapTol:=", "3",         "AlignTol:=", "300um",         "removeDanglingVias:=", True     ])</pre>

	])
--	----

<b>VB Syntax</b>	SnapVias (<NAME:vias>, <Objects>), (<NAME:Setting Contents>, <IsFactorBased>, <SnapTol>, <AlignTol>, <RemoveDanglingVias>)
<b>VB Example</b>	<code>oEditor.SnapVias Array("Name:vias", "via_0", "via_1", "via_2"), Array(NAME:Setting Contents", "IsFactorBased:=", True, "snapTol:=", "3", "AlignTol:=", "300um", "removeDanglingVias:=", True)</code>

### SnapViasOnLayer (IC Editor)

Forces the edges of selected vias to snap to alignment on the target layer.

<b>UI Access</b>	From the <b>Layers</b> window, right-click the target layer and select <b>Snap Vias</b> .		
<b>Parameters</b>	<b>Name</b>	<b>Type</b>	<b>Description</b>
	<Layer_Name>	String	Name of the target layer.
	<Parameters>	Array	Structured array containing these parameters in order: <ul style="list-style-type: none"> <li>&lt;NAME:Setting Contents&gt;</li> <li>&lt;IsFactorBased&gt;</li> <li>&lt;SnapTol&gt;</li> <li>&lt;AlignTol&gt;</li> <li>&lt;RemoveDanglingVias&gt;</li> </ul>
	NAME:Setting Contents	Array	Required first parameter which must have the specified value (i.e., "Setting Contents").
	IsFactorBased	Array	<b>True</b> 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.

			<b>False</b> selected via groups are snapped by a maximum distance of the tolerance.
	<i>SnapTol</i>	Array	Vias at or within this tolerance are snapped.
	<i>AlignTol</i>	Array	Vias at or within this tolerance are aligned.
	<i>RemoveDanglingVias</i>	Array	<b>True</b> removes vias from selection that have no connectivity or have connectivity only to a single layer. <b>False</b> deactivates this check.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	<code>SnapVias(&lt;Layer_Name&gt;, [&lt;NAME:Setting Contents&gt;, &lt;IsFactorBased&gt;, &lt;SnapTol&gt;, &lt;AlignTol&gt;, &lt;RemoveDanglingVias&gt;])</code>
<b>Python Example</b>	<pre> oEditor.SnapViasOnLayer(     ["Trace",         "NAME:Setting Contents",         "IsFactorBased:=", False,         "snapTol:=", "5mm",         "AlignTol:=", "5mm",         "removeDanglingVias:=", True     ] ) </pre>

<b>VB Syn-</b>	<code>SnapVias &lt;Layer_Name&gt;, (&lt;NAME:Setting Contents&gt;, &lt;IsFactorBased&gt;, &lt;SnapTol&gt;, &lt;AlignTol&gt;, &lt;RemoveDanglingVias&gt;)</code>
----------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------

<b>tax</b>	
<b>VB Example</b>	<pre>oEditor.SnapViasOnLayer "Trace", Array("NAME:Setting Contents", "IsFactorBased:=", False, "snapTol:=", "5mm", "AlignTol:=", "5mm", "removeDanglingVias:=", True)</pre>

### UnitePrimitivesOnLayer (IC Editor)

Unites primitives on the target layer.

<b>UI Access</b>	From the <b>Layers</b> window, right-click the target layer and select <b>Unite Primitives</b> .								
<b>Parameters</b>	<table border="1"> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> <tr> <td>&lt;NAME&gt;</td> <td>String</td> <td>Name of the target layer.</td> </tr> </table>			Name	Type	Description	<NAME>	String	Name of the target layer.
Name	Type	Description							
<NAME>	String	Name of the target layer.							
<b>Return Value</b>	None.								

<b>Python Syntax</b>	UnitePrimitivesOnLayer(<NAME>)
<b>Python Example</b>	<pre>oEditor.UnitePrimitivesOnLayer ("Trace")</pre>

<b>VB Syntax</b>	UnitePrimitivesOnLayer<NAME>
<b>VB Example</b>	<pre>oEditor.UnitePrimitivesOnLayer "Trace"</pre>

### WrapGeometries (IC Editor)

Wraps the selected polygons.

<b>UI Access</b>	After choosing the appropriate geometries, select <b>Wrap Geometry</b> from the <b>Layout</b> ribbon.
------------------	-------------------------------------------------------------------------------------------------------

	Name	Type	Description
<b>Parameters</b>	<Parameters>	Array	Structured array containing these parameters in order: <ul style="list-style-type: none"><li>• &lt;NAME:Contents&gt;</li><li>• &lt;Elements&gt;</li><li>• &lt;Objects&gt;</li></ul>
	<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)
<b>Return Value</b>	None.		

<b>Python Syntax</b>	WrapGeometries ([<NAME:Contents>, Elements, Objects])
<b>Python Example</b>	<pre> oEditor.WrapGeometries( [     "NAME:Contents",     "elements:=", ["rect_2577", "rect_2578", "rect_2579"] ]) </pre>

<b>VB Syntax</b>	WrapGeometries (<NAME:Contents>, Elements, Objects)
<b>VB Example</b>	<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.

<b>UI Access</b>	From the <b>Layout</b> ribbon, select <b>Wrap Geometry</b> . Right-click the selected geometry within the design and Adjust the tolerance slider as necessary. Then click <b>Wrap</b> .		
<b>Parameters</b>	Name	Type	Description
	<SEED_OBJECT_NAME>	String	Name of the selected polygon.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	WrapGeometry (<SEED_OBJECT_NAME>, <Snap_Dis_Tolerance>)
<b>Python Example</b>	<pre>oEditor.WrapGeometry("rect_1855", "5um")</pre>

<b>VB Syntax</b>	WrapGeometry <SEED_OBJECT_NAME> <Snap_Dis_Tolerance>
<b>VB Example</b>	<pre>oEditor.WrapGeometry "rect_1855", "5um"</pre>

## IC Mode Commands 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 \(Layout Editor\)](#)

---

[DuplicateAcrossLyr \(Layout Editor\)](#)

[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 \(Layout Editor\)](#)

## **AddObject (Layout Editor and IC Editor)**

Adds a user-selected object to the active design.

<b>UI Access</b>	Not strictly applicable, although objects can be drawn by making the appropriate selection from the <b>Layout</b> ribbon (i.e., <b>Draw polygon</b> , <b>Draw line</b> , <b>Draw rectangle</b> , <b>Draw arc</b> , <b>Draw circle</b> ).		
<b>Parameters</b>	Name <i>&lt;ObjectType&gt;</i>	Type String	Description Type of object to add (i.e., rect (i.e., rectangle), circle, poly (i.e., polygon), line, or 3D line).
	<i>&lt;Parameters&gt;</i>	Array	Structured array containing the parameters required for the type of object selected. The parameters differ for each object. Refer to the following pages: <ul style="list-style-type: none"> <li>• "CreateRectangle (Layout Editor)" on page 30-61</li> <li>• "CreateCircle (Layout Editor)" on page 30-35</li> <li>• "CreatePolygon (Layout Editor)" on page 30-57</li> <li>• "CreateLine (Layout Editor)" on page 30-47</li> <li>• "AddObject ("3D Line") (Layout Editor)" on page 28-146</li> </ul>
<b>Return Value</b>	Returns the name of the newly created object ( <b>IC Mode</b> only).		

<b>Python Syntax</b>	AddObject(<ObjectType>, [<Parameters>])
<b>Python Example</b>	<pre> oEditor.AddObject("rect", [     "NAME:Contents",     "rectGeometry:=", [         "Name:=", "rect_5",         "LayerName:=", "Trace",         "lw:=", "0",     ] ] ) </pre>

```
    "Ax:=", "-84mm",
    "Ay:=", "16mm",
    "Bx:=", "-77mm",
    "By:=", "12mm"]
)
```

<b>VB Syntax</b>	AddObject(<ObjectType>, [<Parameters>])
<b>VB Example</b>	<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>

### AddPinGroupRefPort (Layout Editor)

Assigns a reference port to a pin group.

**Note:**

AddPinGroupRefPort is also called when the command "CreatePinGroupPort (Layout Editor)" on page 30-57 is used.

<b>UI Access</b>	From <b>Layout</b> , select <b>Draw Ports/Sources</b> .		
<b>Parameters</b>	Name	Type	Description
	<PinGroup>	Array	Array consisting of the name of a pingroup that requires a reference port (e.g., PinGroup_2).
<b>Return Value</b>	None		

<b>Python Syntax</b>	AddPinGroupRefPort ([<PinGroup>], [<PortRef>])
<b>Python Example</b>	oEditor.AddPinGroupRefPort(["PinGroup_2"], ["PinGroup_1"])

<b>VB Syntax</b>	AddPinGroupRefPort (<PinGroup>), (<PortRef>)
<b>VB Example</b>	oEditor.AddPinGroupRefPort Array("PinGroup_2"), Array("PinGroup_1")

**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))
```

### **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"

### 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	Create or change a layer in the <b>Edit Layers</b> window.		
Parameters	Name <LayerArray>	Type Array	Description An array that contains these parameters in order: <ul style="list-style-type: none"><li>• &lt;mode&gt;</li><li>• &lt;Layername&gt;</li><li>• &lt;ID&gt;</li><li>• &lt;Type&gt;</li><li>• &lt;TopBottom&gt;</li><li>• &lt;Color&gt;</li><li>• &lt;Transparency&gt;</li><li>• &lt;Pattern&gt;</li><li>• &lt;VisFlag&gt;</li><li>• &lt;Locked&gt;</li></ul>

		<ul style="list-style-type: none"> <li>• &lt;Draw&gt;</li> <li>• &lt;Subarray&gt;</li> </ul>
<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	<p>Sets visibility for all objects on the layer using the following additive code:</p> <ul style="list-style-type: none"> <li>• 1 makes primitives (rectangles, circles, polygons) visible.</li> <li>• 2 makes lines (also known as paths or traces) visible.</li> <li>• 4 makes pads (from padstacks, aka vias) visible.</li> <li>• 8 makes holes (or vias) visible.</li> <li>• 16 makes components visible.</li> <li>• 32 makes the mesh overlay visible if they are plotted.</li> <li>• 64 makes the mesh for the background material visible.</li> </ul> <p>For example, a value of 25 (1+8+16) makes primitives, vias, and components</p>

		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	<p><i>Optional.</i> An array that contains these parameters in order:</p> <ul style="list-style-type: none"> <li>• &lt;Thickness&gt;</li> <li>• &lt;LowerElevation&gt;</li> <li>• &lt;Roughness&gt;</li> <li>• &lt;BotRoughness&gt;</li> <li>• &lt;SideRoughness&gt;</li> <li>• &lt;Material&gt;</li> <li>• &lt;FillMaterial&gt;</li> </ul> <p>This array's placeholder does not appear in the syntax or examples below but instead has its contents listed.</p>
<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.
<b>Return Value</b>	None	

<b>Python Syntax</b>	<pre>oEditor.ChangeLayer(&lt;LayerArray&gt;)</pre>
<b>Python Example</b>	<pre>oEditor = oDesign.SetActiveEditor("Layout") oEditor.ChangeLayer(     ["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",         ],     ], )</pre>

```

        "SideRoughness:="      , "0um",
        "Material:="          , "MQ",
        "FillMaterial:="       , "FR4_epoxy"
    ]
)

```

<b>VB Syntax</b>	<code>oEditor.ChangeLayer &lt;LayerArray&gt;</code>
<b>VB Example</b>	<pre> Set oEditor = oDesign.SetActiveEditor("Layout") 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 (Layout 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 <b>Edit Layers</b> window.		
<b>Parameters</b>	Name	Type	Description
	<LayerArray>	Array	An array that contains these parameters in order: <ul style="list-style-type: none"> <li>• &lt;Layername&gt;</li> <li>• &lt;ID&gt;</li> <li>• &lt;Type&gt;</li> <li>• &lt;TopBottom&gt;</li> <li>• &lt;Color&gt;</li> <li>• &lt;Transparency&gt;</li> <li>• &lt;Pattern&gt;</li> <li>• &lt;VisFlag&gt;</li> <li>• &lt;Locked&gt;</li> <li>• &lt;Draw&gt;</li> <li>• &lt;Subarray&gt;</li> <li>• &lt;Neg&gt;</li> </ul>
	<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	<p>Sets visibility for all objects on the layer using the following additive code:</p> <ul style="list-style-type: none"> <li>• 1 makes primitives (rectangles, circles, polygons) visible.</li> <li>• 2 makes lines (also known as paths or traces) visible.</li> <li>• 4 makes pads (from padstacks, aka vias) visible.</li> <li>• 8 makes holes (or vias) visible.</li> <li>• 16 makes components visible.</li> <li>• 32 makes the mesh overlay visible if they are plotted.</li> <li>• 64 makes the mesh for the background material visible.</li> </ul> <p>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.</p>
<Locked>	Boolean	<i>Optional.</i> Whether the layer is locked.
<Draw>	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.
<Subarray>	Array	<i>Optional.</i> An array that contains these parameters in order: <ul style="list-style-type: none"> <li>• &lt;Thickness&gt;</li> <li>• &lt;LowerElevation&gt;</li> <li>• &lt;Roughness&gt;</li> </ul>

		<ul style="list-style-type: none"> <li>• &lt;BotRoughness&gt;</li> <li>• &lt;SideRoughness&gt;</li> <li>• &lt;Material&gt;</li> <li>• &lt;FillMaterial&gt;</li> </ul> <p>This array's placeholder does not appear in the syntax or examples below but instead has its contents listed.</p>
<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.
<b>Return Value</b>	None	

<b>Python Syntax</b>	<pre> oEditor.ChangeLayers(     [         "NAME:layers",         "Mode:=", "&lt;mode&gt;",         [             "NAME:pps"         ]     ] ) </pre>
----------------------	------------------------------------------------------------------------------------------------------------------------------------------------------

```
        ],
        [
            "NAME:stackup layer"
            "Name:=", "<LayerName>",
            "ID:=", <ID>,
            "Type:=", "<Type>",
            "Top Bottom:=", "<TopBottom>",
            "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>",
            ]
        ]
    ]
```

	<pre>"FillMaterial:=", "&lt;FillMaterial&gt;" ], "Neg:=", &lt;Neg&gt; ] ])</pre>
<b>Python Example</b>	<pre>oEditor = oDesign.SetActiveEditor("Layout") oEditor.ChangeLayers( [     "NAME:layers",     "Mode:=", "Laminate",     [         "NAME:pps"     ],     [         "NAME:stackup layer",         "Name:=", "Cover",         "ID:=", 10,         "Type:=", "signal",         "Top Bottom:=", "neither",         "Color:=", 16711680,</pre>

```
    "Transparency:=", 0,
    "Pattern:=", 1,
    "VisFlag:=", 127,
    "Locked:=", False,
    "DrawOverride:=", 0,
    [
        "NAME:Sublayer",
        "Thickness:=", "0mil",
        "LowerElevation:=", "20mil",
        "Roughness:=", "0um",
        "BotRoughness:=", "0um",
        "SideRoughness:=", "0um",
        "Material:=", "gold",
        "FillMaterial:=", "FR4_epoxy"
    ],
    "Neg:=", True
]
])
```

**VB Syntax**

```
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-
```

	arence:=", <Transparency>, "Pattern:=", <Pattern>, "VisFlag:=", <VisFlag>, "Locked:=", <Locked>, "DrawOverride:=", <Draw>, Array("NAME:Sublayer", "Thickness:=", "<Thickness>", "LowerElevation:=", "<LowerElevation>", "Roughness:=", "<Roughness>", "BotRoughness:=", "<BotRoughness>", "SideRoughness:=", "<SideRoughness>", "Material:=", <Material>,"FillMaterial:=", "<FillMaterial>"),"Neg:=", <Neg>))
<b>VB Example</b>	Set oEditor = oDesign.SetActiveEditor("Layout")  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:=", _ false, "DrawOverride:=", 0, Array("NAME:Sublayer", "Thickness:=", _ "1.6mm", "LowerElevation:=", "0mm", "Roughness:=", "0um", "BotRoughness:=", _ "0um", "SideRoughness:=", "0um", "Material:=", "FR4_epoxy"),"Neg:=", True))

## 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" <sup>1</sup> "DrawingResolution" <sup>2</sup>
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" <sup>3</sup> "MinorColor" <sup>3</sup> "MajorSize" <sup>2</sup> "MinorSize" <sup>2</sup> "ShowGrid" <sup>1</sup>
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" <sup>4</sup>
Background color	Color of the layout background	white	"background color" <sup>3</sup>
Sketch Mode	Fill patterns and center lines are not drawn.	off	"DefaultToSketchMode" <sup>1</sup>
Solid Mode	Objects are filled in with solid color.	off	"fillMode" <sup>1</sup>
Draw Con-	Display pin symbols in the layout.	off	"show connection points" <sup>1</sup>

nection Points			
Draw Rats	Display lines indicating missing physical connections in nets	on	"draw rats" <sup>1</sup>
Label Vertices	Display property text labeling the vertices of selected polygons and lines.	off	"display vertex labels" <sup>1</sup>
Net Color	Color used for highlighting nets	light yellow	"NetColor" <sup>3</sup>
Rectangle Description	Specifies if rectangles are described with two points or center point, width, and height	2 points	"rectangle description" <sup>5</sup>
Default Hole Size	Default size for actual holes in a PC board	25 mil	"defaultholesize" <sup>2</sup>
Align Microwave Components	Snaps components on entry.	on	"snaptoport" <sup>1</sup>
Symbol Footprint Scaling	Used to size symbol footprints placed in layout.	Based on data extent and current length units.	"sym footprint scaling" <sup>6</sup>
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" <sup>3</sup> "secondary selection color" <sup>3</sup>
Preview Selection	Highlight the object that would be selected with a left mouse button click in the current location.	off	"preview selection" <sup>1</sup>
Snapping options	Choose snapping sources and targets and the maximum distance (in pixels) for snapping to occur. Constrain edits to grid if	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	"PixelSnapTolerance" <sup>7</sup> "SnapTargetVertex_on" <sup>1</sup> "SnapTargetEdgeCenter_on" <sup>1</sup>

	desired.  <b>Note:</b> Off-grid objects are ignored if ConstrainToGrid is asserted.	grid.	"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

## 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>,
          <NewPropsArray>,
          <ChangedPropsArray>,
          <DeletedPropsArray>)
```

```
<PropServersArray>
```

```
    Array ("Name:PropServers", <PropServer>,
          <PropServer>, ...)
```

```
<NewPropsArray>
    Array ("Name:NewProps", <PropdataArray>,
           <PropdataArray>, ...)

<ChangedPropsArray>
    Array ("Name:ChangedProps", <PropdataArray>,
           <PropdataArray>, ...)

<DeletedPropsArray>
    Array ("Name:DeletedProps", <PropertyName>,
           <PropertyName>, ...)

<PropdataArray>
    Array ("NAME:<PropertyName>",
           "PropType:=", <PropType>,
           "NewName:=", <string>,
           "Description:=", <string>,
           "NewRowPosition:=", <int>,
           "ReadOnly:=", <bool>,
           "Hidden:=", <bool>,
```

```
<PropTypeSpecificArgs>
```

```
<PropType>
```

Type: string

Identifies the type of property when a new property is added. In HFSS and Maxwell, only separator properties and variable properties can be added.

```
"SeparatorProp"
```

```
"VariableProp"
```

```
"TextProp"
```

```
"NumberProp"
```

```
"ValueProp"
```

```
"CheckboxProp"
```

```
"MenuProp"
```

```
"PointProp"
```

```
"VPointProp"
```

```
"V3DPointProp"
```

```
"ButtonProp"
```

**NewName**

Specify the new name of a property if the property's name is being edited. In HFSS and Maxwell, the name can only be changed for separators and variables.

#### **Description**

Specify a description of the property. In HFSS and Maxwell, the description can only be changed for separators and variables.

#### **NewRowPosition**

Used to reorder rows in the **Property** dialog box.

In HFSS, this only applies to the **Project>Project Variables** panel and the **Hfss>DesignProperties** panel.

In Maxwell, this only applies to the **Project>Project Variables** panel and the **Maxwell3D>DesignProperties**, **Maxwell2D>Design Properties**, or **RMxprt>Design Properties** panels. Specify the new zero-based row index of the variable or separator.

#### **ReadOnly**

Used to mark a property as "read only" so it can not be modified. In HFSS and Maxwell, this flag can only be set for variables and separators.

#### **Hidden**

Used to hide a property so it can not be viewed outside of the **Property** dialog box. In HFSS and Maxwell, this flag can only be set for variables and separators.

#### **<PropTypeSpecificArgs>**

SeparatorProp: no arguments

```
TextProp: "Value:=", <string>
NumberProp: "Value:=", <double>
ValueProp: "Value:=", <value>
CheckboxProp: "Value:=", <bool>
MenuProp: "Value:=", <string>
PointProp "X:=", <double>, "Y:=", <double>
VPointProp: "X:=", <value>, "Y:=", <value>
V3DPointProp: "X:=", <value>, "Y:=", <value>,
               "Z:=", <value>
Material Button: "Material:=", <string>
Color Button: "R:=", <int>, "G:=", <int>, "B:=", <int>
Transparency Button: "Value:=", <double>
```

<PropTypeSpecificArgs> for VariableProps

**Syntax:**

```
"Value:=", <value>, <OptimizationFlagsArray>,
<TuningFlagsArray>, <SensitivityFlagsArray>,
<StatisticsFlagsArray>
```

**Parameters:**

```
<OptimizationFlagsArray>
Array("NAME:Optimization",
"Included:=", <bool>,
"Min:=", <value>,
"Max:=", <value>)
```

```
<Tuning_flagsArray>
Array("NAME:Tuning",
"Included:=", <bool>,
"Step:=", <value>,
"Min:=", <value>,
"Max:=", <value>)
```

```
<SensitivityFlagsArray>
Array("NAME:Sensitivity",
"Included:=", <bool>,
"Min:=", <value>,
"Max:=", <value>,
"IDisp:=", <value> )
```

```
<StatisticsFlagsArray>
```

```
Array("NAME:Statistical",
"Included:=", <bool>,
"Dist:=", <Distribution>,
"StdD:=", <value>,
"Min:=", <value>,
"Max:=", <value>,
"Tol:=", <string>)
```

**<Distribution>**

Type: string

Value should be "Gaussian" or "Uniform"

**StdD**

Standard deviation.

**Min**

Low cut-off for the distribution.

**Max**

High cut-off for the distribution.

**Tol**

Tolerance for uniform distributions. Format is "<int>%".

Example: "20%".

*VB Example:* Adding a new project level variable "\$width":

```
oProject.ChangeProperty Array("NAME:AllTabs",_
    Array("NAME:ProjectVariableTab",_
        Array("NAME:PropServers", "ProjectVariables"),_
        Array("NAME:NewProps",_
            Array("NAME:$width",_
                "PropType:=", "VariableProp",_
                "Value:=", "3mm",_
                "Description:=", "my new variable"))))
```

*VB Example:* Deleting the design level variable "height":

```
oDesign.ChangeProperty Array("NAME:AllTabs",_
    Array("NAME:LocalVariableTab",_
        Array("NAME:PropServers", "DefinitionParameters"),_
```

```
        Array("NAME:DeletedProps", "height"))
```

*VB Example:* Changing a property's value. If the following command were executed, then the value of the property "XSize" of the PropServer "Box1:CreateBox:1" on the "Geometry3DCmdTab" tab would be changed. (oEditor is the Geometry3D editor in HFSS.)

```
oEditor.ChangeProperty Array("NAME:AllTabs", _  
    Array("NAME:Geometry3DCmdTab", _  
        Array("NAME:PropServers","Box1:CreateBox:1"),_  
        Array("NAME:ChangedProps",_  
            Array("NAME:XSize", "Value:=", "1.4mil"))))
```

*VB Example:* Changing the Company Name, Design Name, the background color, and the Axis scaling in a Report.

```
Set oProject = oDesktop.SetActiveProject("wgcombiner")  
Set oDesign = oProject.SetActiveDesign("HFSSDesign2")  
Set oModule = oDesign.GetModule("ReportSetup")  
  
oModule.ChangeProperty Array("NAME:AllTabs", Array("NAME:Header", _ Array("NAME:PropServers", "XY  
Plot1:Header"), _  
    Array("NAME:ChangedProps", Array("NAME:Company Name", _  
        "Value:=", "My Company"))))
```

```
oModule.ChangeProperty Array("NAME:AllTabs", Array("NAME:Header", _ Array("NAME:PropServers", "XY Plot1:Header"), _
Array("NAME:ChangedProps", Array("NAME:Design Name", _
"Value:=", "WG Combiner")))

oModule.ChangeProperty Array("NAME:AllTabs", Array("NAME:General", _ Array("NAME:PropServers",
"XY Plot1:General"), _
Array("NAME:ChangedProps", Array("NAME:Back Color", _
"R:=", 128, "G:=", 255, "B:=", 255)))

oModule.ChangeProperty Array("NAME:AllTabs", Array("NAME:Axis", _ Array("NAME:PropServers", "XY Plot1:AxisX"), _
Array("NAME:ChangedProps", Array("NAME:Axis Scaling", _
"Value:=", "Log"))))
```

### **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", ...)

## 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.

<b>UI Access</b>	From the <b>View</b> menu, select <b>Clip Plane</b> , or from the <b>View</b> ribbon, select <b>Add Clipping Plane</b> .
<b>Parameters</b>	None.
<b>Return Value</b>	None.

<b>Python Syntax</b>	ClipPlane()
<b>Python Example</b>	<pre>oEditor = oDesign.SetActiveEditor("Layout") Editor.ClipPlane ("n1, n2, n3, et cetera") success = oEditor.ClipPlane("n1, n2, n3, et cetera")</pre>

<b>VB Syntax</b>	ClipPlane
<b>VB Example</b>	<pre>oEditor = oDesign.SetActiveEditor("Layout" Editor.ClipPlane "n1, n2, n3, et cetera" Set success = oEditor.ClipPlane "n1, n2, n3, et cetera"</pre>

## 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.

<b>UI Access</b>	After selecting one or more primitives, navigate to <b>Draw &gt; Convert &gt; Primitives to Vias</b> .		
<b>Parameters</b>	Name	Type	Description
	<Parameters>	Array	Structured array containing these parameters in order: <ul style="list-style-type: none"> <li>• &lt;NAME:elements&gt;</li> <li>• &lt;TargetPrimitives&gt;</li> </ul>
	<NAME:elements>	String	Required first parameter which must have the specified value (i.e., "elements").
<b>Return Value</b>	None.		

<b>Python Syntax</b>	ConvertPrimitivestoVias([<NAME:elements>, <TargetPrimitives>])
<b>Python Example</b>	<pre> oEditor.ConvertPrimitivestoVias(     [         "NAME:elements",         "rect_395"     ] ) </pre>

<b>VB Syntax</b>	ConvertPrimitivestoVias(<NAME:elements>, <TargetPrimitives>)
<b>VB Example</b>	<code>oEditor.ConvertPrimitivestoVias Array("NAME:elements", "rect_395")</code>

### Copy (Layout Editor)

*Use:* Copies the referenced objects to the clipboard.

*Syntax:* **Copy** Array (<object\_name>, // 1<sup>st</sup> object

<object\_name>, // 2<sup>nd</sup> object

*Return Value:* Returns the name of the new net.

*VB Example:*

```
oEditor.Copy Array("circle_0", "rect_2")
```

### AddObject ("3D Line") (Layout Editor)

Adds a 3D line to the active design.

<b>UI Access</b>	From the <b>Layout</b> ribbon, select <b>Draw 3D line</b> .		
<b>Parameters</b>	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: <ul style="list-style-type: none"><li>• &lt;NAME:Contents&gt;</li><li>• &lt;3D LineGeometry&gt;</li></ul>
	<NAME:Contents>	String	Identifies the contents of the array (i.e., "Contents").
	<	Array	Structured array containing these parameters in order:

	<code>3DLineGeometry&gt;</code>	<ul style="list-style-type: none"> <li>• &lt;Name&gt;</li> <li>• &lt;LayerName&gt;</li> <li>• &lt;lw&gt;</li> <li>• &lt;gm&gt;</li> </ul>
	<code>&lt;Name&gt;</code>	String Name of the new object.
	<code>&lt;LayerName&gt;</code>	String Name of the target layer.
	<code>&lt;lw&gt;</code>	Integer Line width of the new object.
	<code>&lt;gm&gt;</code>	Structured array containing these parameters in order: <ul style="list-style-type: none"> <li>• &lt;m&gt;</li> <li>• &lt;pnt&gt;</li> </ul>
	<code>&lt;m&gt;</code>	Integer "Placement Mode" - species one of the following: <b>0</b> 3D - points are placed according to user-specified XYZ coordinates. <b>1</b> Top - points are placed on top of the target layer. <b>2</b> Middle - points are placed in the middle of the target layer. <b>3</b> Bottom - points are placed on bottom of the target layer.
	<code>&lt;pnt&gt;</code>	Structured array containing these parameters in order: <ul style="list-style-type: none"> <li>• &lt;lid&gt;</li> <li>• &lt;pos&gt;</li> </ul>
	<code>&lt;lid&gt;</code>	Integer Identifies the layer identification number on which the 3D line's position will be affixed (e.g., "2").
	<code>&lt;pos&gt;</code>	Structured array containing these parameters in order: <ul style="list-style-type: none"> <li>• &lt;X&gt;</li> <li>• &lt;Y&gt;</li> <li>• &lt;Z&gt;</li> </ul>
	<code>&lt;X&gt;</code>	Integer X coordinate of the new object (e.g., -0.08mm).
	<code>&lt;Y&gt;</code>	Integer X coordinate of the new object (e.g., 0.03mm).
	<code>&lt;Z&gt;</code>	Integer Y coordinate of the new object (e.g., 0mm).

<b>Return Value</b>	Returns the name of the newly created 3D line.
<b>Python Syntax</b>	<pre>AddObject(&lt;ObjectType&gt;, [&lt;Parameters&gt;])</pre>
<b>Python Example</b>	<pre>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:=",</pre>

```

        "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:=" ,  

        "0.03mm", "Z:=" , "0"]]  

    ]  

])

```

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"))
```

### **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)
```

### 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("<fooprint stackup layer>:<design stackup layer>", ...),  
"DrawLayers:=", Array("<fooprint 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.

### CreateDifferentialPair (Layout Editor)

Creates a differential pair from two target nets.

<b>UI Access</b>	From the <b>Nets</b> window, <b>Ctrl+click</b> two nets from the list. Then right-click within the <b>Nets</b> window and select <b>Create Differential Pair</b> .		
<b>Parameters</b>	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: <ul style="list-style-type: none"><li>• &lt;Net1&gt;</li><li>• &lt;Net2&gt;</li></ul>
	<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).
<b>Return Value</b>	None		

<b>Python Syntax</b>	CreateDifferentialPair(<DiffPairName>, <Description>, [<Net1>, <Net2>])
<b>Python Example</b>	<code>oEditor.CreateDifferentialPair("DiffPair1", "", ["GND", "Port1"])</code>

<b>VB Syntax</b>	CreateDifferentialPair <DiffPairName>, <Description>, (<Net1>, <Net2>)
<b>VB Example</b>	<code>oEditor.CreateDifferentialPair "DiffPair1", "", Array("GND", "Port1")</code>

### CreateEdgePort (Layout 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:* 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)

*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)*

### **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

UnmiterredJoin = 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')
```

### 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')
```

### 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 <b>Layout Editor</b> and select <b>Port &gt; Create Pin Group</b> .		
Parameters	Name <code>&lt;Parameters&gt;</code>	Type Array	Description Structured array containing these parameters in order: <ul style="list-style-type: none"><li>• <code>&lt;NAME:PinGroupDatas&gt;</code></li><li>• <code>&lt;PinGroup1&gt;</code></li></ul>
	<code>&lt;NAME:PinGroupDatas&gt;</code>	String	Required first parameter which must have the specified value (i.e., "PinGroupDatas").
	<code>&lt;PinGroup1&gt;</code>	Array	Structured array containing these parameters in order, followed by <code>&lt;PinGroup2&gt;</code> , <code>&lt;PinGroup3&gt;</code> , etcetera: <ul style="list-style-type: none"><li>• <code>&lt;NAME:&gt;</code></li></ul>

		<ul style="list-style-type: none"> <li>• &lt;PinNames&gt;</li> </ul>
<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.
<b>Return Value</b>	None.	

<b>Python Syntax</b>	CreatePinGroups([<NAME:PinGroupDatas>, [<NAME:>, <PinNames>]])
<b>Python Example</b>	<pre> oEditor.CreatePinGroups ( [     "NAME:PinGroupDatas",     ["NAME:PinGroup_3",         "via_2",         "via_1"     ] ]) </pre>

<b>VB Syntax</b>	CreatePinGroups(<NAME:PinGroupDatas>, (<NAME:>, <PinNames>))
<b>VB Example</b>	<pre> oEditor.CreatePinGroups ("NAME:PinGroupDatas", ("NAME:PinGroup_3", "via_2", "via_1")) </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") )
```

### 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 <b>Layout Editor</b> and select <b>Port &gt; Create Ports On Component</b> to open the <b>Create Ports</b> window. Select a net from the <b>Net Filter</b> area, then click <b>OK</b> to close the <b>Create Ports</b> window.								
Parameters	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;Parameters&gt;</td> <td>Array</td> <td>Structured array containing these parameters in order: • &lt;NAME:Components&gt;</td> </tr> </tbody> </table>			Name	Type	Description	<Parameters>	Array	Structured array containing these parameters in order: • <NAME:Components>
Name	Type	Description							
<Parameters>	Array	Structured array containing these parameters in order: • <NAME:Components>							

		<ul style="list-style-type: none"> <li>• &lt;ComponentNames&gt;</li> </ul>
<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: <ul style="list-style-type: none"> <li>• &lt;NAME:Nets&gt;</li> <li>• &lt;NetNames&gt;</li> </ul>
<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.
<b>Return Value</b>	None.	

<b>Python Syntax</b>	CreatePortsOnComponentsByNet([<NAME:Components>, <ComponentNames>], [<NAME:Nets>, <NetNames>], <Type>, <R>, <L>, <C>)
<b>Python Example</b>	<pre> oEditor.CreatePortsOnComponentsByNet (     [         "NAME:Components",         "U0"     ] ) </pre>

```

    "U1"
],
[
  "NAME:Nets",
  "trace"
], "Port", "0", "0", "0")

```

<b>VB Syntax</b>	CreatePortsOnComponentsByNet Array(<NAME:Components>, <ComponentNames>), Array(<NAME:Nets>, <NetNames>), <Type>, <R>, <L>, <C>
<b>VB Example</b>	oEditor.CreatePortsOnComponentsByNet ("NAME:Components", "U0", "U1"), ("NAME:Nets", "trace"), "Port", "0", "0", "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"))
```

**CreateViaGroups (Layout Editor)**

Creates via groups on the target layer.

UI Access	After selecting two or more vias, navigate to <b>Draw &gt; Via Groups &gt; Create &gt; Persistent/Non-Persistent</b> .		
Parameters	Name <i>&lt;Parameters&gt;</i>	Type Array	Description Structured array containing these parameters in order: <ul style="list-style-type: none"> <li>• <i>&lt;NAME:vias&gt;</i></li> <li>• <i>&lt;TargetObjects&gt;</i></li> <li>• <i>&lt;GroupName&gt;</i></li> <li>• <i>&lt;Enable&gt;</i></li> </ul>
	<i>&lt;NAME:vias&gt;</i>	String	Required first parameter which must have the specified value (i.e., "vias").
	<i>&lt;TargetObjects&gt;</i>	String	Names of the target primitives to add to a via group, separated by commas.
	<i>&lt;GroupName&gt;</i>	String	Names of the new via group.
	<i>&lt;Enable&gt;</i>	Boolean	<b>True</b> creates persistent via group. <b>False</b> 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", ] ) </pre>

```
"rect_825",
"ViaGroup_1"
], False)
```

<b>VB Syntax</b>	CreateViaGroups(<NAME:vias>, <TargetObjects>, <GroupName>), <Enable>
<b>VB Example</b>	oEditor.CreateViaGroups Array("NAME:vias", "circle_8651", "rect_825", "ViaGroup_1"), False

### 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.

### 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.

<b>UI Access</b>	None.		
<b>Parameters</b>	Name <netName>	Type String	Description Name of unwanted net.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	DeleteNets ([<netName>, ...])
<b>Python Example</b>	<pre>oEditor.DeleteNets (     [         "NET_1",         "NET_2",         "NET_3"     ])</pre>

<b>VB Syntax</b>	DeleteNets Array (<netName>, ...)
<b>VB Example</b>	<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")

**DelNetClass**

*Use:* Delete a net class in a layout.

*Command:* DelNetClass

*Syntax:* DelNetClass (<names>)

*Return Value:* None

*VB Example:* oEditor.DelNetClass Array ("power")

**DissolveComponents (Layout Editor)**

Dissolves the named components.

<b>UI Access</b>	After selecting one or more vias, right-click within the Layout Editor and select <b>Components &gt; Dissolve</b> .		
<b>Parameters</b>	Name	Type	Description
	<Parameters>	Array	Structured array containing these parameters in order: <ul style="list-style-type: none"><li>• &lt;NAME:elements&gt;</li><li>• &lt;TargetObjects&gt;</li></ul>
	<NAME:elements>	String	Required first parameter which must have the specified value (i.e., "vias").
<b>Return Value</b>	None.		

**Python Syntax**

<b>Python Syntax</b>	DissolveComponents([<NAME:elements>, <TargetObjects>])
----------------------	--------------------------------------------------------

**Python Example**

```
oEditor.DissolveComponents ( [ "NAME:elements",  
    "poly_801",  
    "ViaGroup_2"  
]  
)
```

**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)

### 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" ))
```

## EditComponent (Layout Editor)

Modifies a current source.

Edt the selected component to add or remove pins

<b>UI Access</b>	After choosing a component, navigate to the <b>Properties</b> window. From the <b>Pins</b> row, select <b>Edit</b> to open the <b>Edit Component</b> window. From the <b>Edit Component</b> window, remove pins as necessary.		
<b>Parameters</b>	Name	Type	Description
	<Parameters>	Array	Structured array containing these parameters in order: <ul style="list-style-type: none"> <li>• &lt;NAME:Contents&gt;</li> <li>• &lt;definition_name&gt;</li> <li>• &lt;ref_des:=&gt;</li> <li>• &lt;add&gt;</li> <li>• &lt;modified_elements&gt;</li> </ul>
	<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	<b>True</b> adds pins <b>False</b> removes pins
<modified_elements>	Array	Structured array containing the names of pins to add or remove (e.g., via_246, via_247).	
<b>Return Value</b>	None.		

<b>Python Syntax</b>	EditComponent([<NAME:Contents>, <definition_name>, <ref_des>, <add>, <modified_elements>])
<b>Python Example</b>	<pre>oEditor.EditComponent (     [         "NAME:Contents",         "definition_name:=", "Component_0",         "ref_des:=", "U0",         "add:=", "False",         "modified_elements", ["via_246", "via_247"]     ] )</pre>

<b>VB Syntax</b>	EditComponent (<NAME:Contents>, <definition_name>, <ref_des>, <add>, <modified_elements>)
<b>VB Example</b>	<pre>oEditor.EditComponent("NAME:Contents", "definition_name:=", "Component_0", "ref_des:=", "U0", "add:=", "False", "modified_elements", ("via_246", "via_247"))</pre>

### EditPinGroup (Layout Editor)

Allows the user to edit the target pin group.

<b>UI Access</b>	From <b>Draw</b> , select <b>Create/Manage Pin Groups</b> to open the <b>Create/Manage Pin Groups</b> window. Select a
------------------	------------------------------------------------------------------------------------------------------------------------

	group from the <b>Pin Group List</b> , then click <b>Edit Pin Group</b> to open the <b>Edit Pin Group</b> window. Check the adjacent box next to any pins to remove them, then click <b>Delete Pins</b> . Click <b>OK</b> to close the <b>Edit Pin Group</b> window.		
<b>Parameters</b>	Name <i>&lt;NAME&gt;</i>	Type String	Description Name of the target pin group.
	<i>&lt;Parameters1&gt;</i>	Array	Structured array containing these parameters in order: <ul style="list-style-type: none"><li>• <i>&lt;NAME:elements_remove&gt;</i></li><li>• <i>&lt;PinsRemove&gt;</i></li></ul>
	<i>&lt;NAME:elements_remove&gt;</i>	String	Required first parameter which must have the specified value (i.e., "elements_remove").
	<i>&lt;PinsRemove&gt;</i>	String	Names of the pins to remove from the group, separated by commas.
	<i>&lt;Parameters2&gt;</i>	Array	Structured array containing these parameters in order: <ul style="list-style-type: none"><li>• <i>&lt;NAME:elements_add&gt;</i></li><li>• <i>&lt;PinsAdd&gt;</i></li></ul>
	<i>&lt;NAME:elements_add&gt;</i>	String	Required first parameter which must have the specified value (i.e., "elements_add").
<b>Return Value</b>	None.		

<b>Python Syntax</b>	EditPinGroup(<NAME>, [<NAME:elements_remove>, <PinsRemove>], [<NAME:elements_add>, <PinsAdd>])
<b>Python Example</b>	<pre>oEditor.EditPinGroup ("GND_30_SQFP28X28_208_U1",                       ["NAME:elements_remove",                        "U1-50",                        "U1-82"],</pre>

	<pre>[ "Name:elements_add",   "U1-51" ])</pre>
--	------------------------------------------------

<b>VB Syntax</b>	EditPinGroup <NAME>, (<NAME:elements_remove>, <PinsRemove>), (<NAME:elements_add>, <PinsAdd>)
<b>VB Example</b>	oEditor.EditPinGroup "GND_30_SQFP28X28_208_U1", Array("NAME:elements_remove", "U1-50", "U1-82"), Array("Name:elements_add", "U1-51")

### EnableComponents (Layout Editor)

Enables or disables target components.

<b>UI Access</b>	From the <b>Components</b> window, right-click a component and select <b>Enable/Disable</b> or <b>Ctrl+click</b> multiple components, then right-click any of the selected components and click <b>Enable/Disable</b> .																	
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;Parameters&gt;</td> <td>Array</td> <td>Structured array containing these parameters in order:           <ul style="list-style-type: none"> <li>&lt;NAME:Components&gt;</li> <li>&lt;ObjectName&gt;</li> </ul> </td> </tr> <tr> <td>&lt;NAME:Components&gt;</td> <td>String</td> <td>Required first parameter which must have the specified value (i.e., "Components").</td> </tr> <tr> <td>&lt;ObjectName&gt;</td> <td>String</td> <td>List of target objects, separated by commas.</td> </tr> <tr> <td>&lt;Enable&gt;</td> <td>Boolean</td> <td><b>True</b> enables selected components. <b>False</b> disables selected components</td> </tr> </tbody> </table>			Name	Type	Description	<Parameters>	Array	Structured array containing these parameters in order: <ul style="list-style-type: none"> <li>&lt;NAME:Components&gt;</li> <li>&lt;ObjectName&gt;</li> </ul>	<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	<b>True</b> enables selected components. <b>False</b> disables selected components
Name	Type	Description																
<Parameters>	Array	Structured array containing these parameters in order: <ul style="list-style-type: none"> <li>&lt;NAME:Components&gt;</li> <li>&lt;ObjectName&gt;</li> </ul>																
<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	<b>True</b> enables selected components. <b>False</b> disables selected components																

<b>Return Value</b>	None.
---------------------	-------

<b>Python Syntax</b>	EnableComponents ([<NAME:Components>, <ObjectName>], <Enable>)
<b>Python Example</b>	<pre> oEditor.EnableComponents (     [         "NAME:Components",         "U0",         "U1"     ], False) </pre>

<b>VB Syntax</b>	EnableComponents (<NAME:Components>, <ObjectName>), <Enable>
<b>VB Example</b>	<pre> oEditor.EnableComponents Array("NAME:Contents", "U0", "U1"), False </pre>

### ExportStackupXML (Layout Editor)

Exports the stackup in XML format.

<b>UI Access</b>	From the <b>Edit Layers</b> window, select <b>Stackup &gt; Export XML</b> to open an explorer window. Navigate to an appropriate directory, choose a <b>File name</b> , and click <b>Save</b> .								
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;SaveAs&gt;</td> <td>String</td> <td>User-specified directory in which to save the new stackup XML file.</td> </tr> </tbody> </table>			Name	Type	Description	<SaveAs>	String	User-specified directory in which to save the new stackup XML file.
Name	Type	Description							
<SaveAs>	String	User-specified directory in which to save the new stackup XML file.							
<b>Return Value</b>	None.								

<b>Python Syntax</b>	ExportStackupXML(<SaveAs>)
<b>Python Example</b>	<code>oEditor.ExportStackupXML ("C:\\\\Users\\\\USERNAME\\\\Downloads\\\\test.xml")</code>

<b>VB Syntax</b>	ExportStackupXML <SaveAs>
<b>VB Example</b>	<code>oEditor.ExportStackupXML "C:\\\\Users\\\\USERNAME\\\\Downloads\\\\test.xml"</code>

### FilterObjectList (Layout Editor)

Filters a list of object names using a key/value pair. See: [FindObjects](#).

<b>UI Access</b>	N/A.								
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;key&gt;</td> <td>String</td> <td> <p>One of:</p> <ul style="list-style-type: none"> <li>• "<b>Name</b>" to search by object name.</li> <li>• "<b>Type</b>" to search by object type.</li> </ul> <p>Valid &lt;value&gt; 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'</p> <ul style="list-style-type: none"> <li>• "<b>Layer</b>" to search by layer.</li> </ul> <p>An asterisk (*) in the &lt;value&gt; string matches all layers. Vias and pins match using their defined layer range. 'Multi' matches (non via/pin) multi-layer objects.</p> </td> </tr> </tbody> </table>	Name	Type	Description	<key>	String	<p>One of:</p> <ul style="list-style-type: none"> <li>• "<b>Name</b>" to search by object name.</li> <li>• "<b>Type</b>" to search by object type.</li> </ul> <p>Valid &lt;value&gt; 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'</p> <ul style="list-style-type: none"> <li>• "<b>Layer</b>" to search by layer.</li> </ul> <p>An asterisk (*) in the &lt;value&gt; string matches all layers. Vias and pins match using their defined layer range. 'Multi' matches (non via/pin) multi-layer objects.</p>		
Name	Type	Description							
<key>	String	<p>One of:</p> <ul style="list-style-type: none"> <li>• "<b>Name</b>" to search by object name.</li> <li>• "<b>Type</b>" to search by object type.</li> </ul> <p>Valid &lt;value&gt; 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'</p> <ul style="list-style-type: none"> <li>• "<b>Layer</b>" to search by layer.</li> </ul> <p>An asterisk (*) in the &lt;value&gt; string matches all layers. Vias and pins match using their defined layer range. 'Multi' matches (non via/pin) multi-layer objects.</p>							

		<ul style="list-style-type: none"> <li>• "Net" to search by net assignment.</li> </ul>
<value>	String	Specific query. Queries use a basic string match. For instance, an asterisk (*) matches anything. All matching is case insensitive.
<b>Return Value</b>	Filtered list of object names.	

<b>Python Syntax</b>	FilterObjectList (<key>, <value>)
<b>Python Example</b>	RFIN_pins = oLayout.FilterObjectList('Type', 'Pin', oLayout.FindObjects('Net', 'RFIN'))

<b>VB Syntax</b>	FilterObjectList <key>, <value>
<b>VB Example</b>	<pre>Set objectlist = oEditor.FindObjects("Net", "RFIN") Set RFIN_pins = oEditor.FilterObjectList("Type", "Pin", objectlist)</pre>

### FindObjects (Layout Editor)

Returns a list of object names using a key/value pair.

<b>UI Access</b>	N/A.								
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;key&gt;</td> <td>String</td> <td>           One of:           <ul style="list-style-type: none"> <li>• "Name" to search by object name.</li> <li>• "Type" to search by object type.</li> </ul>           Valid &lt;value&gt; strings for this type include: 'pin', 'via', 'rect', 'arc', 'line',         </td> </tr> </tbody> </table>	Name	Type	Description	<key>	String	One of: <ul style="list-style-type: none"> <li>• "Name" to search by object name.</li> <li>• "Type" to search by object type.</li> </ul> Valid <value> strings for this type include: 'pin', 'via', 'rect', 'arc', 'line',		
Name	Type	Description							
<key>	String	One of: <ul style="list-style-type: none"> <li>• "Name" to search by object name.</li> <li>• "Type" to search by object type.</li> </ul> Valid <value> strings for this type include: 'pin', 'via', 'rect', 'arc', 'line',							

		<p>'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'</p> <ul style="list-style-type: none"> <li>• <b>"Layer"</b> to search by layer.</li> </ul> <p>An asterisk (*) in the &lt;value&gt; string matches all layers. Vias and pins match using their defined layer range. 'Multi' matches (non via/pin) multi-layer objects.</p> <ul style="list-style-type: none"> <li>• <b>"Net"</b> to search by net assignment.</li> </ul>
<value>	String	Specific query. Queries use a basic string match. For instance, an asterisk (*) matches anything. All matching is case insensitive.
<b>Return Value</b>	List of object names.	

<b>Python Syntax</b>	FindObjects (<key>, <value>)
<b>Python Example</b>	oLayout.FindObjects ('Type', 'Via')

<b>VB Syntax</b>	FindObjects <key>, <value>
<b>VB Example</b>	Set vias = oLayout.FindObjects "Type", "Via"

### 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(<oPolygopn>, <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.Polyon().AddPoint(p0).AddPoint(p1).AddPoint(p2).AddPoint(p3).SetClosed(True)
objs = oLayout.FindObjectsByPolygon(box, '*')
```

### **GetActiveUnits (Layout Editor)**

Returns the current active unit of measurement.

<b>UI Access</b>	N/A.
<b>Parameters</b>	None.
<b>Return Value</b>	Current active unit of measurement (e.g., mm).

<b>Python Syntax</b>	GetActiveUnits()
<b>Python Example</b>	<pre>oEditor.GetActiveUnits() 'mm'</pre>

<b>VB Syntax</b>	GetActiveUnits
<b>VB Example</b>	<pre>oEditor.GetActiveUnits 'mm'</pre>

### **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")

### **GetEditorName (Layout Editor)**

*Use:* Informational.

*Command:* None.

*Syntax:* GetEditorName()

*Return Value:* Returns the name of the editor.

*VB Example:* dim info  
info = oEditor.GetEditorName

### **GetLayerInfo (Layout Editor)**

Retrieves information on a specified layer in the Layout editor.

<b>UI Access</b>	None
------------------	------

Parameters	<table border="1"> <thead> <tr> <th>Name</th><th>Type</th><th>Description</th></tr> </thead> <tbody> <tr> <td>&lt;LayerName&gt;</td><td>String</td><td>Name of the layer</td></tr> </tbody> </table>	Name	Type	Description	<LayerName>	String	Name of the layer
Name	Type	Description					
<LayerName>	String	Name of the layer					
Return Value	<p>An array of strings, as follows:</p> <ul style="list-style-type: none"> <li>• Type: type of name</li> <li>• TopBottomAssociation: "Top" "Neither" "Bottom" "Template" "Invalid"</li> <li>• Color: integer [representing rgb in hex]</li> <li>• IsVisible: "true" "false". This is true if any these types of objects are visible: <ul style="list-style-type: none"> <li>• IsVisibleShape: "true" "false"[for stackup layer only]</li> <li>• IsVisiblePath: "true" "false"[for stackup layer only]</li> <li>• IsVisiblePad: "true" "false"[for stackup layer only]</li> <li>• IsVisibleHole: "true" "false"[for stackup layer only]</li> <li>• IsVisibleComponent: "true" "false"[for stackup layer only]</li> </ul> </li> <li>• IsLocked: "true" "false"</li> <li>• LayerId: integer [the ID for the layer]</li> </ul> <p>The following are also in the array if the layer is a stackup layer:</p> <ul style="list-style-type: none"> <li>• Index: integer [the stackup order index]</li> <li>• LayerThickness: double [total layer thickness, in meters]</li> <li>• EtchFactor: double [This will not appear if the layer has no etch factor defined.]</li> <li>• IsIgnored: "true" "false"</li> <li>• NumberOfSublayers: 1 [This is always 1.]</li> <li>• Material0: material name</li> </ul>						

- 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', 'LowerElevation0: 62mil', 'Roughness0 Type: Huray', 'Roughness0: 0mil, 2.9', 'BottomRoughness0 Type: Huray', 'BottomRoughness0: 0mil, 2.9', 'SideRoughness0 Type: Huray', 'SideRoughness0: 0mil, 2.9']
```

<b>Python Syntax</b>	GetLayerInfo(<layer_name>)
<b>Python Example</b>	<pre>oEditor = oDesign.SetActiveEditor("Layout") oEditor.GetLayerInfo("TopLayer")</pre>

<b>VB Syntax</b>	GetLayerInfo <layer_name>
<b>VB Example</b>	Set oModule = oDesign.GetModule("Layout") oModule.GetLayerInfo "TopLayer"

### **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
```

### **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")

## GetNets (Layout Editor)

Returns the names of all nets in the active design.

<b>UI Access</b>	N/A.
<b>Parameters</b>	None.
<b>Return Value</b>	Array of the names of the nets in the active design, separated by commas, or None if no nets exist.

<b>Python Syntax</b>	GetNets()
<b>Python Example</b>	<pre>oEditor.GetNets() ['gnd', 'net1', 'net2']</pre>

<b>VB Syntax</b>	GetNets
------------------	---------

<b>VB Example</b>	<pre>oEditor.GetNets 'gnd', 'net1', 'net2'</pre>
-------------------	------------------------------------------------------

### 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')

### GetPolygonVoids (Layout Editor)

Returns the names of the voids inside the selected polygon.

<b>UI Access</b>	N/A.		
<b>Parameters</b>	Name	Type	Description

---

	<code>&lt;PolyName&gt;</code>	String	target name of a polygon in the active design.
<b>Return Value</b>	Array of the names of the polygon voids inside the target polygon, separated by commas, or None if no voids exist.		

<b>Python Syntax</b>	<code>GetPolygonVoids (&lt;PolyName&gt;)</code>
<b>Python Example</b>	<code>oEditor.GetPolygonVoids('rect_0') ['circle void_1', 'circle void_2', 'circle void 3']</code>

<b>VB Syntax</b>	<code>GetPolygonVoids &lt;PolyName&gt;</code>
<b>VB Example</b>	<code>oEditor.GetPolygonVoids 'rect_0' 'circle void_1', 'circle void_2', 'circle void 3'</code>

### 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  
connss = 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 (Layout Editor)

Returns an array of currently selected objects.

<b>UI Access</b>	N/A
<b>Parameters</b>	None.
<b>Return Value</b>	Array containing object IDs

<b>Python Syntax</b>	GetSelections()
<b>Python Example</b>	<code>oEditor.GetSelections()</code>

<b>VB Syntax</b>	GetSelections
<b>VB Example</b>	<code>oEditor.GetSelections</code>

### 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

### HighlightNet (Layout Editor)

Highlights or removes the highlight from all elements in a net.

<b>UI Access</b>	None.		
<b>Parameters</b>	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.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	HighlightNet(["Name:Args", "Name:=", "<netName>", "Hi:=", <highlightFlag>, ...])
<b>Python Example</b>	<pre>oEditor.HighlightNet (     [         "NAME:Args",         "Name:=", "net_0",</pre>

```
"Hi:=", True
])
```

<b>VB Syntax</b>	HighlightNet Array("Name:Args", "Name:=", "<netName>", "Hi:=", <highlightFlag>, ...)
<b>VB Example</b>	<pre>Set oDesign = oProject.SetActiveDesign ("HFSS3D1") Set oEditor = oDesign.SetActiveEditor ("Layout") oEditor.HighlightNet Array ("NAME:Args", "Name:=", "net_0", "Hi:=", false)</pre>

### ImportStackupXML (Layout Editor)

Imports an XML file into an active project.

<b>UI Access</b>	From the <b>Edit Layers</b> window, select <b>Stackup &gt; Import XML</b> .						
<b>Parameters</b>	<table border="1"> <tr> <td>Name</td> <td>Type</td> <td>Description</td> </tr> <tr> <td>&lt;XMLFileName&gt;</td> <td>String</td> <td>Full path to XML file.</td> </tr> </table>	Name	Type	Description	<XMLFileName>	String	Full path to XML file.
Name	Type	Description					
<XMLFileName>	String	Full path to XML file.					
<b>Return Value</b>	None.						

<b>Python Syntax</b>	ImportStackupXML(<XMLFileName>)
<b>Python Example</b>	<pre>oProject = oDesktop.SetActiveProject ("Project_Name") oDesign = oProject.SetActiveDesign ("DesignName") oEditor = oDesign.SetActiveEditor ("Layout") oEditor.ImportStackupXML ("C:/Files/example.xml")</pre>

<b>VB Syntax</b>	ImportStackupXML <XMLFileName>
<b>VB Example</b>	<pre>oProject = oDesktop.SetActiveProject "Project_Name" oDesign = oProject.SetActiveDesign "DesignName" oEditor = oDesign.SetActiveEditor "Layout" oEditor.ImportStackupXML "C:/Files/example.xml"</pre>

### 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")
```

### ModifyDifferentialPair (Layout Editor)

Modifies a differential pair from two target nets.

<b>UI Access</b>	From the <b>Nets</b> window, select the <b>Differential</b> tab. Then right-click a differential pair and select <b>Swap</b> .		
<b>Parameters</b>	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: <ul style="list-style-type: none"><li>• &lt;Net1&gt;</li><li>• &lt;Net2&gt;</li></ul>
	<Net1>	String	A net in the differential pair, presumably now in the swapped position (e.g., Sig_P).
<b>Return Value</b>	<Net2>	String	A net in the differential pair, presumably now in the swapped position (e.g., Sig_N).
	None		

<b>Python Syntax</b>	ModifyDifferentialPair(<DiffPairName>, <NewName>, <Description>, [<Net1>, <Net2>])
<b>Python Example</b>	oEditor.ModifyDifferentialPair("Sig_0", "Sig_0", "", ["Sig_P", "Sig_N"])

<b>VB Syntax</b>	ModifyDifferentialPair <DiffPairName>, <NewName>, <Description>, (<Net1>, <Net2>)
<b>VB Example</b>	oEditor.ModifyDifferentialPair "DiffPair1", "", Array("GND", "Port1")

## 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))

### 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))

### 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()

### RemoveLayer (Layout Editor)

UI Access	From the <b>Layers</b> window, right-click the target layer and select <b>Remove</b> .
-----------	----------------------------------------------------------------------------------------

#### General Mode

Removes the target layer or target stackup layer.

Parameters	Name <i>&lt;LayerName&gt;</i>	Type String	Description Name of the target layer.
Return Value	None.		

<b>Python Syntax</b>	RemoveLayer(<LayerName>)
<b>Python Example</b>	<code>oEditor.RemoveLayer ("Ground")</code>

<b>VB Syntax</b>	RemoveLayer <LayerName>
<b>VB Example</b>	<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
	<Parameters>	Array	Structured array containing these parameters in order: <ul style="list-style-type: none"><li>• &lt;NAME:Contents&gt;</li><li>• &lt;NAME&gt;</li><li>• &lt;RemoveLayer&gt;</li></ul>
	<NAME:Contents>	String	Required first parameter which must have the specified value (i.e., "Contents").
	<LayerName>	String	Name of the target layer.
<RemoveLayer>	Boolean	<b>True</b> removes the layer and all objects on the target layer. <b>False</b> removes only all objects.	
Return Value	None.		

<b>Python Syntax</b>	RemoveLayer([<NAME:Contents>, <LayerName>], <RemoveLayer>)
<b>Python Example</b>	<pre> oEditor.RemoveLayer (     [         "NAME:Contents",         "Ground"     ], False) </pre>

<b>VB Syntax</b>	RemoveLayer(<NAME:Contents>, <LayerName>), <RemoveLayer>
<b>VB Example</b>	<pre> oEditor.RemoveLayer Array(     "NAME:Contents",     "Ground", ), False </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")

### **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

### **RemovePortsOnComponents (Layout Editor)**

Remove ports on port instances of selected components. Multiple nets can be listed.

<b>UI Access</b>	Right click and choose <b>Port &gt; Remove Ports From Component</b> .		
<b>Parameters</b>	Name	Type	Description
	< requiredKeyword>	String	Required parameter that must proceed all element names with the value "NAME:elements".
<b>Return Value</b>	None.		

#### **Python Syntax**

RemovePortsOnComponents (["NAME:elements", <elementName>, ...])

<b>Python Example</b> <pre>oEditor.RemovePortsOnComponents (     [         "NAME:elements",         "0"     ]) </pre>
--------------------------------------------------------------------------------------------------------------------------

<b>VB Syntax</b>	RemovePortsOnComponents Array("NAME:elements", " <element-name>",...)
<b>VB Example</b>	<code>oEditor.RemovePortsOnComponents Array("NAME:elements", "0")</code>

### ResetDifferentialpairs (Layout Editor)

Resets a differential pair to its original state.

<b>UI Access</b>	From the <b>Nets</b> window, select the <b>Differential</b> tab. Then right-click a differential pair and select <b>Edit</b> .		
<b>Parameters</b>	Name <code>&lt;Parameters1&gt;</code>	Type Array	Description Structured array containing these parameters in order: <ul style="list-style-type: none"> <li>• &lt;NAME&gt;New Differential Pairs&gt;</li> <li>• &lt;DiffPair1&gt;</li> <li>• &lt;DiffPair2&gt;</li> </ul>
	<code>&lt;NAME&gt;New Differential Pairs&gt;</code>	String	Required first parameter which must have the specified value (i.e., "New Differential Pairs").
	<code>&lt;DiffPair1&gt;</code>	Array	Array containing the following parameters for the target differential pair in order: <ul style="list-style-type: none"> <li>• &lt;DiffPairName&gt;</li> <li>• &lt;Description1&gt;</li> </ul>

		<ul style="list-style-type: none"> <li>• &lt;DiffPair1Nets&gt;</li> </ul>
<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: <ul style="list-style-type: none"> <li>• &lt;Net1&gt;</li> <li>• &lt;Net2&gt;</li> </ul>
<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: <ul style="list-style-type: none"> <li>• &lt;DiffPairName&gt;</li> <li>• &lt;Description2&gt;</li> <li>• &lt;DiffPair2Nets&gt;</li> </ul>
<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: <ul style="list-style-type: none"> <li>• &lt;Net3&gt;</li> <li>• &lt;Net4&gt;</li> </ul>
<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).
<b>Return Value</b>	None	

<b>Python Syntax</b>	ResetDifferentialPairs([<NAME>New Differential Pairs>, [<DiffPair1>, <Description1>, [<Net1>, <Net2>]], [<DiffPair2>, <Description2>, [<Net3>, <Net4>]]])
----------------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------

**Python Example**

```

oEditor.ResetDifferentialPairs(
    [
        "NAME:New Differential Pairs",
        [
            "1",
            "",
            ["Port1", "Sig_P"]
        ],
        [
            "2",
            "",
            ["Sig_N", "Port2"]
        ]
    ]
)

```

**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")))

```

## Select (Layout Editor)

Chooses a target list of objects to make active.

<b>UI Access</b>	N/A.		
<b>Parameters</b>	Name <ObjectName>	Type Array	Description Array consisting of target object names, separated by commas (e.g., rect_0, rect_1, circle_0).
<b>Return Value</b>	None.		

<b>Python Syntax</b>	Select(<ObjectName>)
<b>Python Example</b>	<code>oEditor.Select(['rect_0', 'rect_1', 'circle_0'])</code>

<b>VB Syntax</b>	Select<ObjName>
<b>VB Example</b>	<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()
```

## SelectNetConnected (Layout Editor)

Select all elements in the active design connected to the target net.

<b>UI Access</b>	From the <b>Layout</b> tab, right-click an object in the active design and select <b>Nets &gt; Select Net Connected</b> .		
<b>Parameters</b>	Name	Type	Description
	<Parameters>	Array	Structured array containing these parameters in order: <ul style="list-style-type: none"><li>• &lt;NAME:elements&gt;</li><li>• &lt;TargetNet&gt;</li></ul>
	<NAME:elements>	String	Required first parameter which must have the specified value (i.e., "elements").
<b>Return Value</b>	None.		

<b>Python Syntax</b>	SelectNetConnected([<NAME:elements>, <TargetNet>])
<b>Python Example</b>	<pre> oEditor.SelectNetConnected (     [         "NAME:elements",         "circle_1288",     ] ) </pre>

<b>VB Syntax</b>	SelectNetConnected(<NAME:elements>, <TargetNet>)
<b>VB Example</b>	<code>oEditor.SelectNetConnected Array ("NAME:elements", "circle_1288")</code>

### SelectPhysicallyConnected (Layout Editor)

Selects all elements physically connected to the target object.

<b>UI Access</b>	From the <b>Layout</b> tab, right-click an object in the active design and select <b>Nets &gt; Select Physically Connected</b> .		
<b>Parameters</b>	Name	Type	Description
	<Parameters>	Array	Structured array containing these parameters in order: <ul style="list-style-type: none"><li>• &lt;NAME:elements&gt;</li><li>• &lt;TargetObject&gt;</li></ul>
	<NAME:elements>	String	Required first parameter which must have the specified value (i.e., "elements").
<b>Return Value</b>	None.		

<b>Python Syntax</b>	SelectPhysicallyConnected([<NAME:elements>, <TargetObject>])
<b>Python Example</b>	<code>oEditor.SelectPhysicallyConnected (</code> [

	<pre>"NAME:elements", "circle_1288", ])</pre>
--	-----------------------------------------------

<b>VB Syntax</b>	SelectPhysicallyConnected(<NAME:elements>, <TargetObject>)
<b>VB Example</b>	<code>oEditor.SelectPhysicallyConnected Array ("NAME:elements", "circle_1288")</code>

### SetActiveUnits (Layout Editor)

Sets the current active unit of measurement to the target unit.

<b>UI Access</b>	N/A						
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;SelectedUnits&gt;</td> <td>Integer</td> <td>User-specified unit of measurement to set to active (e.g., mm).</td> </tr> </tbody> </table>	Name	Type	Description	<SelectedUnits>	Integer	User-specified unit of measurement to set to active (e.g., mm).
Name	Type	Description					
<SelectedUnits>	Integer	User-specified unit of measurement to set to active (e.g., mm).					
<b>Return Value</b>	The previous active unit of measurement.						

<b>Python Syntax</b>	SetActiveUnits(<SelectedUnits>)
<b>Python Example</b>	<code>oEditor.SetActiveUnits ('um')</code> <code>'mm'</code>

<b>VB Syntax</b>	SetActiveUnits <SelectedUnits>
------------------	--------------------------------

**VB Example**

```
oEditor.SetActiveUnits 'um'  
'mm'
```

**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

**SetHfssExtentsVisible (Layout Editor)**

Hides or reveals HFSS extents.

<b>UI Access</b>	From the <b>Layout</b> tab, select <b>HFSS Extents &gt; Show/Hide</b> .		
<b>Parameters</b>	<b>Name</b>	<b>Type</b>	<b>Description</b>
	<Enable>	Boolean	<b>True</b> displays HFSS extents. <b>False</b> hides HFSS extents.
<b>Return Value</b>	None.		

**Python Syntax**

```
SetHfssExtentsVisible(<Enable>)
```

**Python Example**

```
oEditor.SetHfssExtentsVisible(False)
```

<b>VB Syntax</b>	SetHfssExtentsVisible <Enable>
<b>VB Example</b>	<code>oEditor.SetHfssExtentsVisible False</code>

### SetHfssPortsVisible (Layout Editor)

Enables or disables target components.

<b>UI Access</b>	From <b>Layout</b> , select <b>Draw Ports/Sources</b> .		
<b>Parameters</b>	<b>Name</b>	<b>Type</b>	<b>Description</b>
	<Enable>	Boolean	<b>True</b> displays HFSS ports. <b>False</b> hides HFSS ports.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	SetHfssPortsVisible(<Enable>)
<b>Python Example</b>	<code>oEditor.SetHfssPortsVisible(False)</code>

<b>VB Syntax</b>	SetHfssPortsVisible <Enable>
<b>VB Example</b>	<code>oEditor.SetHfssPortsVisible False</code>

### SetNetColor (Layout Editor)

Sets the color of a target net.

<b>UI Access</b>	From the <b>Nets</b> window, right-click a net and select <b>List</b> to open the <b>Design List</b> window. From any row, double-
------------------	------------------------------------------------------------------------------------------------------------------------------------

	click the bar in the <b>Color</b> column to open the <b>Color</b> window. Select a new color, then click <b>OK</b> to change the net's color.		
<b>Parameters</b>	Name	Type	Description
	<Parameters>	Array	Structured array containing these parameters in order: <ul style="list-style-type: none"><li>• &lt;NAME:Args&gt;</li><li>• &lt;Name&gt;</li><li>• &lt;Color&gt;</li></ul>
	<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: <ul style="list-style-type: none"><li>• &lt;R&gt;</li><li>• &lt;G&gt;</li><li>• &lt;B&gt;</li><li>• &lt;Check_CContainment&gt;</li><li>• &lt;Via_Persistence&gt;</li></ul>
	<R>	Integer	Value of red color.
	<G>	Integer	Value of green color.
	<B>	Integer	Value of blue color.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	SetNetColor([<NAME:Args>, <Name>, <Color>, [<R>, <G>, <B>]])
<b>Python Example</b>	<pre> oEditor.SetNetColor( [     "NAME:Args",     "Name:=", "Net_1",     "Color:=", ["R:=", 0, "G:=", 0, "B:=", 160] ]) </pre>

<b>VB Syntax</b>	SetNetColor(<Name:Args>, <Name>, <Color>, (<R>, <G>, <B>))
<b>VB Example</b>	<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.

<b>UI Access</b>	On the <b>Nets</b> tab of the <b>Nets</b> window, highlight and right click on one or more nets. Click <b>Hide</b> or any of the <b>Show</b> options.											
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;requiredParameterKeyword&gt;</td> <td>String</td> <td>Required first parameter which must have the value "Name:Args".</td> </tr> <tr> <td>&lt;requiredNameKeyword&gt;</td> <td>String</td> <td>Required second parameter which must have the value "Name:=". Must be the first parameter for each net.</td> </tr> </tbody> </table>	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.		
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.										

	<code>&lt;netName&gt;</code>	String	Name of the net
	<code>&lt;requiredFlagKeyword&gt;</code>	String	Required fourth parameter which must have the value "Vis:=". Must be the third parameter for each net.
	<code>&lt;visibilityFlag&gt;</code>	Boolean	True if the net should be visible. False if it should be hidden.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	<code>SetNetVisible (["Name:Args", "Name:=", "&lt;netName&gt;", "Vis:=", &lt;highlightFlag&gt;, ...])</code>
<b>Python Example</b>	<pre> oEditor.SetNetVisible( [     "NAME:Args",     "Name:=", "Net_1",     "Vis:=", True ]) oEditor.SetNetVisible( [     "NAME:Args",     "Name:=", "&lt;NO-NET&gt;",     "Vis:=", False,     "Name:=", "Ground",     "Vis:=", False, ] ) </pre>

```

    "Name:=", "Chassis",
    "Vis:=", True
)

```

<b>VB Syntax</b>	SetNetVisible Array ("Name:Args", "Name:=", "<netName>", "Vis:=", <visibilityFlag>, ...)
<b>VB Example</b>	<pre> oEditor.SetNetVisible Array("NAME:Args", "Name:=", "NET_1", "Vis:=", true) oEditor.SetNetVisible Array("NAME:Args", "Name:=", "&lt;NO-NET&gt;", "Vis:=", false, "Name:=", "Ground", "Vis:=", false, "Name:=", "Chassis", "Vis:=", true) </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"
```

### 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")
```

### ToggleNetHighlight (Layout Editor)

Highlights or removes the highlight from all objects in the nets of the target elements.

<b>UI Access</b>	After selecting one or more elements within the <b>Layout Editor</b> , navigate to <b>Layout &gt; Nets &gt; Toggle Net Highlight</b> .		
<b>Parameters</b>	Name <i>&lt;Parameters&gt;</i>	Type Array	Description Structured array containing these parameters in order: • <NAME:elements>

		<ul style="list-style-type: none"> <li>• &lt;Names&gt;</li> </ul>
<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.
<b>Return Value</b>	None.	

<b>Python Syntax</b>	ToggleNetHighlight([<NAME:elements>, <Names>])
<b>Python Example</b>	<pre> oEditor.ToggleNetHighlight ( [     "NAME:elements",     "rect_0",     "rect_2",     "circle_1"     "R96-2" ]) </pre>

<b>VB Syntax</b>	ToggleNetHighlight(<NAME:elements>, <Names>)
<b>VB Example</b>	<pre> oEditor.ToggleNetHighlight("NAME:elements", "rect_0", "rect_2", "circle_1" "R96-2") </pre>

### 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")

### **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>, // 1<sup>st</sup> primitive name

<object\_name>, // 2<sup>nd</sup> 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()

## **Layers Methods**

The following methods are available to manipulate layers:

- [AddLayer \(Layout Editor\)](#)
- [AddStackupLayer \(Layout Editor\)](#)
- [ChangeLayers \(Layout Editor\)](#)
- [ClearLayerMappings \(Layout Editor\)](#)
- CreateBoardBend
- [DuplicateAcrossLyrs \(Layout Editor\)](#)
- [GetAllLayerNames \(Layout Editor\)](#)
- [GetLayerInfo \(Layout Editor\)](#)
- [GetStackupLayerNames \(Layout Editor\)](#)
- [ImportStackupXML](#)
- ProcessBentModelCmd
- [RemoveLayer \(Layout Editor\)](#)

- [SetLayerMapping \(Layout Editor\)](#)
- [SyncRigidFlexSandbox](#)

## AddObject (Layout Editor and IC Editor)

Adds a user-selected object to the active design.

<b>UI Access</b>	Not strictly applicable, although objects <i>can</i> be drawn by making the appropriate selection from the <b>Layout</b> ribbon (i.e., <b>Draw polygon</b> , <b>Draw line</b> , <b>Draw rectangle</b> , <b>Draw arc</b> , <b>Draw circle</b> ).		
<b>Parameters</b>	Name <code>&lt;ObjectType&gt;</code>	Type String	Description Type of object to add (i.e., rect (i.e., rectangle), circle, poly (i.e., polygon), line, or 3D line).
	<code>&lt;Parameters&gt;</code>	Array	Structured array containing the parameters required for the type of object selected. The parameters differ for each object. Refer to the following pages: <ul style="list-style-type: none"><li>• "CreateRectangle (Layout Editor) " on page 30-61</li><li>• "CreateCircle (Layout Editor) " on page 30-35</li><li>• "CreatePolygon (Layout Editor) " on page 30-57</li><li>• "CreateLine (Layout Editor) " on page 30-47</li><li>• "AddObject ("3D Line") (Layout Editor) " on page 28-146</li></ul>
<b>Return Value</b>	Returns the name of the newly created object ( <b>IC Mode</b> only).		

### Python Syntax

```
AddObject(<ObjectType>, [<Parameters>])
```

<b>Python Example</b>	<pre> oEditor.AddObject("rect", [     "NAME:Contents",     "rectGeometry:=", [         "Name:=", "rect_5",         "LayerName:=", "Trace",         "lw:=", "0",         "Ax:=", "-84mm",         "Ay:=", "16mm",         "Bx:=", "-77mm",         "By:=", "12mm"]     ]) </pre>
-----------------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

<b>VB Syntax</b>	<b>AddObject(&lt;ObjectType&gt;, [&lt;Parameters&gt;])</b>
<b>VB Example</b>	<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>

	"By:=", "12mm"] ])
--	-----------------------

## 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: <ul style="list-style-type: none"><li>• &lt;Layername&gt;</li><li>• &lt;Type&gt;</li><li>• &lt;TopBottom&gt;</li><li>• &lt;Color&gt;</li><li>• &lt;Transparency&gt;</li><li>• &lt;Pattern&gt;</li><li>• &lt;VisFlag&gt;</li><li>• &lt;Locked&gt;</li><li>• &lt;Draw&gt;</li></ul>

	<b>&lt;LayerName&gt;</b>	String	Name of layer.
	<b>&lt;Type&gt;</b>	String	Type of layer, such as "user", "soldermask", "solderpaste", "silkscreen", "assembly", "wirebond", "glue", "user", "Postprocessing", or "outline".
	<b>&lt;TopBottom&gt;</b>	String	Optional. Indicates whether the layer is the top or bottom. Acceptable values are "top", "bottom", or "neither".
	<b>&lt;Color&gt;</b>	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.
	<b>&lt;Transparency&gt;</b>	Integer	Optional. A percentage that indicates the transparency of the layer between 0 (opaque) and 100 (completely transparent and not visible).
	<b>&lt;Pattern&gt;</b>	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.
	<b>&lt;VisFlag&gt;</b>	Integer	<p>Sets visibility for all objects on the layer using the following additive code:</p> <ul style="list-style-type: none"> <li>• 1 makes primitives (rectangles, circles, polygons) visible.</li> <li>• 2 makes lines (also known as paths or traces) visible.</li> <li>• 4 makes pads (from padstacks, aka vias) visible.</li> <li>• 8 makes holes (or vias) visible.</li> <li>• 16 makes components visible.</li> <li>• 32 makes the mesh overlay visible if they are plotted.</li> <li>• 64 makes the mesh for the background material visible.</li> </ul> <p>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.</p>
	<b>&lt;Locked&gt;</b>	Boolean	Optional. Whether the layer is locked.
	<b>&lt;Draw&gt;</b>	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.
<b>Return Value</b>	None		

<b>Python Syntax</b>	<pre>oEditor.AddLayer(     [         "NAME:layers",         "Name:=", "&lt;LayerName&gt;",         "Type:=", "&lt;Type&gt;",         "Top Bottom:=", "&lt;TopBottom&gt;",         "Color:=", &lt;Color&gt;,         "Transparency:=", &lt;Transparency&gt;,         "Pattern:=", &lt;Pattern&gt;,         "VisFlag:=", &lt;VisFlag&gt;,         "Locked:=", &lt;Locked&gt;,         "DrawOverride:=", &lt;Draw&gt;,     ])</pre>
<b>Python Example</b>	<pre>oEditor = oDesign.SetActiveEditor("Layout") oEditor.AddLayer(     [         "NAME:layers",         "Name:=", "signal",         "Type:=", "assembly",         "Top Bottom:=", "neither",</pre>

```

    "Color:=", 16711680,
    "Transparency:=", 0,
    "Pattern:=", 1,
    "VisFlag:=", 127,
    "Locked:=", False,
    "DrawOverride:=", 0,
)

```

<b>VB Syntax</b>	<pre> oEditor.AddLayer Array("NAME:stackup layer", "Name:=", "&lt;LayerName&gt;", "Type:=", "&lt;Type&gt;", "Top Bottom:=", "&lt;TopBottom&gt;", "Color:=", &lt;Color&gt;, "Transparency:=", &lt;Transparency&gt;, "Pattern:=", &lt;Pattern&gt;, "VisFlag:=", &lt;VisFlag&gt;, "Locked:=", &lt;Locked&gt;, "DrawOverride:=", &lt;Draw&gt;,) </pre>
<b>VB Example</b>	<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.

<b>UI Access</b>	None		
<b>Parameters</b>	Name	Type	Description
	<LayerArray>	Array	An array that contains these parameters in order: <ul style="list-style-type: none"><li>• &lt;Layername&gt;</li><li>• &lt;Type&gt;</li><li>• &lt;TopBottom&gt;</li><li>• &lt;Color&gt;</li><li>• &lt;Transparency&gt;</li><li>• &lt;Pattern&gt;</li><li>• &lt;VisFlag&gt;</li><li>• &lt;Locked&gt;</li><li>• &lt;Draw&gt;</li><li>• and an array with the parameters:<ul style="list-style-type: none"><li>• &lt;Thickness&gt;</li><li>• &lt;LowerElevation&gt;</li><li>• &lt;Material&gt;</li></ul></li></ul>
	<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: <ul style="list-style-type: none"><li>• 1 makes primitives (rectangles, circles, polygons) visible.</li><li>• 2 makes lines (also known as paths or traces) visible.</li><li>• 4 makes pads (from padstacks, aka vias) visible.</li><li>• 8 makes holes (or vias) visible.</li><li>• 16 makes components visible.</li><li>• 32 makes the mesh overlay visible if they are plotted.</li><li>• 64 makes the mesh for the background material visible.</li></ul> 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.
<b>Return Value</b>	None	

<b>Python Syntax</b>	<pre> oEditor.AddStackupLayers(     [         "NAME:stackup layer",     ] ) </pre>
----------------------	------------------------------------------------------------------------------------

```
"Name:=", "<LayerName>",
"Type:=", "<Type>",
"Top Bottom:=", "<TopBottom>",
"Color:=", <Color>,
"Transparency:=", <Transparency>,
"Pattern:=", <Pattern>,
"VisFlag:=", <VisFlag>,
"Locked:=", <Locked>,
"DrawOverride:=", <Draw>,
[
    "NAME:sublayer",
    "Thickness:=", "<Thickness>",
    "LowerElevation:=", "<LowerElevation>",
    "Material:=", "<Material>",
]
])
```

**Python Example**

```
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,
    [
        "NAME:Sublayer",
        "Thickness:=", "0mil",
        "LowerElevation:=", "20mil",
        "Material:=", "gold",
    ]
)

```

<b>VB Syntax</b>	<code>oEditor.AddStackupLayers Array("NAME:stackup layer" "Name:=", "&lt;LayerName&gt;", "Type:=", "&lt;Type&gt;", "Top Bottom:=", "&lt;TopBottom&gt;", "Color:=", &lt;Color&gt;, "Transparency:=", &lt;Transparency&gt;, "Pattern:=", &lt;Pattern&gt;, "VisFlag:=", &lt;VisFlag&gt;, "Locked:=", &lt;Locked&gt;, "DrawOverride:=", &lt;Draw&gt;, Array("NAME:Sublayer", "Thickness:=", "&lt;Thickness&gt;", "LowerElevation:=", "&lt;LowerElevation&gt;", "Material:=", &lt;Material&gt;))</code>
<b>VB Example</b>	<code>Set oEditor = oDesign.SetActiveEditor("Layout")</code>

```

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"))

```

## ChangeLayers (Layout 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 <b>Edit Layers</b> window.		
Parameters	Name	Type	Description
	<LayerArray>	Array	<p>An array that contains these parameters in order:</p> <ul style="list-style-type: none"> <li>• &lt;Layername&gt;</li> <li>• &lt;ID&gt;</li> <li>• &lt;Type&gt;</li> <li>• &lt;TopBottom&gt;</li> <li>• &lt;Color&gt;</li> <li>• &lt;Transparency&gt;</li> </ul>

		<ul style="list-style-type: none"> <li>• &lt;Pattern&gt;</li> <li>• &lt;VisFlag&gt;</li> <li>• &lt;Locked&gt;</li> <li>• &lt;Draw&gt;</li> <li>• &lt;Subarray&gt;</li> <li>• &lt;Neg&gt;</li> </ul>
<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: <ul style="list-style-type: none"> <li>• 1 makes primitives (rectangles, circles, polygons) visible.</li> <li>• 2 makes lines (also known as paths or traces) visible.</li> <li>• 4 makes pads (from padstacks, aka vias) visible.</li> <li>• 8 makes holes (or vias) visible.</li> <li>• 16 makes components visible.</li> </ul>

		<ul style="list-style-type: none"> <li>• 32 makes the mesh overlay visible if they are plotted.</li> <li>• 64 makes the mesh for the background material visible.</li> </ul> <p>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.</p>
<Locked>	Boolean	<i>Optional.</i> Whether the layer is locked.
<Draw>	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.
<Subarray>	Array	<i>Optional.</i> An array that contains these parameters in order: <ul style="list-style-type: none"> <li>• &lt;Thickness&gt;</li> <li>• &lt;LowerElevation&gt;</li> <li>• &lt;Roughness&gt;</li> <li>• &lt;BotRoughness&gt;</li> <li>• &lt;SideRoughness&gt;</li> <li>• &lt;Material&gt;</li> <li>• &lt;FillMaterial&gt;</li> </ul> <p>This array's placeholder does not appear in the syntax or examples below but instead has its contents listed.</p>
<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.
<b>Return Value</b>	None		

**Python Syntax**

```
oEditor.ChangeLayers(  
    [  
        "NAME:layers",  
        "Mode:=", "<mode>",  
        [  
            "NAME:pps"  
        ],  
        [  
            "NAME:stackup layer"  
            "Name:=", "<LayerName>",  
            "ID:=", <ID>,  
            "Type:=", "<Type>",  
            "Top Bottom:=", "<TopBottom>",  
            "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>
])
```

```
oEditor = oDesign.SetActiveEditor("Layout")
oEditor.ChangeLayers(
    [
        "NAME:layers",
        "Mode:=", "Laminate",
```

**Python Example**

```
[  
    "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",  
    "Thickness:=", "0mil",  
    "LowerElevation:=", "20mil",  
    "Roughness:=", "0um",  
    "BotRoughness:=", "0um",
```

```

        "SideRoughness:=", "0um",
        "Material:=", "gold",
        "FillMaterial:=", "FR4_epoxy"
    ],
    "Neg:=", True
]
)

```

<b>VB Syntax</b> <pre>oEditor.ChangeLayers Array("NAME:layers", "Mode:=", "&lt;mode&gt;", Array("NAME:pps"), Array("NAME:stackup layer", "Name:=", "&lt;LayerName&gt;", "ID:=", &lt;ID&gt;, "Type:=", "&lt;Type&gt;", "Top Bottom:=", "&lt;TopBottom&gt;", "Color:=", &lt;Color&gt;, "Transparency:=", &lt;Transparency&gt;, "Pattern:=", &lt;Pattern&gt;, "VisFlag:=", &lt;VisFlag&gt;, "Locked:=", &lt;Locked&gt;, "DrawOverride:=", &lt;Draw&gt;, Array("NAME:Sublayer", "Thickness:=", "&lt;Thickness&gt;", "LowerElevation:=", "&lt;LowerElevation&gt;", "Roughness:=", "&lt;Roughness&gt;", "BotRoughness:=", "&lt;BotRoughness&gt;", "SideRoughness:=", "&lt;SideRoughness&gt;", "Material:=", &lt;Material&gt;,"FillMaterial:=", "&lt;FillMaterial&gt;"),"Neg:=", &lt;Neg&gt;))</pre>	<b>VB Example</b> <pre>Set oEditor = oDesign.SetActiveEditor("Layout") 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:=", _ false, "DrawOverride:=", 0, Array("NAME:Sublayer", "Thickness:=", _ "1.6mm", "LowerElevation:=", "0mm", "Roughness:=", "0um", "BotRoughness:=", _ "0um", "SideRoughness:=", "0um", "Material:=", "FR4_epoxy"),"Neg:=", True))</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")

## **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.

<b>UI Access</b>	None								
<b>Parameters</b>	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>&lt;LayerName&gt;</td><td>String</td><td>Name of the layer</td></tr></tbody></table>			Name	Type	Description	<LayerName>	String	Name of the layer
Name	Type	Description							
<LayerName>	String	Name of the layer							
<b>Return Value</b>	<p>An array of strings, as follows:</p> <ul style="list-style-type: none"><li>• Type: type of name</li><li>• TopBottomAssociation: "Top" "Neither" "Bottom" "Template" "Invalid"</li><li>• Color: integer [representing rgb in hex]</li><li>• IsVisible: "true" "false". This is true if any these types of objects are visible:<ul style="list-style-type: none"><li>• IsVisibleShape: "true" "false"[for stackup layer only]</li><li>• IsVisiblePath: "true" "false"[for stackup layer only]</li><li>• IsVisiblePad: "true" "false"[for stackup layer only]</li><li>• IsVisibleHole: "true" "false"[for stackup layer only]</li><li>• IsVisibleComponent: "true" "false"[for stackup layer only]</li></ul></li><li>• IsLocked: "true" "false"</li><li>• LayerId: integer [the ID for the layer]</li></ul>								

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', 'LowerElevation0: 62mil', 'Roughness0 Type: Huray', 'Roughness0: 0mil, 2.9', 'BottomRoughness0 Type: Huray', 'BottomRoughness0: 0mil, 2.9', 'SideRoughness0 Type: Huray', 'SideRoughness0: 0mil, 2.9']
```

	Huray', 'SideRoughness0: 0mil, 2.9']
--	--------------------------------------

<b>Python Syntax</b>	GetLayerInfo(<layer_name>)
<b>Python Example</b>	<pre>oEditor = oDesign.SetActiveEditor("Layout") oEditor.GetLayerInfo("TopLayer")</pre>

<b>VB Syntax</b>	GetLayerInfo <layer_name>
<b>VB Example</b>	<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.

<b>UI Access</b>	From the <b>Edit Layers</b> window, select <b>Stackup &gt; Import XML</b> .						
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;XMLFileName&gt;</td> <td>String</td> <td>Full path to XML file.</td> </tr> </tbody> </table>	Name	Type	Description	<XMLFileName>	String	Full path to XML file.
Name	Type	Description					
<XMLFileName>	String	Full path to XML file.					
<b>Return Value</b>	None.						

<b>Python Syntax</b>	ImportStackupXML(<XMLFileName>)
<b>Python Example</b>	<pre> oProject = oDesktop.SetActiveProject("Project_Name") oDesign = oProject.SetActiveDesign("DesignName") oEditor = oDesign.SetActiveEditor("Layout") oEditor.ImportStackupXML("C:/Files/example.xml") </pre>

<b>VB Syntax</b>	ImportStackupXML <XMLFileName>
<b>VB Example</b>	<pre> oProject = oDesktop.SetActiveProject "Project_Name" oDesign = oProject.SetActiveDesign "DesignName" oEditor = oDesign.SetActiveEditor "Layout" oEditor.ImportStackupXML "C:/Files/example.xml" </pre>

## 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 28-207.

## **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	<p>Structured Array containing:</p> <pre>Array ("Name:contents",       &lt;       Geometry1Array       &gt;, &lt;Geometry2Array&gt;, &lt;Geometry3Array&gt;...)</pre> <p><b>Note:</b> 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 <b>Zone Order</b> parameter in Electronics Desktop.</p>
	<Geometry>	Array	<p>Structured Array containing:</p> <pre>"Name:=", &lt;string zone# name&gt;, "LayerName:=", "Outline",</pre>

			<p><b>Note:</b> Depending on the type of geometry ( i.e., <a href="#">polygon</a>, <a href="#">rectangle</a>, <a href="#">circle</a>, <a href="#">line</a>, or board bend), additional parameters may be applicable. For example, adding "Is_Fixed:=true within a &lt;Geometry&gt; array will assign the <b>Fixed</b> 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.</p>
--	--	--	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

<b>Return Value</b>	None.
<b>Python Syntax</b>	SyncRigidFlexSandbox (<Contents>, <Geometry>...)
<b>Python Example</b>	

<b>VB Syntax</b>	SyncRigidFlexSandbox <Contents>, <Geometry>...
<b>VB Example</b>	

## Net and NetClass Methods

The following net and net class primitives are available:

- [CreateNetClass](#)
- [DeleteNets](#)
- [DelNetClass](#)
- [GetNetClasses](#)

- [GetNetClassNets](#)
- [GetNetConnections](#)
- [HighlightNet](#)
- [ModifyNetClass](#)
- [SetNetVisible](#)

## 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.

<b>UI Access</b>	None.		
<b>Parameters</b>	Name <netName>	Type String	Description Name of unwanted net.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	DeleteNets ([<netName>, ...])
<b>Python Example</b>	<pre>oEditor.DeleteNets(     [         "NET_1",         "NET_2",         "NET_3"     ])</pre>

<b>VB Syntax</b>	DeleteNets Array (<netName>, ...)
<b>VB Example</b>	<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")

## 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.

	<i>&lt;netName&gt;</i>	String	Name of the net.
	<i>&lt;requiredFlagKeyword&gt;</i>	String	Required fourth parameter which must have the value "Hi:=". Must be the third parameter for each net.
	<i>&lt;highlightFlag&gt;</i>	Boolean	True if the net should be highlighted. False if it should not.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	HighlightNet(["Name:Args", "Name:=", " <i>&lt;netName&gt;</i> ", "Hi:=", <i>&lt;highlightFlag&gt;</i> , ...])
<b>Python Example</b>	<pre> oEditor.HighlightNet (     [         "NAME:Args",         "Name:=", "net_0",         "Hi:=", True     ] ) </pre>

<b>VB Syntax</b>	HighlightNet Array("Name:Args", "Name:=", " <i>&lt;netName&gt;</i> ", "Hi:=", <i>&lt;highlightFlag&gt;</i> , ...)
<b>VB Example</b>	<pre> Set oDesign = oProject.SetActiveDesign ("HFSS3D1") Set oEditor = oDesign.SetActiveEditor("Layout") oEditor.HighlightNet Array("NAME:Args", "Name:=", "net_0", "Hi:=", false) </pre>

## 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")

## SetNetVisible (Layout Editor)

Shows and highlights or hides specified nets in the Layout Editor. Multiple nets can be highlighted or hidden at once.

<b>UI Access</b>	On the <b>Nets</b> tab of the <b>Nets</b> window, highlight and right click on one or more nets. Click <b>Hide</b> or any of the <b>Show</b> options.		
<b>Parameters</b>	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.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	<code>SetNetVisible (["Name:Args", "Name:=", "&lt;netName&gt;", "Vis:=", &lt;highlightFlag&gt;, ...])</code>
<b>Python Example</b>	<pre> oEditor.SetNetVisible( [     "NAME:Args",     "Name:=", "Net_1",     "Vis:=", True ]) oEditor.SetNetVisible( [     "NAME:Args",     "Name:=", "&lt;NO-NET&gt;",     "Vis:=", False,     "Name:=", "Ground",     "Vis:=", False,     "Name:=", "Chassis",     "Vis:=", True ]) </pre>

<b>VB Syn- tax</b>	<code>SetNetVisible Array ("Name:Args", "Name:=", "&lt;netName&gt;", "Vis:=", &lt;visibilityFlag&gt;, ...)</code>
----------------------------	-------------------------------------------------------------------------------------------------------------------

**VB  
Exa-  
mple**

```
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)
```

## Port Methods

The following port methods are available:

- [AddCircuitRefPort](#)
- [AddPortsToAllNets](#)
- [AddPortsToNets](#)
- [AddRefPort](#)
- [AlignPorts](#)
- [AssignCircuitRefPort](#)
- [AssignRefPort](#)
- [ClearRefPort](#)
- [CreateCircuitPort](#)
- [CreateEdgePort](#)
- [CreateNPort](#)
- [CreatePortInstancePorts](#)
- [CreatePortsOnComponents](#)
- [GetPortInfo](#)
- [RemovePortsFromAllNets](#)

- [RemovePortsFromNet](#)
- [RemovePortsOnComponents](#)

## 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>, // 1<sup>st</sup> object  
    <object\_name>, // 2<sup>nd</sup> 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)

*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:* 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)

```
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)
```

## 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,
"dataemode:=", "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")
```

## **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
connss = oEditor.GetPortInfo("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 <b>Layout</b> window, right click the net and click <b>Port &gt; Remove Ports from Net</b> or on the <b>Nets</b> tab of the <b>Nets</b> window, highlight and right click one or more nets and choose <b>Remove Ports</b> .		
Parameters	Name	Type	Description
	<requiredKeyword>	String	Required parameter that must proceed all element names with the value "NAME:Nets".
Return Value	None.		

Python Syntax	RemovePortsFromNet(["NAME:Nets", <netName>, ...])
---------------	---------------------------------------------------

<b>Python Example</b>	<pre>oEditor.RemovePortsFromNet (     [         "NAME:Nets",         "NET_1",         "NET_2",         "NET_3"     ] )</pre>
-----------------------	------------------------------------------------------------------------------------------------------------------------------

<b>VB Syntax</b>	RemovePortsFromNet Array("NAME:Nets", <netName>, ...)
<b>VB Example</b>	oEditor.RemovePortsFromNet Array ("NAME:Nets", "NET_1", "NET_2", "NET_3")

## RemovePortsOnComponents (Layout Editor)

Remove ports on port instances of selected components. Multiple nets can be listed.

<b>UI Access</b>	Right click and choose <b>Port &gt; Remove Ports From Component</b> .									
<b>Parameters</b>	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>&lt; requiredKeyword&gt;</td><td>String</td><td>Required parameter that must proceed all element names with the value "NAME:elements".</td></tr><tr><td>&lt;elementName&gt;</td><td>String</td><td>Name of a component. Ports are removed from the port instances of the selected components.</td></tr></tbody></table>	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.
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.								
<b>Return Value</b>	None.									

<b>Python Syntax</b>	RemovePortsOnComponents (["NAME:elements", <elementName>, ...])
<b>Python Example</b>	<pre> oEditor.RemovePortsOnComponents (     [         "NAME:elements",         "0"     ] ) </pre>

<b>VB Syntax</b>	RemovePortsOnComponents Array("NAME:elements", " <element-name>", ...)
<b>VB Example</b>	<pre> oEditor.RemovePortsOnComponents Array ("NAME:elements", "0") </pre>

## Miscellaneous Methods

The following miscellaneous methods are available.

- [AlignObjects \(Layout Editor\)](#)
- [ChangeOptions \(Layout Editor\)](#)
- [ClipPlane \(Layout Editor\)](#)
- [CopyToHFSS 3D Layout \(Layout Editor\)](#)
- [CutOutSubDesign \(Layout Editor\)](#)
- [DefeatureObjects \(Layout Editor\)](#)
- [Duplicate \(Layout Editor\)](#)
- [Edit3DComponentDefinition \(Layout Editor\)](#)

- [EraseMeasurements \(Layout Editor\)](#)
  - [ExportDXF \(Layout Editor\)](#)
  - [ExportGDSII \(Layout Editor\)](#)
  - [ExportGerber \(Layout Editor\)](#)
  - [ExportNCDrill \(Layout Editor\)](#)
  - [GetComponentInfo \(Layout Editor\)](#)
  - [GetComlInstanceFromRefDes \(Layout Editor\)](#)
  - [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 \(Layout Editor\)](#)
  - [SelectAll \(Layout Editor\)](#)
  - [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" <sup>1</sup> "DrawingResolution" <sup>2</sup>
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" <sup>3</sup> "MinorColor" <sup>3</sup> "MajorSize" <sup>2</sup> "MinorSize" <sup>2</sup> "ShowGrid" <sup>1</sup>
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" <sup>4</sup>
Background color	Color of the layout background	white	"background color" <sup>3</sup>
Sketch Mode	Fill patterns and center lines are not drawn.	off	"DefaultToSketchMode" <sup>1</sup>
Solid Mode	Objects are filled in with solid color.	off	"fillMode" <sup>1</sup>
Draw Connection Points	Display pin symbols in the layout.	off	"show connection points" <sup>1</sup>
Draw Rats	Display lines indicating missing physical connections in nets	on	"draw rats" <sup>1</sup>
Label Vertices	Display property text labeling the vertices of selected polygons and lines.	off	"display vertex labels" <sup>1</sup>

Net Color	Color used for highlighting nets	light yellow	"NetColor" <sup>3</sup>
Rectangle Description	Specifies if rectangles are described with two points or center point, width, and height	2 points	"rectangle description" <sup>5</sup>
Default Hole Size	Default size for actual holes in a PC board	25 mil	"defaultholesize" <sup>2</sup>
Align Microwave Components	Snaps components on entry.	on	"snaptoport" <sup>1</sup>
Symbol Footprint Scaling	Used to size symbol footprints placed in layout.	Based on data extent and current length units.	"sym footprint scaling" <sup>6</sup>
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" <sup>3</sup> "secondary selection color" <sup>3</sup>
Preview Selection	Highlight the object that would be selected with a left mouse button click in the current location.	off	"preview selection" <sup>1</sup>
Snapping options	Choose snapping sources and targets and the maximum distance (in pixels) for snapping to occur. Constrain edits to grid if desired.	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" <sup>7</sup> "SnapTargetVertex_on" <sup>1</sup> "SnapTargetEdgeCenter_on" <sup>1</sup> "SnapTargetObjCenter_on" <sup>1</sup> "SnapTargetEdge_on" <sup>1</sup> "SnapTargetElecConnection_on" <sup>1</sup> "SnapTargetIntersection_on" <sup>1</sup> "SnapTargetGrid_on" <sup>1</sup> "SnapSourceVertex_on" <sup>1</sup> "SnapSourceEdgeCenter_on" <sup>1</sup>

	<b>Note:</b> Off-grid objects are ignored if ConstrainToGrid is asserted.		"SnapSourceEdge_on" <sup>1</sup> "SnapSourceElecConnection_on" <sup>1</sup> "ConstrainToGrid" <sup>1</sup>
Always Show Merge Layers Dialog	Controls display of the layer merging dialog.	off – Only show dialog when the user must be involved.	"AlwaysShowLayerMergeDlg" <sup>1</sup>
Rotation Increment	During rotation, objects are rotated by multiples of this amount.	Based on current angle units.	"anglesnap" <sup>2</sup>
Allow Drag on first click	Selection and move with one left mouse button click.	false	"AllowDragOnFirstClick" <sup>1</sup>

The footnotes in the table above correspond to the following:

<sup>1</sup> = Integer with a value of 1 for on/true and 0 for off/false

<sup>2</sup> = String containing a amount and units

<sup>3</sup> = Integer representing a Color

<sup>4</sup> = An Array containing (lower left x, lower left y, upper right x, upper right y)

<sup>5</sup> = Integer with a value of 0 for two points, and 1 for center/width/height

<sup>6</sup> = String holding a double.

<sup>7</sup> = Integer

## 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.

<b>UI Access</b>	From the <b>View</b> menu, select <b>Clip Plane</b> , or from the <b>View</b> ribbon, select <b>Add Clipping Plane</b> .
<b>Parameters</b>	None.
<b>Return Value</b>	None.

<b>Python Syntax</b>	ClipPlane()
<b>Python Example</b>	<pre>oEditor = oDesign.SetActiveEditor("Layout") Editor.ClipPlane ("n1, n2, n3, et cetera") success = oEditor.ClipPlane("n1, n2, n3, et cetera")</pre>

<b>VB Syntax</b>	ClipPlane
<b>VB Example</b>	<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")
```

## 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 <b>Edit Definition</b> .		
<b>Parameters</b>	Name <i>&lt;Parameters&gt;</i>	Type Array	Description 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.
<b>Return Value</b>	None.		
<b>Python Syntax</b>	Edit3DComponentDefinition (<Parameters>)		
<b>Python Example</b>	<pre>oEditor.Update3DComponentDefinitions(     [ "NAME:3D Components",         "Definitions:=", [ "My_Component" ],         "IsLocal:=", True,         &gt;EditFilePath:="", ""</pre>		

---

	)
--	---

<b>VB Syntax</b>	Edit3DComponentDefinition <Parameters>
<b>VB Example</b>	<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

## 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:/Anstranslator/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
	<OptionParameters>	Array	<p>Structured array.</p> <pre>Array("NAME:options",       "FileName:=", &lt;string, GDSII export file name.&gt;,       "NumVertices:=", &lt;integer, maximum number of vertices allowed in a polygon (specify zero if unlimited).&gt;,       "ArcTol:=", &lt;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.&gt;,       "LayerMap:=", Array(         "entry:=", Array(           "layer:=", &lt;string, name of a layer to export.&gt;,           "id:=", &lt;integer, GDSII layer ID to assign the exported layer.&gt;,           "include:=", &lt;boolean, if true, the layer is exported.))     )</pre>

<b>Return Value</b>	None.
---------------------	-------

<b>Python Syntax</b>	ExportGDSII(<OptionParameters>)
<b>Python Example</b>	<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,             "include:=", true]     ] ) </pre>

<b>VB Syntax</b>	ExportGDSII <OptionParameters>
------------------	--------------------------------

**VB Example**

```
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)))
```

**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)))
```

## **GetComponentInfo (Layout Editor)**

*Use:* Informational.

*Command:* None.

*Syntax:* GetComponentInfo<complD>

*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")

## **GetComplInstanceFromRefDes (Layout Editor)**

*Use:* Informational.

*Command:* None.

*Syntax:* GetComplInstanceFromRefDes (<object\_name>)

*Return Value:* Returns IDispatch for ComplInstance.

```
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.GetPropertyValues("BaseElementTab",_  
"rect_1", "Name")
```

## GetSelections (Layout Editor)

Returns an array of currently selected objects.

<b>UI Access</b>	N/A
<b>Parameters</b>	None.
<b>Return Value</b>	Array containing object IDs

<b>Python Syntax</b>	GetSelections()
<b>Python Example</b>	<code>oEditor.GetSelections ()</code>

<b>VB Syntax</b>	GetSelections
<b>VB Example</b>	<code>oEditor.GetSelections</code>

## Heal (Layout Editor)

*Use:* Draw > Geometry Healing

*Command:* Heal( [ "NAME:<category>", "Type:=", "<type>", "Tol:=", "<tolerance>", <args>] )

<category> can have these values: Snap, Feature, or Repair

<type> will have these values:

For category *Snap*: Point, Arc, Grid

For category *Feature*: Voids, FloatingBodies

For category *Repair*: Colinear, SelfIntersecting

For the category *Feature*, type *Voids* there is an additional optional argument: "AntiPads:=" true or false.

*Syntax:*

```
Heal( [ "NAME:<category>", "Type:=", "<type>", "Tol:=", "<tolerance>", <args>] )
```

```
Heal( [ "NAME:<category>", "Selection:=", [<obj1>, <obj2>, ... ], "Type:=", "<type>", "Tol:=", "<tolerance>", <args>] )
```

*Return Value:* None

*VB Example:*

```
oProject = oDesktop.SetActiveProject("GeometryHealing")
oDesign = oProject.SetActiveDesign("TestDesign")
oEditor = oDesign.SetActiveEditor("Layout")
oEditor.Heal(
[
    "NAME:Snap",
    "Type:=" , "Point",
    "Tol:=" , "0.1mm"
])
oEditor.Heal(
[
    "NAME:Snap",
    "Type:=" , "Arc",
```

```
"Tol:=" , "0.1mm"
])
oEditor.Heal(
[
"NAME:Snap",
"Type:=" , "Grid",
"Tol:=" , "0.1mm"
])
oEditor.Heal(
[
"NAME:Feature",
"Type:=" , "Voids",
"AntiPads:=" , False,
"Tol:=" , "4mm2"
])
oEditor.Heal(
[
"NAME:Feature",
"Type:=" , "FloatingBodies",
"AntiPads:=" , False,
"Tol:=" , "4mm2"
```

```
])
oEditor.Heal(
[
"NAME:Repair",
"Type:=" , "Colinear",
"Tol:=" , "1e-007mm"
])
oEditor.Heal(
[
"NAME:Repair",
"Type:=" , "SelfIntersecting",
"Tol:=" , "1e-007mm"
])
```

## **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	<ul style="list-style-type: none"> <li>• Layout Editor</li> <li>• Schematic Editor</li> <li>• Select a Nexxim solution in a 3D Layout design, right click, and choose <b>Push Excitations</b>.</li> </ul>									
Parameters	<table border="1" data-bbox="460 1049 1890 1390"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td data-bbox="460 1049 834 1220">&lt;Reference Designation&gt;</td><td data-bbox="834 1049 960 1220">String</td><td data-bbox="960 1049 1890 1220"> <p>"&lt;refdes&gt;"</p> <p>This argument is empty when excitations are pushed from a Nexxim solution.</p> </td></tr> <tr> <td data-bbox="460 1220 834 1390">&lt;PushExcitations Parameters&gt;</td><td data-bbox="834 1220 960 1390">Array</td><td data-bbox="960 1220 1890 1390"> <p>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.</p> </td></tr> </tbody> </table>	Name	Type	Description	<Reference Designation>	String	<p>"&lt;refdes&gt;"</p> <p>This argument is empty when excitations are pushed from a Nexxim solution.</p>	<PushExcitations Parameters>	Array	<p>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.</p>
Name	Type	Description								
<Reference Designation>	String	<p>"&lt;refdes&gt;"</p> <p>This argument is empty when excitations are pushed from a Nexxim solution.</p>								
<PushExcitations Parameters>	Array	<p>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.</p>								

For a linear frequency solution, use this array:

```
Array("NAME:options",
      "CalcThevenin =", <true|false>,
      "Sol:=", "<solution name>")
```

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:

```
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>")
```

`winType` can have the following values:

- Rectangular
- Bartlett

			<ul style="list-style-type: none"> <li>• Blackman</li> <li>• Hamming</li> <li>• Hanning</li> <li>• Kaiser</li> <li>• Welch</li> <li>• Weber</li> <li>• Lanzcos</li> </ul>
<b>Return Value</b>	None		

**Note:**

For more information about the component name, schematic IDs, and formats, see the section [Editor Scripting IDs](#).

<b>VB Syntax</b>	PushExcitations "<refdes>", Array("NAME:options", "transient:=", ["CalcThevenin =", <true>], Array("start:=", <start time>, "stop:=", <stop time>, "maxHarmonics:=", <max harmonics>, "winType:=", <>window>, ["widthPct:=", <width percentage>], ["kaiser:=", <Kaiser value>], ["correctCoherentGain:=", true]), "Sol:=", "<solution name>")
<b>VB Example</b>	<p>For a transient solution:</p> <pre>Set oEditor = oDesign.SetActiveEditor("Layout") oEditor.PushExcitations "U3", Array("NAME:options", _ "transient:=", Array("start:=", 0, "stop:=", 5E-005, _ "maxHarmonics:=", 100, "winType:=", "Rectangular", _</pre>

```
"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")
```

## SelectAll (Layout Editor)

*Use:* Select all elements in the layout editor.

*Command:* None.

*Syntax:* `SelectAll()`

*Parameters:* None

*VB Example:* `Dim removedDefs`

```
removedDefs = oDefinitionEditor.SelectAll()
```

## 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

**GetPropertyValues** 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.

# 29 - 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](#)

[GetPropertyValues](#)

[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",  
""Radian", "Thermal", or <VHDLPackageName>

```
<VHDLPackageName>
<string> // in the form <Library>.<Package>
```

```
<Library>
<string> // name of the VHDL library
```

```
<Package>
<string> // name of the VHDL package
```

```
<VariableInfo>
Array(<string>, // name
<FlagLetters>,
<string>, // description
"CB:=", <string>, // optional - script for call back
<string>) // value: number, variable, or expression
```

```
<FlagLetters>
<string> // "D" - has description parameter,
// "RD" - readonly & has description parameter,
```

```
// or "RHD" - readonly, hidden, & has description parameter
```

```
<CheckBoxInfo>:  
    Array(<string>, // name  
          <FlagLetters>,  
          <string>, // description  
          "CB:=", <string>, // optional - script for call back  
          <bool>) // value: true or false
```

```
<ButtonInfo>:  
    Array(<string>, // name  
          <FlagLetters>,  
          <string>, // description  
          <string>, // button title  
          <string>, // extra text  
          <ClientID>,  
          "ButtonPropClientData:= ", <ClientdataArray>)
```

```
<ClientID>:  
    <int> // specifies Button Prop Client  
    // 0 - unknown, ButtonPropClientData  
    // array will be empty
```

```
// 1 - Netlist Prop Client  
// 2 - not used  
// 3 - File Name Prop Client
```

<ClientdataArray>:

varies with <ClientID>

<ClientId> is 0 or 1: empty array

Array()

<ClientID> is 3:

Array("InternalFormatText:", "<prefix><RelativePath>")

<prefix>:

<string> // "<Project>", "<PersonalLib>", "<UserLib>", or "<SysLib>"

<RelativePath>:

<string> // relative path to file from <prefix>

<TextInfo>:

```
Array(<string>, // name  
<FlagLetters>,  
<string>, // description  
"CB:=", <string>, // optional - script for call back  
<string>) // value: a text string
```

```
<NumberInfo>:  
Array(<string>, // name  
<FlagLetters>,  
<string>, // description  
"CB:=", <string>, // optional - script for call back  
<real>, // value: a number  
<string>) // units
```

```
<SeparatorInfo>:  
Array(<string>, // name  
<FlagLetters>,  
<string>, // description  
"CB:=", <string>, // optional - script for call back  
<string>) // value: a text string
```

```
<ValueInfo>:
```

```
Array(<string>, // name
      <FlagLetters>,
      <string>, // description
      "CB:=", <string>, // optional - script for call back
      <string>) // value: a number, variable or expression

<MenuPropInfo>:
  Array(<string>, // name
        <FlagLetters>,
        <string>, // description
        <string>, // menu choices - separated by commas
        <int>) // 0 based index of current menu choice

<VPointInfo>:
  Array(<string>, // name
        <FlagLetters>,
        <string>, // description
        "CB:=", <string>, // optional - script for call back
        <string>, // x value: number with length units
        <string>) // y value: number with length units
```

```
<PointInfo>:  
  Array(<string>, // name  
        <FlagLetters>,  
        <string>, // description  
        "CB:=", <string>, // optional - script for call back  
        <real>, // x value  
        <real>) // y value
```

```
<QuantityPropInfo>:  
  Array(<string>, // name  
        <FlagLetters>,  
        <string>, // description  
        <string>, // value  
        <TypeString>,  
        <TypeStringDependentInfo>)
```

```
<TypeString>:  
  <string> // "Across", "Through", or "Free"
```

```
<TypeStringDependentInfo>:
```

```
<TypeString> is "Free" :
<string>, // direction: "In", "Out", "InOut", or "DontCare"
// Following <string> is not present if direction is "DontCare"
<string> // when to calculate: "BeforeAnalogSolver",
// "BeforeStateGraph", "AfterStateGraph", or "DontCareWhen"

<TypeString> is "Across" or "Through":
<int>, // terminal 1
<int> // terminal 2

<CosimDefInfo>:
Array("NAME:CosimDefinition",
"CosimulatorType:=", <int>,
"CosimDefName:=", <string> // "HFSS 3D Layout", "Circuit",
// "Custom", or "Netlist"
"IsDefinition:=", <bool>,
final array member(s) vary with CosimDefName)

final array members for HFSS 3D Layout:
"CosimStackup:=", <string>,
"CosimDmbedRatio:=", <int>
```

final array members for Circuit:

```
"ExportAsNport:=", <int>,  
"UsePjt:=", <int>
```

final array member for Custom:

```
"DefinitionCompName:=", <string>
```

final array member for Netlist:

```
"NetlistString:=", <string>
```

*VB Example:*

```
Dim name  
  
oComponentManager.Add (Array("NAME:MyComponent", _  
"Info:=", Array("Type:=", 4294901767, _  
"NumTerminals:=", 2, _  
"DataSource:=", "", _  
"ModifiedOn:=", 1071096503, _  
"Manufacturer:=", "Ansys", _  
"Symbol:=", "bendo", _  
"Footprint:=", "BENDO", _
```

```
"Description:=", "", _  
"InfoTopic:=", "", _  
"InfoHelpFile:=", "", _  
"IconFile:=", "", _  
"LibraryName:=", "", _  
"OriginalLocation:=", "Project", _  
"Author:=", "", _  
"OriginalAuthor:=", "", _  
"CreationDate:= ", 1147460679), _  
"Refbase:=", "U", _  
"NumParts:=", 1, _  
"OriginalComponent:=", "", _  
"Terminal:=", Array("n1", _  
"n1", _  
"A", _  
false, _  
0, _  
1, _  
"", _  
"Electrical"), _
```

```
"Terminal:=", Array("n2", _  
"n2", _  
"A", _  
false, _  
1, _  
0, _  
"", _  
"Electrical"), _  
Array("NAME:Parameters", _  
"MenuProp:=", Array("CoSimulator", _  
"D", _  
"", _  
"Default, HFSS 3D Layout,Circuit,Custom,Netlist", _  
0), _  
"ButtonProp:=", Array("CosimDefinition", _  
"D", _  
"", _  
"", _  
"Edit", _  
0, _  
"ButtonPropClientData:=", Array()), _
```

```
Array("NAME:CosimDefinitions", _  
    Array("NAME:CosimDefinition", _  
        "CosimulatorType:=", 0, _  
        "CosimDefName:=", "HFSS 3D Layout", _  
        "IsDefinition:=", true, _  
        "CosimStackup:=", "Layout stackup", _  
        "CosimDmbedRatio:=", 3), _  
    Array("NAME:CosimDefinition", _  
        "CosimulatorType:=", 1, _  
        "CosimDefName:=", "Circuit", _  
        "IsDefinition:=", true, _  
        "ExportAsNport:=", 0, _  
        "UsePjt:=", 0), _  
    Array("NAME:CosimDefinition", _  
        "CosimulatorType:=", 2, _  
        "CosimDefName:=", "Custom", _  
        "IsDefinition:=", true, _  
        "DefinitionCompName:=", ""), _  
    Array("NAME:CosimDefinition", _  
        "CosimulatorType:=", 3, _
```

```
"CosimDefName:=", "Netlist", _  
"IsDefinition:=", true, _  
"NetlistString:=", "") ))
```

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.**

**Python Syntax**

```
["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:=", &lt;CheckBoxInfo&gt;, "TextProp:=", &lt;TextInfo&gt;, "NumberProp:=", &lt;NumberInfo&gt;, "SeparatorProp:=", &lt;SeparatorInfo&gt;, "ValueProp:=", &lt;ValueInfo&gt;, "MenuProp:=", &lt;MenuInfo&gt;, "VPointProp:=", &lt;VPointInfo&gt;, "PointProp:=", &lt;PointInfo&gt;], ["Quantities", "QuantityProp:=", &lt;QuantityPropInfo&gt; ...], ["NAME:CosimDefinitions", &lt;CosimDefInfo&gt;, &lt;CosimDefInfo&gt;...]]</pre>
<b>Python Example</b>	<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
	< <i>DatasetdataArray</i> >	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 < <i>DatasetdataArray</i> >
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>
<b>VB Example</b>	<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.

<b>UI Access</b>	N/A		
<b>Parameters</b>	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
<b>Return Value</b>	A property scripting object for the definition.		

P-y-t-h-o-n-S-y-n-t-

AddDefinitionFromBlock(<defBlock>,<defFolderName>,<newTimeStamp>,<replaceExisting>)

ax	
P- y- t- 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>, ... ]

		<pre>"component1:=", &lt;value&gt;, "component2:=", &lt;value&gt;, "component3:=", &lt;value&gt;)) ]</pre>
<i>&lt;PropertyName&gt;</i>	String	<p>Should be one of the following (depending on the material, design, and solution types):</p> <p>Electromagnetic (Maxwell-exclusive material properties omitted, see Maxwell Scripting help):</p> <ul style="list-style-type: none"> <li>"permittivity", "permeability", "conductivity", "dielectric_loss_tangent",</li> <li>"magnetic_loss_tangent", "electric_coercivity", "magnetic_coercivity",</li> <li>"saturation_mag", "lande_g_factor", "delta_H", "delta_h_freq",</li> <li>"mass_density"</li> </ul> <p>Thermal (including solids, Icepak fluid flow, and Mechanical rotating fluid modeling):</p> <ul style="list-style-type: none"> <li>"thermal_conductivity", "mass_density", "specific_heat",</li> <li>"thermal_expansion_coefficient", "thermal_material_type", "viscosity",</li> <li>"diffusivity", "molecular_mass", "clarity_type"</li> </ul> <p>Structural:</p> <ul style="list-style-type: none"> <li>"mass_density", "youngs_modulus", "poissons_ratio",</li> <li>"thermal_expansion_coefficient"</li> </ul>

	<code>&lt;Unit&gt;</code>	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"
<b>Return Value</b>	None		

<b>Python Syntax</b>	<code>AddMaterial (["NAME:&lt;MaterialName&gt;","&lt;MatProperty&gt;, &lt;MatProperty&gt;, ...])</code>
<b>Python Example</b>	<code>oDefinitionManager.AddMaterial(["permittivity:=", "2.2", "0.002"])</code>

```

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") ]

```

<b>VB Syntax</b>	AddMaterial Array("NAME:<MaterialName>",     <MatProperty>, <MatProperty>, ...)
<b>VB Example</b>	<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",_ </pre>

```
"component1:=", "11", _  
"component2:=", "22", _  
"component3:=", "33"), _  
"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:lgm",

```
"lay:=", <string>, // definition layer name  
"id:=", <int>, // definition layer id  
"pad:=", <PadInfo>, // pad  
"ant:=", <PadInfo>, // antipad  
"thm:=", <PadInfo>, // themal pad  
"X:=", <string>, // pad x connection, real with units  
"Y:=", <string>, // pad y connection, real with units  
"dir:=", <DirectionString>) // pad connection direction
```

<PadInfo>:

```
Array("shp:=", <PadShape>,  
"Szs:=", <DimensionArray>,  
"X:=", <string>, // x offset, real with units  
"Y:=", <string>, // y offset, real with units  
"R:=", <string>) // rotation, real with units
```

<PadShape>:

```
<string> one of these choices  
"No" // no pad  
"Cir" // Circle
```

```
"Sq" // Square  
"Rct" // Rectangle  
"Ov" // Oval  
"Blt" // Bullet  
"Ply" // Polygons  
"R45" // Round 45 thermal  
"R90" // Round 90 thermal  
"S45" // Square 45 thermal  
"S90" // Square 90 thermal
```

<DimensionArray>:

```
Array(<string>, ...) // each string is a real with units for one of the dimensions of the shape
```

<DirectionString>:

<string> one of these choices

```
"No" // no direction  
"Any" // any direction  
"0" // 0 degrees  
"45" // 45 degrees  
"90" // 90 degrees  
"135" // 135 degrees
```

```
"180" // 180 degrees
"225" // 225 degrees
"270" // 270 degrees
"315" // 315 degrees

<HoleRange>:
<string> one of these choices
"Thr" // through all layout layers
"Beg" // from upper pad layer to lowest layout layer
"End" // from upper layout layer to lowest pad layer
"UTL" // from upper pad layer to lowest pad layer

<SolderballShape>:
<string> one of these choices
"None" // no solderball
"Cyl" // cylinder solderball
"Sph" // spheroid solderball

<SolderballPlacement>:
<string> one of these choices
"abv" // above padstack
"blw" // below padstack
```

<PadPortLayerArray>:

Array( <int>, <int>,....) where each int is a layer id

*VB Example:*

```

oPadstackManager.Add Array("NAME:Circle - through3", "ModTime:=", 1235765635, "Library:=", _
"", "LibLocation:=", "Project", Array("NAME:psd", "nam:=", "Circle - through3", "lib:=", _
 "", "mat:=", "", "plt:=", "0", Array("NAME:pds", Array("NAME:lgm", "lay:=", "Top Signal",
"id:=", _
0, "pad:=", Array("shp:=", "Cir", "Szs:=", Array("2.5mm"), "X:=", "0mm", "Y:=", _
"0mm", "R:=", "0deg"), "ant:=", Array("shp:=", "Cir", "Szs:=", Array("3.5mm"), "X:=", _
"0mm", "Y:=", "0mm", "R:=", "0"), "thm:=", Array("shp:=", "No", "Szs:=", Array(), "X:=", _
"0mm", "Y:=", "0mm", "R:=", "0"), "X:=", "0mm", "Y:=", "0mm", "dir:=", "Any"), Array("NAME:lgm",
"lay:=", _
"SignalA", "id:=", 1, "pad:=", Array("shp:=", "Cir", "Szs:=", Array("2mm"), "X:=", _
"0mm", "Y:=", "0mm", "R:=", "0deg"), "ant:=", Array("shp:=", "Cir", "Szs:=", Array( _
"3mm"), "X:=", "0mm", "Y:=", "0mm", "R:=", "0"), "thm:=", Array("shp:=", "No", "Szs:=", Array(),
"X:=", _
"0mm", "Y:=", "0mm", "R:=", "0"), "X:=", "0mm", "Y:=", "0mm", "dir:=", "Any"), Array("NAME:lgm",
"lay:=", _
"SignalB", "id:=", 2, "pad:=", Array("shp:=", "Cir", "Szs:=", Array("2mm"), "X:=", _
"0mm", "Y:=", "0mm", "R:=", "0deg"), "ant:=", Array("shp:=", "Cir", "Szs:=", Array( _
"3mm"), "X:=", "0mm", "Y:=", "0mm", "R:=", "0deg"), "thm:=", Array("shp:=", "No", "Szs:=", Array(),
"X:=", _

```

```
"0mm", "Y:=", "0mm", "R:=", "0"), "X:=", "0mm", "Y:=", "0mm", "dir:=", "Any"), Array("NAME:lgm",
"lay:=", _
"Ground", "id:=", 3, "pad:=", Array("shp:=", "No", "Szs:=", Array(), "X:=", _
"0mm", "Y:=", "0mm", "R:=", "0deg"), "ant:=", Array("shp:=", "No", "Szs:=", Array(), "X:=", _
"0mm", "Y:=", "0mm", "R:=", "0"), "thm:=", Array("shp:=", "R90", "Szs:=", Array( _
"3mm", "0.75mm", "1mm"), "X:=", "0mm", "Y:=", "0mm", "R:=", "0"), "X:=", "0mm", "Y:=", _
"0mm", "dir:=", "Any"), Array("NAME:lgm", "lay:=", "Bottom signal", "id:=", 5, "pad:=", Array(
("shp:=", _
"Cir", "Szs:=", Array("1mm"), "X:=", "0mm", "Y:=", "0mm", "R:=", "0deg"), "ant:=", Array("shp:=", ,
-
"Cir", "Szs:=", Array("2mm"), "X:=", "0mm", "Y:=", "0mm", "R:=", "0deg"), "thm:=", Array("shp:=", ,
-
"No", "Szs:=", Array(), "X:=", "0mm", "Y:=", "0mm", "R:=", "0"), "X:=", "0mm", "Y:=", _
"0mm", "dir:=", "Any)), "hle:=", Array("shp:=", "Cir", "Szs:=", Array("1.5mm"), "X:=", ,
"0mm", "Y:=", "0mm", "R:=", "0deg"), "hRg:=", "End", "sbsh:=", "Sph", "sbpl:=", ,
"abv", "sbr:=", "750um", "sb2:=", "1200um", "1200um", "sbn:=", "solder"), "ppl:=", Array( _
0, 1, 2, 3, 5))
```

## 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.

<b>UI Access</b>	None.		
<b>Parameters</b>	Name material_data	Type <string>	Description 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
<b>Return Value</b>	Boolean: <ul style="list-style-type: none"> <li>• <b>True</b> – named materials have equivalent major properties.</li> <li>• <b>False</b> – named materials do not have equivalent major properties.</li> </ul>		

<b>Python Syntax</b>	AreMaterialPropertiesEqual(<material_data>, '<MaterialName>')
<b>Python Example</b>	<code>oDefinitionManager.AreMaterialPropertiesEqual(material_data, 'Mold_Material')</code>

<b>VB Syntax</b>	AreMaterialPropertiesEqual <material_data>, <MaterialName>
<b>VB Example</b>	<code>oDefinitionManager.AreMaterialPropertiesEqual material_data, "modified_epoxy"</code>

## AreSurfaceMaterialPropertiesEqual

Checks whether named surface material compared to added surface material data have equivalent major properties.

<b>UI Access</b>	None.		
<b>Parameters</b>	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"]
<b>Return Value</b>	<b>Boolean:</b> <ul style="list-style-type: none"><li>• <b>True</b> – named surface materials have equivalent major properties.</li></ul>		

- |  |                                                                                                                                     |
|--|-------------------------------------------------------------------------------------------------------------------------------------|
|  | <ul style="list-style-type: none"> <li>• <b>False</b> – named surface materials do not have equivalent major properties.</li> </ul> |
|--|-------------------------------------------------------------------------------------------------------------------------------------|

<b>Python Syntax</b>	AreSurfaceMaterialPropertiesEqual(<material_data>, '<MaterialName>')
<b>Python Example</b>	<pre>oDefinitionManager.AreSurfaceMaterialPropertiesEqual (material_data, 'Mold_Material')</pre>

<b>VB Syntax</b>	AreMaterialPropertiesEqual <material_data>, <MaterialName>
<b>VB Example</b>	<pre>oDefinitionManager.AreMaterialPropertiesEqual material_data, "modified_epoxy"</pre>

## ChangeProperty (Symbol Editor)

**Use:** Changes to properties are scripted using the **ChangeProperty** command. This command can be executed by the **oEditor** to change editor properties, by the **oDesign** to change design level properties, and by the **oProject** to change project level properties. The command can be used to create, edit, and/or remove properties. In Circuit, only Variable and Separator properties can be deleted.

**Command:** None

**Syntax:** ChangeProperty Array("Name:AllTabs", <PropTabArray>, <PropTabArray>, ...)

**Return Value:** None

**Parameters:** <PropTabArray>

```
Array("Name:<PropTab>",
      <PropServersArray>,
      <NewPropsArray>,
      <ChangedPropsArray>,
```

```
<DeletedPropsArray>
```

```
<PropServersArray>
```

```
Array("Name:PropServers", <PropServer>,
<PropServer>, ...)
```

```
<NewPropsArray>
```

```
Array("Name:NewProps", <PropdataArray>,
<PropdataArray>, ...)
```

```
<ChangedPropsArray>
```

```
Array("Name:ChangedProps", <PropdataArray>,
<PropdataArray>, ...)
```

```
<DeletedPropsArray>
```

```
Array("Name:DeletedProps", <PropName>,
<PropName>, ...)
```

OR (for PropDisplay deletions only)

```
Array("Name:DeletedProps", <PropdataArray>,
<PropdataArray>, ...)
```

```
<PropdataArray>
  Array("NAME:<PropName>",
        "PropType:=", <PropType>,
        "NewName:=", <string>,
        "Description:=", <string>,
        "Callback:=", <string>,
        "NewRowPosition:=", <int>,
        "ReadOnly:=", <bool>,
        "Hidden:=", <bool>,
        <PropTypeSpecificArgs>)OR (for PropDisplays only)
  Array("Name:<PropName>",<PropDisplayData>

<PropDisplayData>
  for adding, changing, deleting PropDisplays
  <PropDisplayAttributes>
    for changing PropDisplays only
    <PropDisplayNewAttributes>

<PropDisplayAttributes>
```

## Layer & Location only used for PropDisplays in layout

For adding PropDisplays, this will add a single PropDisplay with attributes as shown; if an attribute is missing, a default value will be assigned. Adding PropDisplay to schematic with attributes that are identical to one already existing there will fail without an error message.

For deleting PropDisplays, these attributes are used to identify an existing PropDisplay to delete. If there doesn't exist a PropDisplay that matches the given attributes, then nothing will be deleted. If multiple PropDisplays match the given attributes, then all of them will be deleted. If an attribute is missing, then all PropDisplays match that missing attribute. For example, if Layer is missing, then PropDisplays on all layers that match the remaining given attributes will be deleted.

For changing PropDisplays, these attributes are used to identify an existing PropDisplay to change. If no PropDisplay matching the attributes is found, no changes will be made. If multiple PropDisplays match the attributes, all of them will be changed. If an attribute is missing, it matches all PropDisplays. For example, to change the format of PropDisplays that are on the bottom, but have any layer, style or format to show the name only, this command should have Location set to "Bottom" and all other attributes omitted.

```
"Format:=", <PropDisplayType>,
"Location:=", <PropDisplayLocation>,
"Layer:=", <string>,
"Style:=", <string>

<PropDisplayNewAttributes>
```

## NewLayer & NewLocation only used for PropDisplays in layout

For changing PropDisplays, these attributes are used to identify which attributes to change and what the new value is. If the attribute should not be changed, the corresponding entry should be omitted.

```
"NewName:=", <string>,  
"NewFormat:=", <PropDisplayType>,  
"NewLocation:=", <PropDisplayLocation>,  
"NewLayer:=", <string>,  
"NewStyle:=", <string>
```

<PropDisplayType>

Type: string

Identifies the format of PropDisplay.

"Name"

"Value"

"NameAndValue"

"EvaluatedValue"

"NameAndEvaluatedValue"

<PropDisplayLocation>

Type: string

Identifies where PropDisplay is located with respect to object

"Left"

"Top"

"Right"

"Bottom"

"Custom"

<PropType>

Type: string

Identifies the type of property when a new property is added. In Circuit, only separator properties and variable properties can be added.

"SeparatorProp"

"VariableProp"

"TextProp"

"NumberProp"

"ValueProp"

"CheckboxProp"

"MenuProp"

"PointProp"

"VPointProp"

"ButtonProp"

NewName

Specify the new name of a property if the property's name is being edited. In Circuit, the name can only be changed for separators and variables.

#### Description

Specify a description of the property. In Circuit, the description can only be changed for separators and variables.

#### Callback

Specify the name of the script callback to be run when the property value is changed.

#### NewRowPosition

Used to reorder rows in the **Property** dialog box. In Circuit, this only applies to the **Project>Project Variables** panel and the **Designer>DesignProperties** panel. Specify the new zero-based row index of the variable or separator.

#### ReadOnly

Used to mark a property as "read only" so it can not be modified. In Circuit, this flag can only be set for variables and separators.

#### Hidden

Used to hide a property so it can not be viewed outside of the **Property** dialog box. In Circuit, this flag can only be set for variables and separators.

#### <PropTypeSpecificArgs>

**SeparatorProp:** no arguments

TextProp: "Value:=", <string>  
NumberProp: "Value:=", <double>  
ValueProp: "Value:=", <value>  
CheckboxProp: "Value:=", <bool>  
MenuProp: "Value:=", <string>  
PointProp "X:=", <double>, "Y:=", <double>  
VPointProp: "X:=", <value>, "Y:=", <value>  
Material Button: "Material:=", <string>  
Color Button: "R:=", <int>, "G:=", <int>, "B:=", <int>  
Transparency Button: "Value:=", <double>

<PropTypeSpecificArgs> for MenuProps  
Syntax for NewProps array: "AllChoices:=",  
<"choice1,choice2,..."> or <Array("choice1" "choice2", ... )>,  
"Value:=", <string>  
Syntax for ChangedProps array: "Value:=", <string>

<PropTypeSpecificArgs> for VariableProps  
Syntax:  
**"Value:=", <value>, <OptimizationFlagsArray>,**

```
<TuningFlagsArray>, <SensitivityFlagsArray>,
<StatisticsFlagsArray>
```

Parameters:

```
<OptimizationFlagsArray>
Array("NAME:Optimization",
"Included:=", <bool>,
"Min:=", <value>,
"Max:=", <value>)
```

```
<TuningFlagsArray>
Array("NAME:Tuning",
"Included:=", <bool>,
"Step:=", <value>,
"Min:=", <value>,
"Max:=", <value>)
```

```
<SensitivityFlagsArray>
Array("NAME:Sensitivity",
"Included:=", <bool>,
"Min:=", <value>,
"Max:=", <value>,
```

"IDisp:=", <value> )

<StatisticsFlagsArray>

Array("NAME:Statistical",

"Included:=", <bool>,

"Dist:=", <Distribution>,

"StdD:=", <value>,

"Min:=", <value>,

"Max:=", <value>,

"Tol:=", <string>)

<Distribution>

Type: string

Value should be "**Gaussian**" or "**Uniform**"

StdD

Standard deviation.

Min

Low cut-off for the distribution.

**Max**

High cut-off for the distribution.

**Tol**

Tolerance for uniform distributions. Format is "<int>%".

Example: "**20%**".

*VB Example:*

Adding a new project level variable "**\$width**":

```
oProject.ChangeProperty Array("NAME:AllTabs",_
Array("NAME:ProjectVariableTab",_
Array("NAME:PropServers", "ProjectVariables"),_
Array("NAME:NewProps",_
Array("NAME:$width",_
"PropType:=", "VariableProp",_
"value:=", "3mm",_
"description:=", "my new variable")))
```

*VB Example:* Deleting the design level variable "height":

```
oDesign.ChangeProperty Array("NAME:AllTabs",_
```

```
Array("NAME:LocalVariableTab",_
Array("NAME:PropServers", "DefinitionParameters"),_
Array("NAME:DeletedProps", "height"))
```

Changing a property's value. If the following command were executed, then the value of the property "XSize" of the PropServer

"Box1>CreateBox:1" on the "Geometry3DCmdTab" tab would be changed. (oEditor is the Geometry3D editor in Circuit.)

```
oEditor.ChangeProperty Array("NAME:AllTabs",_
Array("NAME:Geometry3DCmdTab",_
Array("NAME:PropServers", "Box1>CreateBox:1"),_
Array("NAME:ChangedProps",_
Array("NAME:XSize", "Value:=", "1.4mil"))))
```

*VB Example:* Changing a property's value. If the following command were executed, then the values of Callback and L on the PassedParameterTab would be changed.

```
oEditor.ChangeProperty Array("NAME:AllTabs", _
Array("NAME:PassedParameterTab", _
Array("NAME:PropServers", "CHOKE2"), _
Array("NAME:ChangedProps", _
```

```
Array("NAME:L", "Callback:=", "ac", "OverridingDef:=", true), _  
Array("NAME:L", "Value:=", "1nH"))))
```

*VB Example:* Changing the Company Name, Design Name, the background color, and the Axis scaling in a Report.

```
Set oProject = oDesktop.SetActiveProject("wgcombiner")  
Set oDesign = oProject.SetActiveDesign("CircuitDesign2")  
Set oModule = oDesign.GetModule("ReportSetup")  
oModule.ChangeProperty Array("NAME:AllTabs", Array("NAME:Header", _ Array("NAME:PropServers",  
"XY Plot1:Header")), _  
Array("NAME:ChangedProps", Array("NAME:Company Name", _  
"Value:=", "My Company"))))  
oModule.ChangeProperty Array("NAME:AllTabs", Array("NAME:Header", _ Array("NAME:PropServers", "XY  
Plot1:Header")), _  
Array("NAME:ChangedProps", Array("NAME:Design Name", _  
"Value:=", "WG Combiner"))))  
oModule.ChangeProperty Array("NAME:AllTabs", Array("NAME:General", _ Array("NAME:PropServers",  
"XY Plot1:General")), _  
Array("NAME:ChangedProps", Array("NAME:Back Color", _  
"R:=", 128, "G:=", 255, "B:=", 255))))  
oModule.ChangeProperty Array("NAME:AllTabs", Array("NAME:Axis", _ Array("NAME:PropServers", "XY  
Plot1:AxisX")), _
```

```
Array("NAME:ChangedProps", Array("NAME:Axis Scaling", _  
"Value:=", "Log")))
```

*VB Example:* Changing a property's value. Note that the AllChoices parameter is only used when the MenuProp is being added. Also note that either a string of choices separated by commas or an Array("choice1", "choice2", "choice3") works for the AllChoices parameter.

```
Set oEditor = oDesign.SetActiveEditor("SchematicEditor")  
  
oEditor.ChangeProperty Array("NAME:AllTabs", Array("NAME:PassedParameterTab", Array  
("NAME:PropServers", _  
"CompInst@CAP_2"), Array("NAME>NewProps", Array("NAME:xxxx", "PropType:=", _  
"MenuProp", "AllChoices:=", Array("aa", "bb", "cc", "dd"), "Value:=", "bb"))))
```

## CloneMaterial

Clones a local material.

<b>UI Access</b>	N/A									
<b>Parameters</b>	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>&lt;matName&gt;</td><td>String</td><td>Name of existing material.</td></tr><tr><td>&lt;newName&gt;</td><td>String</td><td>Name for newly cloned material.</td></tr></tbody></table>	Name	Type	Description	<matName>	String	Name of existing material.	<newName>	String	Name for newly cloned material.
Name	Type	Description								
<matName>	String	Name of existing material.								
<newName>	String	Name for newly cloned material.								
<b>Return Value</b>	Boolean: <ul style="list-style-type: none"><li>• 1 - Material is cloned.</li></ul>									

- |  |                                                                                                                               |
|--|-------------------------------------------------------------------------------------------------------------------------------|
|  | <ul style="list-style-type: none"> <li>• 0 - Existing material not found or a conflict with the new material name.</li> </ul> |
|--|-------------------------------------------------------------------------------------------------------------------------------|

<b>Python Syntax</b>	CloneMaterial (<matName>, <newName>)
----------------------	--------------------------------------

<b>Python Example</b>	<code>oDefinitionManager.CloneMaterial ("copper1", "copper3")</code>
-----------------------	----------------------------------------------------------------------

<b>VB Syntax</b>	CloneMaterial <matName>, <newName>
------------------	------------------------------------

<b>VB Example</b>	<code>oDefinitionManager.CloneMaterial "copper1", "copper3"</code>
-------------------	--------------------------------------------------------------------

## DeleteDataset

Deletes a specified dataset. This can be executed by the oProject, or oDesign variables.

<b>UI Access</b>	<b>Project &gt; Datasets &gt; Remove.</b>								
<b>Parameters</b>	<table border="1"> <tr> <td>Name</td> <td>Type</td> <td>Description</td> </tr> <tr> <td>&lt;DatasetName&gt;</td> <td>String</td> <td>Name of the dataset found in the project.</td> </tr> </table>			Name	Type	Description	<DatasetName>	String	Name of the dataset found in the project.
Name	Type	Description							
<DatasetName>	String	Name of the dataset found in the project.							
<b>Return Value</b>	None.								

<b>Python Syntax</b>	DeleteDataset (<DatasetName>)
----------------------	-------------------------------

<b>Python Example</b>	<code>oProject.DeleteDataset ('\$ds1')</code> <code>oDesign.DeleteDataset ('\$ds1')</code>
-----------------------	-----------------------------------------------------------------------------------------------

<b>VB Syntax</b>	DeleteDataset <DatasetName>
<b>VB Example</b>	<pre>oProject.DeleteDataset "\$ds1" oDesign.DeleteDataset "\$ds1"</pre>

## DoesMaterialExist

Checks for the presence of a material in the library by name

<b>UI Access</b>	None.		
<b>Parameters</b>	Name <MaterialName>	Type <String>	Description Name of the material to search for in the material database
<b>Return Value</b>	<p>Boolean:</p> <ul style="list-style-type: none"><li>• <b>True</b> – specified material exists.</li><li>• <b>False</b> – specified material does not exist.</li></ul>		

<b>Python Syntax</b>	DoesMaterialExist(<MaterialName>)
<b>Python Example</b>	<pre>oDefinitionManager.DoesMaterialExist("modified_epoxy")</pre>

<b>VB Syntax</b>	DoesMaterialExist <MaterialName>
<b>VB Example</b>	<pre>oDefinitionManager.DoesMaterialExist "modified_epoxy"</pre>

## Edit [component manager]

Modifies an existing component

*Command:* Tools > Edit Configured Libraries > Components > Edit Component

*Syntax:* Edit <ComponentName>,

```
Array("NAME:<NewComponentName>",
  "Info:=", <ComponentInfo>,
  "RefBase:=", <string>, // reference designator
  "NumParts:=", <int>, // parts per component
  "OriginalComponent:=", <string>
  "Terminal:=", <TerminalInfo>,
  "Terminal:=", <TerminalInfo>, ...
  // The remaining parameters are optional
  Array("NAME:Parameters", // any combo of the following
    "VariableProp:=", <VariableInfo>,
    "CheckboxProp:=", <CheckBoxInfo>,
    "ButtonProp:=", <ButtonInfo>,
    "TextProp:=", <TextInfo>,
    "NumberProp:=", <NumberInfo>,
    "SeparatorProp:=", <SeparatorInfo>,
    "ValueProp:=", <ValueInfo>,
    "MenuProp:=", <MenuInfo>),
```

```
Array("NAME:Properties", // any combo of the following
      "CheckboxProp:=", <CheckBoxInfo>,
      "TextProp:=", <TextInfo>,
      "NumberProp:=", <NumberInfo>,
      "SeparatorProp:=", <SeparatorInfo>,
      "ValueProp:=", <ValueInfo>,
      "MenuProp:=", <MenuInfo>),
      "VPointProp:=", <VPointInfo>,
      "PointProp:=", <PointInfo>),
      Array("Quantities",
            "QuantityProp:=", <QuantityPropInfo>...),
      Array("NAME:CosimDefinitions",
            <CosimDefInfo>,
            <CosimDefInfo>...)
```

*Return Value:* <string>

```
// composite name of the component.  
// If the name requested conflicts with the name of an existing  
// component, the requested name is altered to be unique.  
// The name returned reflects any change made to be unique.
```

*Parameters:* <ComponentName>:

<string> // [composite name](#) of the component to edit

<NewComponentName>:

<string> // new [simple name](#) for the component

<ComponentInfo>:

Array("Type:=", <TypeInfo>,

"NumTerminals:=", <int>,

"DataSource:=", <string>,

"ModifiedOn:=", <ModifiedOnInfo>,

"Manufacturer:=", "<string>,

"Symbol:=", <string>,

"Footprint:=", <string>,

"Description:=", <string>,

"InfoTopic:=", <string>,

"InfoHelpFile:=", <string>,

"IconFile:=", <string>,

"LibraryName:=", <string>,

"OriginalLocation:=", <string>, // Project Location

"Author:=", <string>,

```
"OriginalAuthor:=", <string>,
"CreationDate:=", <int>)
```

<TypeInfo>:

An integer that is the or-ing of bits for each product listed below. The default setting is 0xffffffff (4294967295) which indicates valid for all products. In the component editing dialog, checking different boxes in the "Specify products for which this component is valid" grid control sets specific flags that correspond to the following hex/decimal settings:

Nexxim -- 100 binary, 4 decimal, 0x4  
SIwaveDeNovo -- 1000 binary, 8 decimal, 0x8  
Simplorer -- 10000 binary, 16 decimal, 0x10  
MaxwellCircuit -- 100000 binary, 32 decimal, 0x20

<ModifiedOnInfo>:

An integer that corresponds to the number of seconds that have elapsed since 00:00 hours, Jan 1, 1970 UTC from the system clock.

<TerminalInfo>:

```
Array(<string>, // symbol pin
      <string> // footprint pin
      <string>, // gate name
      <bool>, // shared
```

```
<int>, // equivalence number  
<int>, // what to do if unconnected: flag as error:0, ignore:1  
<string>, // description  
<Nature>)
```

<Nature>:  
<string> // content varies as follows

Nexxim/Circuit:  
"Electrical" // the only choice

Simplorer:  
// several choices  
"Electrical", "Magnetic", "Fluidic", "Translational",  
"Translational\_V", "Rotational", "Rotational\_V",  
""Radian", "Thermal", or <VHDLPackageName>

<VHDLPackageName>:  
<string> // in the form <Library>.<Package>

<Library>:

```
<string> // name of the VHDL library
```

```
<Package>:
```

```
<string> // name of the VHDL package
```

```
<VariableInfo>:
```

```
Array(<string>, // name
```

```
<FlagLetters>,
```

```
<string>, // description
```

```
"CB:=", <string>, // optional - script for call back
```

```
<string>) // value: number, variable, or expression
```

```
<FlagLetters>:
```

```
<string> // "D" - has description parameter,
```

```
// "RD" - readonly & has description parameter,
```

```
// or "RHD" - readonly, hidden, & has description parameter
```

```
<CheckBoxInfo>:
```

```
Array(<string>, // name
```

```
<FlagLetters>,
<string>, // description
"CB:=", <string>, // optional - script for call back
<bool>) // value: true or false
```

```
<ButtonInfo>:
Array(<string>, // name
<FlagLetters>,
<string>, // description
<string>, // button title
<string>, // extra text
<ClientID>,
"ButtonPropClientData:= ", <ClientdataArray>)
```

```
<ClientID>:
<int> // specifies Button Prop Client
// 0 - unknown, "ButtonPropClientData"
// array will be empty
// 1 - Netlist Prop Client
// 2 - not used
// 3 - File Name Prop Client
```

<ClientdataArray>:

varies with <ClientID>

<ClientID> is 0 or 1: empty array

Array()

<ClientID> is 3:

Array("InternalFormatText:", "<prefix><RelativePath>")

<prefix>:

<string> // "<Project>", "<PersonalLib>", "<UserLib>", or "<SysLib>"

<RelativePath>:

<string> // relative path to file from <prefix>

<TextInfo>:

Array(<string>, // name

<FlagLetters>,

<string>, // description

```
"CB:=", <string>, // optional - script for call back  
<string>) // value: a text string
```

```
<NumberInfo>:  
  Array(<string>, // name  
        <FlagLetters>,  
        <string>, // description  
        "CB:=", <string>, // optional - script for call back  
        <real>, // value: a number  
        <string>) // units
```

```
<SeparatorInfo>:  
  Array(<string>, // name  
        <FlagLetters>,  
        <string>, // description  
        "CB:=", <string>, // optional - script for call back  
        <string>) // value: a text string
```

```
<ValueInfo>:  
  Array(<string>, // name  
        <FlagLetters>,  
        <string>, // description
```

```
"CB:=", <string>, // optional - script for call back  
<string>) // value: a number, variable or expression
```

```
<MenuPropInfo>:  
  Array(<string>, // name  
        <FlagLetters>,  
        <string>, // description  
        <string>, // menu choices - separated by commas  
        <int>) // 0 based index of current menu choice
```

```
<VPointInfo>:  
  Array(<string>, // name  
        <FlagLetters>,  
        <string>, // description  
        "CB:=", <string>, // optional - script for call back  
        <string>, // x value: number with length units  
        <string>) // y value: number with length units
```

```
<PointInfo>:  
  Array(<string>, // name  
        <FlagLetters>,
```

```
<string>, // description
"CB:=", <string>, // optional - script for call back
<real>, // x value
<real>) // y value

<QuantityPropInfo>:
Array(<string>, // name
<FlagLetters>,
<string>, // description
<string>, // value
<TypeString>,
<TypeStringDependentInfo>

<TypeString>:
<string> // "Across", "Through", or "Free"

<TypeStringDependentInfo>:

"Free":
<string>, // direction: "In", "Out", "InOut", or "DontCare"
// Following <string> is not present if direction is "DontCare"
```

```
<string> // when to calculate: "BeforeAnalogSolver",
// "BeforeStateGraph", "AfterStateGraph", or "DontCareWhen"

"Across" or "Through"
<int>, // terminal 1
<int> // terminal 2

<CosimDefInfo>:
Array("NAME:CosimDefinition",
"CosimulatorType:=", <int>,
"CosimDefName:=", <string> // "HFSS3D", "Circuit",
// "Custom", or "Netlist"
"IsDefinition:=", <bool>,
final array member(s) vary with CosimDefName)

final array members for HFSS 3D Layout:
"CosimStackup:=", <string>,
"CosimDmbedRatio:=", <int>

final array members for Circuit:
"ExportAsNport:=", <int>,
```

"UsePjt:=", <int>

final array member for Custom:

"DefinitionCompName:=", <string>

final array member for Netlist:

"NetlistString:=", <string>

*VB Example:*

```
Dim name

name = oComponentManager.Edit ("Nexxim Circuit Elements\BJTs:Level01_NPN", _
Array("NAME:Level01_NPN", _
"Info:=", Array("Type:=", 4294901764, _
"NumTerminals:=", 3, _
"DataSource:=", "Ansys built-in component", _
"ModifiedOn:=", 1152722112, _
"Manufacturer:=", "", _
"Symbol:=", "nexx_bjt_npn", _
"Footprint:=", "", _
"Description:=", "BJT, GP, NPN", _
"InfoTopic:=", "NXBJT1.htm", _
"InfoHelpFile:=", "nexximcomponents.chm", _
```

```
"IconFile:=", "bjtsn.bmp", _
"Library:=", "Nexxim Circuit Elements\BJTs", _
"OriginalLocation:=", "SysLibrary ", _
"Author:=", "", _
"OriginalAuthor:=", "", _
"CreationDate:=", 1152722102), _
"Refbase:=", "Q", _
"NumParts:=", 1, _
"Terminal:=", Array("collector", _
"collector", _
"A", _
false, _
6, _
0, _
"", _
"Electrical"), _
"Terminal:=", Array("base", _
"base", _
"A", _
false, _
```

```
7, _
0, _
"", _
"Electrical"), _
"Terminal:=", Array("emitter", _
"emitter", _
"A", _
false, _
8, _
0, _
"", _
"Electrical"), _
Array("NAME:Parameters", _
"TextProp:=", Array("LabelID", _
"HD", _
"Property string for netlist ID", _
"Q@ID"), _
"TextProp:=", Array("MOD", _
"D", _
"Name of model data reference", _
"required") , _
```

```
"VariableProp:=", Array("AREA", _  
"D", _  
"Emitter area multiplying factor, which affects currents, resistances, and capacitances", _ "1"), _  
-  
"VariableProp:", Array("AREAB", _  
"D", _  
"Base AREA", _  
"1"), _  
"VariableProp:", Array( "AREAC", _  
"D", _  
Collector AREA", _  
"1"), _  
"VariableProp:", Array("DTEMP", _  
"D", _  
"The difference between element and circuit temperature deg Cel)", _  
"0"), _  
"VariableProp:", Array("M", _  
"D", _  
"Multiplier factor to simulate multiple BJTs in parallel", _ "1"), _  
"ButtonProp:", Array("NexximNetlist", _  
"HD", _
```

```

""" , _
"Q@ID %0 %1 %2 *MOD (@MOD) *AREA (AREA=@AREA) " & _
" *AREAB (AREAB=@AREAB) *AREAC (AREAC=@" & _
"AREAC) *DTEMP (DTEMP=@DTEMP) *M (M=@M) ", _
"Q@ID %0 %1 %2 *MOD (@MOD) " & _ "*AREA (AREA=@AREA) AREAB (AREAB=@AREAB) *AREAC (AREAC=@" & _
"AREAC) *DTEMP (DTEMP=@DTEMP) *M (M=@M) ", _
1, _
"ButtonPropClientData:=", Array(), _
"TextProp:=", Array( "modelName", _
"HD", _
""" , _
"Q"))

```

*VB Example:*

```

Dim name2

name2 = oComponentManager.Edit "MyComponent", _
(Array("NAME:MyOtherComponent", _
"Info:=", Array("Type:=", 4294901767, _
"NumTerminals:=", 2, _
"DataSource:=", "", _
"ModifiedOn:=", 1071096503, _
"Manufacturer:=", "Ansys", _

```

```
"Symbol:=", "bendo", _  
"Footprint:=", "BENDO", _  
"Description:=", "", _  
"InfoTopic:=", "", _  
"InfoHelpFile:=", "", _  
"IconFile:=", "", _  
"LibraryName:=", "", _  
"OriginalLocation:=", "Project", _  
"Author:=", "", _  
"OriginalAuthor:=", "", _  
"CreationDate:= ", 1147460679), _  
"Refbase:=", "U", _  
"NumParts:=", 1, _  
"OriginalComponent:=", "", _  
"Terminal:=", Array("n1", _  
"n1", _  
"A", _  
false, _  
0, _  
0, _
```

```
""" , _  
"Electrical") , _  
"Terminal:=", Array("n2", _  
"n2", _  
"A", _  
false, _  
1, _  
0, _  
""" , _  
Electrical") , _  
Array("NAME:Parameters", _  
"MenuProp:=", Array("CoSimulator", _  
"D", _  
""" , _  
"Default, HFSS3D, Circuit, Custom, Netlist", _  
0), _  
"ButtonProp:=", Array("CosimDefinition", _  
"D", _  
""" , _  
""" , _  
"Edit", _
```

```
0, _  
"ButtonPropClientData:=", Array()), _  
Array("NAME:CosimDefinitions", _  
Array("NAME:CosimDefinition", _  
"CosimulatorType:=", 0, _  
"CosimDefName:=", "HFSS3D", _  
"IsDefinition:=", true, _  
"CosimStackup:=", "Layout stackup", _  
"CosimDmbedRatio:=", 3), _  
Array("NAME:CosimDefinition", _  
"CosimulatorType:=", 1, _  
"CosimDefName:=", "Circuit", _  
"IsDefinition:=", true, _  
"ExportAsNport:=", 0, _  
"UsePjt:=", 0), _  
Array("NAME:CosimDefinition", _  
"CosimulatorType:=", 2, _  
"CosimDefName:=", "Custom", _  
"IsDefinition:=", true, _  
"DefinitionCompName:=", ""), _
```

```
Array("NAME:CosimDefinition", _  
"CosimulatorType:=", 3, _  
"CosimDefName:=", "Netlist", _  
"IsDefinition:=", true, _  
"NetlistString:=", "")))
```

<b>Python Syntax</b>	<pre>Edit &lt;ComponentName&gt;,       ["NAME:&lt;NewComponentName&gt;",        "Info:=, &lt;ComponentInfo&gt;,        "RefBase:=, &lt;string&gt;, // reference designator        "NumParts:=, &lt;int&gt;, // parts per component        "OriginalComponent:=, &lt;string&gt;        "Terminal:=, &lt;TerminalInfo&gt;,        "Terminal:=, &lt;TerminalInfo&gt;, ...       #The remaining parameters are optional       ["NAME:Parameters", // any combo of the following        "VariableProp:=, &lt;VariableInfo&gt;,        "CheckboxProp:=, &lt;CheckBoxInfo&gt;,        "ButtonProp:=, &lt;ButtonInfo&gt;,        "TextProp:=, &lt;TextInfo&gt;,        "NumberProp:=, &lt;NumberInfo&gt;,</pre>
----------------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

	<pre>"SeparatorProp:=", &lt;SeparatorInfo&gt;, "ValueProp:=", &lt;ValueInfo&gt;, "MenuProp:=", &lt;MenuInfo&gt;], ["NAME:Properties", # any combo of the following "CheckboxProp:=", &lt;CheckBoxInfo&gt;, "TextProp:=", &lt;TextInfo&gt;, "NumberProp:=", &lt;NumberInfo&gt;, "SeparatorProp:=", &lt;SeparatorInfo&gt;, "ValueProp:=", &lt;ValueInfo&gt;, "MenuProp:=", &lt;MenuInfo&gt;), "VPointProp:=", &lt;VPointInfo&gt;, "PointProp:=", &lt;PointInfo&gt;), ["Quantities", "QuantityProp:=", &lt;QuantityPropInfo&gt;...], ["NAME:CosimDefinitions", &lt;CosimDefInfo&gt;, &lt;CosimDefInfo&gt;...])</pre>
<b>Python Example</b>	<pre>name = oComponentManager.Edit ("Simplorer Circuit Elements\BJTs:Level01_NPN", __ ["NAME:Level01_NPN", "Info:=", ["Type:=", 4294901764, __ "NumTerminals:=", 3, "DataSource:=", "Ansoft built-in component", __</pre>

```
"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  
currents, resistances, and capacitances", "1"], _  
"VariableProp:=", ["AREAB", "D", "Base AREA", _
```

```

    "1"], "VariableProp:=", ["AREAC", "D", "Collector AREA", _  

    "1"], "VariableProp:=", ["DTEMP", "D", _  

    "The difference between element and circuit temperature (deg Cel)", _  

    "0"], "VariableProp:=", ["M", "D", _  

    "Multiplier factor to simulate multiple BJTs in parallel", _  

    "1"], "ButtonProp:=", ["NexximNetlist", "HD", "", _  

    "Q@ID %0 %1 %2 *MOD(@MOD) *AREA(AREA=@AREA) "& _  

    " *AREAB(AREAB=@AREAB) *AREAC(AREAC=@" &  

    "AREAC) *DTEMP(DTEMP=@DTEMP) *M(M=@M)", _  

    "Q@ID %0 %1 %2 *MOD(@MOD) " & "*AREA(AREA=@AREA)  

    *AREAB(AREAB=@AREAB) *AREAC(AREAC=@" &  

    "AREAC) *DTEMP(DTEMP=@DTEMP) *M(M=@M)", 1, _  

    "ButtonPropClientData:=", []],  

    "TextProp:=", ["modelName", "HD", "", "Q"]])
  
```

<b>Python Example 2</b>	<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:=", ""]])
```

## EditDataset

Modifies a dataset. This can be executed by the oProject, or oDesign variables.

<b>UI Access</b>	<b>Project &gt; Datasets &gt; Edit.</b>									
<b>Parameters</b>	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td><i>&lt;OriginalName&gt;</i></td><td>String</td><td>Name of the original dataset.</td></tr><tr><td><i>&lt;DatasetdataArray&gt;</i></td><td>Array</td><td>Data for the modified dataset.</td></tr></tbody></table>	Name	Type	Description	<i>&lt;OriginalName&gt;</i>	String	Name of the original dataset.	<i>&lt;DatasetdataArray&gt;</i>	Array	Data for the modified dataset.
Name	Type	Description								
<i>&lt;OriginalName&gt;</i>	String	Name of the original dataset.								
<i>&lt;DatasetdataArray&gt;</i>	Array	Data for the modified dataset.								
<b>Return Value</b>	None.									

<b>Python Syntax</b>	EditDataset (<OriginalName> <DatasetdataArray>)
<b>Python Example</b>	<pre>oProject.EditDataset ("ds1"     [ "NAME:ds2",         [ "NAME:Coordinates",             [                 "NAME:Coordinate",                 "X:=", 1, "Y:=", 2             ],             [                 "NAME:Coordinate",                 "X:=", 3, "Y:=", 4             ]         ]     ] )</pre>

```

)
oDesign.EditDataset ("ds1"
[ "NAME:ds2",
[ "NAME:Coordinates",
[
    "NAME:Coordinate",
    "X:=", 1, "Y:=", 2
],
[
    "NAME:Coordinate",
    "X:=", 3, "Y:=", 4
]
]
)

```

<b>VB Syntax</b>	EditDataset <OriginalName> <DatasetdataArray>
<b>VB Example</b>	<pre> oProject.EditDataset "ds1" Array("NAME:ds2", Array("NAME:Coordinates",Array("NAME:Coordinate", "X:=", 1, "Y:=", 2), Array("NAME:Coordinate", "X:=", 3, "Y:=", 4))) </pre>

```

oDesign.EditDataset "ds1" Array("NAME:ds2",
    Array("NAME:Coordinates",Array("NAME:Coordinate",
        "X:=", 1, "Y:=", 2), Array("NAME:Coordinate",
        "X:=", 3, "Y:=", 4)))

```

## EditMaterial

Modifies an existing material.

UI Access	View/Edit Materials command in the material editor		
Parameters	Name <i>&lt;OriginalName&gt;</i>	Type String	Description Name of the material before editing.

Name	Type	Description
<i>&lt;OriginalName&gt;</i>	String	Name of the material before editing.
<i>&lt;MatProperties&gt;</i>	Array	Structured array containing material properties:  [ "NAME:<New material name>", "CoordinateSystemType:=", <string>, "BulkOrSurfaceType:=", <integer>, [ "NAME:PhysicsTypes", "set:=", <array containing string physics types> ], <Optional ModifierdataArray>,   ]

		<pre> "permeability:=" , &lt;string containing value&gt;, "conductivity:=" , &lt;string containing value&gt;, "thermal_conductivity:=" , &lt;string containing value&gt;, "mass_density:=" , &lt;string containing value&gt;, "specific_heat:=" , &lt;string containing value&gt;, "youngs_modulus:=" , &lt;string containing value&gt;, "poissons_ratio:=" , &lt;string containing value&gt;, "thermal_expansion_coefficient:=" , &lt;string containing value&gt; ] </pre>
<i>&lt;ModifierdataArray&gt;</i>	Array	<p>Optional structured array containing thermal or spatial modifiers:</p> <pre> [     "NAME:ModifierData",     [         "NAME:&lt;ThermalModifierData or SpatialModifierData&gt;",         "modifier_data:=" , &lt;"thermal_modifier_data" or "spatial_modifier_data"&gt;,         [             "NAME:&lt;all_thermal_modifiers or all_spatial_modifiers&gt;",             [                 "NAME:&lt;modifierName&gt;",                 "Property:::=" , &lt;string property being mod- </pre>

		<pre> ified&gt;,       "Index::=" , &lt;integer&gt;,       "prop_modifier::=" , &lt;"thermal_modifier" or "spatial_modifier"&gt;,       "use_free_form::=" , &lt;Boolean&gt;,       "free_form_value::=" , &lt;string modifier value&gt;,     ]   ] ] </pre>
<b>Return Value</b>	None.	

<b>Python Syntax</b>	EditMaterial (<OriginalName>, <MatProperties>)
<b>Python Example</b>	<p><b>Without Modifiers:</b></p> <pre>oDefinitionManager.EditMaterial("alumina_92pct", [     "NAME:alumina_92pct",     "CoordinateSystemType:=" , "Cartesian",     "Color:=" , "Red",     "Transparency:=" , 100,</pre>

```
"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"
]
)
```

**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

```
# -----
# Script Recorded by Ansys Electronics Desktop Version 2023.2.0
# 14:23:56 Jun 17, 2022
# -----
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",
            "PlasmaMaintainFrequency:=" , "2.31174231183139GHz",
        ]
    ]
]
```

```

    "NeutralDensity:=" , 2.65164580488373E+20,
    "ElectronDensity:=" , 2.65164580488373E+17,
    "CollisionRateConstant:=", 2.05499505485618E-15
]
]
)

```

<b>VB Syntax</b>	EditMaterial <OriginalName>, <MatProperties>
<b>VB Example</b>	<p><b>Without Modifiers:</b></p> <pre> oDefinitionManager&gt;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", "thermal_conductivity:=", "26", "mass_density:=", "3720", </pre>

```
"specific_heat:=", "790",
"youngs_modulus:=", "267000000000",
"poissons_ratio:=", "0.26",
"thermal_expansion_coeffcient:=", "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:=" , "12000000000",
"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.

<b>UI Access</b>	<b>Project &gt; Datasets &gt; Export.</b>		
<b>Parameters</b>	Name	Type	Description
	<datasetFilePath>	String	The full path to the file.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	ExportDataset (<datasetFileFullPath>)
<b>Python Example</b>	<pre>oProject.ExportDataset('e:/tmp/dsdata.txt') oDesign.ExportDataset('e:/tmp/dsdata.txt')</pre>

<b>VB Syntax</b>	ExportDataset <datasetFileFullPath>
<b>VB Example</b>	<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.		

<b>Python Syntax</b>	ExportMaterial (<ExportData>, <LibraryLocation>)
<b>Python Example</b>	<pre>oDefinitionManager.ExportMaterial ([ "NAME:mylib", _ "Material1", "Material2", "Material3"], "PersonalLib")</pre>

<b>VB Syntax</b>	ExportMaterial <ExportData>, <LibraryLocation>
<b>VB Example</b>	<pre>oDefinitionManager.ExportMaterial Array ("NAME:mylib", _ "Material1", "Material2", "Material3"), "PersonalLib"</pre>

## 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"
```

<b>Python Syntax</b>	ExportScript( <ExportData>,<Library location>)
<b>Python Example</b>	<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.

<b>UI Access</b>	N/A								
<b>Parameters</b>	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>None</td><td></td><td></td></tr></tbody></table>			Name	Type	Description	None		
Name	Type	Description							
None									
<b>Return Value</b>	String names of the materials in the active project.								

<b>Python Syntax</b>	GetProjectMaterialNames()
<b>Python Example</b>	<pre>oProject = oDesktop.GetActiveProject() oDefinitionManager = oProject.GetDefinitionManager() materials = oDefinitionManager.GetProjectMaterialNames() AddWarningMessage(str(materials))</pre>

## 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 <i>&lt;PropTab&gt;</i>	Type String	Description One of the following, where tab titles are shown in parentheses: <ul style="list-style-type: none"><li>• PassedParameterTab ("Parameter Values")</li><li>• DefinitionParameterTab (Parameter Defaults")</li><li>• LocalVariableTab ("Variables" or "Local Variables")</li><li>• ProjectVariableTab ("Project variables")</li><li>• ConstantsTab ("Constants")</li><li>• BaseElementTab ("Symbol" or "Footprint")</li><li>• ComponentTab ("General")</li><li>• Component("Component")</li><li>• CustomTab ("Intrinsic Variables")</li><li>• Quantities ("Quantities")</li><li>• Signals ("Signals")</li></ul>
	<i>&lt;PropServer&gt;</i>	String	An object identifier, generally returned from another script method, such as CompInst@R;2;3
	<i>&lt;PropName&gt;</i>	String	Name of the property.

<b>Return Value</b>	String value of the property.
---------------------	-------------------------------

<b>Python Syntax</b>	GetPropertyValue (<PropTab>, <PropServer>, <PropName>)
<b>Python Example</b>	<pre>selectionArray = oEditor.GetSelections() for k in selectionArray:     val = oEditor.GetPropertyValues("PassedParameterTab", k, "R")     ...     ...</pre>

<b>VB Syntax</b>	GetPropertyValues (<PropTab>, <PropServer>, <PropName>)
<b>VB Example</b>	<pre>selectionArray = oEditor.GetSelections for k in selectionArray:     val = oEditor.GetPropertyValues("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.

<b>UI Access</b>	<b>Project &gt; Datasets &gt; Import.</b>		
<b>Parameters</b>	Name	Type	Description
	<code>&lt;datasetFilePath&gt;</code>	String	The full path to the file containing the dataset values. *.tab files recommended (see <a href="#">note</a> below).
<b>Return Value</b>	None.		

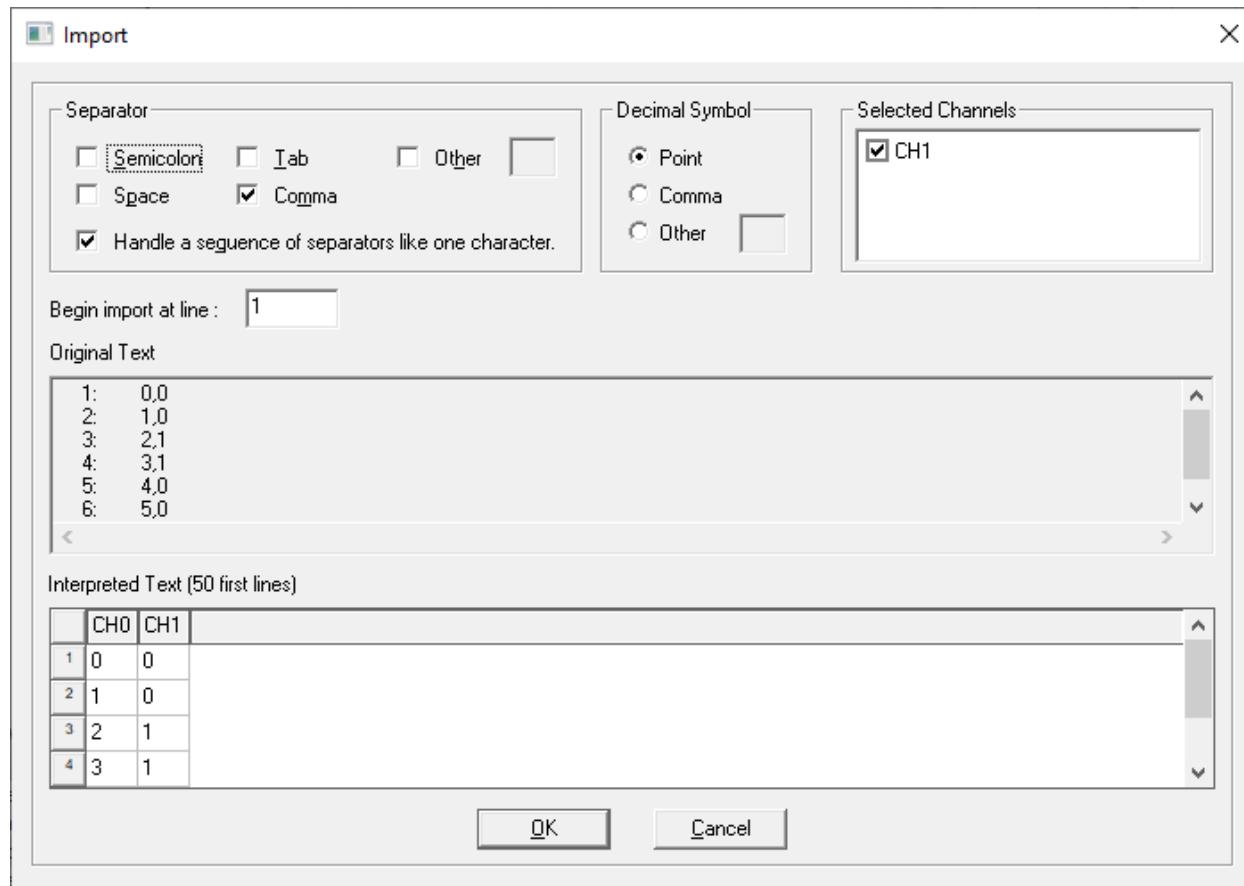
<b>Python Syntax</b>	<code>ImportDataset (&lt;datasetFilePath&gt;,&lt;optionalDatasetName&gt;)</code>
<b>Python Example</b>	<pre> oProject.ImportDataset('e:\tmp\dsdata.tab') oDesign.ImportDataset('e:\tmp\dsdata.tab') oProject.ImportDataset('e:\tmp\dsdata.tab', 'MyDatasetName') oDesign.ImportDataset('e:\tmp\dsdata.tab', 'MyDatasetName')</pre>

<b>VB Syntax</b>	<code>ImportDataset &lt;datasetFilePath&gt;,&lt;optionalDatasetName&gt;</code>
<b>VB Example</b>	<pre> oProject.ImportDataset "e:\tmp\dsdata.tab" oDesign.ImportDataset "e:\tmp\dsdata.tab" oProject.ImportDataset "e:\tmp\dsdata.tab", "MyDatasetName" oDesign.ImportDataset "e:\tmp\dsdata.tab", "MyDatasetName"</pre>

## Note About File Types:

Tab-delimited or space-delimited files with the extension \*.tab are the recommended file type. When using ImportDataset at the Design level, \*.tab is the only file type supported.

At the Project level, other file types are supported (for example, \*.csv). However, after calling the command, you must configure the file import format manually through the Electronics Desktop GUI by selecting **Project > Datasets** and clicking **Import**.



## Remove [component manager]

Remove a component from a library

*Command:* Tools > Edit Configured Libraries > Components > Remove Component

*Syntax:* Remove <ComponentName>,

```
<IsProjectComponent>,  
<LibraryName>,  
<LibraryLocation>
```

*Return Value:* None

*Parameters:* <ComponentName>:

```
<string> // composite name of the component to remove
```

```
<IsProjectComponent>:  
<bool>
```

```
<LibraryName>:  
<string> // name of the library
```

```
<LibraryLocation>:  
<string> // location of the library in <LibraryName>  
// One of "Project", "PersonalLib", or "UserLib"
```

*VB Example:*

```
oComponentManager.Remove "Nexxim Circuit Elements\BJTs:Level01_NPN", _ true, "Project"
```

<b>Python Syntax</b>	Remove (<ComponentName>, <IsProjectComponent>, <LibraryName>, <LibraryLocation>)
<b>Python Example</b>	<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>)
---------------	-------------------------------------------------------------------------------------------

<b>Python Example</b>	<pre>oDefinitionManager.RemoveMaterial ([     "Material1", false, "mo0907", "UserLib"])</pre>
-----------------------	---------------------------------------------------------------------------------------------------

<b>VB Syntax</b>	RemoveMaterial <MaterialName>, <IsProjectMaterial>, <LibraryName>, <LibraryLocation>
<b>VB Example</b>	<pre>oDefinitionManager.RemoveMaterial     "Material1", false, "mo0907", "UserLib"</pre>

## 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"

```

<b>Python Syntax</b>	RemoveScript (<ScriptName>,<IsProjectScript>, <LibraryName>,<LibraryLocation>)
<b>Python Example</b>	<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 <Definitions>	Type Array	Description Definitions to be removed, such as materials and surface materials.
Return Value	None.		

<b>Python Syntax</b>	RemoveUnusedDefinitions(<Definitions>)
<b>Python Example</b>	<pre> oProject.RemoveUnusedDefinitions (     [         [             "NAME:Materials",             "Al-Extruded"         ],         [             "NAME:SurfaceMaterials",             "Steel-oxidised-surface"         ]     ] ) </pre>

<b>VB Syntax</b>	RemoveUnusedDefinitions <Definitions>
<b>VB Example</b>	<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	<p>One of the following, where tab titles are shown in parentheses:</p> <ul style="list-style-type: none"><li>• PassedParameterTab ("Parameter Values")</li><li>• DefinitionParameterTab (Parameter Defaults")</li><li>• LocalVariableTab ("Variables" or "Local Variables")</li><li>• ProjectVariableTab ("Project variables")</li><li>• ConstantsTab ("Constants")</li><li>• BaseElementTab ("Symbol" or "Footprint")</li><li>• ComponentTab ("General")</li><li>• Component("Component")</li><li>• CustomTab ("Intrinsic Variables")</li><li>• Quantities ("Quantities")</li></ul>

		<ul style="list-style-type: none"> <li>• Signals ("Signals")</li> </ul>
<propServer>	String	An object identifier, generally returned from another script method, such as ComplInst@R;2;3
<propName>	String	Name of the property.
<propValue>	String	The value for the property
<b>Return Value</b>	None.	

<b>Python Syntax</b>	SetPropertyValue(<propTab>, <propServer>, <propName>, <propValue>)
<b>Python Example</b>	oEditor SetPropertyValue ("PassedParameterTab", "k", "R", "2200")

<b>VB Syntax</b>	SetPropertyValue <propTab>, <propServer>, <propName>, <propValue>
<b>VB Example</b>	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.

<b>UI Access</b>	N/A		
<b>Parameters</b>	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.
<b>Return Value</b>	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-p-l-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 = ''</pre>

```

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 + "\""
# 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 'MatAppearanceData'
$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

<b>UI Access</b>	N/A
------------------	-----

	Name	Type	Description
<b>Parameters</b>	<code>&lt;targetDefName&gt;</code>	String	Name of the target definition, i.e. the name of the material to update.
	<code>&lt;desiredDefName&gt;</code>	String	New name to use for a rename. If this is the same as <code>targetDefName</code> , it means the definition is not being renamed. If this is blank, the block name will be used for this purpose instead.
	<code>&lt;defBlock&gt;</code>	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.
	<code>&lt;defFolderName&gt;</code>	String	Library type (by definition, folder name).
	<code>&lt;newTimeStamp&gt;</code>	String	New timestamp string (time_t as integer, number of seconds since 1/1/1970 12:00am), default is current time.
	<code>&lt;continuelfConflict&gt;</code>	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.
<b>Return Value</b>	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.)		

<b>Python Syntax</b>	<code>UpdateDefFromBlockEx(&lt;targetDefName&gt;, &lt;desiredDefName&gt;, &lt;defBlock&gt;, &lt;defFolderName&gt;, &lt;newTimeStamp&gt;, &lt;continuelfConflict&gt;)</code>
<b>Python Example</b>	<pre># Note that the material block name will be ignored in this case defBlock = "\$begin 'unused_name' \$begin 'AttachedData' \$begin \</pre>

```

'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'

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.

<b>UI Access</b>	<b>Tools &gt; Project Tools &gt; Update Definitions.</b> Click <b>Select All</b> , then <b>Update</b> .
<b>Parameters</b>	None.
<b>Return Value</b>	None.

<b>Python Syntax</b>	UpdateDefinitions()
<b>Python Example</b>	<code>oProject.UpdateDefinitions ()</code>

<b>VB Syntax</b>	UpdateDefinitions
<b>VB Example</b>	<code>oProject.UpdateDefinitions</code>

## 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  
SIwaveDeNovo -- 1000 binary, 8 decimal, 0x8  
Simplorer -- 10000 binary, 16 decimal, 0x10  
MaxwellCircuit -- 100000 binary, 32 decimal, 0x20

**<ModifiedOnInfo>:**

An integer that corresponds to the number of seconds that have elapsed since 00:00 hours, Jan 1, 1970 UTC from the system clock.

**<TerminalInfo>:**

```
Array(<string>, // symbol pin  
<string> // footprint pin
```

```
<string>, // gate name  
<bool>, // shared  
<int>, // equivalence number  
<int>, // what to do if unconnected: flag as error:0, ignore:1  
<string> // description  
<Nature>)
```

<Nature>:  
<string> // content varies as follows

Nexxim/Circuit:  
"Electrical" // the only choice

Simplorer:  
// several choices  
"Electrical", "Magnetic", "Fluidic", "Translational",  
"Translational\_V", "Rotational", "Rotational\_V",  
""Radian", "Thermal", or <VHDLPackageName>

<VHDLPackageName>:  
<string> // in the form <Library>.<Package>

<Library>:

<string> // name of the VHDL library

<Package>:

<string> // name of the VHDL package

<VariableInfo>:

Array(<string>, // name

<FlagLetters>,

<string>, // description

"CB:=", <string>, // optional - script for call back

<string>) // value: number, variable, or expression

<FlagLetters>:

<string> // "D" - has description parameter,

// "RD" - readonly & has description parameter,

// or "RHD" - readonly, hidden, & has description parameter

<CheckBoxInfo>:

```
Array(<string>, // name  
<FlagLetters>,  
<string>, // description  
"CB:=", <string>, // optional - script for call back  
<bool>) // value: true or false
```

```
<ButtonInfo>:  
Array(<string>, // name  
<FlagLetters>,  
<string>, // description  
<string>, // button title  
<string>, // extra text  
<ClientID>,  
"ButtonPropClientData:= ", <ClientdataArray>)
```

```
<ClientID>:  
<int> // specifies Button Prop Client  
// 0 - unknown, ButtonPropClientData  
// array will be empty  
// 1 - Netlist Prop Client  
// 2 - not used
```

// 3 - File Name Prop Client

<ClientdataArray>:

varies with <ClientID>

<ClientID> is 0 or 1: empty array

Array()

<ClientID> is 3:

Array("InternalFormatText:", "<prefix><RelativePath>")

<prefix>:

<string> // "<Project>", "<PersonalLib>", "<UserLib>", or "<SysLib>"

<RelativePath>:

<string> // relative path to file from <prefix>

<TextInfo>:

Array(<string>, // name

<FlagLetters>,

```
<string>, // description  
"CB:=", <string>, // optional - script for call back  
<string>) // value: a text string
```

```
<NumberInfo>:  
Array(<string>, // name  
<FlagLetters>,  
<string>, // description  
"CB:=", <string>, // optional - script for call back  
<real>, // value: a number  
<string>) // units
```

```
<SeparatorInfo>:  
Array(<string>, // name  
<FlagLetters>,  
<string>, // description  
"CB:=", <string>, // optional - script for call back  
<string>) // value: a text string
```

```
<ValueInfo>:  
Array(<string>, // name  
<FlagLetters>,
```

```
<string>, // description  
"CB:=", <string>, // optional - script for call back  
<string>) // value: a number, variable or expression
```

```
<MenuPropInfo>:  
Array(<string>, // name  
<FlagLetters>,  
<string>, // description  
<string>, // menu choices - separated by commas  
<int>) // 0 based index of current menu choice
```

```
<VPointInfo>:  
Array(<string>, // name  
<FlagLetters>,  
<string>, // description  
"CB:=", <string>, // optional - script for call back  
<string>, // x value: number with length units  
<string>) // y value: number with length units
```

```
<PointInfo>:  
Array(<string>, // name
```

```
<FlagLetters>,
<string>, // description
"CB:=", <string>, // optional - script for call back
<real>, // x value
<real>) // y value

<QuantityPropInfo>:
Array(<string>, // name
<FlagLetters>,
<string>, // description
<string>, // value
<TypeString>,
<TypeStringDependentInfo>

<TypeString>:
<string> // "Across", "Through", or "Free"

<TypeStringDependentInfo>:

<TypeString> is "Free" :
<string>, // direction: "In", "Out", "InOut", or "DontCare"
```

```
// Following <string> is not present if direction is "DontCare"
<string> // when to calculate: "BeforeAnalogSolver",
// "BeforeStateGraph", "AfterStateGraph", or "DontCareWhen"

<TypeString> is "Across" or "Through":
<int>, // terminal 1
<int> // terminal 2

<CosimDefInfo>:
Array("NAME:CosimDefinition",
"CosimulatorType:=", <int>,
"CosimDefName:=", <string> // "HFSS 3D Layout", "Circuit",
// "Custom", or "Netlist"
"IsDefinition:=", <bool>,
final array member(s) vary with CosimDefName)

final array members for HFSS 3D Layout:
"CosimStackup:=", <string>,
"CosimDmbedRatio:=", <int>

final array members for Circuit:
```

```
"ExportAsNport:=", <int>,
"UsePjt:=", <int>
```

final array member for Custom:

```
"DefinitionCompName:=", <string>
```

final array member for Netlist:

```
"NetlistString:=", <string>
```

*VB Example:*

```
Dim name
oComponentManager.Add (Array("NAME:MyComponent", _
"Info:=", Array("Type:=", 4294901767, _
"NumTerminals:=", 2, _
"DataSource:=", "", _
"ModifiedOn:=", 1071096503, _
"Manufacturer:=", "Ansys", _
"Symbol:=", "bendo", _
"Footprint:=", "BENDO", _
"Description:=", "", _
"InfoTopic:=", "", _
```

```
"InfoHelpFile:="", "", _
"IconFile:="", "", _
"LibraryName:="", "", _
"OriginalLocation:=", "Project", _
"Author:="", "", _
"OriginalAuthor:="", "", _
"CreationDate:= ", 1147460679), _
"Refbase:=", "U", _
"NumParts:=", 1, _
"OriginalComponent:="", "", _
"Terminal:=", Array("n1", _
"n1", _
"A", _
false, _
0, _
1, _
"", _
"Electrical"), _
"Terminal:=", Array("n2", _
"n2", _
```

```
"A", _  
false, _  
1, _  
0, _  
"", _  
"Electrical"), _  
Array("NAME:Parameters", _  
"MenuProp:=", Array("CoSimulator", _  
"D", _  
"", _  
"Default, HFSS 3D Layout, Circuit, Custom, Netlist", _  
0), _  
"ButtonProp:=", Array("CosimDefinition", _  
"D", _  
"", _  
"", _  
"Edit", _  
0, _  
"ButtonPropClientData:=", Array()), _  
Array("NAME:CosimDefinitions", _  
Array("NAME:CosimDefinition", _
```

```
"CosimulatorType:=", 0, _  
"CosimDefName:=", "HFSS 3D Layout", _  
"IsDefinition:=", true, _  
"CosimStackup:=", "Layout stackup", _  
"CosimDmbedRatio:=", 3), _  
Array("NAME:CosimDefinition", _  
"CosimulatorType:=", 1, _  
"CosimDefName:=", "Circuit", _  
"IsDefinition:=", true, _  
"ExportAsNport:=", 0, _  
"UsePjt:=", 0), _  
Array("NAME:CosimDefinition", _  
"CosimulatorType:=", 2, _  
"CosimDefName:=", "Custom", _  
"IsDefinition:=", true, _  
"DefinitionCompName:=", ""), _  
Array("NAME:CosimDefinition", _  
"CosimulatorType:=", 3, _  
"CosimDefName:=", "Netlist", _  
"IsDefinition:=", true, _
```

```
"NetlistString:=", "")) )
```

	<pre>Add [("NAME:&lt;ComponentName&gt;", "Info:=", &lt;ComponentInfo&gt;, "RefBase:=", &lt;string&gt;, // reference designator "NumParts:=", &lt;int&gt;, // parts per component "OriginalComponent:=", &lt;string&gt; "Terminal:=", &lt;TerminalInfo&gt;, "Terminal:=", &lt;TerminalInfo&gt;, ...  <b>The remaining parameters are optional.</b> ["NAME:Parameters", // any combo of the following "VariableProp:=", &lt;VariableInfo&gt;, "CheckboxProp:=", &lt;CheckBoxInfo&gt;, "ButtonProp:=", &lt;ButtonInfo&gt;, "TextProp:=", &lt;TextInfo&gt;, "NumberProp:=", &lt;NumberInfo&gt;, "SeparatorProp:=", &lt;SeparatorInfo&gt;, "ValueProp:=", &lt;ValueInfo&gt;, "MenuProp:=", &lt;MenuInfo&gt;], ["NAME:Properties", Any combination of the following: "CheckboxProp:=", &lt;CheckBoxInfo&gt;,</pre>
--	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

	<pre>"TextProp:=", &lt;TextInfo&gt;, "NumberProp:=", &lt;NumberInfo&gt;, "SeparatorProp:=", &lt;SeparatorInfo&gt;, "ValueProp:=", &lt;ValueInfo&gt;, "MenuProp:=", &lt;MenuInfo&gt;, "VPointProp:=", &lt;VPointInfo&gt;, "PointProp:=", &lt;PointInfo&gt;], ["Quantities", "QuantityProp:=", &lt;QuantityPropInfo&gt;...], ["NAME:CosimDefinitions", &lt;CosimDefInfo&gt;, &lt;CosimDefInfo&gt;...]]</pre>
<b>Python Example</b>	<pre>oComponentManager.Add( [ "NAME:Component", "Info:=", [ "Type:=", 0, "NumTerminals:=", 0, "DataSource:=", "",  "ModifiedOn:=", 1467910752,</pre>

```
"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  
])
```

## AddDynamicNPortData [component manager]

Adds a component using the specified data

*Command:* Project Menu > Add Model > Add 2DExtractor Model

Project Menu > Add Model > Add HFSS Model

Project Menu > Add Model > Add Nexsys Matlab Model

Project Menu > Add Model > Add Nport Model

Project Menu > Add Model > Add Parametric Model

Project Menu > Add Model > Add Q3D Model

Project Menu > Add Model > Add SIwave Model

Project Menu > Add Model > Add State-space Model

*Syntax:* AddDynamicNPortData Array("NAME:<ComponentDataName>","  
"ComponentDataType:=", <DataType>,  
"name:=", <string>, // Name of the item  
"filename:=", <string>, // Path to the file to find the data>  
"numberofports:=", <int>,  
"DesignName:=", <string>, // Name of the internal design  
"SolutionName:=", <string>, // Name of the solution to reference  
"Simulate:=", <bool>,  
"CloseProject:=", <bool>,  
"SaveProject:=", <bool>,

```
"RefNode:=", <bool>,
"InterpY:=", <bool>, // true to choose interpolating
"InterpAlg:=", <InterpolationAlgorithm>,
"NewToOldMap:=", <NewToOldMapPairs>,
"OldToNewMap:=", <OldToNewMapPairs>,
"PinNames:=", Array(<string>, <string>...))
```

*Return Value:* <string>

```
// composite name of the component.
// If the name requested conflicts with the name of an existing
// component, the requested name is altered to be unique.
// The name returned reflects any change made to be unique.
```

*Parameters:* <ComponentDataName>:

```
<string> // simple name of the component
```

```
<DataType>:
```

```
<string> // "DesignerData" or "Q3DDData"
```

```
<InterpolationAlgorithm>:
```

```
<string> // "auto", "lin", "shadH", or "shadNH"
```

```
<NewToOldMapPairs>:
```

Array of name/value pairs such as "V1", "V2" that will have the new variable name first and the old variable name second.

<OldToNewMapPairs>:

Array of name/value pairs such as "V1", "V2" that will have the old variable name first and the new variable name second.

*VB Example:*

```
oComponentManager.AddDynamicNPortData  
Array("NAME:ComponentData", _  
"ComponentDataType:=", "DesignerData", _  
"name:=", "DesignerData", _  
"filename:=", _  
"C:/Program Files/AnsysEM/Designer/Examples/Projects/optiguides/optiguides.adsn", _  
"numberofports:=", 2, _  
"DesignName:=", "DesignerModel1", _  
"SolutionName:=", "Setup1 : Adaptive_1", _  
"Simulate:=", true, _  
"CloseProject:=", false, _  
"SaveProject:=", true, _  
"RefNode:=", false, _  
"InterpY:=", true, _  
"InterpAlg:=", "auto", _
```

```
"NewToOldMap:=", Array("nport_height:=", "$height", _  
"nport_length:=", "$length", _  
"nport_width:=", "$width"), _  
"OldToNewMap:=", Array("$height:=", "nport_height", _  
"$length:=", "nport_length", _  
"$width:=", "nport_width"), _  
"PinNames:=", Array( "WavePort1:1", "WavePort2:1"))
```

## AddNPortData [component manager]

*Use:* Adds a component using the specified data

*Command:* Project Menu > Add Model > Add Nport Model

*Syntax:* AddNPortData Array("NAME:<ComponentDataName>","

```
    "ComponentDataType:=", "NportData",  
    "name:=", <string>, // Name of the item  
    "filename:=", <string>, // Path to the file to find the data  
    "numberofports:=", <int>,  
    "filelocation:=", <LocationType>,  
    "domain:=", <string>, // "time" or "frequency"  
    "datemode:=", <string> // "EnterData", "Import", or "Link"  
    "devicename:=", <string>,  
    "ImpedanceTab:=", <bool>,  
    "NoiseDataTab:=", <bool>,
```

```
"DCBehaviorTab:=", <bool>,  
"SolutionName:=", <string>,  
"displayformat:=", <DisplayInfo>,  
"datatype:=", <string>, // "SMatrix", "YMatrix", or "ZMatrix"  
"ShowRefPin:=", <bool>,  
"RefNodeCheckbox:=", <bool>, ...  
<ProductOptionsInfo>
```

*Return Value:* <string>

```
// composite name of the component.  
// If the name requested conflicts with the name of an existing  
// component, the requested name is altered to be unique.  
// The name returned reflects any change made to be unique.
```

*Parameters:* <ComponentDataName>:

```
<string> // simple name of the component
```

<LocationType>:

```
<string> // one of "UsePath", "PersonalLib", "UserLib", "SysLib",  
// or "Project".
```

```
<dclInfo>
<string> // one of "DCOpen", "DCShort", "DCShShort",
// "DCNone", or "DCEmpty".

<DisplayInfo>
<string> // one of "MagnitudePhase", "ReallImaginary",
// or "DbPhase".

<ProductOptionsInfo>
// The remaining parameters differ by product

// HFSS 3D Layout - doesn't support interpolation/DC behavior
"DCOption:=", -1,
"InterpOption:=", -1,
"ExtrapOption:=", -1,
"DataType:=", 0

// Nexxim
"DCOption:=", <NexximDCOption>,
"InterpOption:=", <NexximInterpOption>,
"ExtrapOption:=", <NexximExtrapOption>,
```

```
"DataType:=", 2

<NexximDCOption>:
<int> // 0 : Zero Padding
// 1 : Same as last point
// 2 : Linear extrapolation from last 2 points
// 3 : Constant magnitude, linear phase extrapolation
// 4 : Leave all signal lines open circuited
// 5 : Short all signal lines together
// 6 : Short all signal lines to ground

<NexximInterpOption>:
<int> // 0 : Step
// 1 : Linear

<NexximExtrapOption>:
<int> // 0 : Zero padding
// 1 : Same as last point
// 2 : Linear extrapolation from last 2 points
// 3 : Constant magnitude, linear phase extrapolation
```

```
<CircuitDCOption>:  
    <int> // 0 : Leave all signal lines open circuited  
    // 1 : Short all signal lines together  
    // 2 : Short all signal lines to ground  
    // 3 : Extrapolate from data provided (not recommended)
```

```
<CircuitInterpOption>:  
    <int> // 0 : Linear  
    // 1 : Cubic spline  
    // 2 : Rational polynomial
```

```
<CircuitExtrapOption>:  
    <int> // 0 : Same as interpolation  
    // 1 : Zero padding  
    // 2 : Same as last point
```

*VB Example:*

```
oComponentManager.AddNPortData Array("NAME:ComponentData", _  
"ComponentDataType:=", "NPortData", _  
"name:=", "NportData", _  
"filename:=", "", _
```

```
"numberofports:=", 2, _
"filelocation:=", "UsePath", _
"domain:=", "frequency", _
"datemode:=", "Import", _
"devicename:=", "", _
"ImpedanceTab:=", true, _
"NoiseDataTab:=", true, _
"DCBehaviorTab:=", true, _
"SolutionName:=", "", _
"displayformat:=", "MagnitudePhase", _
"datatype:=", "SMatrix", _
>ShowRefPin:=", true, _
"RefNodeCheckbox:=", false, _
"DCOption:=", 3, _
"InterpOption:=", 1, _
"ExtrapOption:=", 3, _
"DataType:=", 2, _
"DCOption:=", 3, _
"InterpOption:=", 1, _
"ExtrapOption:=", 3, _
```

```
"DataType:=", 2)
```

## AddSolverOnDemandModel

*Use:* This method looks for a local component of the name passed in, and to this component it adds an SOD model definition using the information passed in the VARIANT. It returns the name of the SOD model added.

*Parameters:* BSTR component name.

*Parameters:* VARIANT which is the SOD model data.

*Return Value:* Returns the name of the model added.

## ClearSolutionCache [component manager]

*Use:* Clear the solution cache for dynamic link component.

*Command:* Each of the following commands will clear the solution cache:

- Dynamic Link Item RCM > Clear Solution Cache
- Dynamic Link Component in schematic RCM > Clear Solution Cache

*Syntax:* ClearSolutionCache <Component Name>

*Return Value:* None

*Parameters:* <component name> is the name of the dynamic link component

*VB Example:*

```
oComponentManager.ClearSolutionCache "TeeModel1"
```

## Edit [component manager]

Modifies an existing component

*Command:* Tools > Edit Configured Libraries > Components > Edit Component

*Syntax:* Edit <ComponentName>,

```
Array("NAME:<NewComponentName>","
```

```
"Info:=", <ComponentInfo>,  
"RefBase:=", <string>, // reference designator  
"NumParts:=", <int>, // parts per component  
"OriginalComponent:=", <string>  
"Terminal:=", <TerminalInfo>,  
"Terminal:=", <TerminalInfo>, ...  
// The remaining parameters are optional  
Array("NAME:Parameters", // any combo of the following  
"VariableProp:=", <VariableInfo>,  
"CheckboxProp:=", <CheckBoxInfo>,  
"ButtonProp:=", <ButtonInfo>,  
"TextProp:=", <TextInfo>,  
"NumberProp:=", <NumberInfo>,  
"SeparatorProp:=", <SeparatorInfo>,  
"ValueProp:=", <ValueInfo>,  
"MenuProp:=", <MenuInfo>),  
Array("NAME:Properties", // any combo of the following  
"CheckboxProp:=", <CheckBoxInfo>,  
"TextProp:=", <TextInfo>,  
"NumberProp:=", <NumberInfo>,
```

```
"SeparatorProp:=", <SeparatorInfo>,
"ValueProp:=", <ValueInfo>,
"MenuProp:=", <MenuInfo>),
"VPointProp:=", <VPointInfo>,
"PointProp:=", <PointInfo>),
Array("Quantities",
"QuantityProp:=", <QuantityPropInfo>...),
Array("NAME:CosimDefinitions",
<CosimDefInfo>,
<CosimDefInfo>...)
```

*Return Value:* <string>

```
// composite name of the component.  
// If the name requested conflicts with the name of an existing  
// component, the requested name is altered to be unique.  
// The name returned reflects any change made to be unique.
```

*Parameters:* <ComponentName>:

```
<string> // composite name of the component to edit
```

```
<NewComponentName>:
```

```
<string> // new simple name for the component
```

```
<ComponentInfo>
  Array("Type:=", <TypeInfo>,
        "NumTerminals:=", <int>,
        "DataSource:=", <string>,
        "ModifiedOn:=", <ModifiedOnInfo>,
        "Manufacturer:=", "<string>,
        "Symbol:=", <string>,
        "Footprint:=", <string>,
        "Description:=", <string>,
        "InfoTopic:=", <string>,
        "InfoHelpFile:=", <string>,
        "IconFile:=", <string>,
        "LibraryName:=", <string>,
        "OriginalLocation:=", <string>, // Project Location
        "Author:=", <string>,
        "OriginalAuthor:=", <string>,
        "CreationDate:=", <int>)
```

```
<TypeInfo>
```

An integer that is the or-ing of bits for each product listed below. The default setting is 0xffffffff (4294967295) which indicates valid for all products. In the component editing dialog, checking different boxes in the "Specify products for which this component is valid" grid control sets specific flags that correspond to the following hex/decimal settings:

Nexxim -- 100 binary, 4 decimal, 0x4

SIwaveDeNovo -- 1000 binary, 8 decimal, 0x8

Simplorer -- 10000 binary, 16 decimal, 0x10

MaxwellCircuit -- 100000 binary, 32 decimal, 0x20

**<ModifiedOnInfo>:**

An integer that corresponds to the number of seconds that have elapsed since 00:00 hours, Jan 1, 1970 UTC from the system clock.

**<TerminalInfo>:**

```
Array(<string>, // symbol pin  
<string> // footprint pin  
<string>, // gate name  
<bool>, // shared  
<int>, // equivalence number  
<int>, // what to do if unconnected: flag as error:0, ignore:1  
<string>, // description  
<Nature>)
```

**<Nature>:**

<string> // content varies as follows

Nexxim/Circuit:

"Electrical" // the only choice

Simplorer:

// several choices

"Electrical", "Magnetic", "Fluidic", "Translational",  
"Translational\_V", "Rotational", "Rotational\_V",  
""Radian", "Thermal", or <VHDLPackageName>

<VHDLPackageName>:

<string> // in the form <Library>.<Package>

<Library>:

<string> // name of the VHDL library

<Package>:

<string> // name of the VHDL package

```
<VariableInfo>
  Array(<string>, // name
        <FlagLetters>,
        <string>, // description
        "CB:=", <string>, // optional - script for call back
        <string>) // value: number, variable, or expression
```

```
<FlagLetters>
  <string> // "D" - has description parameter,
  // "RD" - readonly & has description parameter,
  // or "RHD" - readonly, hidden, & has description parameter
```

```
<CheckBoxInfo>
  Array(<string>, // name
        <FlagLetters>,
        <string>, // description
        "CB:=", <string>, // optional - script for call back
        <bool>) // value: true or false
```

```
<ButtonInfo>
```

```
Array<string>, // name  
<FlagLetters>,  
<string>, // description  
<string>, // button title  
<string>, // extra text  
<ClientID>,  
"ButtonPropClientData:= ", <ClientdataArray>)
```

```
<ClientID>:  
<int> // specifies Button Prop Client  
// 0 - unknown, "ButtonPropClientData  
// array will be empty  
// 1 - Netlist Prop Client  
// 2 - not used  
// 3 - File Name Prop Client
```

```
<ClientdataArray>:  
varies with <ClientID>
```

<ClientID> is 0 or 1: empty array

Array()

<ClientID> is 3:

Array("InternalFormatText:=", "<prefix><RelativePath>")

<prefix>:

<string> // "<Project>", "<PersonalLib>", "<UserLib>", or "<SysLib>"

<RelativePath>:

<string> // relative path to file from <prefix>

<TextInfo>:

Array(<string>, // name  
<FlagLetters>,  
<string>, // description  
"CB:=", <string>, // optional - script for call back  
<string>) // value: a text string

<NumberInfo>:

Array(<string>, // name  
<FlagLetters>,

```
<string>, // description
"CB:=", <string>, // optional - script for call back
<real>, // value: a number
<string>) // units

<SeparatorInfo>:
Array(<string>, // name
<FlagLetters>,
<string>, // description
"CB:=", <string>, // optional - script for call back
<string>) // value: a text string

<ValueInfo>:
Array(<string>, // name
<FlagLetters>,
<string>, // description
"CB:=", <string>, // optional - script for call back
<string>) // value: a number, variable or expression

<MenuPropInfo>:
Array(<string>, // name
```

```
<FlagLetters>,
<string>, // description
<string>, // menu choices - separated by commas
<int>) // 0 based index of current menu choice
```

```
<VPointInfo>:
Array(<string>, // name
<FlagLetters>,
<string>, // description
"CB:=", <string>, // optional - script for call back
<string>, // x value: number with length units
<string>) // y value: number with length units
```

```
<PointInfo>:
Array(<string>, // name
<FlagLetters>,
<string>, // description
"CB:=", <string>, // optional - script for call back
<real>, // x value
<real>) // y value
```

```
<QuantityPropInfo>:
```

```
Array<string>, // name  
<FlagLetters>,  
<string>, // description  
<string>, // value  
<TypeString>,  
<TypeStringDependentInfo>  
  
<TypeString>:  
<string> // "Across", "Through", or "Free"  
  
<TypeStringDependentInfo>:  
  
"Free":  
<string>, // direction: "In", "Out", "InOut", or "DontCare"  
// Following <string> is not present if direction is "DontCare"  
<string> // when to calculate: "BeforeAnalogSolver",  
// "BeforeStateGraph", "AfterStateGraph", or "DontCareWhen"  
  
"Across" or "Through"  
<int>, // terminal 1  
<int> // terminal 2
```

```
<CosimDefInfo>
  Array("NAME:CosimDefinition",
    "CosimulatorType:=", <int>,
    "CosimDefName:=", <string> // "HFSS3D", "Circuit",
    // "Custom", or "Netlist"
    "IsDefinition:=", <bool>,
    final array member(s) vary with CosimDefName)
```

final array members for HFSS 3D Layout:

```
"CosimStackup:=", <string>,
"CosimDmbedRatio:=", <int>
```

final array members for Circuit:

```
"ExportAsNport:=", <int>,
"UsePjt:=", <int>
```

final array member for Custom:

```
"DefinitionCompName:=", <string>
```

final array member for Netlist:

"NetlistString:=", <string>

*VB Example:*

```
Dim name

name = oComponentManager.Edit ("Nexxim Circuit Elements\BJTs:Level01_NPN", _
Array("NAME:Level01_NPN", _
"Info:=", Array("Type:=", 4294901764,_
"NumTerminals:=", 3, _
"DataSource:=", "Ansys built-in component", _
"ModifiedOn:=", 1152722112, _
"Manufacturer:="", "", _
"Symbol:=", "nexx_bjt_npn", _
"Footprint:="", "", _
"Description:=", "BJT, GP, NPN", _
"InfoTopic:=", "NXBJT1.htm", _
"InfoHelpFile:=", "nexximcomponents.chm", _
"IconFile:=", "bjtsn.bmp", _
"Library:=", "Nexxim Circuit Elements\BJTs", _
"OriginalLocation:=", "SysLibrary ", _
"Author:="", "", _
"OriginalAuthor:="", "", _
```

```
"CreationDate:=", 1152722102), _  
"Refbase:=", "Q", _  
"NumParts:=", 1, _  
"Terminal:=", Array("collector", _  
"collector", _  
"A", _  
false, _  
6, _  
0, _  
"", _  
"Electrical"), _  
"Terminal:=", Array("base", _  
"base", _  
"A", _  
false, _  
7, _  
0, _  
"", _  
"Electrical"), _  
"Terminal:=", Array("emitter", _  
"emitter", _
```

```
"A", _  
false, _  
8, _  
0, _  
"", _  
"Electrical"), _  
Array("NAME:Parameters", _  
"TextProp:=", Array("LabelID", _  
"HD", _  
"Property string for netlist ID", _  
"Q@ID"), _  
"TextProp:=", Array("MOD", _  
"D", _  
"Name of model data reference", _  
"required"), _  
"VariableProp:=", Array("AREA", _  
"D", _  
"Emitter area multiplying factor, which affects currents, resistances, and capacitances", _ "1"), _  
_  
"VariableProp=", Array("AREAB", _  
"D", _
```

```
"Base AREA", _  
"1"), _  
"VariableProp:=", Array( "AREAC", _  
"D", _  
Collector AREA", _  
"1"), _  
"VariableProp:=", Array("DTEMP", _  
"D", _  
"The difference between element and circuit temperature deg Cel)", _  
"0"), _  
"VariableProp:=", Array("M", _  
"D", _  
"Multiplier factor to simulate multiple BJTs in parallel", _ "1"), _  
"ButtonProp:=", Array("NexximNetlist", _  
"HD", _  
"", _  
"Q@ID %0 %1 %2 *MOD(@MOD) *AREA(AREA=@AREA)" & _  
" *AREAB(AREAB=@AREAB) *AREAC(AREAC=@" & _  
"AREAC) *DTEMP(DTEMP=@DTEMP) *M(M=@M)", _  
"Q@ID %0 %1 %2 *MOD(@MOD) " & _ "*AREA(AREA=@AREA) AREAB(AREAB=@AREAB) *AREAC(AREAC=@" & _  
"AREAC) *DTEMP(DTEMP=@DTEMP) *M(M=@M)", _
```

```
1, _
"ButtonPropClientData:=", Array(), _
"TextProp:=", Array( "modelName", _
"HD", _
"", _
"Q")))
```

*VB Example:*

```
Dim name2
name2 = oComponentManager.Edit "MyComponent", _
(Array("NAME:MyOtherComponent", _
"Info:=", Array("Type:=", 4294901767, _
"NumTerminals:=", 2, _
"DataSource:=", "", _
"ModifiedOn:=", 1071096503, _
"Manufacturer:=", "Ansys", _
"Symbol:=", "bendo", _
"Footprint:=", "BENDO", _
"Description:=", "", _
"InfoTopic:=", "", _
"InfoHelpFile:=", "")
```

```
"IconFile:="", "", _
"LibraryName:="", "", _
"OriginalLocation:=", "Project", _
"Author:="", "", _
"OriginalAuthor:="", "", _
"CreationDate:= ", 1147460679), _
"Refbase:=", "U", _
"NumParts:=", 1, _
"OriginalComponent:="", "", _
"Terminal:=", Array("n1", _
"n1", _
"A", _
false, _
0, _
0, _
"", _
"Electrical"), _
"Terminal:=", Array("n2", _
"n2", _
"A", _
false, _
```

```
1, _
0, _
"", _
Electrical"), _
Array("NAME:Parameters", _
"MenuProp:=", Array("CoSimulator", _
"D", _
"", _
"Default, HFSS3D, Circuit, Custom, Netlist", _
0), _
"ButtonProp:=", Array("CosimDefinition", _
"D", _
"", _
"", _
>Edit", _
0, _
"ButtonPropClientData:=", Array()), _
Array("NAME:CosimDefinitions", _
Array("NAME:CosimDefinition", _
"CosimulatorType:=", 0, _
```

```
"CosimDefName:=", "HFSS3D", _  
"IsDefinition:=", true, _  
"CosimStackup:=", "Layout stackup", _  
"CosimDmbedRatio:=", 3), _  
Array("NAME:CosimDefinition", _  
"CosimulatorType:=", 1, _  
"CosimDefName:=", "Circuit", _  
"IsDefinition:=", true, _  
"ExportAsNport:=", 0, _  
"UsePjt:=", 0), _  
Array("NAME:CosimDefinition", _  
"CosimulatorType:=", 2, _  
"CosimDefName:=", "Custom", _  
"IsDefinition:=", true, _  
"DefinitionCompName:=", ""), _  
Array("NAME:CosimDefinition", _  
"CosimulatorType:=", 3, _  
"CosimDefName:=", "Netlist", _  
"IsDefinition:=", true, _  
"NetlistString:=", "") ))
```

**Python Syntax**

```
Edit <ComponentName>,
    ["NAME:<NewComponentName>",
     "Info:=", <ComponentInfo>,
     "RefBase:=", <string>, // reference designator
     "NumParts:=", <int>, // parts per component
     "OriginalComponent:=", <string>
     "Terminal:=", <TerminalInfo>,
     "Terminal:=", <TerminalInfo>, ...
#The remaining parameters are optional
["NAME:Parameters", // any combo of the following
 "VariableProp:=", <VariableInfo>,
 "CheckboxProp:=", <CheckBoxInfo>,
 "ButtonProp:=", <ButtonInfo>,
 "TextProp:=", <TextInfo>,
 "NumberProp:=", <NumberInfo>,
 "SeparatorProp:=", <SeparatorInfo>,
 "ValueProp:=", <ValueInfo>,
 "MenuProp:=", <MenuInfo>],
["NAME:Properties", # any combo of the following
 "CheckboxProp:=", <CheckBoxInfo>,
```

	<pre> "TextProp:=", &lt;TextInfo&gt;, "NumberProp:=", &lt;NumberInfo&gt;, "SeparatorProp:=", &lt;SeparatorInfo&gt;, "ValueProp:=", &lt;ValueInfo&gt;, "MenuProp:=", &lt;MenuInfo&gt;), "VPointProp:=", &lt;VPointInfo&gt;, "PointProp:=", &lt;PointInfo&gt;), ["Quantities", "QuantityProp:=", &lt;QuantityPropInfo&gt;...], ["NAME:CosimDefinitions", &lt;CosimDefInfo&gt;, &lt;CosimDefInfo&gt;...]) </pre>
Python Example	<pre> 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:=", "", _  </pre>

```
"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  
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"])])

```

```

name2 = oComponentManager.Edit ("MyComponent", _  

(["NAME:MyOtherComponent", "Info:=", ["Type:=", 4294901767, _  

"NumTerminals:=", 2, "DataSource:=", "", _  

"ModifiedOn:=", 1071096503, "Manufacturer:=", "Ansoft", _  

"Symbol:=", "bendo", "Footprint:=", "BENDO", _
```

## Python Example 2

```
"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"), "Per-
sonalLib"

```

<b>Python Syntax</b>	Export(["NAME:<LibraryName>", <ComponentName>, <ComponentName>...], <LibraryLocation>)
<b>Python Example</b>	<pre> oComponentManager.Export ([ "NAME:mylib", "Simpler Circuit Elements\BJTs:Level01_NPN"] , "PersonalLib") </pre>

## GetData [component manager]

Gets data that describes the definition.

*Command:* None

*Syntax:* GetData(<DefinitionName>)

*Return Value:* <DefinitionData> This is an array of data for the definition.

*Parameters:* <DefinitionName>:

<string> // [composite name](#) of the definition to edit

*VB Example:*

```
Dim compData  
compData = oComponentManager.GetData("Level01_NPN")
```

**Note:**

GetData allows the user to access definition information, make modifications, and then use the Edit or EditWithComps script commands to save the modified definition. Accordingly, for each type of definition, the array data returned to GetData should be the same array information that is supplied to the [Edit](#) or [EditWithComps](#) commands.

<b>Python Syntax</b>	GetData(<DefinitionName>)
<b>Python Example</b>	compData = oComponentManager.GetData("Level01_NPN")

## GetNames [component manager]

Returns the names of the components (used and unused) in a design. The following script command, **IsUsed**, can then be used to separate used and unused components.

*Command:* None

*Syntax:* GetNames()

*Return Value:* An array of strings

*Parameters:* None

*VB Example:*

```
Dim componentNames  
  
componentNames = oComponentManager.GetNames()
```

<b>Python Syntax</b>	GetNames()
<b>Python Example</b>	componentNames = oComponentManager.GetNames()

## **GetNPortData [component manager]**

Returns NPort data for the component with the specified name.

*Command:* None

*Return Value:* Variant array, whose contents depend on the type of component. The array will be empty if the component does not have NPort data. See the syntax for AddDynamicNPortData and AddNPortData for descriptions of the array contents for components with those types of NPort data.

*Parameters:* <ComponentName>:

<string>

*Example:* This script displays each item in the returned array for each component in the design.

```
Sub DisplayVariant(x)  
  
Dim index  
  
index = 0
```

```
For Each info In x
    curr = x(index)
    If TypeName(curr) <> "Variant()" Then
        If TypeName(curr) <> "Empty" And TypeName(curr) <> "Null" Then
            str = CStr(curr)
            If str = "" Then
                str = ChrW(34) & ChrW(34)
            End If
            MsgBox str
        Else
            str = "Empty/Null item."
            MsgBox str
        End If
    Else
        DisplayVariant curr
    End If

    index = index + 1
Next

End Sub
```

```
Dim oAnsoftApp
Dim oDesktop
Dim oProject
Dim dnpInfo
Dim index

Set oAnsoftApp = CreateObject("Ansoft.ElectronicsDesktop")
Set oDesktop = oAnsoftApp.GetAppDesktop()
oDesktop.RestoreWindow
Set oProject = oDesktop.GetActiveProject()
Set oDefinitionManager = oProject.GetDefinitionManager()

Set oComponentManager = oDefinitionManager.GetManager("Component")

Dim componentNames
componentNames = oComponentManager.GetNames()

index = 0
For Each name In componentNames
```

```
name = componentNames(index)

message = "NPort data for component " + name

Msgbox message

dnpInfo = oComponentManager.GetNPortData(name)

DisplayVariant dnpInfo

index = index + 1

Next
```

## **GetSolverOnDemandData**

*Use:* This method looks for a local component of the name passed in, and in this component it looks for an SOD model of the name passed in and returns the SOD data pertaining to that model.

*Parameters:* BSTR component name.

*Parameters:* BSTR SOD model name

*Return Value:* VARIANT which is the SOD data.

## **GetSolverOnDemandModelList**

*Use:* This method looks for a local component of the name passed in, and returns a list of SOD model names defined in the component.

*Parameters:* BSTR component name.

*Return Value:* VARIANT which is a list of SOD model names.

## IsUsed [component manager]

Used to determine if a component is used in the design.

*Command:* None

*Syntax:* IsUsed(<ComponentName>)

*Return Value:* <Boolean> // true if the specified component is used in the design

*Parameters:* <ComponentName>:

<string>

*VB Example:*

```
Dim isUsed
isUsed = oComponentManager.IsUsed("MyComponent")
```

<b>Python Syntax</b>	IsUsed(<ComponentName>)
<b>Python Example</b>	IsUsed = oComponentManager.IsUsed("MyComponent")

## Remove [component manager]

Remove a component from a library

*Command:* Tools > Edit Configured Libraries > Components > Remove Component

*Syntax:* Remove <ComponentName>,

- <IsProjectComponent>,
- <LibraryName>,
- <LibraryLocation>

*Return Value:* None

*Parameters:* <ComponentName>:

<string> // [composite name](#) of the component to remove

<IsProjectComponent>:

<bool>

<LibraryName>:

<string> // name of the library

<LibraryLocation>:

<string> // location of the library in <LibraryName>

// One of "Project", "PersonalLib", or "UserLib"

*VB Example:*

```
oComponentManager.Remove "Nexxim Circuit Elements\BJTs:Level01_NPN", _ true, "Project"
```

<b>Python Syntax</b>	Remove (<ComponentName>, <IsProjectComponent>, <LibraryName>, <LibraryLocation>)
<b>Python Example</b>	<pre>oComponentManager.Remove ("Simplorer Circuit Elements\BJTs:Level01_NPN", _ true, "Project")</pre>

## RemoveSolverOnDemandModel

*Use:* This method looks for a local component of the name passed in, and in this component it looks for an SOD model of the name passed in and deletes the SOD model definition from the component.

*Parameters:* BSTR component name.

*Parameters:* BSTR SOD model name

*Return Value:* None.

## RemoveUnused [component manager]

Removes components that are not used in the design.

*Command:* Project->Remove Unused Definitions is similar but operates slightly different and does not record script commands.

*Syntax:* RemoveUnused()

*Return Value:* <bool> True if one or more components are removed.

*Parameters:* None

*VB Example:*

```
Dim removedDefs  
removedDefs = oComponentManager.RemoveUnused()
```

**Note:**

The order of calls to RemoveUnused is significant. As a result, removing definitions in an unordered fashion may cause other components in dependent definitions to be rendered unusable.

Also, the symbol and footprint of an unused component are not unusable until after the component itself is removed using the Component Manager Remove script.

<b>Python Syntax</b>	RemoveUnused()
<b>Python Example</b>	<pre>removedDefs = oComponentManager.RemoveUnused()</pre>

## **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:Lyr$,  
"Layer:=", <LayerArray>,  
"Layer:=", <LayerArray>...,  
"SLayer:=", <StackupLayerArray>,  
"SLayer:=", <StackupLayerArray>...),  
"ActLyr:=", <string>, // name of active layer  
"Tol:=", <ToleranceArray> // optional  
<PrimitivesInfo>, // optional  
<PinsInfo>, // optional  
<ViasInfo>, // optional  
<EdgeportsInfo>, // optional  
<ComponentPropertyInfo>,  
<ScriptInfo>) // optional, specified for scripted footprints
```

*Return Value:* <string>

// [composite name](#) of the footprint.

```
// If the name requested conflicts with the name of an existing  
// footprint, the requested name is altered to be unique.  
// The name returned reflects any change made to be unique.
```

*Parameters:* <FootprintName>:

```
<string> // simple name of footprint to create
```

```
<ModifiedOnInfo>:
```

An integer that corresponds to the number of seconds that have elapsed  
since 00:00 hours, Jan 1, 1970 UTC from the system clock.

```
<UnitType>:
```

```
<string> // default length units to use if units are not specified in other  
// parameters
```

```
<LayerArray>:
```

```
Array("N:=", <string>, // layer name  
"ID:=", <int> ,  
"T:=", <LayerTypeInfo>, // layer type  
"TB:=", <TopBottomInfo>,  
"Col:=", <int>, // optional - color  
"Pat:=", <int>, // optional - fill pattern
```

```
"Vis:=", <bool>, // optional - are objects on layer visible
"Sel:=", <bool>, // optional - are objects on layer selectable
"L:=", <bool>) // optional
// are objects on layer locked (can't be edited)

<LayerTypeInfo>:
<string> // one of: signal, dielectric, metalized signal, assembly, silkscreen, soldermask, solderpaste, glue, or user

<TopBottomInfo>:
<string> // one of: top, neither, bottom, or template

<StackupLayerArray>:
Array(<LayerArray>,
"Elev:=", <ElevationInfo>,
"SubL:=", Array("Th:=", <Dimension>,
"LElev:=", <Dimension>,
"R:=", <Dimension>,
"M:=", <MaterialInfo>))

<ElevationInfo>:
<string> // "top" - snap to top
```

```
// "mid" - snap to middle
// "bot" - snap to bottom
// "edit" - manual edit
// "none"

<Dimension>:
<string> // real number, may include units

<MaterialInfo>:
<string> // name of the layer material

<ToleranceArray>:
Array(<real>, // distance tolerance
      <real>, // angle tolerance (radians)
      <real>) // dimensionless tolerance

<PrimitivesInfo>:
Array("NAME:Prims",
      // one or more of the following
      <RectInfo>,
      <CircleInfo>,
```

```
<ArcInfo>,
<LineInfo>,
<PolygonInfo>,
<TextInfo>,
<ImageInfo>

<RectInfo>:
"Rect:=", Array(<real>, // line width
<int>, // fill pattern
<int>, // color
<real>, // angle, in radians
<real>, // x position of center
<real>, // y position of center
<real>, // width
<real>) // height

<CircleInfo>:
"Circle:=", Array(<real>, // line width
<int>, // fill pattern
<int>, // color
```

```
<real>, // x position of center  
<real>, // y position of center  
<real>) // radius
```

<ArcInfo>:

```
"Arc:=", Array(<real>, // line width  
<int>, // line pattern  
<int>, // color  
<real>, // x position of center  
<real>, // y position of center  
<real>, // radius  
<real>, // start angle, in radians  
<end>) // end angle, in radians
```

<LineInfo>:

```
"Line:=", Array(<real>, // line width  
<int>, // line pattern  
<int>, // color  
<PointInfo>, // must specify at least 2 points  
<PointInfo>...)  
<PointInfo>:
```

```
<real>, // x position  
<real> // y position  
  
<PolygonInfo>:  
"Polygon:=", Array(<real>, // line width  
<int>, // fill pattern  
<int>, // color  
<PointInfo>, // must specify at least 3 points  
<PointInfo>...)  
  
<TextInfo>:  
"Text:=", Array(<real>, // x position  
<real>, // y position  
<real>, // angle, in radians  
<Justification>,  
<bool>, // is plotter font  
<string>, // font name  
<int>, // color  
<string>) // text string  
<Justification>: <int>
```

```
// 0 : left top  
// 1 : left base  
// 2 : left bottom  
// 3 : center top  
// 4 : center base  
// 5 : center bottom  
// 6 : right top  
// 7 : right base  
// 8 : right bottom
```

```
<ImageInfo>:  
"Image:=", Array(<RectInfo>,  
<ImageData>,  
<bool>) // is mirrored
```

```
<ImageData>:  
<string>, // file path  
<int>, // 0 : use the file path and link to it  
// 1 : ignore file path and use next parameter  
<string> // text data, only present if preceding int is 1
```

<PinsInfo>:

```
Array("NAME:Pins",
"P:=", <PinArray>,
"P:=", <PinArray>,...)
```

<PinArray>:

```
Array("Port:=", Array("Id:=", <int>,
"Clr:=", <real>, // optional - clearance
"N:=", <string>), // pin name
"Pos:=", Array("x:=", <Location>, // padstack (x,y) position
"y:=", <Location>),
"VRt:=", <Angle>, // optional - rotation
"HD:=", <Size>, // optional - hole diameter
<PadstackImplementationInfo>)
```

<Location>:

<string> specifying real number and units (may use variables)

<Angle>:

<string> specifying angle with a real number and units

<Size>:

<string> specifying size with a real number and units

<PadstackImplementationInfo>:

If another with the same implementation has already been specified:

"Ref:=", <int> // id of the other

If not:

"Ref:=", <string>, // name

"Frm:=", <int>, // id of highest layer

"To:=", <int>, // id of lowest layer

<LayerPlacementInfo>,

"Man:=", <int>, // optional, 1 if manually placed, 0 if not (default)

"Use:=", Array(<PadUseInfo>, <PadUseInfo>...) // array may be empty

<LayerPlacementInfo>:

"Lyr:=", Array("Mrg:=", <int>, // optional,

// 1 if all layers have been merged (default)

// 0 if layer are not merged

"Flp:=", <int>, // optional,

// 1 if placed bottom up

```
// 0 if not (default)
"Map:=", <ParentToLocalLayerInfo>

<ParentToLocalLayerInfo>:
Array("U:=", <DirectionOfUniqueness>,
"F:=", Array(<int>, <int>, ...), // forward mapping
// -1 is not mapped
"B:=", Array(<int>, <int>, ...)) // backward mapping
// -1 is not mapped

<PadUseInfo>:
"Pad:=", Array("Lid:=", <int>, // layer id
"T:=", <string>, // type : "connected", "thermal",
// "no_pad, "not_connected",
// or "not_connected_thermal"
"Man:=", <int>) // optional, 1 if manually placed
// 0 if not (default)

<DirectionOfUniqueness>:
<string> // one of "forward", "backward", or "two ways"
```

```
<ViasInfo>:  
    Array("NAME:Vias", <VialInfo>, <VialInfo>...)
```

```
<VialInfo>:  
    "V:=", Array("Id:=", <int>,  
    "N:=", <string>, // name  
    "Pos:=", Array("x:=", <Location>, // via (x,y) position  
    "y:=", <Location>),  
    "VRt:=", <Angle>, // optional - rotation  
    <ImplementationInfo>
```

```
<EdgeportsInfo>:  
    Array("NAME:EPorts", <EdgePortArray>, <EdgePortArray>...)
```

```
<EdgePortArray>:  
    Array("NAME:EP",  
    "LP:=", Array("Id:=", <int>, // port id  
    "N:=", <string>), // port name  
    "Eo:=", Array(<edge description>, <edge description>,...))
```

<edge description> for primitive edges

"et:=", "pe", "pr:=", <id>, "ei:=", <edge#>

<id>: integer that is the primitive id

<edge#>: integer that is the edge number on the primitive

<edge description> for via edges

"et:=", "pse", "layer:=", <layer id>, "se:=", <via id>,  
"sx:=", <start X location>, "sy:=", <start Y location>, "ex:=", <end X location>,  
"ey:=", <end Y location>, "h:=", <arc height>, "rad:=", <radians>

<via id>: an integer that is the id of the via to use

<layer id>: an integer that is the id of the layer of the pad of the via to use

<start X location>

<start Y Location>:

doubles that are the X, Y location of the start point of the edge arc <end X location>

<end Y Location>:

doubles that are the X, Y location of the end point of the edge arc

<arc height>: double giving the height of the edge arc (0 for a straight edge)

<radians>: double giving the arc size in radians (0 for a straight edge)

<ComponentPropertyInfo>:

```
Array("NAME:CProps",
"VariableProp:=", <VariableInfo>,
"VariableProp:=", <VariableInfo>,
...)
```

<VariableInfo>:

```
Array(<string>, // name
<FlagLetters>,
<string>, // description
"CB:=", <string>, // optional - script for call back
<string>) // value: number, variable, or expression
```

<ScriptInfo>:

```
Array("NAME:script",
```

```
"language:=", <string>, // one of "javascript" or "vbscript"  
"UsesScript:=", true,  
"script:=", <string>) // contents of script
```

*VB Example:*

```
oFootprintManager.Add (Array("NAME:BCL", _  
"ModTime:=", 1023388445, _  
"Library:=", "", _  
"LibLocation:=", "Project", _  
"OkayToMirror:=", false, _  
"DefUnits:=", "mm", _  
Array("NAME:Lyrs", _  
"Layer:=", Array("N:=", "Measures", _  
"ID:=", 8, _  
"T:=", "measures", _  
"Col:=", 4144959, _  
"Pat:=", 1), _  
"Layer:=", Array("N:=", "Rats", _  
"ID:=", 1, _  
"T:=", "rat", _
```

```
"Col:=", 16711680, _  
"Pat:=", 1), _  
"Layer:=", Array("N:=", "Errors", _  
"ID:=", 2, _  
"T:=", "error", _  
"Col:=", 255, _  
"Pat:=", 1, _  
"L:=", true), _  
"Layer:=", Array("N:=", "Symbols", _  
"ID:=", 3, _  
"T:=", "symbol", _  
"Col:=", 8323199, _  
"Pat:=", 4), _  
"Layer:=", Array("N:=", "Assembly", _  
"ID:=", 4, _  
"T:=", "assembly", _  
"TB:=", "top", _  
"Col:=", 16711680, _  
Pat:=", 3), _  
"Layer:=", Array("N:=", "Silkscreen", _  
"ID:=", 5, _
```

```
"T:=", "silkscreen", _
TB:=", "top", _
"Col:=", 8454143, _
"Pat:=", 6), _
SLayer:=", Array("Layer:=", Array("N:=", "Cover", _
"ID:=", 12, _
"T:=", "metalizedsignal", _
"Col:=", 32639, _
Pat:=", 7), _
"Elev:=", "mid", _
"SubL:=", Array("Th:=", "0mm", _
"LElev:=", "118.110236220472mil", _
"R:=", "0mm", _
"Mat:="", "")), _
"SLayer:=", Array("Layer:=", Array("N:=", "Dielec3", _
"ID:=", 11, _
"T:=", "dielectric", _
"Col:=", 8323199, _
"Pat:=", 6), _
"Elev:=", "none", _
```

```
"SubL:=", Array("Th:=", "1mm", _  
"LElev:=", "78.740157480315mil", _  
"R:=", "0mm", _  
"Mat:="", "")), _  
"SLayer:=", Array("Layer:=", Array("N:=", "Trace2", _  
"ID:=", 10, _  
"T:=", "signal", _  
"Col:=", 8355584, _  
"Pat:=", 5), _  
"Elev:=", "mid", _  
"SubL:=", Array("Th:=", "0mm", _  
"LElev:=", "78.740157480315mil", _  
"R:=", "0mm", _  
"Mat:="", "")), _  
"SLayer:=", Array("Layer:=", Array("N:=", "Dielec2", _  
"ID:=", 6, _  
"T:=", "dielectric", _  
"Col:=", 8323072, _  
"Pat:=", 4), _  
"Elev:=", "none", _  
"SubL:=", Array("Th:=", "1mm", _
```

```
"LElev:=", "39.3700787401575mil", _  
"R:=", "0mm", _  
"Mat:="", "") ), _  
"SLayer:=", Array("Layer:=", Array("N:=", "Trace1", _  
"ID:=", 0, _  
"T:=", "signal", _  
"Col:=", 32512, _  
"Pat:=", 3), _  
"Elev:=", "mid", _  
"SubL:=", Array("Th:=", "0mm", _  
"LElev:=", "39.3700787401575mil", _  
"R:=", "0mm", _  
"Mat:="", "") ), _  
"SLayer:=", Array("Layer:=", Array("N:=", "Dielec1", _  
"ID:=", 7, _  
"T:=", "dielectric", _  
"Col:=", 127, _  
"Pat:=", 7), _  
"Elev:=", "none", _  
"SubL:=", Array("Th:=", "1mm", _
```

```
"LElev:=", "0mil", _  
"R:=", "0mil", _  
"Mat:="", "") ), _  
"SLayer:=", Array("Layer:=", Array("N:=", "Ground", _  
"ID:=", 9, _  
"T:=", "metalizedsignal", _  
"Col:=", 4144959, _  
"Pat:=", 6), _  
"Elev:=", "mid", _  
"SubL:=", Array("Th:=", "0mil", _  
"LElev:=", "0mil", _  
"R:=", "0mil", _  
"Mat:="", "") )) , _  
"ActLyr:=", "Trace2", _  
Array("NAME:Prims",  
"rect:=", Array("Id:=", 10003, _  
"Lyr:=", 10, _  
"N:=", "rect101", _  
"x:=", "0mm", _  
"y:=", "0mm", _  
"w:=", "P", _
```

```
"h:=", "W", _
"vds:=", Array(), _
"rect:=", Array("Id:=", 10103, _
"lyr:=", 0, _
"N:=", "rect103", _
"x:=", "0mm", _
"y:=", "0mm", _
"w:=", "P", _
"h:=", "W", _
"vds:=", Array()), _
Array("NAME: Pins",
"P:=", Array("Port:=", Array("Id:=", 3, "N:=", "n1"), _
"Pos:=", Array("x:=", "-P/2", "y:=", "0mm"), _
"VRt:=", "180deg", _
"Ref:=", "NoPad SMT East", _
"frm:=", 12, _
"To:=", 9, _
"lyr:=", Array("Map:=", Array("U:=", "forward", _"
"F:=", Array(-1, -1, -1, -1, _"
-1, -1, -1, -1, _"
```

```
-1, -1, 0, -1, -1), _
"B:=", Array(10)), _
"Use:", Array()), _
"P:=", Array("Port:=", Array("Id:=", 4, "N:=", "n4"), _
"Pos:=", Array("x:=", "P/2", "y:=", "0mm"), _
"HD:=", "1mm", _
"Ref:=", 3), _
"P:=", Array("Port:=", Array("Id:=", 5, "N:=", "n2"), _
"Pos:=", Array("x:=", "-P/2", "y:=", "0mm"), _
"VRt:=", "180deg", _
"Ref:=", "NoPad SMT East", _
" Frm:=", 12, _
"To:=", 9, _
" Lyr:=", Array("Map:=", Array("U:=", "forward", _
" F:=", Array(0, -1, -1, -1, -1, -
-1, -1, -1, -1, -
-1, -1, -1, -1), -
" B:=", Array(0))), -
" Use:", Array()), -
"P:=", Array("Port:=", Array("Id:=", 6, "N:=", "n3"), -
"Pos:=", Array("x:=", "P/2", "y:=", "0mm"), -
```

```
"HD:=", "1mm", _  
"Ref:=", 5)), _  
Array("NAME:Nets"), _  
Array("NAME:CProps",  
"VariableProp:=", Array("W", _  
"UD", _  
"", _  
"1.5mm"), _  
"VariableProp:=", Array("P", _  
"UD", _  
"", _  
"10mm"))))
```

## Edit [footprint manager]

Deprecated command — please use [EditWithComps](#).

## EditWithComps [footprint manager]

*Use:* Edit an existing footprint.

*Command:* None

*Syntax:* EditWithComps <FootprintName>,

```
        Array("NAME:<NewFootprintName>,
```

```
"ModTime:=", <ModifiedOnInfo>,
"Library:=", "", 
"LibLocation:=", "Project",
"OkayToMirror:=", <bool>,
"DefUnits:=", <UnitType>,
Array(NAME:Lrys",
"Layer:=", <LayerArray>,
"Layer:=", <LayerArray>...,
"SLayer:=", <StackupLayerArray>,
"SLayer:=", <StackupLayerArray>...),
"ActLyr:=", <string>, // name of active layer
"Tol:=", <ToleranceArray> // optional
<PrimitivesInfo>, // optional
<PinsInfo>, // optional
<ViasInfo>, // optional
<EdgeportsInfo>, // optional
<ComponentPropertyInfo>,
<ScriptInfo>, // optional, specified for scripted footprints
Array(<ListofComponentNames>) // Component names
```

*Return Value:* <string>

// [composite name](#) of the footprint.

```
// If the name requested conflicts with the name of an existing  
// footprint, the requested name is altered to be unique.  
// The name returned reflects any change made to be unique.
```

*Parameters:* <FootprintName>:

```
<string> // composite name of the footprint being edited
```

<NewFootprintName>:

```
<string> // new simple name for the footprint
```

<ModifiedOnInfo>:

An integer that corresponds to the number of seconds that have elapsed  
since 00:00 hours, Jan 1, 1970 UTC from the system clock.

<UnitType>:

```
<string> // default length units to use if units are not specified in other  
// parameters
```

<LayerArray>:

```
Array("N:=", <string>, // layer name
```

```
"ID:=", <int> ,  
"T:=", <LayerTypeInfo>, // layer type  
"TB:=", <TopBottomInfo>,  
"Col:=", <int>, // optional - color  
"Pat:=", <int>, // optional - fill pattern  
"Vis:=", <bool>, // optional - are objects on layer visible  
"Sel:=", <bool>, // optional - are objects on layer selectable  
"L:=", <bool>) // optional  
// are objects on layer locked (can't be edited)
```

**<LayerTypeInfo>:**

```
<string> // one of: signal, dielectric, metalized signal, assembly, silkscreen, soldermask, solderpaste, glue, or user
```

**<TopBottomInfo>:**

```
<string> // one of: top, neither, bottom, or template
```

**<StackupLayerArray>:**

```
Array(<LayerArray>,  
"Elev:=", <ElevationInfo>,  
"SubL:=", Array("Th:=", <Dimension>,  
"LElev:=", <Dimension>,
```

```
"R:=", <Dimension>,  
"M:=", <MaterialInfo>))
```

**<ElevationInfo>:**

```
<string> // "top" - snap to top  
// "mid" - snap to middle  
// "bot" - snap to bottom  
// "edit" - manual edit  
// "none"
```

**<Dimension>:**

```
<string> // real number, may include units
```

**<MaterialInfo>:**

```
<string> // name of the layer material
```

**<ToleranceArray>:**

```
Array(<real>, // distance tolerance  
<real>, // angle tolerance (radians)  
<real>) // dimensionless tolerance
```

**<PrimitivesInfo>:**

```
Array("NAME:Prims",
// one or more of the following
<RectInfo>,
<CircleInfo>,
<ArcInfo>,
<LineInfo>,
<PolygonInfo>,
<TextInfo>,
<ImageInfo>)
```

**<RectInfo>:**

```
"Rect:=", Array(<real>, // line width
<int>, // fill pattern
<int>, // color
<real>, // angle, in radians
<real>, // x position of center
<real>, // y position of center
<real>, // width
<real>) // height
```

**<CircleInfo>:**

```
"Circle:=", Array(<real>, // line width  
<int>, // fill pattern  
<int>, // color  
<real>, // x position of center  
<real>, // y position of center  
< real>) // radius
```

**<ArcInfo>:**

```
"Arc:=", Array(<real>, // line width  
<int>, // line pattern  
<int>, // color  
<real>, // x position of center  
<real>, // y position of center  
< real>, // radius  
<real>, // start angle, in radians  
<end>) // end angle, in radians
```

**<LineInfo>:**

```
"Line:=", Array(<real>, // line width  
<int>, // line pattern  
<int>, // color  
<PointInfo>, // must specify at least 2 points  
<PointInfo>...)  
<PointInfo>:  
<real>, // x position  
<real> // y position
```

#### **<PolygonInfo>:**

```
"Polygon:=", Array(<real>, // line width  
<int>, // fill pattern  
<int>, // color  
<PointInfo>, // must specify at least 3 points  
<PointInfo>...)
```

#### **<TextInfo>:**

```
"Text:=", Array(<real>, // x position  
<real>, // y position  
<real>, // angle, in radians  
<Justification>,
```

```
<bool>, // is plotter font  
<string>, // font name  
<int>, // color  
<string>) // text string  
<Justification>: <int>  
    // 0 : left top  
    // 1 : left base  
    // 2 : left bottom  
    // 3 : center top  
    // 4 : center base  
    // 5 : center bottom  
    // 6 : right top  
    // 7 : right base  
    // 8 : right bottom
```

**<ImageInfo>:**

```
"Image:=", Array(<RectInfo>,  
<ImageData>,  
<bool>) // is mirrored
```

**<ImageData>:**

```
<string>, // file path  
<int>, // 0 : use the file path and link to it  
// 1 : ignore file path and use next parameter  
<string> // text data, only present if preceding int is 1
```

**<PinsInfo>:**

```
Array("NAME:Pins",  
"P:=", <PinArray>,  
"P:=", <PinArray>,...)
```

**<PinArray>:**

```
Array("Port:=", Array("Id:=", <int>,  
"Clr:=", <real>, // optional - clearance  
"N:=", <string>), // pin name  
"Pos:=", Array("x:=", <Location>, // padstack (x,y) position  
"y:=", <Location>),  
"VRt:=", <Angle>, // optional - rotation  
"HD:=", <Size>, // optional - hole diameter  
<PadstackImplementationInfo>)
```

**<Location>:**

<string> specifying real number and units (may use variables)

**<Angle>:**

<string> specifying angle with a real number and units

**<Size>:**

<string> specifying size with a real number and units

**<PadstackImplementationInfo>:**

If another with the same implementation has already been specified:

"Ref:=", <int> // id of the other

If not:

"Ref:=", <string>, // name

"Frm:=", <int>, // id of highest layer

"To:=", <int>, // id of lowest layer

<LayerPlacementInfo>,

"Man:=", <int>, // optional, 1 if manually placed, 0 if not (default)

"Use:=", Array(<PadUseInfo>, <PadUseInfo>...) // array may be empty

**<LayerPlacementInfo>:**

```
"Lyr:=", Array("Mrg:=", <int>, // optional,  
// 1 if all layers have been merged (default)  
// 0 if layer are not merged  
"Flp:=", <int>, // optional,  
// 1 if placed bottom up  
// 0 if not (default)  
"Map:=", <ParentToLocalLayerInfo>)
```

**<ParentToLocalLayerInfo>:**

```
Array("U:=", <DirectionOfUniqueness>,  
"F:=", Array(<int>, <int>, ...), // forward mapping  
// -1 is not mapped  
"B:=", Array(<int>, <int>, ...)) // backward mapping  
// -1 is not mapped
```

**<PadUseInfo>:**

```
"Pad:=", Array("Lid:=", <int>, // layer id  
"T:=", <string>, // type : "connected", "thermal",  
// "no_pad, "not_connected",  
// or "not_connected_thermal"
```

```
"Man:=", <int>) // optional, 1 if manually placed  
// 0 if not (default)
```

**<DirectionOfUniqueness>:**

```
<string> // one of "forward", "backward", or "two ways"
```

**<ViasInfo>:**

```
Array("NAME:Vias", <VialInfo>, <VialInfo>...)
```

**<VialInfo>:**

```
"V:=", Array("Id:=", <int>,  
"N:=", <string>, // name  
"Pos:=", Array("x:=", <Location>, // via (x,y) position  
"y:=", <Location>),  
"VRt:=", <Angle>, // optional - rotation  
<ImplementationInfo>
```

**<EdgeportsInfo>:**

```
Array("NAME:EPorts", <EdgePortArray>, <EdgePortArray>...)
```

**<EdgePortArray>**

```
Array("NAME:EP",
"LP:=", Array("Id:=", <int>, // port id
"N:=", <string>), // port name
"Eo:=", Array(<edge description>, <edge description>, ...))
```

<edgedescription> for primitive edges

```
"et:=", "pe", "pr:=", <id>, "ei:=", <edge#>
```

<id>: integer that is the primitive id

<edge#>: integer that is the edge number on the primitive

<edge description> for via edges

```
"et:=", "pse", "sel:=", <"via">, "layer:=", <layer id>,
"sx:=", <start X location>, "sy:=", <start Y location>, "ex:=", <end X location>, "ey:=", <end Y location>, "h:=", <arc height>,
"rad:=", <radians>
```

<"via">: text that is the name of the via to use

<layer id>: an integer that is the id of the layer of the pad of the via to use

<start X location>, <start Y Location>:

doubles that are the X, Y location of the start point of the edge arc

<end X location>, <end Y Location>:

doubles that are the X, Y location of the end point of the edge arc

<arc height>: double giving the height of the edge arc (0 for a straight edge)

<radians>: double giving the arc size in radians (0 for a straight edge)

**<ComponentPropertyInfo>:**

```
Array("NAME:CProps",
"VariableProp:=", <VariableInfo>,
"VariableProp:=", <VariableInfo>,
...)
```

**<VariableInfo>:**

```
Array(<string>, // name
<FlagLetters>,
<string>, // description
"CB:=", <string>, // optional - script for call back
```

```
<string>) // value: number, variable, or expression
```

```
<ScriptInfo>:
```

```
Array("NAME:script",
"language:=", <string>, // one of "javascript" or "vbscript"
"UsesScript:=", true,
"script:=", <string>) // contents of script
```

```
<ListOfComponentNames>:
```

```
<string>,<string> ...
// The list may be empty. When not empty, each string that is listed is a component
// that references the footprint to be edited. Prior to editing, a clone of the footprint is
// made, and the components that are listed are modified so that they now refer to
// the clone.
```

*VB Example:*

```
Dim nam
oFootprintManager>EditWithComps _
("Nexxim Circuit Elements\ Distributed\ Distributed:bendo", _
Array("NAME:bendo new name", _
"ModTime:=", 1152722165, _
```

```
"Library:=", "Nexxim Circuit Elements\ Distributed\ Distributed", _  
"LibLocation:=", "SysLibrary", _  
Array("NAME:PinDef", _  
"Pin:=", Array( "n1", _  
-0.00254, _  
0.00254, _  
0, _  
"N", _  
0, _  
0, _  
false, _  
0, _  
true, _  
"", _  
false, _  
false)), _  
Array("NAME:PinDef", _  
"Pin:=", Array( "n2", _  
0.00254, _  
-0.00254, _
```

```
1.5708, _
"N", _
0, _
0, _
false, _
0, _
true, _
"", _
false, _
false)), _
Array("NAME:Graphics", _
"Polygon:=", Array( 0, 0, 12566272, 0, 0.00381, 0, .00127, 0.00127, _
0.00127, 0.00127, 0, 0.00381, 0, 0.00381, _
0.002032, 0.00127, 0.00381), _
"Line:=", Array(0, 1, 12566272, -0.00254, 0.00254, 0, .00254), _
"Line:=", Array(0, 1, 12566272, 0.00254, -0.00254, .00254, 0)), _
Array("NAME:PropDisplayMap", _
"W:=", Array(3, _
5, _
0, _
"Text:=", Array( 2.1684E-019, _
```

```
0.00504119, _  
0, _  
1, _  
5, _  
false, _  
"Arial", _  
0, _  
"W=***") )) ), _  
Array("MY_COMP")
```

## **Export [footprint manager]**

*Use:* Export a footprint to a library

*Command:* Tools > Edit Configured Libraries > Footprints > Export to Library

*Syntax:* Export Array("NAME:<LibraryName>,"

```
<FootprintName>,  
<FootprintName>...),  
<LibraryLocation>
```

*Return Value:* None

*Parameters:* <LibraryName>:

```
<string> // name of the library
```

```
<FootprintName>
<string> // composite name of footprint to export

<LibraryLocation>
<string> // location of the library in <LibraryName>
// One of "Project", "PersonalLib", or "UserLib"
```

*VB Example:*

```
oFootprintManager.Export Array("NAME:mylib", "Distributed Footprints:BPAD"), "PersonalLib"
```

## **GetData [footprint manager]**

### **FootprintGetData**

*Use:* Gets data that describes the definition.

*Command:* None

*Syntax:* GetData(<DefinitionName>)

*Return Value:* <DefinitionData> This is an array of data for the definition.

*Parameters:* <DefinitionName>:

```
<string> // composite name of the definition to edit
```

*VB Example:*

```
Dim footprintData
footprintData = oFootprintManager.GetData("Nexxim Circuit Elements\ Distributed\Nexxim_Foot-
prints:MCPL13_Nexx")
```

---

**Note:**

GetData allows the user to access definition information, make modifications, and then use the Edit or EditWithComps script commands to save the modified definition. Accordingly, for each type of definition, the array data returned to GetData should be the same array information that is supplied to the [Edit](#) or [EditWithComps](#) commands.

## **GetNames [footprint manager]**

*Use:* Returns the names of the footprints (used and unused) in a design. The following script command, **IsUsed**, can then be used to separate used and unused footprints.

*Command:* None

*Syntax:* GetNames()

*Return Value:* An array of strings

*Parameters:* None

*VB Example:*

```
Dim footprintNames  
footprintNames = oFootprintManager.GetNames()
```

## **IsUsed [footprint manager]**

*Use:* Used to determine if a footprint is used in the design.

*Command:* None

*Syntax:* IsUsed(<FootprintName>)

*Return Value:* <Boolean> // true if the specified footprint is used in the design

*Parameters:* <FootprintName>:

<string>

*VB Example:*

```
Dim isUsed  
isUsed = oFootprintManager.IsUsed("MyFootprint")
```

## Remove [footprint manager]

*Use:* Removes a footprint from a library

*Command:* Tools > Edit Configured Libraries > Footprints > Remove Footprint

*Syntax:* Remove <FootprintName>,

```
<IsProjectFootprint>,  
<LibraryName>,  
<LibraryLocation>
```

*Return Value:* None

*Parameters:* <FootprintName>:

<string> // [composite name](#) of the footprint to remove

<IsProjectFootprint>:

<bool>

<LibraryName>:

<string> // name of the library

```
<LibraryLocation>:  
    <string> // location of the library in <LibraryName>  
    // One of "Project", "PersonalLib", or "UserLib"
```

*VB Example:*

```
oFootprintManager.Remove "BPAD", true, "Distributed Footprints", "Project"  
oFootprintManager.Remove "BPAD", false, "MyLib", "PersonalLib"
```

## **RemoveUnused [footprint manager]**

*Use:* Removes footprints that are not used in the design.

*Command:* **Project->Remove Unused Definitions** is similar but operates slightly different and does not record script commands.

*Syntax:* RemoveUnused()

*Return Value:* <bool> True if one or more footprints are removed.

*Parameters:* None

*VB Example:*

```
Dim removedDefs  
removedDefs = oFootprintManager.RemoveUnused()
```

**Note:**

The order of calls to RemoveUnused is significant. As a result, removing definitions in an unordered fashion may cause other footprints in dependent definitions to be rendered unusable.

Also, the symbol and footprint of an unused component are not unusable until after the component itself is removed using the Component ManagerRemove script.

## Material Manager Script Commands

The material manager provides access to materials in a project. The manager object is accessed via the definition manager.

```
Set oDefinitionManager = oProject.GetDefinitionManager()  
Set oMaterialManager = oDefinitionManager.GetManager("Material")
```

**The topics for this section include:**

[GetData](#)  
[GetNames](#)  
[GetProperties](#)  
[IsUsed](#)  
[RemoveUnused](#)

### GetData [material manager]

*Use:* Get material data

*Command:* None

*Syntax:* GetData <material\_name>

*Return Value:* Array of material data

*Parameters:* <material\_name>

Type: string

Value: Name of the project material

*VB Example:*

```
materialData = oMaterialMgr.GetData("MyMaterial2")
```

<b>Python Syntax</b>	GetData(<SimpleName>)
<b>Python Example</b>	<pre>dataArray = oMaterialManager.GetData ("mymaterial")</pre>

## GetNames [material manager]

*Use:* Get the names of the materials in a project

*Command:* None

*Syntax:* GetNames

*Return Value:* Names of the materials in a project

*Parameters:* None

*Example:*

```
materialNames = oMaterialMgr.GetNames()
```

```
Dim oAnsoftApp
Dim oDesktop
Dim oProject
Dim oDesign
Dim oEditor
Dim oModule

Set oAnsoftApp = CreateObject("Ansoft.ElectronicsDesktop")
Set oDesktop = oAnsoftApp.GetAppDesktop()
oDesktop.RestoreWindow

Set oProject = oDesktop.NewProject
oProject.InsertDesign "Nexxim Circuit", "Nexxim1", "", ""

Set oMaterialMgr = oProject.GetDefinitionManager().GetManager("Material")
oMaterialMgr.Add( Array("NAME:MyMaterial1", "CoordinateSystemType:=", "Cartesian", "permittivity:=", "0.123") )
oMaterialMgr.Add( Array("NAME:MyMaterial2", "CoordinateSystemType:=", "Cartesian", "permittivity:=", "0.456") )

Dim materialNames
materialNames = oMaterialMgr.GetNames()
```

<b>Python Syntax</b>	GetNames()
<b>Python Example</b>	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 <material_name>	Type string Boolean	Description 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" )
```

<b>Python Syntax</b>	IsUsed(<ComboName>)
<b>Python Example</b>	isused = oMaterialManager.IsUsed("mylib:mymaterial")

## RemoveUnused [material manager]

**Use:** Remove all unused materials from the project.

**Command:** None

**Syntax:** RemoveUnused

**Return Value:** None

**Parameters:** None

**VB Example:**

```
oMaterialMgr.RemoveUnused()
```

<b>Python Syntax</b>	RemoveUnused()
<b>Python Example</b>	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>,
```

```
<ArclInfo>,
```

```
<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\Elements\Elements\bendo:mylib", _ "myModel"), "Per-  
sonalLib"
```

<b>Python Syntax</b>	Export (["NAME:<LibraryName>", <ComboName>, <ComboName>...], <LibraryLocation>)
<b>Python Example</b>	oModelManager.Export ( ["NAME:mylib", "model1", "model2"] )

## GetData [model manager]

*Use:* Gets data that describes the definition.

*Command:* None

*Syntax:* GetData(<DefinitionName>)

*Return Value:* <DefinitionData> This is an array of data for the definition.

*Parameters:* <DefinitionName>:

<string> // composite name of the definition to edit

*VB Example:*

```
Dim ModelData
ModelData = oModelManager.GetData("Nexxim Circuit Elements\Hspice:nexx_bjt_npn")
```

### Note:

GetData allows the user to access definition information, make modifications, and then use the Edit or EditWithComps script commands to save the modified definition. Accordingly, for each type of definition, the array data returned to GetData should be the same array information that is supplied to the [Edit](#) or [EditWithComps](#) commands.

<b>Python Syntax</b>	GetData(<ComboName>)
<b>Python Example</b>	dataArray = oModelManager.GetData

	("mylib:mymodel")
--	-------------------

## GetNames [model manager]

*Use:* Returns the names of the models (used and unused) in a design. The following script command, **IsUsed**, can then be used to separate used and unused models.

*Command:* None

*Syntax:* GetNames()

*Return Value:* An array of strings

*Parameters:* None

*VB Example:*

```
Dim modelNames
```

```
modelNames = oModelManager.GetNames()
```

<b>Python Syntax</b>	GetNames()
<b>Python Example</b>	modelnames = oModelManager.GetNames()

## IsUsed [model manager]

*Use:* Used to determine if a model is used in the design.

*Command:* None

*Syntax:* IsUsed(<ModelName>)

*Return Value:* <Boolean> // true if the specified model is used in the design

*Parameters:* <ModelName>:

<string>

*VB Example:*

```
Dim isUsed
isUsed = oModelManager.IsUsed("MyModel")
```

<b>Python Syntax</b>	IsUsed(<ComboName>)
<b>Python Example</b>	isused = oModelManager.IsUsed ("mylib:mymodel")

## Remove [model manager]

*Use:* Removes a model from a library

*Command:* Tools > Edit Configured Libraries > Models > Remove Model

*Syntax:* Remove <ModelName>,

- <IsProjectModel>,
- <LibraryName>,
- <LibraryLocation>

*Return Value:* None

*Parameters:* <ModelName>:

<string> // composite name of the model to remove

**<IsProjectModel>:**

<bool>

**<LibraryName>:**

```
<string> // name of the library
```

**<LibraryLocation>:**

```
<string> // location of the library in <LibraryName>  
// One of "Project", "PersonalLib", or "UserLib"
```

*VB Example:*

```
oModelManager.Remove "Nexxim Circuit Elements\ Distributed\ Distributed:bendo", true, "Project"
```

<b>Python Syntax</b>	Remove (<ModelName>, <IsLocal>, <LibraryName>, <LibraryLocation>)
<b>Python Example</b>	<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.

<b>Python Syntax</b>	RemoveUnused()
<b>Python Example</b>	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.

<b>UI Access</b>	N/A		
<b>Parameters</b>	Name	Type	Description
	<DesignVariationKey>	String	Design variation key. Pass empty string for the current nominal variation.

		sweep.
<SolnSelector>	String	Solution setup name and solution name, separated by a colon.
<FileFormat>	Integer	<p>File format value.</p> <p>2 : Tab delimited spreadsheet format (.tab)</p> <p>3 : Touchstone (.sNp)</p> <p>4 : CitiFile (.cit)</p> <p>7 : Matlab (.m)</p> <p>8 : Terminal Z0 spreadsheet</p>
<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	<p>Optional. Type: "0", "1", or "2"</p> <p>The format to use for the exported data.</p> <p>0 = Magnitude/Phase.</p> <p>1= Real/Imaginary.</p> <p>2= db/Phase.</p>

	<i>&lt;Precision&gt;</i>	Integer	Optional. Touchstone number of digits precision. Default if not specified is 15.
	<i>&lt;UseExportFreqs&gt;</i>	Boolean	Specifies whether to use export frequencies.
	<i>&lt;IncludeGammaComments&gt;</i>	Boolean	Specifies whether to include Gamma and Impedance comments.
	<i>&lt;SupportNonStdExport&gt;</i>	Boolean	Specifies whether to support non-standard Touchstone extensions for mixed reference impedance.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	ExportNetworkData(<DesignVariationKey>, <SolnSelectionArray>, <SolnSelector>, <FileFormat>, <OutFile>, <FreqsArray>, <DoRenorm>, <RenormImped>, [Optional <DataType>], [Optional <pass>], [Optional <ComplexFormat>], [Optional <Precision>], [Optional <UseExportFreqs>], [Optional <IncludeGammaComments>], [Optional <SupportNonStdExport>])
<b>Python Example</b>	<pre> oModule.ExportNetworkData("", ["Setup1:Sweep1"], 3, "C://Documents/package_HFSSDesign1.s4p", ['all'], True, 50, "S", -1, 0, 15, True, True, True </pre>

<b>VB Syntax</b>	ExportNetworkData <DesignVariationKey>, <SolnSelectionArray>, <SolnSelector>, <FileFormat>, <OutFile>, <FreqsArray>, <DoRenorm>, <RenormImped>, [Optional <DataType>], [Optional <pass>], [Optional <ComplexFormat>], [Optional <Precision>], [Optional <UseExportFreqs>], [Optional <IncludeGammaComments>], [Optional <SupportNonStdExport>]
<b>VB Example</b>	<pre> oModule.ExportNetworkData "width='2in'", _     Array("Setup1:Sweep1"), 2, "c:\mydir\out.tab", _     Array("all"), false, 0  oModule.ExportNetworkData "width='2in'", _ </pre>

```
Array("Setup1:Sweep1", "Setup1:Sweep2"), 3, _  
"c:\mydir\out.s2p", Array(1.0e9, 1.5e9, 2.0e9), _  
true, 50.0
```

## ExportNMFData

**Use:** Exports s-parameters in neutral file format.

**Command:** In the **matrix** tab of the **Solution** dialog box, click **Export->S-Parameter**. Then select the **Neutral Model** file format.

**Syntax:** ExportNetworkData <SolutionName> <FileName> <ReduceMatrix><Reference Impedance> <FrequencyArray> <DesignVariation> <Format> <Length> <PassNumber>

**Return Value:** None

*Parameters:* <SolutionName>

Type: <String>

Format: <SetupName>:<SolutionName>

Solution that is exported.

<FileName>

Type:<String>

The name of the file. The file extension will determine the file format.

<ReduceMatrix>

Type: <string>

Either "Original" or one of the reduce matrix setup name

<Reference Impedance>

Type: <Double>

Reference impedance

<FrequencyArray>

Type: <Array of doubles>

Value: Array(<double>, <double>....)

Frequency points in the sweep that is exported.

<DesignVariation>

Type: <String>

Design variation at which the solution is exported.

<Format>

Type: <String>

Values: "MagPhase", "RealImag", "DbPhase".

The format in which the sparameters will be exported.

<Length>

Type:<String>

Length for exporting sparameters.

<PassNumber>

Type:<integer>

Pass number.

*VB Example:*

```
oDesign.ExportNMFDATA "Setup7 : LastAdaptive", "C:/temp/neu.nmf", "Original", _ 50, Array  
(10000000, 20000000, 30000000, 40000000, 50000000, 60000000, 70000000, _ 80000000, 90000000,  
100000000), "", "MagPhase", "7meter"
```

**Python Syntax**

```
ExportNMFDATA(  
    "",  
    # Empty string.  
    1,  
    #The number 1.  
    "Name",  
    # This is the full path of the file from which the solution is loaded  
    "ExportFile",  
    # full path of file to export to  
    ["NAME:Frequencies"],  
    #optional, if none defined all frequencies are used  
    ["NAME:NMFOptions"],  
    #Export NMF options object  
    "DataTypes:=", ["S"],  
    #DataTypes can be "S", "Y", "Z", "G", and "Z0", for S , Y, Z matrix, Gamma and Z0 (zero)  
    "DisplayFormat:=", "MA",
```

```
#DisplayFormat "MA", "RI", "DB"
    "FileType:=", "",

# Export File Type
2 - Spreadsheet(*.tab)
3 - Touchstone(*.sNp)
4 - Citifile(*.cit)
6 - Neutral format(*.nmf)
7 - Matlab format(*.m)

    "Renormalize:=", false,
#Renormalize true/false
    "RefImpedance:=", 50,
# Reference Impedance
    "Precision:=", 8,
# Number of digits Precision
    "Variables:=", [ "FF", "cap", "Rs"]
# Array of variables
    "Variations:=", [ "", "", ""]
# Array of variations to export solutions for
    [ "NAME:ConstantVars"]

#Array of variables that are constant, can be empty
    [ "NAME: DependentVars"]
```

	<pre>#Array of variables that are dependent, can be empty     "MatrixSize:=", 2, #Matrix size, optional (used in nmf file header)     "CreateNPortModel:=", true #Create a model based on the exported file true/false     ])</pre>
<b>Python Example</b>	<pre>oTool = oDesktop.GetTool("NdExplorer") oTool.ExportNetworkData("", True, "C:/.../1000OHM.S2P", "\$SYSLIB/Test.s2p", "",  [     "NAME:Frequencies",     500000000,     1000000000,     10000000000,     25000000000,     50000000000,     75000000000,     100000000000 ],</pre>

```
[  
    "NAME:TSOptions",  
    "DataTypes:=" , [ "S" ],  
    "DisplayFormat:=" , "DB",  
    "FileType:=" , 3,  
    "Renormalize:=" , True,  
    "RefImpedance:=" , 50,  
    "Precision:=" , 6,  
    "UseMultipleCores:=" , False,  
    "NumberOfCores:=" , 1,  
    "Comments:=" , True,  
    "Noise:=" , False  
])
```

## 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....),
"hlc:=", <PadInfo>
"hrg:=", <HoleRange>,
"sbsh:=", <SolderballShape>,
"sbpl:=", <SolderballPlacement>,
"sbr:=", <string>, // solderball diameter, real with units
"sb2:=", <string>, // solderball mid diameter, real with units
"sbm:=", <string>), // name of solderball material
"ppl:=", <PadPortLayerArray>)
```

*Return Value:* [simple name](#) of the added padstack

```
// If the name requested conflicts with the name of an existing
// padstack, the requested name is altered to be unique.
// The name returned reflects any change made to be unique.
```

*Parameters:* <PadstackName>:

```
<string> // simple name of padstack to create
```

<ModifiedOnInfo>:

An integer that corresponds to the number of seconds that have elapsed  
since 00:00 hours, Jan 1, 1970 UTC from the system clock.

```
<LayerGeometryArray>
  Array("Name:lgm",
    "lay:=", <string>, // definition layer name
    "id:=", <int>, // definition layer id
    "pad:=", <PadInfo>, // pad
    "ant:=", <PadInfo>, // antipad
    "thm:=", <PadInfo>, // themal pad
    "X:=", <string>, // pad x connection, real with units
    "Y:=", <string>, // pad y connection, real with units
    "dir:=", <DirectionString>) // pad connection direction

<PadInfo>
  Array("shp:=", <PadShape>,
    "Szs:=", <DimensionArray>,
    "X:=", <string>, // x offset, real with units
    "Y:=", <string>, // y offset, real with units
    "R:=", <string>) // rotation, real with units

<PadShape>
```

<string> one of these choices

"No" // no pad

"Cir" // Circle

"Sq" // Square

"Rct" // Rectangle

"Ov" // Oval

"Blt" // Bullet

"Ply" // Polygons

"R45" // Round 45 thermal

"R90" // Round 90 thermal

"S45" // Square 45 thermal

"S90" // Square 90 thermal

<DimensionArray>:

Array(<string>, ...) // each string is a real with units for one of the dimensions of the shape

<DirectionString>:

<string> one of these choices

"No" // no direction

"Any" // any direction

"0" // 0 degrees

"45" // 45 degrees

"90" // 90 degrees

"135" // 135 degrees

"180" // 180 degrees

"225" // 225 degrees

"270" // 270 degrees

"315" // 315 degrees

<HoleRange>:

<string> one of these choices

"Thr" // through all layout layers

"Beg" // from upper pad layer to lowest layout layer

"End" // from upper layout layer to lowest pad layer

"UTL" // from upper pad layer to lowest pad layer

<SolderballShape>:

<string> one of these choices

"None" // no solderball

"Cyl" // cylinder solderball

"Sph" // spheroid solderball

<SolderballPlacement>:

<string> one of these choices

"abv" // above padstack

"blw" // below padstack

<PadPortLayerArray>:

Array( <int>, <int>,....) where each int is a layer id

*VB Example:*

```

oPadstackManager.Add Array("NAME:Circle - through3", "ModTime:=", 1235765635, "Library:=", _
"", "LibLocation:=", "Project", Array("NAME:psd", "nam:=", "Circle - through3", "lib:=", _
"", "mat:="", "", "plt:=", "0", Array("NAME:pds", Array("NAME:lgm", "lay:=", "Top Signal",
"id:=", _
0, "pad:=", Array("shp:=", "Cir", "Szs:=", Array("2.5mm"), "X:=", "0mm", "Y:=", _
"0mm", "R:=", "0deg"), "ant:=", Array("shp:=", "Cir", "Szs:=", Array("3.5mm"), "X:=", _
"0mm", "Y:=", "0mm", "R:=", "0"), "thm:=", Array("shp:=", "No", "Szs:=", Array(), "X:=", _
"0mm", "Y:=", "0mm", "R:=", "0"), "X:=", "0mm", "Y:=", "0mm", "dir:=", "Any"), Array("NAME:lgm",
"lay:=", _
"SignalA", "id:=", 1, "pad:=", Array("shp:=", "Cir", "Szs:=", Array("2mm"), "X:=", _
"0mm", "Y:=", "0mm", "R:=", "0deg"), "ant:=", Array("shp:=", "Cir", "Szs:=", Array( _
"3mm"), "X:=", "0mm", "Y:=", "0mm", "R:=", "0"), "thm:=", Array("shp:=", "No", "Szs:=", Array(),
"X:=", _
"0mm", "Y:=", "0mm", "R:=", "0"), "X:=", "0mm", "Y:=", "0mm", "dir:=", "Any"), Array("NAME:lgm",
"lay:=", _
"SignalB", "id:=", 2, "pad:=", Array("shp:=", "Cir", "Szs:=", Array("2mm"), "X:=", _

```

```
"0mm", "Y:=", "0mm", "R:=", "0deg"), "ant:=", Array("shp:=", "Cir", "Szs:=", Array( _  
"3mm"), "X:=", "0mm", "Y:=", "0mm", "R:=", "0deg"), "thm:=", Array("shp:=", "No", "Szs:=", Array()  
( ), "X:=", _  
"0mm", "Y:=", "0mm", "R:=", "0"), "X:=", "0mm", "Y:=", "0mm", "dir:=", "Any"), Array("NAME:lgm",  
"lay:=", _  
"Ground", "id:=", 3, "pad:=", Array("shp:=", "No", "Szs:=", Array()), "X:=", _  
"0mm", "Y:=", "0mm", "R:=", "0deg"), "ant:=", Array("shp:=", "No", "Szs:=", Array()), "X:=", _  
"0mm", "Y:=", "0mm", "R:=", "0"), "thm:=", Array("shp:=", "R90", "Szs:=", Array( _  
"3mm", "0.75mm", "1mm"), "X:=", "0mm", "Y:=", "0mm", "R:=", "0"), "X:=", "0mm", "Y:=", _  
"0mm", "dir:=", "Any"), Array("NAME:lgm", "lay:=", "Bottom signal", "id:=", 5, "pad:=", Array  
("shp:=", _  
"Cir", "Szs:=", Array("1mm"), "X:=", "0mm", "Y:=", "0mm", "R:=", "0deg"), "ant:=", Array("shp:=",  
_  
"Cir", "Szs:=", Array("2mm"), "X:=", "0mm", "Y:=", "0mm", "R:=", "0deg"), "thm:=", Array("shp:=",  
_  
"No", "Szs:=", Array(), "X:=", "0mm", "Y:=", "0mm", "R:=", "0"), "X:=", "0mm", "Y:=", _  
"0mm", "dir:=", "Any)), "hle:=", Array("shp:=", "Cir", "Szs:=", Array("1.5mm"), "X:=", _  
"0mm", "Y:=", "0mm", "R:=", "0deg"), "hRg:=", "End", "sbsh:=", "Sph", "sbpl:=", _  
"abv", "sbr:=", "750um", "sb2:=", "1200um", "1200um", "sbn:=", "solder"), "ppl:=", Array( _  
0, 1, 2, 3, 5))
```

## Edit [padstack manager]

*Use:* Edit an existing padstack.

*Command:* Tools > Edit Configured Libraries > Padstacks > Edit Padstack

*Syntax:* Edit <PadstackName>,

```
Array("NAME:<NewPadstackName>",
"ModTime:=", <ModifiedOnInfo>,
"Library:=", "", // name of the library
"LibLocation:=", "Project", // location of the named library
Array("NAME:psd",
"nam:=", <PadstackName>,
"lib:=", "", // name of the library
"mat:=", "", // hole plating material
"plt:=", "0", // percent of hole's radius filled by plating
Array("NAME:pds",
<LayerGeometryArray>,
<LayerGeometryArray....>),
"hole:=", <PadInfo>
"holeRg:=", <HoleRange>,
"sbsh:=", <SolderballShape>,
"sbpl:=", <SolderballPlacement>,
"sbr:=", <string>, // solderball diameter, real with units
"sb2:=", <string>, // solderball mid diameter, real with units
"sbm:=", <string>), // name of solderball material
```

```
"ppl:=", <PadPortLayerArray>)
```

*Return Value:* <string> // [composite name](#) of the padstack

```
// If the name requested conflicts with the name of an existing  
// padstack, the requested name is altered to be unique.  
// The name returned reflects any change made to be unique.
```

*Parameters:* <PadstackName>:

```
<string> // composite name of padstack to edit
```

<NewPadstackName>:

```
<string> // new simple name for padstack
```

<ModifiedOnInfo>:

An integer that corresponds to the number of seconds that have elapsed  
since 00:00 hours, Jan 1, 1970 UTC from the system clock.

<LayerGeometryArray>:

```
Array("Name:lgm",
```

```
"lay:=", <string>, // definition layer name
```

```
"id:=", <int>, // definition layer id
```

```
"pad:=", <PadInfo>, // pad
"ant:=", <PadInfo>, // antipad


<PadInfo>:


```

```
Array("shp:=", <PadShape>,
"Szs:=", <DimensionArray>,
"X:=", <string>, // x offset, real with units
"Y:=", <string>, // y offset, real with units
"R:=", <string>) // rotation, real with units
```

<PadShape>:

```
<string> one of these choices
"No" // no pad
"Cir" // Circle
" Sq" // Square
"Rct" // Rectangle
"Ov" // Oval
```

```
"Blt" // Bullet  
"Ply" // Polygons  
"R45" // Round 45 thermal  
"R90" // Round 90 thermal  
"S45" // Square 45 thermal  
"S90" // Square 90 thermal
```

```
<DimensionArray>:  
Array(<string>, ...) // each string is a real with units for one of the  
// dimensions of the shape
```

```
<DirectionString>:  
<string> one of these choices  
"No" // no direction  
"Any" // any direction  
"0" // 0 degrees  
"45" // 45 degrees  
"90" // 90 degrees  
"135" // 135 degrees  
"180" // 180 degrees
```

```
"225" // 225 degrees
"270" // 270 degrees
"315" // 315 degrees

<HoleRange>:
<string> one of these choices
"Thr" // through all layout layers
"Beg" // from upper pad layer to lowest layout layer
"End" // from upper layout layer to lowest pad layer
"UTL" // from upper pad layer to lowest pad layer

<SolderballShape>:
<string> one of these choices
"None" // no solderball
"Cyl" // cylinder solderball
"Sph" // spheroid solderball

<SolderballPlacement>:
<string> one of these choices
"abv" // above padstack
"blw" // below padstack

<PadPortLayerArray>:
```

Array(<int>, <int>, ....) where each int is a layer id

*VB Example:*

```
oPadstackManager.Edit "Circle - through1", Array("NAME:Circle - through1", "ModTime:=", _  
1235765635, "Library:=", "", "LibLocation:=", "Project", Array("NAME:psd", "nam:=", _  
"Circle - through1", "lib:=", "", "mat:=", "", "plt:=", "0", Array("NAME:pds", Array("NAME:lgm",  
"lay:=", _  
"Top Signal", "id:=", 0, "pad:=", Array("shp:=", "Cir", "Szs:=", Array("2.5mm"), "X:=", _  
"0mm", "Y:=", "0mm", "R:=", "0deg"), "ant:=", Array("shp:=", "Cir", "Szs:=", Array( _  
"3.5mm"), "X:=", "0mm", "Y:=", "0mm", "R:=", "0"), "thm:=", Array("shp:=", "No", "Szs:=", Array()  
( ), "X:=", _  
"0mm", "Y:=", "0mm", "R:=", "0"), "X:=", "0mm", "Y:=", "0mm", "dir:=", "Any"), Array("NAME:lgm",  
"lay:=", _  
"SignalA", "id:=", 1, "pad:=", Array("shp:=", "Cir", "Szs:=", Array("2mm"), "X:=", _  
"0mm", "Y:=", "0mm", "R:=", "0deg"), "ant:=", Array("shp:=", "Cir", "Szs:=", Array( _  
"3mm"), "X:=", "0mm", "Y:=", "0mm", "R:=", "0"), "thm:=", Array("shp:=", "No", "Szs:=", Array()  
( ), "X:=", _  
"0mm", "Y:=", "0mm", "R:=", "0"), "X:=", "0mm", "Y:=", "0mm", "dir:=", "Any"), Array("NAME:lgm",  
"lay:=", _  
"SignalB", "id:=", 2, "pad:=", Array("shp:=", "Cir", "Szs:=", Array("2mm"), "X:=", _  
"0mm", "Y:=", "0mm", "R:=", "0deg"), "ant:=", Array("shp:=", "Cir", "Szs:=", Array( _
```

```

"3mm"), "X:=", "0mm", "Y:=", "0mm", "R:=", "0deg"), "thm:=", Array("shp:=", "No", "Szs:=", Array(), "X:=", _
"0mm", "Y:=", "0mm", "R:=", "0"), "X:=", "0mm", "Y:=", "0mm", "dir:=", "Any"), Array("NAME:lgm",
"lay:=", _
"Ground", "id:=", 3, "pad:=", Array("shp:=", "No", "Szs:=", Array(), "X:=", _
"0mm", "Y:=", "0mm", "R:=", "0deg"), "ant:=", Array("shp:=", "No", "Szs:=", Array(), "X:=", _
"0mm", "Y:=", "0mm", "R:=", "0"), "thm:=", Array("shp:=", "R90", "Szs:=", Array( _
"3mm", "0.75mm", "1mm"), "X:=", "0mm", "Y:=", "0mm", "R:=", "0"), "X:=", "0mm", "Y:=", _
"0mm", "dir:=", "Any"), Array("NAME:lgm", "lay:=", "Bottom signal", "id:=", 5, "pad:=", Array(
"shp:=", _
"Cir", "Szs:=", Array("1mm"), "X:=", "0mm", "Y:=", "0mm", "R:=", "0deg"), "ant:=", Array(
"shp:=", _
"Cir", "Szs:=", Array("2mm"), "X:=", "0mm", "Y:=", "0mm", "R:=", "0deg"), "thm:=", Array(
"shp:=", _
"No", "Szs:=", Array(), "X:=", "0mm", "Y:=", "0mm", "R:=", "0"), "X:=", "0mm", "Y:=", _
"0mm", "dir:=", "Any)), "hle:=", Array("shp:=", "Cir", "Szs:=", Array("1.5mm"), "X:=", _
"0mm", "Y:=", "0mm", "R:=", "0deg"), "hRg:=", "End", "sbsh:=", "Sph", "sbpl:=", _
"abv", "sbr:=", "750um", "sb2:=", "1200um", "1200um", "sbn:=", "solder"), "ppl:=", Array( _
0, 1, 2, 3, 5))

```

## EditWithComps [padstack manager]

*Use:* Edit an existing padstack.

*Command:* None

*Syntax:* EditWithComps <PadstackName>,

```
Array("NAME:<NewPadstackName>",
"ModTime:=", <ModifiedOnInfo>,
"Library:=", "", // name of the library
"LibLocation:=", "Project", // location of the named library
Array("NAME:psd",
"nam:=", <PadstackName>,
"lib:=", "", // name of the library
"mat:=", "", // hole plating material
"plt:=", "0", // percent of hole's radius filled by plating
Array("NAME:pds",
<LayerGeometryArray>,
<LayerGeometryArray....>),
"hlc:=", <PadInfo>
"hrG:=", <HoleRange>,
"sbsh:=", <SolderballShape>,
"sbpl:=", <SolderballPlacement>,
"sbr:=", <string>, // solderball diameter, real with units
"sb2:=", <string>, // solderball mid diameter, real with units
"sbn:=", <string>), // name of solderball material
"ppl:=", <PadPortLayerArray>,
```

```
Array(<ListOfComponentNames>) // Component names
```

*Return Value:* <string>

```
// composite name of the padstack.  
// If the name requested conflicts with the name of an existing  
// padstack, the requested name is altered to be unique.  
// The name returned reflects any change made to be unique.
```

*Parameters:* <PadstackName>:

```
<string> // composite name of the padstack being edited
```

<NewPadstackName>:

```
<string> // new simple name for the padstack
```

<ModifiedOnInfo>:

An integer that corresponds to the number of seconds that have elapsed  
since 00:00 hours, Jan 1, 1970 UTC from the system clock.

<LayerGeometryArray>:

```
Array("Name:lgm",  
"lay:=", <string>, // definition layer name
```

```
"id:=", <int>, // definition layer id  
"pad:=", <PadInfo>, // pad  
"ant:=", <PadInfo>, // antipad  
"thm:=", <PadInfo>, // themal pad  
"X:=", <string>, // pad x connection, real with units  
"Y:=", <string>, // pad y connection, real with units  
"dir:=", <DirectionString>) // pad connection direction
```

```
<PadInfo>:  
Array("shp:=", <PadShape>,  
"Szs:=", <DimensionArray>,  
"X:=", <string>, // x offset, real with units  
"Y:=", <string>, // y offset, real with units  
"R:=", <string>) // rotation, real with units
```

```
<PadShape>:  
<string> one of these choices  
"No" // no pad  
"Cir" // Circle  
"Sq" // Square
```

```
"Rct" // Rectangle  
"Ov" // Oval  
"Blt" // Bullet  
"Ply" // Polygons  
"R45" // Round 45 thermal  
"R90" // Round 90 thermal  
"S45" // Square 45 thermal  
"S90" // Square 90 thermal
```

```
<DimensionArray>:  
Array(<string>, ...) // each string is a real with units for one of the  
// dimensions of the shape
```

```
<DirectionString>:  
<string> one of these choices  
"No" // no direction  
"Any" // any direction  
"0" // 0 degrees  
"45" // 45 degrees  
"90" // 90 degrees  
"135" // 135 degrees
```

"180" // 180 degrees

"225" // 225 degrees

"270" // 270 degrees

"315" // 315 degrees

<HoleRange>:

<string> one of these choices

"Thr" // through all layout layers

"Beg" // from upper pad layer to lowest layout layer

"End" // from upper layout layer to lowest pad layer

"UTL" // from upper pad layer to lowest pad layer

<SolderballShape>:

<string> one of these choices

"None" // no solderball

"Cyl" // cylinder solderball

"Sph" // spheroid solderball

<SolderballPlacement>:

<string> one of these choices

"abv" // above padstack

```
"blw" // below padstack

<PadPortLayerArray>:
    Array( <int>, <int>,....) where each int is a layer id

<ListOfComponentNames>:
    <string>,<string> ...
    // The list may be empty. When not empty, each string that is listed is a component
    // that references the padstack to be edited. Prior to editing, a clone of the padstack is
    // made, and the components that are listed are modified so that they now refer to
    // the clone.
```

*VB Example:*

```
oPadstackManager>EditWithComps "Circle - through1", Array("NAME:Circle - through1", "ModTime:=", _
-
1235765635, "Library:=", "", "LibLocation:=", "Project", Array("NAME:psd", "nam:=", _
"Circle - through1", "lib:=", "", "mat:=", "", "plt:=", "0", Array("NAME:pds", Array("NAME:lgm",
"lay:=", _
"Top Signal", "id:=", 0, "pad:=", Array("shp:=", "Cir", "Szs:=", Array("2.5mm"), "X:=", _
"0mm", "Y:=", "0mm", "R:=", "0deg"), "ant:=", Array("shp:=", "Cir", "Szs:=", Array( _
"3.5mm"), "X:=", "0mm", "Y:=", "0mm", "R:=", "0"), "thm:=", Array("shp:=", "No", "Szs:=", Array(
()), "X:=", _
```

```
"0mm", "Y:=", "0mm", "R:=", "0"), "X:=", "0mm", "Y:=", "0mm", "dir:=", "Any"), Array("NAME:lgm",
"lay:=", _
"SignalA", "id:=", 1, "pad:=", Array("shp:=", "Cir", "Szs:=", Array("2mm"), "X:=", _
"0mm", "Y:=", "0mm", "R:=", "0deg"), "ant:=", Array("shp:=", "Cir", "Szs:=", Array( _
"3mm"), "X:=", "0mm", "Y:=", "0mm", "R:=", "0"), "thm:=", Array("shp:=", "No", "Szs:=", Array(), "X:=", _
"0mm", "Y:=", "0mm", "R:=", "0"), "X:=", "0mm", "Y:=", "0mm", "dir:=", "Any"), Array("NAME:lgm",
"lay:=", _
"SignalB", "id:=", 2, "pad:=", Array("shp:=", "Cir", "Szs:=", Array("2mm"), "X:=", _
"0mm", "Y:=", "0mm", "R:=", "0deg"), "ant:=", Array("shp:=", "Cir", "Szs:=", Array( _
"3mm"), "X:=", "0mm", "Y:=", "0mm", "R:=", "0deg"), "thm:=", Array("shp:=", "No", "Szs:=", Array(), "X:=", _
"0mm", "Y:=", "0mm", "R:=", "0"), "X:=", "0mm", "Y:=", "0mm", "dir:=", "Any"), Array("NAME:lgm",
"lay:=", _
"Ground", "id:=", 3, "pad:=", Array("shp:=", "No", "Szs:=", Array(), "X:=", _
"0mm", "Y:=", "0mm", "R:=", "0deg"), "ant:=", Array("shp:=", "No", "Szs:=", Array(), "X:=", _
"0mm", "Y:=", "0mm", "R:=", "0"), "thm:=", Array("shp:=", "R90", "Szs:=", Array( _
"3mm", "0.75mm", "1mm"), "X:=", "0mm", "Y:=", "0mm", "R:=", "0"), "X:=", "0mm", "Y:=", _
"0mm", "dir:=", "Any"), Array("NAME:lgm", "lay:=", "Bottom signal", "id:=", 5, "pad:=", Array(
"shp:=", _
"Cir", "Szs:=", Array("1mm"), "X:=", "0mm", "Y:=", "0mm", "R:=", "0deg"), "ant:=", Array("shp:=",
```

---

```
"Cir", "Szs:=", Array("2mm"), "X:=", "0mm", "Y:=", "0mm", "R:=", "0deg"), "thm:=", Array  
("shp:=", __  
"No", "Szs:=", Array(), "X:=", "0mm", "Y:=", "0mm", "R:=", "0"), "X:=", "0mm", "Y:=", __  
"0mm", "dir:=", "Any")), "hle:=", Array("shp:=", "Cir", "Szs:=", Array("1.5mm"), "X:=", __  
"0mm", "Y:=", "0mm", "R:=", "0deg"), "hRg:=", "End", "sbsh:=", "Sph", "sbpl:=", __  
"abv", "sbr:=", "750um", "sb2:=", "1200um", "1200um", "sbn:=", "solder"), "ppl:=", Array( __  
0, 1, 2, 3, 5), Array("")
```

## Export [padstack manager]

*Use:* Export a padstack to a library

*Command:* Tools > Edit Configured Libraries > Padstacks > Export to Library

*Syntax:* Export Array("NAME:<LibraryName>","

```
<PadstackName>,  
<PadstackName>...),  
<LibraryLocation>
```

*Return Value:* None

*Parameters:* <LibraryName>:

```
<string> // name of the library
```

```
<PadstackName>:
```

```
<string> // simple name of padstack to export
```

```
<LibraryLocation>:
```

```
<string> // location of the library in <LibraryName>  
// One of "Project", "PersonalLib", or "UserLib"
```

*VB Example:*

```
oPadstackManager.Export Array("NAME:mylib", "myPadstack"), "PersonalLib"
```

## **GetData [padstack manager]**

*Use:* Gets data that describes the definition.

*Command:* None

*Syntax:* GetData(<DefinitionName>)

*Return Value:* <DefinitionData> This is an array of data for the definition.

*Parameters:* <DefinitionName>:

```
<string> // composite name of the definition to edit
```

*VB Example:*

```
Dim padstackData  
padstackData = oPadstackManager.GetData("NoPad SMT East")
```

**Note:**

GetData allows the user to access definition information, make modifications, and then use the Edit or EditWithComps script commands to save the modified definition. Accordingly, for each type of definition, the array data returned to GetData should be the same array information that is supplied to the [Edit](#) or [EditWithComps](#) commands.

## GetNames [padstack manager]

*Use:* Returns the names of the padstack (used and unused) in a design. The following script command, **IsUsed**, can then be used to separate used and unused padstacks.

*Command:* None

*Syntax:* GetNames()

*Return Value:* An array of strings

*Parameters:* None

*VB Example:*

```
Dim padstackNames  
padstackNames = oPadstackManager.GetNames()
```

## IsUsed [padstack manager]

*Use:* Used to determine if a component is used in the design.

*Command:* None

*Syntax:* IsUsed(<PadstackName>)

*Return Value:* <Boolean> // true if the specified padstack is used in the design

*Parameters:* <PadstackName>:

<string>

*VB Example:*

```
Dim isUsed  
isUsed = oPadstackManager.IsUsed("MyPadstack")
```

## **Remove [padstack manager]**

*Use:* Removes a padstack from a library

*Command:* Tools > Edit Configured Libraries > Padstacks > Remove Padstacks

*Syntax:* Remove <PadstackName>,

```
<IsProjectPadstack>,  
<LibraryName>,  
<LibraryLocation>
```

*Return Value:* None

*Parameters:* <PadstackName>:

```
<string> // simple name of the padstack to remove
```

```
<IsProjectPadstack>:
```

```
<bool>
```

```
<LibraryName>:
```

```
<string> // name of the library
```

```
<LibraryLocation>:
```

```
<string> // location of the library in <LibraryName>
// One of "Project", "PersonalLib", or "UserLib"
```

*VB Example:*

```
oPadstackManager.Remove "Polygon SMT", true, "Padstacks", "Project"
oPadstackManager.Remove "Polygon SMT", false, "MyLib", "PersonalLib"
```

## RemoveUnused [padstack manager]

*Use:* Removes padstacks that are not used in the design.

*Command:* Project->**Remove Unused Definitions** is similar but operates slightly different and does not record script commands.

*Syntax:* RemoveUnused()

*Return Value:* <bool> True if one or more padstacks are removed.

*Parameters:* None

*VB Example:*

```
Dim removedDefs
removedDefs = oPadstackManager.RemoveUnused()
```

### Note:

The order of calls to RemoveUnused is significant. As a result, removing definitions in an unordered fashion may cause other padstacks in dependent definitions to be rendered unusable.

## Script and Library Scripts

The definition manager provides access to materials in a project. The manager object is accessed via the definition manager.

```
Set oDefinitionManager = oProject.GetDefinitionManager()
```

The script and library script commands are listed below.

[AddScript](#)

[EditScript](#)

[ExportScript](#)

[RemoveScript](#)

[ModifyLibraries](#)

### AddScript

Adds a script to the definition manager.

<b>UI Access</b>	N/A		
<b>Parameters</b>	Name <i>&lt;AddScriptArray&gt;</i>	Type Array	Description Structured array.  Array ("NAME:<string script name>", "ScriptLang:=", <string "vbscript" or "javascript"> "ScriptText:=, <string text of script>")
<b>Return Value</b>	None.		

<b>Python Syntax</b>	AddScript(<AddScriptArray>)
<b>Python Example</b>	<pre> oDefinitionManager.AddScript(     [         "NAME:MyScript",         "ScriptLang:=", "vbscript",         "ScriptText:=", "MsgBox(\\"HelloWorld\\")"     ] ) </pre>

<b>VB Syntax</b>	AddScript <AddScriptArray>
<b>VB Example</b>	<pre> oDefinitionManager.AddScript Array("NAME:MyScript", "ScriptLang:=", "vbscript", "ScriptText:=", "MsgBox("""HelloWorld""")") </pre>

## EditScript

Edits a script in the definition manager.

<b>UI Access</b>	N/A											
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;OriginalName&gt;</td> <td>String</td> <td>Name of the script to be edited.</td> </tr> <tr> <td>&lt;EditScriptArray&gt;</td> <td>Array</td> <td>           Structured array.            Array("NAME:&lt;string script name&gt;",           "ScriptLang:=&lt;string vbscript or javascript&gt;",           "ScriptText:=&lt;string text of script&gt;")         </td> </tr> </tbody> </table>	Name	Type	Description	<OriginalName>	String	Name of the script to be edited.	<EditScriptArray>	Array	Structured array. Array("NAME:<string script name>",           "ScriptLang:=<string vbscript or javascript>",           "ScriptText:=<string text of script>")		
Name	Type	Description										
<OriginalName>	String	Name of the script to be edited.										
<EditScriptArray>	Array	Structured array. Array("NAME:<string script name>",           "ScriptLang:=<string vbscript or javascript>",           "ScriptText:=<string text of script>")										
<b>Return Value</b>	None.											

<b>Python Syntax</b>	EditScript(<OriginalName>, <EditScriptArray>)
<b>Python Example</b>	<pre>oDefinitionManager.EditScript("MyScript",                                ["NAME:MyNewScript",                                 "ScriptLang:=", "vbscript",                                 "ScriptText:=", "MsgBox(\"HelloAgain\")"                                ])                            )</pre>

<b>VB Syntax</b>	EditScript <OriginalName> <EditScriptArray>
<b>VB Example</b>	<pre>oDefinitionManager.EditScript "MyScript" Array("NAME:MyNewScript",                                                "ScriptLang:=", "vbscript", "ScriptText:=", "MsgBox(""HelloAgain"")")</pre>

## ExportScript

*Use:* Export to Library in the script definition manager

*Command:* None

*Syntax:* ExportScript <ExportData>,<Library location>

*Return Value:* None

*Parameters:* <ExportData>

Array("NAME:<LibraryName>",<ScriptName>,<ScriptName>,...)

*VB Example:*

```
oProject.ExportComponent Array("NAME:mylib", "myscript"), "PersonalLib"
```

<b>Python Syntax</b>	ExportScript( <ExportData>,<Library location>)
<b>Python Example</b>	<pre>oProject.ExportComponent ([ "NAME:mylib", "myscript" ], "PersonalLib")</pre>

## ModifyLibraries

*Use:* Configure Libraries on the Tools menu

*Command:* None

*Syntax:* ModifyLibraries <DesignName>,Array(<ConfigLibArray>)

*Return Value:* None

*Parameters:* <DesignName>

*Type:* <string>

<ConfigLibArray>

Array("NAME:<LibraryType>,<ConfiguredLib>,<ConfiguredLib>,...),...

<ConfiguredLib> // blank to leave unchanged

<DefinitionType>

Array("<libraryname >","<libraryname>,...)

*VB Example:*

```
oDefinitionManager.ModifyLibraries "MyCircuit", _  
Array("NAME:PersonalLib"), _  
Array("NAME:UserLib"), _
```

```
Array("NAME:SystemLib", _  
"Symbols:=", Array( "Circuit Elements", "Symbols", _  
"ParamExtraElements\PE_Symbols", _  
"Vendor Elements\Nonlinear"))
```

<b>Python Syntax</b>	ModifyLibraries (<DesignName>,[<ConfigLibArray>])
<b>Python Example</b>	<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"
```

<b>Python Syntax</b>	RemoveScript (<ScriptName>,<IsProjectScript>, <LibraryName>,<LibraryLocation>)
<b>Python Example</b>	<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.

This page intentionally  
left blank.

# 30 - 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](#)

[CreateComponent](#)

[CreateEdgePort](#)

[CreateGroundPortComponent](#)

[CreateGroupSelected](#)

[CreateHole](#)

[CreateInterfaceGround](#)

[CreateInterfacePort](#)

[CreateInterfacePortComponent](#)

[CreateLine](#)

[CreateLineFromPolygon](#)[CreateLineVoid](#)[CreateMeasure](#)[CreateNPort](#)[CreateNetClass](#)[CreateObjectFromPolygon](#)[CreatePin](#)[CreatePinGroup](#)[CreatePinGroupPort](#)[CreatePolygon](#)[CreatePolygonVoid](#)[CreatePortInstancePorts](#)[CreatePortsOnComponents](#)[CreateRectangle](#)[CreateRectangleVoid](#)[Create Text](#)[CreateTrace](#)[CreateVia](#)[CutOutSubDesign](#)[DeactivateOpen](#)[DeactivateShort](#)

[DefeatureObjects](#)

[DelNetClass](#)

[Delete](#)

[DeleteNets](#)

[DeletePinGroup](#)

[Disconnect](#)

[Duplicate](#)

[DuplicateAcrossLyrs](#)

[Edit](#)

[Edit3DComponentDefinition](#)

[EraseMeasurements](#)

[Expand](#)

[ExpandWithPorts](#)

[ExportDXF](#)

[ExportGDSII](#)

[ExportGerber](#)

[ExportImage](#)

[ExportNCDrill](#)

[ExportToHFSS](#)

[ExportToQ3D](#)

[FilterObjectList](#)  
[FindObjectsByPoint](#)  
[FindObjectsByPolygon](#)  
[FlipHorizontal](#)  
[FlipVertical](#)  
[GeometryCheckAndAutofix](#)  
[GetAllLayerNames](#)  
[GetBBox](#)  
[GetCSObjects](#)  
[GetComlInstanceFromRefDes](#)  
[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: <ul style="list-style-type: none"><li>• &lt;Layername&gt;</li><li>• &lt;Type&gt;</li><li>• &lt;TopBottom&gt;</li><li>• &lt;Color&gt;</li><li>• &lt;Transparency&gt;</li><li>• &lt;Pattern&gt;</li></ul>

		<ul style="list-style-type: none"> <li>• &lt;VisFlag&gt;</li> <li>• &lt;Locked&gt;</li> <li>• &lt;Draw&gt;</li> </ul>
<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	<p>Sets visibility for all objects on the layer using the following additive code:</p> <ul style="list-style-type: none"> <li>• 1 makes primitives (rectangles, circles, polygons) visible.</li> <li>• 2 makes lines (also known as paths or traces) visible.</li> <li>• 4 makes pads (from padstacks, aka vias) visible.</li> <li>• 8 makes holes (or vias) visible.</li> <li>• 16 makes components visible.</li> <li>• 32 makes the mesh overlay visible if they are plotted.</li> <li>• 64 makes the mesh for the background material visible.</li> </ul> <p>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.</p>
<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.

<b>Return Value</b>	None
---------------------	------

<b>Python Syntax</b>	<pre>oEditor.AddLayer(     [         "NAME:layers",         "Name:=", "&lt;LayerName&gt;",         "Type:=", "&lt;Type&gt;",         "Top Bottom:=", "&lt;TopBottom&gt;",         "Color:=", &lt;Color&gt;,         "Transparency:=", &lt;Transparency&gt;,         "Pattern:=", &lt;Pattern&gt;,         "VisFlag:=", &lt;VisFlag&gt;,         "Locked:=", &lt;Locked&gt;,         "DrawOverride:=", &lt;Draw&gt;,     ])</pre>
<b>Python Example</b>	<pre>oEditor = oDesign.SetActiveEditor("Layout") oEditor.AddLayer(     [         "NAME:layers",     ])</pre>

```

    "Name:=", "signal",
    "Type:=", "assembly",
    "Top Bottom:=", "neither",
    "Color:=", 16711680,
    "Transparency:=", 0,
    "Pattern:=", 1,
    "VisFlag:=", 127,
    "Locked:=", False,
    "DrawOverride:=", 0,
)

```

<b>VB Syntax</b>	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>,)
<b>VB Example</b>	<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.

<b>UI Access</b>	Right-click an object in the History Tree and select <b>Properties</b> .		
<b>Parameters</b>	Name <code>&lt;propertyArgs&gt;</code>	Type Array	Description Structured array. The properties vary depending on the object. Due to the number of potential configurations, it is recommended that you generate this script using the UI's <b>Automation</b> tab.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	<code>ChangeProperty(&lt;propertyArgs&gt;)</code>
<b>Python Example</b>	<p><b>Example Changing the Position of a Box:</b></p> <pre>oEditor.ChangeProperty( [     "NAME:AllTabs",     [         "NAME:Geometry3DCmdTab",         [             "NAME:PropServers" ,             "Box1&gt;CreateBox:1"         ], </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  
            ]  
        ]  
    ]
```

	])
--	----

<b>VB Syntax</b>	ChangeProperty < <i>propertyArgs</i> >
<b>VB Example</b>	<p><b>Example Changing the Position of a Box:</b></p> <pre>oEditor.ChangeProperty Array("NAME:AllTabs",       Array("NAME:Geometry3DCmdTab",             Array("NAME:PropServers" ,                   "Box1&gt;CreateBox:1"             ),             Array("NAME:ChangedProps",                   Array("NAME:Position",                         "X:="      , "0.35in",                         "Y:="      , "0.55in",                         "Z:="      , "0in"                   )             )       ) )</pre> <p><b>Example Changing a Box's Material and Wireframe Display:</b></p> <pre>oEditor.ChangeProperty( Array("NAME:AllTabs",       Array("NAME:Geometry3DAttributeTab",             Array("NAME:PropServers",                   "Box1"             ),             Array("NAME:ChangedProps", </pre>

```
        Array("NAME:Material",
      "Value:=",  "\\"vacuum\\\""
    ),
    Array("NAME:Display Wireframe",
      "Value:=",  True
    )
  )
)
```

## 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>, // 1<sup>st</sup> object

<object\_name>, // 2<sup>nd</sup> 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)
```

## 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("<fooprint stackup layer>:<design stackup layer>", ...),  
"DrawLayers:=", Array("<fooprint 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.

## CreateEdgePort (Layout 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: 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)  
  
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)
```

## 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
```

```
UnmiterredJoin = 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.

<b>UI Access</b>	Right-click <b>Circuit Elements</b> > <b>Create Single Port Model...</b> or <b>Circuit Elements</b> > <b>Create Multi-Terminal Model...</b>								
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;NPortArray&gt;</td> <td>Array</td> <td> <p>Structured array.</p> <p>Array ("NAME:&lt;NPortName&gt;","Definition:=", &lt;string&gt;, &lt;AssignmentArray&gt;)</p> </td> </tr> </tbody> </table>			Name	Type	Description	<NPortArray>	Array	<p>Structured array.</p> <p>Array ("NAME:&lt;NPortName&gt;","Definition:=", &lt;string&gt;, &lt;AssignmentArray&gt;)</p>
Name	Type	Description							
<NPortArray>	Array	<p>Structured array.</p> <p>Array ("NAME:&lt;NPortName&gt;","Definition:=", &lt;string&gt;, &lt;AssignmentArray&gt;)</p>							
<b>Return Value</b>	None.								

<b>Python Syntax</b>	CreateNportCircuitElement (<NPortArray>)
<b>Python Example</b>	oModule.CreateNportCircuitElement (

```
[ "NAME:Nport1",
  "Definition:=", "Model",
  [ "NAME:Assignments",
    [ "NAME:Model",
      "Assign:=", "1"]
    ]
  ]
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"],
    ]
  ]
```

	]
--	---

<b>VB Syntax</b>	CreateNportCircuitElement <NPortArray>
<b>VB Example</b>	<pre> oModule.CreateNportCircuitElement _     Array("NAME:Nport1", _         "Definition:=", "Model", _         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"))) </pre>

## 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" ] )
```

## 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",
"n:=", "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>, "10:=",
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>:

double 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 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.

## 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>, // 1<sup>st</sup> component name  
<object\_name>, // 2<sup>nd</sup> 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.

<b>UI Access</b>	None.		
<b>Parameters</b>	Name <netName>	Type String	Description Name of unwanted net.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	DeleteNets ([<netName>, ...])
<b>Python Example</b>	<pre>oEditor.DeleteNets (     [</pre>

```
    "NET_1",
    "NET_2",
    "NET_3"
])
```

<b>VB Syntax</b>	DeleteNets Array (<netName>, ...)
<b>VB Example</b>	<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>, // 1<sup>st</sup> object

```
<object_name>, // 2nd object, if any  
...) // etc  
VB Example: oEditor.Disconnect 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.

<b>UI Access</b>	<b>Layout &gt; 3D Component Definitions.</b> Select component and click <b>Edit Definition</b> .						
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><i>&lt;Parameters&gt;</i></td> <td>Array</td> <td> <p>Structured array.</p> <pre>Array ("NAME:3D Components",       "Definitions:=" , [&lt;string component name&gt;],       "IsLocal:=" , &lt;boolean True for local definition; False to save to file&gt;,       "EditFilePath:=" , &lt;string path of new component file location&gt;)  If saving locally or leaving the file path unaltered, pass empty string for EditFilePath.</pre> </td> </tr> </tbody> </table>	Name	Type	Description	<i>&lt;Parameters&gt;</i>	Array	<p>Structured array.</p> <pre>Array ("NAME:3D Components",       "Definitions:=" , [&lt;string component name&gt;],       "IsLocal:=" , &lt;boolean True for local definition; False to save to file&gt;,       "EditFilePath:=" , &lt;string path of new component file location&gt;)  If saving locally or leaving the file path unaltered, pass empty string for EditFilePath.</pre>
Name	Type	Description					
<i>&lt;Parameters&gt;</i>	Array	<p>Structured array.</p> <pre>Array ("NAME:3D Components",       "Definitions:=" , [&lt;string component name&gt;],       "IsLocal:=" , &lt;boolean True for local definition; False to save to file&gt;,       "EditFilePath:=" , &lt;string path of new component file location&gt;)  If saving locally or leaving the file path unaltered, pass empty string for EditFilePath.</pre>					
<b>Return Value</b>	None.						
<b>Python Syntax</b>	<code>Edit3DComponentDefinition (&lt;Parameters&gt;)</code>						
<b>Python Example</b>	<pre> oEditor.Update3DComponentDefinitions(   ["NAME:3D Components",    "Definitions:=" , ["My_Component"],    "IsLocal:=" , True,    "EditFilePath:=" , ""   ) </pre>						
<b>VB Syntax</b>	<code>Edit3DComponentDefinition &lt;Parameters&gt;</code>						

<b>VB Example</b>	<pre><code>oEditor.Update3DComponentDefinitions Array("NAME:3D Components", _ "Definitions:=", Array("My_Component"), "IsLocal:=", True, "EditFilePath:=", "")</code></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: <ul style="list-style-type: none"><li>• "ROUND"</li><li>• "MITER"</li><li>• "CORNER"</li></ul>
	<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>"
<b>Return Value</b>	None.	

<b>Python Syntax</b>	Expand (<expansion>, <join>, <replace>, <items>)
<b>Python Example</b>	<code>oEditor.Expand ("2mm", "ROUND", True, ["NAME:elements", "circle_0"])</code>

<b>VB Syntax</b>	Expand <expansion>, <join>, <replace>, <items>
<b>VB Example</b>	<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.		
<b>Parameters</b>	Name	Type	Description
	<expansion>	String	Expansion amount, including unit. Value can be negative to shrink an object.
	<join>	String	Corner style. One of: <ul style="list-style-type: none"> <li>• "ROUND"</li> <li>• "MITER"</li> <li>• "CORNER"</li> </ul>
	<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>")
<b>Return Value</b>	None.		

<b>Python Syntax</b>	ExpandWithPorts (<expansion>, <join>, <replace>, <items>)
<b>Python Example</b>	oEditor.ExpandWithPorts("2mm", "ROUND", True, ["NAME:elements", "circle_0"])

<b>VB Syntax</b>	ExpandWithPorts <expansion>, <join>, <replace>, <items>
<b>VB Example</b>	oEditor.ExpandWithPorts "2mm", "ROUND", true, Array("NAME:elements", "poly_0")

## 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:/Anstranslator/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 <code>&lt;OptionParameters&gt;</code>	Type Array	Description <pre>Structured array.  Array("NAME:options",       "FileName:=", &lt;string, GDSII export file name.&gt;,       "NumVertices:=", &lt;integer, maximum num- ber of vertices allowed in a polygon (spe- cify zero if unlimited).&gt;,       "ArcTol:=", &lt;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.&gt;,       "LayerMap:=", Array(       "entry:=", Array(       "layer:=", &lt;string, name of a layer to export.&gt;,       "id:=", &lt;integer, GDSII layer ID to assign the exported layer.&gt;,       "include:=", &lt;boolean, if true, the layer is exported.))</pre>

<b>Return Value</b>	None.
---------------------	-------

<b>Python Syntax</b>	ExportGDSII(<OptionParameters>)
<b>Python Example</b>	<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,             "include:=", true]     ] ) </pre>

<b>VB Syntax</b>	ExportGDSII <OptionParameters>
------------------	--------------------------------

**VB Example**

```
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)))
```

## 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	<p>One of:</p> <ul style="list-style-type: none"><li>• "<b>Name</b>" to search by object name.</li><li>• "<b>Type</b>" to search by object type.</li></ul> <p>Valid &lt;value&gt; 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'</p> <ul style="list-style-type: none"><li>• "<b>Layer</b>" to search by layer.</li></ul> <p>An asterisk (*) in the &lt;value&gt; string matches all layers. Vias and pins match using their defined layer range. 'Multi' matches (non via/pin) multi-layer objects.</p>

		<ul style="list-style-type: none"> <li>• "Net" to search by net assignment.</li> </ul>
<value>	String	Specific query. Queries use a basic string match. For instance, an asterisk (*) matches anything. All matching is case insensitive.
<b>Return Value</b>	Filtered list of object names.	

<b>Python Syntax</b>	FilterObjectList (<key>, <value>)
<b>Python Example</b>	RFIN_pins = oLayout.FilterObjectList('Type', 'Pin', oLayout.FindObjects('Net', 'RFIN'))

<b>VB Syntax</b>	FilterObjectList <key>, <value>
<b>VB Example</b>	<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(<oPolygopn>, <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.Polyon().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")

## GetComplInstanceFromRefDes (Layout Editor)

*Use:* Informational.

*Command:* None.

*Syntax:* GetComplInstanceFromRefDes (<object\_name>)

*Return Value:* Returns IDispatch for ComplInstance.

*Parameters:* <object\_name> // string is refDes

*VB Example:* oEditor.GetCompInstanceFromRefDes ("A1")

## GetComponentInfo (Layout Editor)

*Use:* Informational.

*Command:* None.

*Syntax:* GetComponentInfo<complID>

*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

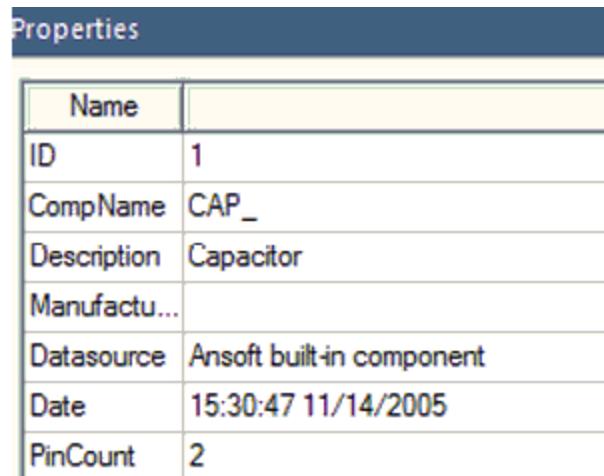
*Parameters:* The LayoutComp's ID  
CompInst@<ComponentName>;<CompInstID>

*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.



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  
connss = 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.GetPropertyValues("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)

*Use:* Draw > Geometry Healing

*Command:* Heal( [ "NAME:<category>", "Type:=", "<type>", "Tol:=", "<tolerance>", <args>] )

<category> can have these values: Snap, Feature, or Repair

<type> will have these values:

For category *Snap*: Point, Arc, Grid

For category *Feature*: Voids, FloatingBodies

For category *Repair*: Colinear, SelfIntersecting

For the category *Feature*, type *Voids* there is an additional optional argument: "AntiPads:=" true or false.

*Syntax:*

Heal( [ "NAME:<category>", "Type:=", "<type>", "Tol:=", "<tolerance>", <args>] ))

Heal( [ "NAME:<category>", "Selection:=", [ "<obj1>", "<obj2>", ... ], "Type:=", "<type>", "Tol:=", "<tolerance>", <args>] ))

*Return Value:* None

*VB Example:*

```
oProject = oDesktop.SetActiveProject("GeometryHealing")
oDesign = oProject.SetActiveDesign("TestDesign")
oEditor = oDesign.SetActiveEditor("Layout")
```

```
oEditor.Heal(
[
  "NAME:Snap",
  "Type:=" , "Point",
  "Tol:=" , "0.1mm"
])
oEditor.Heal(
[
  "NAME:Snap",
  "Type:=" , "Arc",
  "Tol:=" , "0.1mm"
])
oEditor.Heal(
[
  "NAME:Snap",
  "Type:=" , "Grid",
  "Tol:=" , "0.1mm"
])
oEditor.Heal(
[
```

```
"NAME:Feature",
"Type:=" , "Voids",
"AntiPads:=" , False,
"Tol:=" , "4mm2"
])

oEditor.Heal(
[

"NAME:Feature",
"Type:=" , "FloatingBodies",
"AntiPads:=" , False,
"Tol:=" , "4mm2"
)

oEditor.Heal(
[

"NAME:Repair",
"Type:=" , "Colinear",
"Tol:=" , "1e-007mm"
)

oEditor.Heal(
[

"NAME:Repair",
```

```
"Type:=" , "SelfIntersecting",
" Tol:=" , "1e-007mm"
])
```

## HighlightNet (Layout Editor)

Highlights or removes the highlight from all elements in a net.

<b>UI Access</b>	None.		
<b>Parameters</b>	Name	Type	Description
	<requiredPara-meterKeyword>	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.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	HighlightNet (["Name:Args", "Name:=", "<netName>", "Hi:=", <highlightFlag>, ...])
<b>Python Example</b>	<pre>oEditor.HighlightNet (     [</pre>

```

    "NAME:Args",
    "Name:=", "net_0",
    "Hi:=", True
)

```

<b>VB Syntax</b>	HighlightNet Array("Name:Args", "Name:=", "<netName>", "Hi:=", <highlightFlag>, ...)
<b>VB Example</b>	<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.								
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;Parameters&gt;</td> <td>Array</td> <td> <p>Structured array.</p> <pre> Array ("NAME:options",       "FileName:=" , &lt;string&gt;,       "Scale:=" , &lt;float&gt;,       "AutoDetectClosed:=" , &lt;boolean&gt;,       "SelfStitch:=" , &lt;boolean&gt;,       "DefeatureGeometry:=" , &lt;boolean&gt;, </pre> </td></tr> </tbody> </table>	Name	Type	Description	<Parameters>	Array	<p>Structured array.</p> <pre> Array ("NAME:options",       "FileName:=" , &lt;string&gt;,       "Scale:=" , &lt;float&gt;,       "AutoDetectClosed:=" , &lt;boolean&gt;,       "SelfStitch:=" , &lt;boolean&gt;,       "DefeatureGeometry:=" , &lt;boolean&gt;, </pre>		
Name	Type	Description							
<Parameters>	Array	<p>Structured array.</p> <pre> Array ("NAME:options",       "FileName:=" , &lt;string&gt;,       "Scale:=" , &lt;float&gt;,       "AutoDetectClosed:=" , &lt;boolean&gt;,       "SelfStitch:=" , &lt;boolean&gt;,       "DefeatureGeometry:=" , &lt;boolean&gt;, </pre>							

			<pre>         "DefeatureDistance:=" , &lt;integer&gt;,         "RoundCoordinates:=" , &lt;boolean&gt;,         "RoundNumDigits:=" , &lt;integer&gt;,         "WritePolyWithWidthAsFilledPoly:=" , &lt;boolean&gt;,         "ImportMethod:=" , &lt;integer&gt;,         "2DSheetBodies:=" , &lt;boolean&gt;,         &lt;layersArray&gt;     </pre>
	<i>&lt;layersArray&gt;</i>	Array	<p>Structured array.</p> <pre>         Array ("NAME:LayerInfo",                &lt;layer&gt;, &lt;layer&gt;, &lt;layer&gt;, ... )     </pre>
	<i>&lt;layer&gt;</i>	Array	<p>Structured array.</p> <pre>         Array ("NAME:&lt;layerName&gt;",                "source:=" , &lt;string&gt;,                "display_source:=" , &lt;string&gt;,                "import:=" , &lt;boolean&gt;,                "dest:=" , &lt;string&gt;,                "dest_selected:=" , &lt;boolean&gt;,                "layer_type:=" , &lt;string&gt;)     </pre>
<b>Return Value</b>	None.		

<b>Python Syntax</b>	<code>ImportDXF(&lt;Parameters&gt;)</code>
<b>Python Example</b>	<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,                "dest:=" , "0",                "dest_selected:=" , False,                ...            ]          ]        ]      ] ) </pre>

```
        "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"
    ]
]
])
```

<b>VB Syntax</b>	ImportDXF <Parameters>
<b>VB Example</b>	<pre> oEditor.ImportDXF Array("NAME:options", "FileName:=", _ "C:/Users/jdoe/Desktop/export.dxf", "Scale:=", 0.001, "AutoDetectClosed:=", _ true, "SelfStitch:=", true, "DefeatureGeometry:=", false, "DefeatureDistance:=", _ 0, "RoundCoordinates:=", false, "RoundNumDigits:=", 4, "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.

<b>UI Access</b>	<b>File &gt; Import &gt; GDSII.</b>								
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;gdsiiFileName&gt;</td> <td>String</td> <td>Full path of GDSII file.</td> </tr> </tbody> </table>			Name	Type	Description	<gdsiiFileName>	String	Full path of GDSII file.
Name	Type	Description							
<gdsiiFileName>	String	Full path of GDSII file.							

	<code>&lt;outputPathName&gt;</code>	String	Full path of EDB file to create during import.
	<code>&lt;controlFileName&gt;</code>	String	Optional. Full path of XML control file. Pass empty string if none.
	<code>&lt;propertyFileName&gt;</code>	String	Optional. Full path to property mapping file. Pass empty string if none.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	<code>ImportGDSII(&lt;gdsiiFileName&gt;, &lt;outputPathName&gt;, &lt;controlFileName&gt;, &lt;propertyFileName&gt;)</code>
<b>Python Example</b>	<pre>oDesktop.RestoreWindow()  Set oTool = oDesktop.GetTool('ImportExport')  oTool.ImportGDSII('C:/Files/test.gds', 'C:/Files/test.aedb.edb', 'C:/Files/test.xml', 'C:/Files/test.txt')</pre>

<b>VB Syntax</b>	<code>ImportGDSII &lt;gdsiiFileName&gt;, &lt;outputPathName&gt;, &lt;controlFileName&gt;, &lt;propertyFileName&gt;</code>
<b>VB Example</b>	<pre>Set oTool = oDesktop.GetTool("ImportExport")  oTool.ImportGDSII "C:/Files/test.gds", "C:/Files/test.aedb.edb", "C:/Files/test.xml", "C:/Files/test.txt"</pre>

## 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.

<b>UI Access</b>	<ul style="list-style-type: none"><li>Layout Editor</li><li>Schematic Editor</li><li>Select a Nexxim solution in a 3D Layout design, right click, and choose <b>Push Excitations</b>.</li></ul>									
<b>Parameters</b>	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>&lt;Reference Designation&gt;</td><td>String</td><td>"&lt;refdes&gt;"  This argument is empty when excitations are pushed from a Nexxim solution.</td></tr><tr><td>&lt;PushExcitations Parameters&gt;</td><td>Array</td><td>This parameter changes depending on whether the excitations comes from a transient or linear frequency solution. The keyword</td></tr></tbody></table>	Name	Type	Description	<Reference Designation>	String	"<refdes>"  This argument is empty when excitations are pushed from a Nexxim solution.	<PushExcitations Parameters>	Array	This parameter changes depending on whether the excitations comes from a transient or linear frequency solution. The keyword
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:

```
Array ("NAME:options",
       "CalcThevenin =", <true|false>,
       "Sol:=", "<solution name>")
```

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:

```
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>")
```

winType can have the following values:

- Rectangular

			<ul style="list-style-type: none"> <li>• Bartlett</li> <li>• Blackman</li> <li>• Hamming</li> <li>• Hanning</li> <li>• Kaiser</li> <li>• Welch</li> <li>• Weber</li> <li>• Lanzcos</li> </ul>
<b>Return Value</b>	None		

**Note:**

For more information about the component name, schematic IDs, and formats, see the section [Editor Scripting IDs](#).

<b>VB Syntax</b>	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>")
<b>VB Example</b>	<p>For a transient solution:</p> <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")

```

## 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 28-207.

## 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 <b>Remove Ports</b> .		
<b>Parameters</b>	Name	Type	Description
	< requiredKeyword>	String	Required parameter that must proceed all element names with the value "NAME:Nets".
<b>Return Value</b>	None.		

<b>Python Syntax</b>	RemovePortsFromNet(["NAME:Nets", <netName>, ...])
<b>Python Example</b>	<pre> oEditor.RemovePortsFromNet ( [     "NAME:Nets",     "NET_1",     "NET_2",     "NET_3" ]) </pre>

<b>VB Syntax</b>	RemovePortsFromNet Array("NAME:Nets", <netName>, ...)
<b>VB Example</b>	<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.

<b>UI Access</b>	Right click and choose <b>Port &gt; Remove Ports From Component</b> .		
<b>Parameters</b>	Name	Type	Description
	< requiredKeyword>	String	Required parameter that must proceed all element names with the value "NAME:elements".
<b>Return Value</b>	None.		

<b>Python Syntax</b>	RemovePortsOnComponents (["NAME:elements", <elementName>, ...])
<b>Python Example</b>	<pre>oEditor.RemovePortsOnComponents (     [         "NAME:elements",         "0"     ])</pre>

<b>VB Syntax</b>	RemovePortsOnComponents Array("NAME:elements", " <element-name>", ...)
<b>VB Example</b>	<pre>oEditor.RemovePortsOnComponents Array("NAME:elements", "0")</pre>

## 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.

<b>UI Access</b>	On the <b>Nets</b> tab of the <b>Nets</b> window, highlight and right click on one or more nets. Click <b>Hide</b> or any of the <b>Show</b> options.		
<b>Parameters</b>	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.

	<code>&lt;netName&gt;</code>	String	Name of the net
	<code>&lt;requiredFlagKeyword&gt;</code>	String	Required fourth parameter which must have the value "Vis:=". Must be the third parameter for each net.
	<code>&lt;visibilityFlag&gt;</code>	Boolean	True if the net should be visible. False if it should be hidden.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	<code>SetNetVisible (["Name:Args", "Name:=", "&lt;netName&gt;", "Vis:=", &lt;highlightFlag&gt;, ...])</code>
<b>Python Example</b>	<pre> oEditor.SetNetVisible( [     "NAME:Args",     "Name:=", "Net_1",     "Vis:=", True ]) oEditor.SetNetVisible( [     "NAME:Args",     "Name:=", "&lt;NO-NET&gt;",     "Vis:=", False,     "Name:=", "Ground",     "Vis:=", False,     "Name:=", "Chassis", ] ) </pre>

	"Vis:=", True ])
--	---------------------

<b>VB Syntax</b>	SetNetVisible Array ("Name:Args", "Name:=", "<netName>", "Vis:=", <visibilityFlag>, ...)
<b>VB Example</b>	<pre>oEditor.SetNetVisible Array("NAME:Args", "Name:=", "NET_1", "Vis:=", true) oEditor.SetNetVisible Array("NAME:Args", "Name:=", "&lt;NO-NET&gt;", "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>, // 1<sup>st</sup> primitive name

<object\_name>, // 2<sup>nd</sup> 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
```

<b>Python Syntax</b>	AlignHorizontal (["NAME:Selections", "Selections:=", [<selections>]], 0)
<b>Python Example</b>	<pre> oSymbolEditor = oProject.SetActiveDefinitionEditor("SymbolEditor", _ "Simplorer Elements\basic Elements\Circuit\PassiveElements:R")  oSymbolEditor.AlignHorizontal (["NAME:Selections", _</pre>

```
    "Selections:=", [ "SchObj@11" ]], 0)
```

## AlignVertical (Symbol Editor)

Align items vertically

*Syntax:* AlignVertical(

```
Array("NAME:Selections", _  
"Page:=", page number, _ // [Opt=1] Page number  
"Selections:=", IDs to modify)),  
Array("NAME:AlignParameters", _  
"Disconnect:=", bool _ // [Opt=0] Should wires disconnect  
"Rubberband:=", bool) _ // [Opt=1] Should wires staircase  
// Note: Alignment occurs relative  
to the first item in ids
```

<b>Python Syntax</b>	AlignVertical (["NAME:Selections", "Selections:=", [<selections>]], 0)
<b>Python Example</b>	<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:** <stri<Object>

<string> // object to bring to the front

**Example:**

```
oDefinitionEditor.BringToFront Array("NAME:Selections", "Selections:=", Array( "SchObj@10"))
```

**ChangeProperty (Symbol Editor)**

**Use:** Changes to properties are scripted using the **ChangeProperty** command. This command can be executed by the **oEditor** to change editor properties, by the **oDesign** to change design level properties, and by the **oProject** to change project level properties. The command can be used to create, edit, and/or remove properties. In Circuit, only Variable and Separator properties can be deleted.

**Command:** None

**Syntax:** ChangeProperty Array("Name:AllTabs", <PropTabArray>, <PropTabArray>, ...)

**Return Value:** None

**Parameters:** <PropTabArray>

```
Array("Name:<PropTab>",  
<PropServersArray>,  
<NewPropsArray>,  
<ChangedPropsArray>,  
<DeletedPropsArray>)
```

```
<PropServersArray>
  Array("Name:PropServers", <PropServer>,
        <PropServer>, ...)

<NewPropsArray>
  Array("Name>NewProps", <PropdataArray>,
        <PropdataArray>,...)

<ChangedPropsArray>
  Array("Name:ChangedProps",<PropdataArray>,
        <PropdataArray>, ...)

<DeletedPropsArray>
  Array("Name:DeletedProps", <PropName>,
        <PropName>, ...)
  OR (for PropDisplay deletions only)
  Array("Name:DeletedProps",<PropdataArray>,
        <PropdataArray>, ...)
```

```
<PropdataArray>
  Array("NAME:<PropName>",
        "PropType:=", <PropType>,
        "NewName:=", <string>,
        "Description:=", <string>,
        "Callback:=", <string>,
        "NewRowPosition:=", <int>,
        "ReadOnly:=", <bool>,
        "Hidden:=", <bool>,
        <PropTypeSpecificArgs>)OR (for PropDisplays only)
  Array("Name:<PropName>",<PropDisplayData>

<PropDisplayData>
  for adding, changing, deleting PropDisplays
  <PropDisplayAttributes>
    for changing PropDisplays only
    <PropDisplayNewAttributes>

<PropDisplayAttributes>
  Layer & Location only used for PropDisplays in layout
```

For adding PropDisplays, this will add a single PropDisplay with attributes as shown; if an attribute is missing, a default value will be assigned. Adding PropDisplay to schematic with attributes that are identical to one already existing there will fail without an error message.

For deleting PropDisplays, these attributes are used to identify an existing PropDisplay to delete. If there doesn't exist a PropDisplay that matches the given attributes, then nothing will be deleted. If multiple PropDisplays match the given attributes, then all of them will be deleted. If an attribute is missing, then all PropDisplays match that missing attribute. For example, if Layer is missing, then PropDisplays on all layers that match the remaining given attributes will be deleted.

For changing PropDisplays, these attributes are used to identify an existing PropDisplay to change. If no PropDisplay matching the attributes is found, no changes will be made. If multiple PropDisplays match the attributes, all of them will be changed. If an attribute is missing, it matches all PropDisplays. For example, to change the format of PropDisplays that are on the bottom, but have any layer, style or format to show the name only, this command should have Location set to "Bottom" and all other attributes omitted.

```
"Format:=", <PropDisplayType>,  
"Location:=", <PropDisplayLocation>,  
"Layer:=", <string>,  
"Style:=", <string>  
  
<PropDisplayNewAttributes>
```

NewLayer & NewLocation only used for PropDisplays in layout

For changing PropDisplays, these attributes are used to identify which attributes to change and what the new value is. If the attribute should not be changed, the corresponding entry should be omitted.

```
"NewName:=", <string>,  
"NewFormat:=", <PropDisplayType>,
```

```
"NewLocation:=", <PropDisplayLocation>,
```

```
"NewLayer:=", <string>,
```

```
"NewStyle:=", <string>
```

```
<PropDisplayType>
```

Type: string

Identifies the format of PropDisplay.

```
"Name"
```

```
"Value"
```

```
"NameAndValue"
```

```
"EvaluatedValue"
```

```
"NameAndEvaluatedValue"
```

```
<PropDisplayLocation>
```

Type: string

Identifies where PropDisplay is located with respect to object

```
"Left"
```

```
"Top"
```

```
"Right"
```

```
"Bottom"
```

```
"Custom"
```

<PropType>

Type: string

Identifies the type of property when a new property is added. In Circuit, only separator properties and variable properties can be added.

"SeparatorProp"

"VariableProp"

"TextProp"

"NumberProp"

"ValueProp"

"CheckboxProp"

"MenuProp"

"PointProp"

"VPointProp"

"ButtonProp"

NewName

Specify the new name of a property if the property's name is being edited. In Circuit, the name can only be changed for separators and variables.

Description

Specify a description of the property. In Circuit, the description can only be changed for separators and variables.

#### Callback

Specify the name of the script callback to be run when the property value is changed.

#### NewRowPosition

Used to reorder rows in the **Property** dialog box. In Circuit, this only applies to the **Project>Project Variables** panel and the **Designer>DesignProperties** panel. Specify the new zero-based row index of the variable or separator.

#### ReadOnly

Used to mark a property as "read only" so it can not be modified. In Circuit, this flag can only be set for variables and separators.

#### Hidden

Used to hide a property so it can not be viewed outside of the **Property** dialog box. In Circuit, this flag can only be set for variables and separators.

#### <PropTypeSpecificArgs>

**SeparatorProp:** no arguments

TextProp: "Value:=", <string>

NumberProp: "Value:=", <double>

ValueProp: "Value:=", <value>

CheckboxProp: "Value:=", <bool>

MenuProp: "Value:=", <string>  
PointProp "X:=", <double>, "Y:=", <double>  
VPointProp: "X:=", <value>, "Y:=", <value>  
Material Button: "Material:=", <string>  
Color Button: "R:=", <int>, "G:=", <int>, "B:=", <int>  
Transparency Button: "Value:=", <double>

<PropTypeSpecificArgs> for MenuProps  
Syntax for NewProps array: "AllChoices:=",  
<"choice1,choice2,..."> or <Array("choice1" "choice2", ... )>,  
"Value:=", <string>  
Syntax for ChangedProps array: "Value:=", <string>

<PropTypeSpecificArgs> for VariableProps  
Syntax:  
**"Value:=", <value>, <OptimizationFlagsArray>,**  
**<TuningFlagsArray>, <SensitivityFlagsArray>,**  
**<StatisticsFlagsArray>**

Parameters:

```
<OptimizationFlagsArray>
  Array("NAME:Optimization",
    "Included:=", <bool>,
    "Min:=", <value>,
    "Max:=", <value>)
```

```
<TuningFlagsArray>
  Array("NAME:Tuning",
    "Included:=", <bool>,
    "Step:=", <value>,
    "Min:=", <value>,
    "Max:=", <value>)
```

```
<SensitivityFlagsArray>
  Array("NAME:Sensitivity",
    "Included:=", <bool>,
    "Min:=", <value>,
    "Max:=", <value>,
    "IDisp:=", <value> )
```

```
<StatisticsFlagsArray>
  Array("NAME:Statistical",
```

```
"Included:=", <bool>,  
"Dist:=", <Distribution>,  
"StdD:=", <value>,  
"Min:=", <value>,  
"Max:=", <value>,  
"Tol:=", <string>)
```

<Distribution>

Type: string

Value should be "**Gaussian**" or "**Uniform**"

StdD

Standard deviation.

Min

Low cut-off for the distribution.

Max

High cut-off for the distribution.

**Tol**

Tolerance for uniform distributions. Format is "<int>%".

Example: "20%".

*VB Example:*

Adding a new project level variable "\$width":

```
oProject.ChangeProperty Array("NAME:AllTabs",_
Array("NAME:ProjectVariableTab",_
Array("NAME:PropServers", "ProjectVariables"),_
Array("NAME:NewProps",_
Array("NAME:$width",_
"PropType:=", "VariableProp",_
"value:=", "3mm",_
"description:=", "my new variable")))
```

*VB Example:* Deleting the design level variable "height":

```
oDesign.ChangeProperty Array("NAME:AllTabs",_
Array("NAME:LocalVariableTab",_
Array("NAME:PropServers", "DefinitionParameters"),_
Array("NAME:DeletedProps", "height")))
```

Changing a property's value. If the following command were executed, then the value of the property "XSize" of the PropServer

"Box1>CreateBox:1" on the "Geometry3DCmdTab" tab would be changed. (oEditor is the Geometry3D editor in Circuit.)

```
oEditor.ChangeProperty Array("NAME:AllTabs", _  
    Array("NAME:Geometry3DCmdTab", _  
        Array("NAME:PropServers", "Box1>CreateBox:1"), _  
        Array("NAME:ChangedProps", _  
            Array("NAME:XSize", "Value:=", "1.4mil"))))
```

*VB Example:* Changing a property's value. If the following command were executed, then the values of Callback and L on the PassedParameterTab would be changed.

```
oEditor.ChangeProperty Array("NAME:AllTabs", _  
    Array("NAME:PassedParameterTab", _  
        Array("NAME:PropServers", "CHOKE2"), _  
        Array("NAME:ChangedProps", _  
            Array("NAME:L", "Callback:=", "ac", "OverridingDef:=", true), _  
            Array("NAME:L", "Value:=", "1nH"))))
```

*VB Example:* Changing the Company Name, Design Name, the background color, and the Axis scaling in a Report.

```
Set oProject = oDesktop.SetActiveProject("wgcombiner")
Set oDesign = oProject.SetActiveDesign("CircuitDesign2")
Set oModule = oDesign.GetModule("ReportSetup")
oModule.ChangeProperty Array("NAME:AllTabs", Array("NAME:Header", _ Array("NAME:PropServers", "XY Plot1:Header"), _
Array("NAME:ChangedProps", Array("NAME:Company Name", _ "Value:=", "My Company"))))

oModule.ChangeProperty Array("NAME:AllTabs", Array("NAME:Header", _ Array("NAME:PropServers", "XY Plot1:Header"), _
Array("NAME:ChangedProps", Array("NAME:Design Name", _ "Value:=", "WG Combiner"))))

oModule.ChangeProperty Array("NAME:AllTabs", Array("NAME:General", _ Array("NAME:PropServers", "XY Plot1:General"), _
Array("NAME:ChangedProps", Array("NAME:Back Color", _ "R:=", 128, "G:=", 255, "B:=", 255)))))

oModule.ChangeProperty Array("NAME:AllTabs", Array("NAME:Axis", _ Array("NAME:PropServers", "XY Plot1:AxisX"), _
Array("NAME:ChangedProps", Array("NAME:Axis Scaling", _ "Value:=", "Log"))))
```

*VB Example:* Changing a property's value. Note that the AllChoices parameter is only used when the MenuProp is being added. Also note that either a string of choices separated by commas or an Array("choice1", "choice2", "choice3") works for the AllChoices parameter.

```
Set oEditor = oDesign.SetActiveEditor("SchematicEditor")
oEditor.ChangeProperty Array("NAME:AllTabs", Array("NAME:PassedParameterTab", Array
("NAME:PropServers", _
"CompInst@CAP_2"), Array("NAME>NewProps", Array("NAME:xxxx", "PropType:=", _
"MenuProp", "AllChoices:=", Array("aa", "bb", "cc", "dd"), "Value:=", "bb"))))
```

## **CloseEditor (Symbol Editor)**

Closes the specified editor.

*Command:* None

*Syntax:* CloseEditor (VARIANT ptDelta)

*Return Value:* None

<b>Python Syntax</b>	CloseEditor()
<b>Python Example</b>	oDefinitionEditor.CloseEditor()

## **Copy (Symbol Editor)**

Copy items for pasting.

*Command:* Edit>Copy

*Syntax:* Copy Array("NAME:Selections", "Selections:=", Array(<selections>))

*Return Value:* None.

*Parameters:* <selections>

Type: Comma-separated list of strings

Identifiers for each selected item to be copied, in the form SchObj@<id>, where id is the value of the SchematicID property for each element.

*Example:* Dim oSymbolEditor

```
Set oSymbolEditor = oProject.SetActiveDefinitionEditor("SymbolEditor", _  
"Simplorer Elements\basic Elements\Circuit\Passive Elements:R")  
oSymbolEditor .Copy Array("NAME:Selections", "Selections:=", Array("SchObj@11", "SchObj@34"))
```

## CreateArc (Symbol Editor)

Draws an arc in the symbol editor

*Command:* None

*Syntax:* CreateArc

*Return Value:* None

*VB Example:*

```
oDefinitionEditor.CreateArc Array("NAME:ArcData", _  
"X:=", -0.004318, "Y:=", -0.00127, _  
"Radius:=", 0.00297299377732279, _  
"StartAng:=", 1.9195673303788, _  
"EndAng:=", 3.32144615338227, _  
"Id:=", 10), _
```

```
Array("NAME:Attributes", "Page:=", 1)
```

Python Syntax	CreateArc()
<b>Python Example</b>	<pre>oDefinitionEditor.CreateArc ([ "NAME:ArcData",   "X:=", -0.004318, "Y:=", -0.00127, _   "Radius:=", 0.00297299377732279, _   "StartAng:=", 1.9195673303788, _   "EndAng:=", 3.32144615338227, _   "Id:=", 10], _   ["NAME:Attributes", "Page:=", 1])</pre>

## CreateCircle (Symbol Editor)

Draws a circle in the symbol editor

*Command:* None

*Syntax:* CreateCircle

*Return Value:* None

*VB Example:*

```
oDefinitionEditor.CreateCircle Array("NAME:CircleData", _
```

```
"X:=", -0.004572, "Y:=", -0.000508, _
"Radius:=", 0.001778, "Id:=", 12), _
Array("NAME:Attributes", "Page:=", 1)
```

<b>Python Syntax</b>	CreateCircle()
<b>Python Example</b>	<pre>oDefinitionEditor.CreateCircle ( [ "NAME:CircleData", _   "X:=", -0.004572, "Y:=", -0.000508, _   "Radius:=", 0.001778, "Id:=", 12], _   [ "NAME:Attributes", "Page:=", 1])</pre>

## CreateImage (Symbol Editor)

Create picture from a BMP or GIF file.

**Command:**Draw>Image

**Syntax:** CreateImage(

**Return Value:** None.

**Parameters:** Array ("NAME:ImageData",
 "X1:", double, // the image rectangle left X value, in meters
 "Y1:", double, // the image rectangle upper Y value, in meters
 "X2:", double, // the image rectangle right X value, in meters
 "Y2:", double, // the image rectangle lower Y value, in meters
 "File:", string, // the name of the BMP or GIF file

```
"DisplayBorder:=", bool, // true or false, whether to display the image border  
"LineWidth:=", double, // the width of the image border, in meters  
"Color:=", int, // the RBG value of the image border color  
"Id:=", int), // Id for this item  
Array("NAME:Attributes",  
"Page:=", int) // [Opt=1] The page number (one-based)
```

*Example:* oDefinitionEditor.CreateImage Array("NAME:ImageData", "X1:=", 0.00229684174875881,  
"Y1:=", -0.00100608189091032, "X2:=", 0.00890589813779289, "Y2:=", -0.00432967005619977,  
"File:=", "C:/Documents and Settings/user/Desktop/logo.gif", "DisplayBorder:=", false,  
"LineWidth:=", 0, "Color:=", 0, "Id:=", 22), Array("NAME:Attributes", "Page:=", 1)

## CreateLine (Symbol Editor)

Draws a line in the symbol editor

*Command:* None

*Syntax:* CreateLine

*VB Example:*

```
oDefinitionEditor.CreateLine Array("NAME:LineData", _Return Value: None  
"Points:=", Array( "(0.001778, -0.001270)", _  
"(0.004572, 0.002794)", "(0.003048, 0.003556") ), _  
"Id:=", 14), Array("NAME:Attributes", "Page:=", 1)
```

<b>Python Syntax</b>	CreateLine()
<b>Python Example</b>	<pre> oDefinitionEditor.CreateLine ([ "NAME:LineData", _Return Value: None "Points:=", [ "(0.001778, -0.001270)",_ "(0.004572, 0.002794)", "(0.003048, 0.003556)" ],_ "Id:=", 14], [ "NAME:Attributes", "Page:=", 1]) </pre>

## CreatePin (Symbol Editor)

Draws a pin in the symbol editor

*Command:* None

*Syntax:* CreatePin

*Return Value:* None

*VB Example:*

```

oDefinitionEditor.CreatePin Array("NAME:PinData", _
"Name:=", "newpin3"), Array("NAME:PinParams", _
"X:=", -0.00762, "Y:=", 0.00254, "Angle:=", 0, "Flip:=", false)

```

<b>Python Syntax</b>	CreatePin()
<b>Python Example</b>	<pre> oDefinitionEditor.CreatePin ([ "NAME:PinData", _  "Name:=", "newpin3"], [ "NAME:PinParams", _  "X:=", -0.00762, "Y:=", 0.00254, "Angle:=", _  </pre>

```
    0, "Flip:=", false])
```

## CreatePolygon (Symbol Editor)

Draws a polygon in the symbol editor

*Command:* None

*Syntax:* CreatePolygon

*Return Value:* None

*VB Example:*

```
oDefinitionEditor.CreatePolygon Array("NAME:PolygonData", _  
"Points:=", Array( "(0.004826, -0.000508)", "(0.003048, -0.003048)", _  
"(0.006858, -0.003302)", "Id:=",16), _  
Array("NAME:Attributes", "Page:=", 1)
```

<b>Python Syntax</b>	CreatePolygon()
<b>Python Example</b>	<pre>oDefinitionEditor.CreatePolygon ([ "NAME:PolygonData", _ "X1:=", -0.001524, "Y1:=", 0.000508, _ "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)
```

<b>Python Syntax</b>	CreateRectangle()
<b>Python Example</b>	<pre>oDefinitionEditor.CreateRectangle ([ "NAME:RectData", _ "X1:=", -0.001524, "Y1:=", 0.000508, _ "X2:=", 0.002286, "Y2:=", -0.002032, "Id:=", 18], _ [ "NAME:Attributes", "Page:=", 1])</pre>

## CreateText (Symbol Editor)

Draws text in the symbol editor

*Command:* None

*Syntax:* CreateText

*Return Value:* None

*VB Example:*

```
oDefinitionEditor.CreateText Array("NAME:TextData", _
```

```
"X:=", 0.001524, "Y:=", 0.00127, _  
"Text:=","My text", "Id:=", 20), Array("NAME:Attributes", "Page:=", 1)
```

<b>Python Syntax</b>	CreateText()
<b>Python Example</b>	<pre>oDefinitionEditor.CreateText( [ "NAME:TextData","X:=", 0.001524, "Y:=", 0.00127, "Text:=","My text", "Id:=", 20], [ "NAME:Attributes", "Page:=", 1])</pre>

## Cut (Symbol Editor)

Cut page selection.

*Command:* None

*Syntax:* Cut(PageSize, Selections)

*Return Value:* None

*VB Example:*

```
oDefinitionEditor.CreateText Array("NAME:Selections", _  
"Page:=", page number, _ // [Opt=1] Page number  
"Selections:=", IDs to modify) 1)
```

<b>Python Syntax</b>	Cut(["NAME:Selections", "Selections:=",[<selections>]])
<b>Python Example</b>	<pre>oSymbolEditor = oProject.SetActiveDefinitionEditor("SymbolEditor", _</pre>

```
"Simplorer Elements\basic Elements\Circuit\PassiveElements:R")
oSymbolEditor.Cut ([ "NAME:Selections",
"Selections:=", [ "SchObj@11", "SchObj@3" ] ])
```

## 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:=", 0xffffffff, // 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	<code>oDefinitionEditor.Pan([-0.00922653869663931, 0.00686839904222147])</code>

VB Syntax	Pan <argSelections>
VB Example	<code>oDefinitionEditor.Pan Array(-0.00922653869663931, 0.00686839904222147)</code>

## Paste (Symbol Editor)

To paste copied objects to a specified X and Y location.

**Command:****Edit>Paste**

**Syntax:** Paste Array("NAME:Attributes", "X:=", <xval>, "Y:=", <yval>)

**Return Value:** None.

*Parameters:* <xval>

Type: Real number

The X placement value.

<yval>

Type: Real number

The Y placement value.

*Example:* Dim oSymbolEditor

```
Set oSymbolEditor = oProject.SetActiveDefinitionEditor("SymbolEditor", _
    "Simplorer Elements\basic Elements\Circuit\Passive Elements:R")
oSymbolEditor.Paste Array("NAME:Attributes", "X:=", -0.00254, "Y:=", 0.00254)
```

## Redo (Symbol Editor)

Redo the last operation

*Command:* Click **Edit>Redo**

*Syntax:* Redo

*Return Value:* None

*Parameters:* None

*VB Example:* oDesign.Redo

## RemoveLevels (Symbol Editor)

Removes one or more graphics levels from a symbol.

<b>UI Access</b>	<b>Tools &gt; Edit Libraries &gt; Symbols.</b> From Symbol editor, click <b>View &gt; Levels &gt; Remove Level.</b>		
<b>Parameters</b>	Name </levels>	Type Array of integers	Description The levels to be deleted.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	RemoveLevels (</levels>)
<b>Python Example</b>	<code>oSymbolEditor.RemoveLevels ([3, 4])</code>

<b>VB Syntax</b>	RemoveLevels </levels>
<b>VB Example</b>	<pre>Dim oSymbolEditor  Set oSymbolEditor = oProject.SetActiveDefinitionEditor("SymbolEditor", -     "Simplorer Elements\basic Elements\Circuit\Passive     Elements:R")  oldLevel = oSymbolEditor.RemoveLevels(Array(3, 4))</pre>

## RemovePort (Symbol Editor)

Selected pins are changed to vias. Selected edge terminal ports are removed.

*Syntax:* RemovePort <NAME:elements>, <object\_name> ... // objects to be removed

*Return Value:* None.

*Parameters:* <object\_name>

*Type:* <String>

*VB Example:* oEditor.RemovePort Array("NAME:elements", "via\_195", "Port1")

## Rotate (Symbol Editor)

To rotate the specified objects about their common center.

**Command:****Draw>Rotate**

**Syntax:** Rotate Array("NAME:Selections", "Selections:=", Array(<selections>)), Array("NAME:FlipParameters", <angle>)

**Return Value:** None.

*Parameters:* <selections>

Type: Comma-separated list of strings

Identifiers for each selected item to be rotated, in the form SchObj@<id>, where id is the value of the SchematicID property for each element.

<angle>

Type: A comma-separated string and real number

Either "Angle:=", <rad>, where <rad> is the angle in radians, or "Degrees:=", <deg>, where <deg> is the angle in degrees.

*Example:* Dim oSymbolEditor

```
Set oSymbolEditor = oProject.SetActiveDefinitionEditor("SymbolEditor",
    "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()
```

<b>Python Syntax</b>	SelectAll()
<b>Python Example</b>	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"))

<b>Python Syntax</b>	SendToBack (["NAME:Selections", "Selections:=",[<Object>, <Object>, ...]])
<b>Python Example</b>	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", _
"Simpler 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.

<b>UI Access</b>	<b>Tools &gt; Edit Libraries &gt; Symbols.</b> From Symbol editor, click <b>View &gt; Levels</b> . Select the <b>Default</b> check box for desired levels.								
<b>Parameters</b>	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>&lt;/levels&gt;</td><td>Array of integers</td><td>The levels to be selected.</td></tr></tbody></table>			Name	Type	Description	</levels>	Array of integers	The levels to be selected.
Name	Type	Description							
</levels>	Array of integers	The levels to be selected.							
<b>Return Value</b>	None.								

<b>Python Syntax</b>	SetInitialLevels (</levels>)
<b>Python Example</b>	<pre>oSymbolEditor.SetInitialLevels([3, 4])</pre>

<b>VB Syntax</b>	SetInitialLevels </levels>
<b>VB Example</b>	<pre>Dim oSymbolEditor  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.

<b>UI Access</b>	<b>Tools &gt; Edit Libraries &gt; Symbols.</b> From Symbol editor, click <b>View &gt; Levels</b> . Select the <b>Visible</b> check box for the desired levels.								
<b>Parameters</b>	<table border="1"> <tr> <td>Name</td> <td>Type</td> <td>Description</td> </tr> <tr> <td>&lt;/levels&gt;</td> <td>Array of integers</td> <td>The levels to be made visible.</td> </tr> </table>			Name	Type	Description	</levels>	Array of integers	The levels to be made visible.
Name	Type	Description							
</levels>	Array of integers	The levels to be made visible.							
<b>Return Value</b>	None.								

<b>Python Syntax</b>	SetVisibleLevels (<levels>)
<b>Python Example</b>	<pre>oSymbolEditor.SetVisibleLevels([3, 4])</pre>

<b>VB Syntax</b>	SetVisibleLevels <levels>
<b>VB Example</b>	<pre>Dim oSymbolEditor  Set oSymbolEditor = oProject.SetActiveDefinitionEditor("SymbolEditor", -     "Simplorer Elements\basic Elements\Circuit\Passive     Elements:R")  oldLevel = oSymbolEditor.SetVisibleLevels(Array(3, 4))</pre>

## **ToggleLevel (Symbol Editor)**

To toggle display of the graphics level in the symbol editor.

*Command:*"Visible" checkbox for a level in the **View>Levels** dialog.

*Syntax:* ToggleLevel <level>

*Return Value:* None.

*Parameters:* <level>

Type: Integer

The level to turn on or off.

*Example:* Dim oSymbolEditor

```
Set oSymbolEditor = oProject.SetActiveDefinitionEditor("SymbolEditor", _
    "Simplorer Elements\basic Elements\Circuit\Passive Elements:R")
oSymbolEditor.ToggleLevel (2)
```

## **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

<b>Python Syntax</b>	Undo()
<b>Python Example</b>	<pre>oSymbolEditor = oProject.SetActiveDefinitionEditor("SymbolEditor", _     "Simplorer Elements\basic Elements\Circuit\PassiveElements:R")</pre>

	<code>oSymbolEditor.Undo()</code>
--	-----------------------------------

## **ZoomArea (Symbol Editor)**

To change the current graphics view area.

**Command:****View>Zoom Area**

**Syntax:** `ZoomArea Array(<xpt>, <ypt>), Array(<xdelta>, <ydelta>)`

**Return Value:** None.

*Parameters:* `<xpt>`

Type: Real number

The X value of the zoom area box, in meters.

`<ypt>`

Type: Real number

The X value of the zoom area box, in meters.

`<xdelta>`

Type: Real number

The width of the zoom area box, in meters.

`<ydelta>`

Type: Real number

The height of the zoom area box, in meters.

*Example:* `Dim oSymbolEditor`

```
Set oSymbolEditor = oProject.SetActiveDefinitionEditor("SymbolEditor", _
```

```
"Simplorer Elements\basic Elements\Circuit\Passive Elements:R")
oSymbolEditor.ZoomArea Array(0, 0), Array(0.004, 0.004)
```

## **ZoomIn (Symbol Editor)**

To zoom in a standard amount to the current graphics view area.

*Command:* View>Zoom In

*Syntax:* ZoomIn

*Return Value:* None.

*Parameters:* None.

*Example:* Dim oSymbolEditor

```
Set oSymbolEditor = oProject.SetActiveDefinitionEditor("SymbolEditor",
"Simplorer Elements\basic Elements\Circuit\Passive Elements:R")
oSymbolEditor.ZoomIn
```

## **ZoomOut (Symbol Editor)**

To zoom out a standard amount to the current graphics view area.

*Command:* View>Zoom Out

*Syntax:* ZoomOut

*Return Value:* None.

*Parameters:* None.

*Example:* Dim oSymbolEditor

```
Set oSymbolEditor = oProject.SetActiveDefinitionEditor("SymbolEditor",
"Simplorer Elements\basic Elements\Circuit\Passive Elements:R")
```

`oSymbolEditor.ZoomOut`

## **ZoomPrevious (Symbol Editor)**

To zoom the current graphics view area to the previous extents.

*Command:* View>Zoom Previous

*Syntax:* ZoomPrevious

*Return Value:* None.

*Parameters:* None.

*Example:* Dim oSymbolEditor

```
Set oSymbolEditor = oProject.SetActiveDefinitionEditor("SymbolEditor",
    "Simplorer Elements\basic Elements\Circuit\Passive Elements:R")
oSymbolEditor.ZoomPrevious
```

## **ZoomToFit (Symbol Editor)**

Set the current symbol zoom to fit the contents of the currently visible page

*Command:* None

*Syntax:* ZoomToFit()

*Return Value:* None

<b>Python Syntax</b>	<code>ZoomToFit()</code>
<b>Python Example</b>	<code>oSymbolEditor = oProject.SetActiveDefinitionEditor("SymbolEditor", _</code>

```
"Simplorer Elements\basic Elements\Circuit\PassiveElements:R")
oSymbolEditor.ZoomToFit()
```

## 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: <ul style="list-style-type: none"><li>• &lt;Layername&gt;</li><li>• &lt;Type&gt;</li><li>• &lt;TopBottom&gt;</li><li>• &lt;Color&gt;</li><li>• &lt;Transparency&gt;</li><li>• &lt;Pattern&gt;</li><li>• &lt;VisFlag&gt;</li><li>• &lt;Locked&gt;</li><li>• &lt;Draw&gt;</li></ul>
	<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: <ul style="list-style-type: none"><li>• 1 makes primitives (rectangles, circles, polygons) visible.</li><li>• 2 makes lines (also known as paths or traces) visible.</li><li>• 4 makes pads (from padstacks, aka vias) visible.</li><li>• 8 makes holes (or vias) visible.</li><li>• 16 makes components visible.</li><li>• 32 makes the mesh overlay visible if they are plotted.</li><li>• 64 makes the mesh for the background material visible.</li></ul> 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.
<b>Return Value</b>	None	

<b>Python Syntax</b>	<pre> oEditor.AddLayer(     [         "NAME:layers",         "Name:=", "&lt;LayerName&gt;",     ] ) </pre>
----------------------	------------------------------------------------------------------------------------------------------------

	<pre>"Type:=", "&lt;Type&gt;", "Top Bottom:=", "&lt;TopBottom&gt;", "Color:=", &lt;Color&gt;, "Transparency:=", &lt;Transparency&gt;, "Pattern:=", &lt;Pattern&gt;, "VisFlag:=", &lt;VisFlag&gt;, "Locked:=", &lt;Locked&gt;, "DrawOverride:=", &lt;Draw&gt;, ])</pre>
<b>Python Example</b>	<pre>oEditor = oDesign.SetActiveEditor("Footprint") oEditor.AddLayer( [     "NAME:layers",     "Name:=", "signal",     "Type:=", "signal",     "Top Bottom:=", "neither",     "Color:=", 16711680,     "Transparency:=", 0,     "Pattern:=", 1,     "VisFlag:=", 127,     "Locked:=", False,</pre>

	<pre>"DrawOverride:=", 0, ])</pre>
--	------------------------------------

<b>VB Syntax</b>	<pre>oEditor.AddLayer Array("NAME:stackup layer", "Name:=", "&lt;LayerName&gt;", "Type:=", "&lt;Type&gt;", "Top Bottom:=", "&lt;TopBottom&gt;", "Color:=", &lt;Color&gt;, "Transparency:=", &lt;Transparency&gt;, "Pattern:=", &lt;Pattern&gt;, "VisFlag:=", &lt;VisFlag&gt;, "Locked:=", &lt;Locked&gt;, "DrawOverride:=", &lt;Draw&gt;,,)</pre>
<b>VB Example</b>	<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.

<b>UI Access</b>	None		
<b>Parameters</b>	Name	Type	Description

<LayerArray>

Array

An array that contains these parameters in order:

- <Layername>
- <Type>

		<ul style="list-style-type: none"> <li>• &lt;TopBottom&gt;</li> <li>• &lt;Color&gt;</li> <li>• &lt;Transparency&gt;</li> <li>• &lt;Pattern&gt;</li> <li>• &lt;VisFlag&gt;</li> <li>• &lt;Locked&gt;</li> <li>• &lt;Draw&gt;</li> <li>• and an array with the parameters:           <ul style="list-style-type: none"> <li>• &lt;Thickness&gt;</li> <li>• &lt;LowerElevation&gt;</li> <li>• &lt;Material&gt;</li> </ul> </li> </ul>
<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: <ul style="list-style-type: none"> <li>• 1 makes primitives (rectangles, circles, polygons) visible.</li> <li>• 2 makes lines (also known as paths or traces) visible.</li> </ul>

		<ul style="list-style-type: none"> <li>• 4 makes pads (from padstacks, aka vias) visible.</li> <li>• 8 makes holes (or vias) visible.</li> <li>• 16 makes components visible.</li> <li>• 32 makes the mesh overlay visible if they are plotted.</li> <li>• 64 makes the mesh for the background material visible.</li> </ul> <p>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.</p>
<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.
<b>Return Value</b>	None	

<b>Python Syntax</b>	<pre> oEditor.AddStackupLayers( [     "NAME:stackup layer",     "Name:=", "&lt;LayerName&gt;",     "Type:=", "&lt;Type&gt;",     "Top Bottom:=", "&lt;TopBottom&gt;", </pre>
----------------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

	<pre> "Color:=", &lt;Color&gt;, "Transparency:=", &lt;Transparency&gt;, "Pattern:=", &lt;Pattern&gt;, "VisFlag:=", &lt;VisFlag&gt;, "Locked:=", &lt;Locked&gt;, "DrawOverride:=", &lt;Draw&gt;, [     "NAME:sublayer",     "Thickness:=", "&lt;Thickness&gt;",     "LowerElevation:=", "&lt;LowerElevation&gt;",     "Material:=", "&lt;Material&gt;", ] ]) </pre>
<b>Python Example</b>	<pre> oEditor = oDesign.SetActiveEditor("Footprint") oEditor.AddStackupLayer( [     "NAME:stackup layer",     "Name:=", "Cover",     "Type:=", "signal",     "Top Bottom:=", "neither",     "Color:=", 16711680, ] ) </pre>

```
"Transparency:=", 0,  
"Pattern:=", 1,  
"VisFlag:=", 127,  
"Locked:=", False,  
"DrawOverride:=", 0,  
[  
    "NAME:Sublayer",  
    "Thickness:=", "0mil",  
    "LowerElevation:=", "20mil",  
    "Material:=", "gold",  
]  
])
```

<b>VB Syntax</b>	<pre>oEditor.AddStackupLayers Array("NAME:stackup layer", "Name:=", "&lt;LayerName&gt;", "Type:=", "&lt;Type&gt;", "Top Bottom:=", "&lt;TopBottom&gt;", "Color:=", &lt;Color&gt;, "Transparency:=", &lt;Transparency&gt;, "Pattern:=", &lt;Pattern&gt;, "VisFlag:=", &lt;VisFlag&gt;, "Locked:=", &lt;Locked&gt;, "DrawOverride:=", &lt;Draw&gt;, Array("NAME:Sublayer", "Thickness:=", "&lt;Thickness&gt;", "LowerElevation:=", "&lt;LowerElevation&gt;", "Material:=", &lt;Material&gt;))</pre>
<b>VB Example</b>	<pre>Set oEditor = oDesign.SetActiveEditor("Footprint") oEditor.AddStackupLayers Array("NAME:stackup layer", "Name:=", __     "NewLayer", "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", "Material:=", "FR4_epoxy"))</pre>
--	----------------------------------------------------------------------------------------------------------------------------------------------------

## 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 <b>Edit Layers</b> window.		
Parameters	Name	Type	Description
	<LayerArray>	Array	An array that contains these parameters in order: <ul style="list-style-type: none"><li>• &lt;mode&gt;</li><li>• &lt;Layername&gt;</li><li>• &lt;ID&gt;</li><li>• &lt;Type&gt;</li><li>• &lt;TopBottom&gt;</li><li>• &lt;Color&gt;</li><li>• &lt;Transparency&gt;</li><li>• &lt;Pattern&gt;</li><li>• &lt;VisFlag&gt;</li><li>• &lt;Locked&gt;</li></ul>

		<ul style="list-style-type: none"> <li>• &lt;Draw&gt;</li> <li>• &lt;Subarray&gt;</li> </ul>
<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	<p>Sets visibility for all objects on the layer using the following additive code:</p> <ul style="list-style-type: none"> <li>• 1 makes primitives (rectangles, circles, polygons) visible.</li> <li>• 2 makes lines (also known as paths or traces) visible.</li> <li>• 4 makes pads (from padstacks, aka vias) visible.</li> <li>• 8 makes holes (or vias) visible.</li> <li>• 16 makes components visible.</li> <li>• 32 makes the mesh overlay visible if they are plotted.</li> <li>• 64 makes the mesh for the background material visible.</li> </ul> <p>For example, a value of 25 (1+8+16) makes primitives, vias, and components</p>

		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	<p><i>Optional.</i> An array that contains these parameters in order:</p> <ul style="list-style-type: none"> <li>• &lt;Thickness&gt;</li> <li>• &lt;LowerElevation&gt;</li> <li>• &lt;Roughness&gt;</li> <li>• &lt;BotRoughness&gt;</li> <li>• &lt;SideRoughness&gt;</li> <li>• &lt;Material&gt;</li> <li>• &lt;FillMaterial&gt;</li> </ul> <p>This array's placeholder does not appear in the syntax or examples below but instead has its contents listed.</p>
<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.
<b>Return Value</b>	None	

<b>Python Syntax</b>	<code>oEditor.ChangeLayer(&lt;LayerArray&gt;)</code>
<b>Python Example</b>	<pre> oEditor = oDesign.SetActiveEditor("Footprint")  oEditor.ChangeLayer(     ["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",         ]     ] ) </pre>

```

        "SideRoughness:="      , "0um",
        "Material:="          , "MQ",
        "FillMaterial:="       , "FR4_epoxy"
    ]
)

```

<b>VB Syntax</b>	<code>oEditor.ChangeLayer &lt;LayerArray&gt;</code>
<b>VB Example</b>	<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 <b>Edit Layers</b> window.		
<b>Parameters</b>	Name	Type	Description
	<LayerArray>	Array	An array that contains these parameters in order: <ul style="list-style-type: none"> <li>• &lt;Layername&gt;</li> <li>• &lt;ID&gt;</li> <li>• &lt;Type&gt;</li> <li>• &lt;TopBottom&gt;</li> <li>• &lt;Color&gt;</li> <li>• &lt;Transparency&gt;</li> <li>• &lt;Pattern&gt;</li> <li>• &lt;VisFlag&gt;</li> <li>• &lt;Locked&gt;</li> <li>• &lt;Draw&gt;</li> <li>• &lt;Subarray&gt;</li> <li>• &lt;Neg&gt;</li> </ul>
	<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	<p>Sets visibility for all objects on the layer using the following additive code:</p> <ul style="list-style-type: none"> <li>• 1 makes primitives (rectangles, circles, polygons) visible.</li> <li>• 2 makes lines (also known as paths or traces) visible.</li> <li>• 4 makes pads (from padstacks, aka vias) visible.</li> <li>• 8 makes holes (or vias) visible.</li> <li>• 16 makes components visible.</li> <li>• 32 makes the mesh overlay visible if they are plotted.</li> <li>• 64 makes the mesh for the background material visible.</li> </ul> <p>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.</p>
<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: <ul style="list-style-type: none"> <li>• &lt;Thickness&gt;</li> <li>• &lt;LowerElevation&gt;</li> <li>• &lt;Roughness&gt;</li> </ul>

		<ul style="list-style-type: none"> <li>• &lt;BotRoughness&gt;</li> <li>• &lt;SideRoughness&gt;</li> <li>• &lt;Material&gt;</li> <li>• &lt;FillMaterial&gt;</li> </ul> <p>This array's placeholder does not appear in the syntax or examples below but instead has its contents listed.</p>
<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.
<b>Return Value</b>	None	

<b>Python Syntax</b>	<pre> oEditor.ChangeLayers(     [         "NAME:layers",         "Mode:=", "&lt;mode&gt;",         [             "NAME:pps"         ]     ] ) </pre>
----------------------	------------------------------------------------------------------------------------------------------------------------------------------------------

```
        ],
        [
            "NAME:stackup layer",
            "Name:=", "<LayerName>",
            "ID:=", <ID>,
            "Type:=", "<Type>",
            "Top Bottom:=", "<TopBottom>",
            "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>",


```

	<pre>        "FillMaterial:=", "&lt;FillMaterial&gt;"     ],     "Neg:=", &lt;Neg&gt; ] ])</pre>
<b>Python Example</b>	<pre>oEditor = oDesign.SetActiveEditor("Footprint") oEditor.ChangeLayers(     [         "NAME:layers",         "Mode:=", "Laminate",         [             "NAME:pps"         ],         [             "NAME:stackup layer",             "Name:=", "Cover",             "ID:=", 10,             "Type:=", "signal",             "Top Bottom:=", "neither",             "Color:=", 16711680,</pre>

```
"Transparency:=", 0,  
"Pattern:=", 1,  
"VisFlag:=", 127,  
"Locked:=", False,  
"DrawOverride:=", 0,  
[  
    "NAME:Sublayer",  
    "Thickness:=", "0mil",  
    "LowerElevation:=", "20mil",  
    "Roughness:=", "0um",  
    "BotRoughness:=", "0um",  
    "SideRoughness:=", "0um",  
    "Material:=", "gold",  
    "FillMaterial:=", "FR4_epoxy"  
,  
    "Neg:=", True  
]  
])
```

**VB Syntax**

```
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-
```

	arence:=", <Transparency>, "Pattern:=", <Pattern>, "VisFlag:=", <VisFlag>, "Locked:=", <Locked>, "DrawOverride:=", <Draw>, Array("NAME:Sublayer", "Thickness:=", "<Thickness>", "LowerElevation:=", "<LowerElevation>", "Roughness:=", "<Roughness>", "BotRoughness:=", "<BotRoughness>", "SideRoughness:=", "<SideRoughness>", "Material:=", <Material>,"FillMaterial:=", "<FillMaterial>"),"Neg:=", <Neg>))
<b>VB Example</b>	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:=", _ false, "DrawOverride:=", 0, Array("NAME:Sublayer", "Thickness:=", _ "1.6mm", "LowerElevation:=", "0mm", "Roughness:=", "0um", "BotRoughness:=", _ "0um", "SideRoughness:=", "0um", "Material:=", "FR4_epoxy"),"Neg:=", True))

## 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 Resolution	Controls drawing of segments for arcs - either dynamic and based on current zoom or fixed to specified value	Dynamic	"useFixedDrawingResolution" <sup>1</sup> "DrawingResolution" <sup>2</sup>
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" <sup>3</sup> "MinorColor" <sup>3</sup> "MajorSize" <sup>2</sup> "MinorSize" <sup>2</sup> "ShowGrid" <sup>1</sup>
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" <sup>4</sup>
Background color	Color of the layout background	white	"background color" <sup>3</sup>
Sketch Mode	Fill patterns and center lines are not drawn.	off	"DefaultToSketchMode" <sup>1</sup>
Solid Mode	Objects are filled in with solid color.	off	"fillMode" <sup>1</sup>

Draw Connection Points	Display pin symbols in the layout.	off	"show connection points" <sup>1</sup>
Draw Rats	Display lines indicating missing physical connections in nets	on	"draw rats" <sup>1</sup>
Label Vertices	Display property text labeling the vertices of selected polygons and lines.	off	"display vertex labels" <sup>1</sup>
Net Color	Color used for highlighting nets	light yellow	"NetColor" <sup>3</sup>
Rectangle Description	Specifies if rectangles are described with two points or center point, width, and height	2 points	"rectangle description" <sup>5</sup>
Default Hole Size	Default size for actual holes in a PC board	25 mil	"defaultholesize" <sup>2</sup>
Align Microwave Components	Snaps components on entry.	on	"snaptoport" <sup>1</sup>
Symbol Footprint Scaling	Used to size symbol footprints placed in layout.	Based on data extent and current length units.	"sym footprint scaling" <sup>6</sup>
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" <sup>3</sup> "secondary selection color" <sup>3</sup>
Preview Selection	Highlight the object that would be selected with a left mouse button click in the current location.	off	"preview selection" <sup>1</sup>
Snapping options	Choose snapping sources and targets and the maximum dis-	Snap distance is 20 pixels. Snapping sources are vertex, edge center, object center, & elec. conn.	"PixelSnapTolerance" <sup>7</sup> "SnapTargetVertex_on" <sup>1</sup>

	tance (in pixels) for snapping to occur. Constrain edits to grid if desired.	Snapping targets are vertex, edge center, object center, elec. conn, & grid. Edits are constrained to grid.	<sup>1</sup> "SnapTargetElecConnection_on" "SnapTargetIntersection_on" "SnapTargetGrid_on" "SnapSourceVertex_on" "SnapSourceEdgeCenter_on" "SnapSourceObjCenter_on" "SnapSourceEdge_on" "SnapSourceElecConnection_on" "ConstrainToGrid"  <sup>2</sup> "AlwaysShowLayerMergeDlg"  <sup>3</sup> "anglesnap"  <sup>4</sup> "AllowDragOnFirstClick"
Always Show Merge Layers Dialog	Controls display of the layer merging dialog.	off – Only show dialog when the user must be involved.	<sup>1</sup> "AlwaysShowLayerMergeDlg"
Rotation Increment	During rotation, objects are rotated by multiples of this amount.	Based on current angle units.	<sup>2</sup> "anglesnap"
Allow Drag on first click	Selection and move with one left mouse button click.	false	<sup>4</sup> "AllowDragOnFirstClick" <sup>5</sup> "0"

The footnotes in the table above correspond to the following:

<sup>1</sup> = Integer with a value of 1 for on/true and 0 for off/false

<sup>2</sup> = String containing a amount and units

<sup>3</sup> = Integer representing a Color

<sup>4</sup> = An Array containing (lower left x, lower left y, upper right x, upper right y)

<sup>5</sup> = 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:=", "1.43564445897937mm", "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.

<b>UI Access</b>	None						
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;LayerName&gt;</td> <td>String</td> <td>Name of the layer</td> </tr> </tbody> </table>	Name	Type	Description	<LayerName>	String	Name of the layer
Name	Type	Description					
<LayerName>	String	Name of the layer					
<b>Return Value</b>	<p>An array of strings, as follows:</p> <ul style="list-style-type: none"> <li>• Type: typename</li> <li>• TopBottomAssociation: "Top" "Neither" "Bottom" "Template" "Invalid"</li> <li>• Color: integer [representing rgb in hex]</li> </ul>						

- 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', 'IsVisibleHole: true', 'IsVisibleComponent: true', 'IsLocked: false', 'LayerId: 7', 'Index: 0', 'LayerThickness: 0', 'IsIgnored: false', 'NumberOfSublayers: 1', 'Material0: copper', 'FillMaterial0: FR4_epoxy', 'Thickness0: 0mil', 'LowerElevation0: 62mil', 'Roughness0 Type: Huray', 'Roughness0: 0mil, 2.9', 'BottomRoughness0 Type: Huray', 'BottomRoughness0: 0mil, 2.9', 'SideRoughness0 Type: Huray', 'SideRoughness0: 0mil, 2.9']
```

<b>Python Syntax</b>	GetLayerInfo(<layer_name>)
<b>Python Example</b>	<pre>oModule = oDesign.GetModule("Footprint") oModule.GetLayerInfo ("TopLayer")</pre>

<b>VB Syntax</b>	GetLayerInfo <layer_name>
<b>VB Example</b>	<pre>Set oModule = oDesign.GetModule("Footprint")</pre>

```
oModule.GetLayerInfo "TopLayer"
```

## 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>, // 1<sup>st</sup> primitive name  
                  <object\_name>, // 2<sup>nd</sup> 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

**GetPropertyValues** 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>, // 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")

## **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.

# 31 - 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 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.

# 32 - Core Global Script Context Commands

To run these commands:

```
import CoreGlobalScriptContextFunctions
CoreGlobalScriptContextFunctions.[CommandName]
```

The following are general script commands recognized by the **CoreGlobalScriptContextFunctions** object:

- [AddErrorMessage](#)
- [AddFatalMessage](#)
- [AddInfoMessage](#)
- [AddWarningMessage](#)
- [LogDebug](#)
- [LogError](#)

## AddErrorMessage

Adds an error message to the **Message Manager** window. AddErrorMessage is a function of CoreGlobalScriptContextFunctions.

<b>UI Access</b>	N/A		
<b>Parameters</b>	Name <i>&lt;message&gt;</i>	Type String	Description Error message.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	AddErrorMessage(<message>)
<b>Python Example</b>	<pre>import CoreGlobalScriptContextFunctions CoreGlobalScriptContextFunctions.AddErrorMessage('My error message.')</pre>

<b>VB Syntax</b>	N/A
<b>VB Example</b>	N/A. Script cannot be run in VBScript.

## AddFatalMessage

Adds a fatal error message to the **Message Manager** window. AddFatalMessage is a function of CoreGlobalScriptContextFunctions.

<b>UI Access</b>	N/A		
<b>Parameters</b>	Name <i>&lt;message&gt;</i>	Type String	Description Error message.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	AddFatalMessage( <i>&lt;message&gt;</i> )
<b>Python Example</b>	<pre>import CoreGlobalScriptContextFunctions CoreGlobalScriptContextFunctions.AddFatalMessage('My fatal error message.')</pre>

<b>VB Syntax</b>	N/A
<b>VB Example</b>	N/A. Script cannot be run in VBScript.

## AddInfoMessage

Adds an informational message to the **Message Manager** window. AddInfoMessage is a function of CoreGlobalScriptContextFunctions.

<b>UI Access</b>	N/A		
<b>Parameters</b>	Name <i>&lt;message&gt;</i>	Type String	Description Informational message.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	AddInfoMessage( <i>&lt;message&gt;</i> )
<b>Python Example</b>	<pre>import CoreGlobalScriptContextFunctions CoreGlobalScriptContextFunctions.AddInfoMessage('My info.')</pre>

<b>VB Syntax</b>	N/A
<b>VB Example</b>	N/A. Script cannot be run in VBScript.

## AddWarningMessage

Adds a warning message to the **Message Manager** window. AddWarningMessage is a function of CoreGlobalScriptContextFunctions.

<b>UI Access</b>	N/A								
<b>Parameters</b>	<table border="1"> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> <tr> <td>&lt;message&gt;</td> <td>String</td> <td>Warning message.</td> </tr> </table>			Name	Type	Description	<message>	String	Warning message.
Name	Type	Description							
<message>	String	Warning message.							
<b>Return Value</b>	None.								

<b>Python Syntax</b>	AddWarningMessge(<message>)
<b>Python Example</b>	<pre>import CoreGlobalScriptContextFunctions CoreGlobalScriptContextFunctions.AddWarningMessage('My warning.')</pre>

<b>VB Syntax</b>	N/A
<b>VB Example</b>	N/A. Script cannot be run in VBScript.

## LogDebug

Adds a debug line to the log specified at **Tools > Debug Logging**. LogDebug is a function of CoreGlobalScriptContextFunctions.

<b>UI Access</b>	N/A								
<b>Parameters</b>	<table border="1"> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> <tr> <td>&lt;message&gt;</td> <td>String</td> <td>Debug message.</td> </tr> </table>			Name	Type	Description	<message>	String	Debug message.
Name	Type	Description							
<message>	String	Debug message.							
<b>Return Value</b>	None.								

<b>Python Syntax</b>	LogDebug(<message>)
----------------------	---------------------

<b>Python Example</b>	<pre>import CoreGlobalScriptContextFunctions CoreGlobalScriptContextFunctions.LogDebug('My debug message.')</pre>
-----------------------	-------------------------------------------------------------------------------------------------------------------

<b>VB Syntax</b>	N/A
<b>VB Example</b>	N/A. Script cannot be run in VBScript.

## .LogError

Adds an error line to the log specified at **Tools > Debug Logging**. LogError is a function of CoreGlobalScriptContextFunctions.

<b>UI Access</b>	N/A		
<b>Parameters</b>	Name <error>	Type String	Description Error to log.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	.LogError(<error>)
<b>Python Example</b>	<pre>import CoreGlobalScriptContextFunctions CoreGlobalScriptContextFunctions.LogError('My error.')</pre>

<b>VB Syntax</b>	N/A
<b>VB Example</b>	N/A. Script cannot be run in VBScript.

# 33 - 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" & swn,cfreq, "")  
z = val  
'  
' Create the line of text to send to the file and write it to the  
file.  
'  
line = x & chr(9) & y & chr(9) &z  
ofile.WriteLine line  
Next  
'  
' Delete the 2 output variables before finishing.  
'  
oDesign.DeleteOutputVariable "re_S"  
oDesign.DeleteOutputVariable "im_S"  
'  
' Close the file.  
'  
ofile.close  
End if
```

### 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 Impedance")
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 Ele-
ment")

Layer_Number_of_Layers = oBoundary.GetPropValue("Number of
Layers")

Layer_L1_Thickness = oBoundary.GetPropValue("Layer 1/Thick-
ness")

Layer_L1_Material = oBoundary.GetPropValue("Layer 1/Ma-
terial")

Layer_L2_Thickness = oBoundary.GetPropValue("Layer 2/Thick-
ness")

Layer_L2_Material = oBoundary.GetPropValue("Layer 2/Ma-
terial")

MsgBox ("Name = " & Layer_Name & vbCrLf _ 
& "Type = " & Layer_Type & vbCrLf _ 
& "Roughness = " & Layer_Roughness & vbCrLf _ 
& "Inf Ground = " & Layer_Inf_Ground_Plane & vbCrLf _ 
& "Two-Sided = " & Layer_Two_Sided & vbCrLf _ 
& "Shell Element = " & Layer_Shell_Element & vbCrLf _ 
& "Number of Layers = " & Layer_Number_of_Layers & vbCrLf

— 

& "Layer 1 Thickness = " & Layer_L1_Thickness & vbCrLf _ 
& "Layer 1 Material = " & Layer_L1_Material & vbCrLf _ 
& "Layer 2 Thickness = " & Layer_L2_Thickness & vbCrLf _ 
& "Layer 2 Material = " & Layer_L2_Material)

Next

End Sub
```

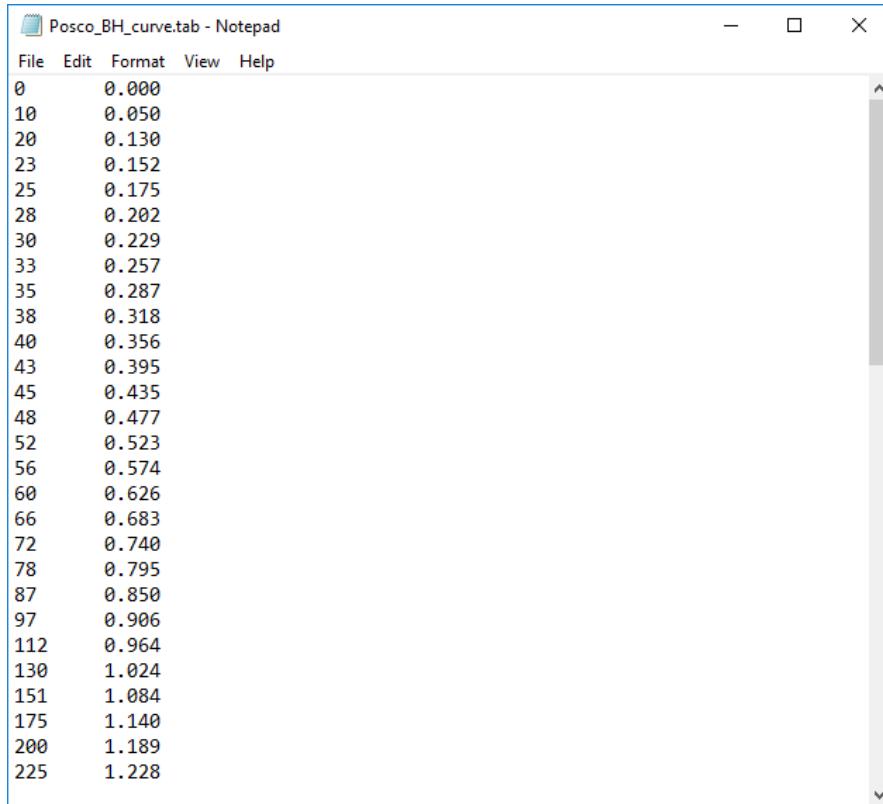
# IronPython Example Scripts

IronPython Examples:

- [BH Coordinates Python Script](#)
- [Equation Based Curve Python Script](#)

## BH Coordinates Python Script

This sample Python script adds a material ("Posco 35PN250") with BH coordinates from a specified file (Posco\_BH\_curve.tab):



The screenshot shows a Windows Notepad window titled "Posco\_BH\_curve.tab - Notepad". The menu bar includes File, Edit, Format, View, and Help. The main text area contains a list of numerical pairs, each consisting of a coordinate value followed by its corresponding BH coordinate value. The data starts at index 0 and ends at index 225.

Index	BH Coordinate
0	0.000
10	0.050
20	0.130
23	0.152
25	0.175
28	0.202
30	0.229
33	0.257
35	0.287
38	0.318
40	0.356
43	0.395
45	0.435
48	0.477
52	0.523
56	0.574
60	0.626
66	0.683
72	0.740
78	0.795
87	0.850
97	0.906
112	0.964
130	1.024
151	1.084
175	1.140
200	1.189
225	1.228

To recreate this tab file, paste [the text below the script](#) into a text editor and save as Posco\_BH\_curve.tab.

The script itself includes comment lines, which are preceded by # and offer explanations for the subsequent line(s).

### Script Contents

```
# specify path to the file with BH coordinates
path_to_file = r"D:\Posco_BH_curve.tab"
```

```
# specify name of the material
material_name = "Posco 35PN250"
oProject = oDesktop.GetActiveProject()
oDefinitionManager = oProject.GetDefinitionManager()

""" create list with B and H points to pass to AddMaterial command
bh_coordinates list is a three dimensional array: 1D: name, 2D:
Coordinate list, 3D: BH point"""

bh_coordinates = ["NAME:BHCoordinates", ["NAME:DimUnits", "", ""]]
with open(path_to_file) as input_file:

    for line in input_file:
        h, b = line.split()

        bh_coordinates.append(["NAME:Coordinate", [
            "NAME:CoordPoint",
            float(h),
            float(b)
        ]])
    )

# create a new magnetic material with BH curve
oDefinitionManager.AddMaterial(
    [
        "NAME:" + material_name,
        "CoordinateSystemType:=", "Cartesian",
        "BulkOrSurfaceType:=" , 1,
        [
            "NAME:PhysicsTypes",
            "set:=" , ["Electromagnetic"]
        ],
        [
            "NAME:permeability",

```

```
"property_type:=" , "nonlinear",
"BTypeForSingleCurve:=" , "normal",
"HUnit:=" , "A_per_meter", # unit can be specified as
variable
"BUnit:=" , "tesla", # unit can be specified as variable
"IsTemperatureDependent:=", False,
bh_coordinates,
[
    "NAME:Temperatures"
]
],
"conductivity:=" , "1818181.82",
[
    "NAME:magnetic_coercivity",
    "property_type:=" , "VectorProperty",
    "Magnitude:=" , "0A_per_meter",
    "DirComp1:=" , "1",
    "DirComp2:=" , "0",
    "DirComp3:=" , "0"
]
])
])
```

### **Posco\_BH\_Curve.tab Contents**

0	0.000
10	0.050
20	0.130
23	0.152
25	0.175
28	0.202
30	0.229
33	0.257

35	0.287
38	0.318
40	0.356
43	0.395
45	0.435
48	0.477
52	0.523
56	0.574
60	0.626
66	0.683
72	0.740
78	0.795
87	0.850
97	0.906
112	0.964
130	1.024
151	1.084
175	1.140
200	1.189
225	1.228
252	1.258
282	1.283
317	1.304
357	1.324
402	1.343
450	1.360
500	1.375
550	1.387
602	1.398
657	1.407

---

717        1.417  
784        1.426  
867        1.437  
974        1.448  
1117      1.461  
1299      1.478  
1515      1.495  
1752      1.513  
2000      1.529  
2253      1.543  
2521      1.557  
2820      1.570  
3167      1.584  
3570      1.600  
4021      1.617  
4503      1.634  
5000      1.652  
5503      1.669  
6021      1.685  
6570      1.701  
7167      1.717  
7839      1.733  
8667      1.749  
9745      1.769  
11167     1.793  
12992     1.820  
15146     1.851  
17518     1.882  
20000     1.909  
22552     1.931

---

```
25417    1.948
28906    1.961
33333    1.971
38932    1.981
```

## Equation Based Curve Python Script

This sample Python script creates an equation based curve that produces a helix.

```
from math import pi, sin, cos

oProject = oDesktop.GetActiveProject()
oDesign = oProject.GetActiveDesign()
oEditor = oDesign.SetActiveEditor("3D Modeler")

Start_t = 0
End_t = pi*2

Npoint = 128
Nsection = Npoint-1

d_t = (End_t-Start_t)/Nsection

for n in range(1,Nsection):

    P1 = Start_t+d_t*(n-1)
    P2 = P1+d_t
    X_t1 = cos(P1*6)
    Y_t1 = sin(P1*6)
    Z_t1 = P1
    X_t2 = cos(P2*6)
    Y_t2 = sin(P2*6)
    Z_t2 = P2
```

```
oEditor.CreatePolyline(
[
    "NAME:PolylineParameters",
    "IsPolylineCovered:=" , True,
    "IsPolylineClosed:=" , False,
    [
        "NAME:PolylinePoints",
        [
            "NAME:PLPoint",
            "X:=" , '1mm*' + str(X_t1),
            "Y:=" , '1mm*' + str(Y_t1),
            "Z:=" , '1mm*' + str(Z_t1)
        ],
        [
            "NAME:PLPoint",
            "X:=" , '1mm*' + str(X_t2),
            "Y:=" , '1mm*' + str(Y_t2),
            "Z:=" , '1mm*' + str(Z_t2)
        ]
    ],
    [
        "NAME:PolylineSegments",
        [
            "NAME:PLSegment",
            "SegmentType:=" , "Line",
            "StartIndex:=" , 0,
            "NoOfPoints:=" , 2
        ]
    ]
]
```

```
],  
[  
    "NAME:PolylineXSection",  
    "XSectionType:=" , "None",  
    "XSectionOrient:=" , "Auto",  
    "XSectionWidth:=" , "0mm",  
    "XSectionTopWidth:=" , "0mm",  
    "XSectionHeight:=" , "0mm",  
    "XSectionNumSegments:=" , "0",  
    "XSectionBendType:=" , "Corner"  
]  
,  
[  
    "NAME:Attributes",  
    "Name:=" , "Polyline"+str(n),  
    "Flags:=" , "",  
    "Color:=" , "(132 132 193)",  
    "Transparency:=" , 0,  
    "PartCoordinateSystem:=" , "Global",  
    "UDMID:=" , "",  
    "MaterialValue:=" , "\"vacuum\"",  
    "SurfaceMaterialValue:=" , "\"\"",  
    "SolveInside:=" , True,  
    "IsMaterialEditable:=" , True,  
    "UseMaterialAppearance:=" , False,  
    "IsLightweight:=" , False  
])
```

## HFSS Waveguide Array Python Script

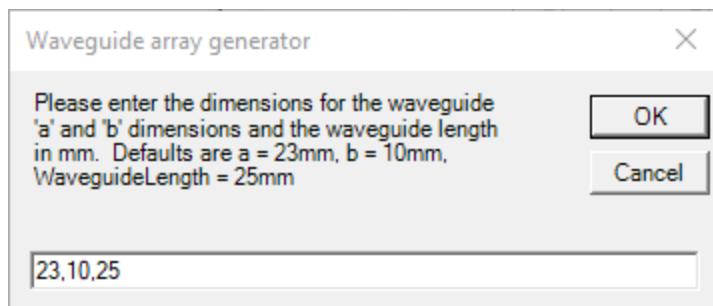
This sample Python script creates an HFSS Waveguide array. The script includes comment lines, which are preceded by an apostrophe ( ' ), that offer explanations for each subsequent line

---

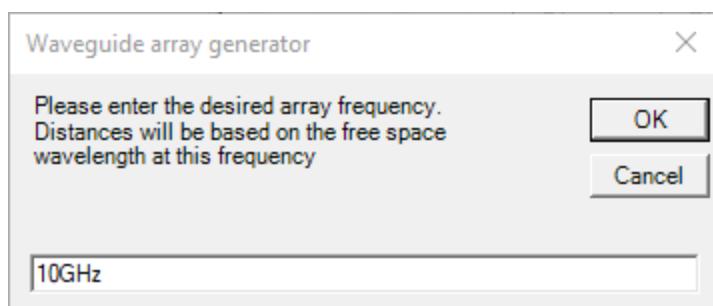
or lines. The script includes examples of creating a 3D model, boundaries and excitation, solution setup and sweep, as well as reports.

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.

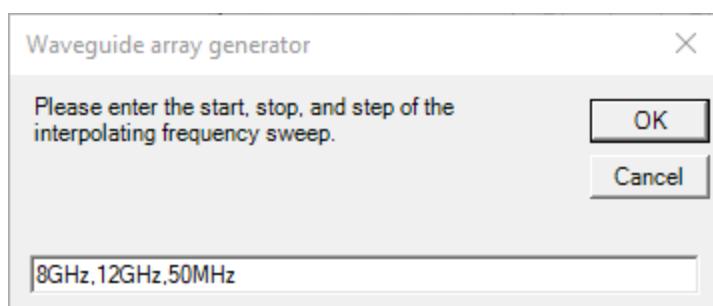
The first dialog asks for input for a and b dimensions and the waveguide length.



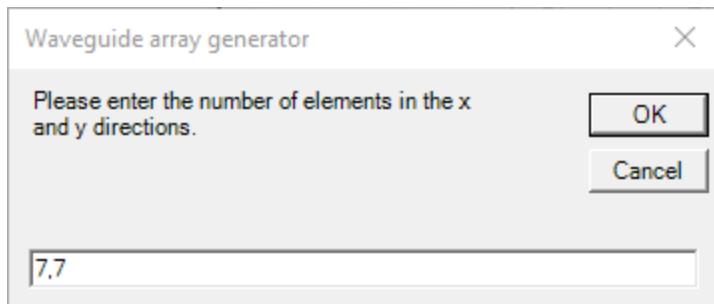
The next asks for the array frequency.



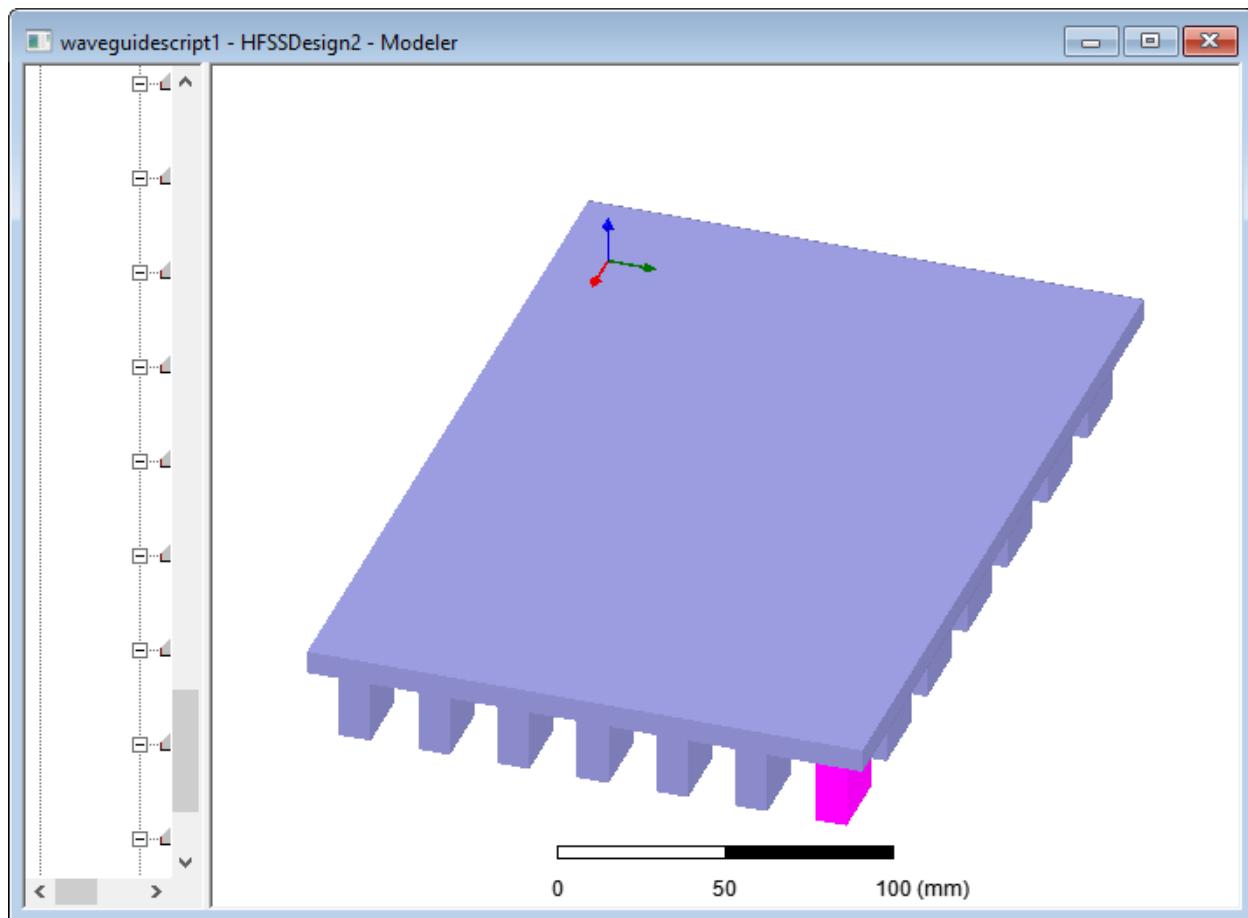
The next asks for start, stop and step values for in an interpolating frequency sweep.



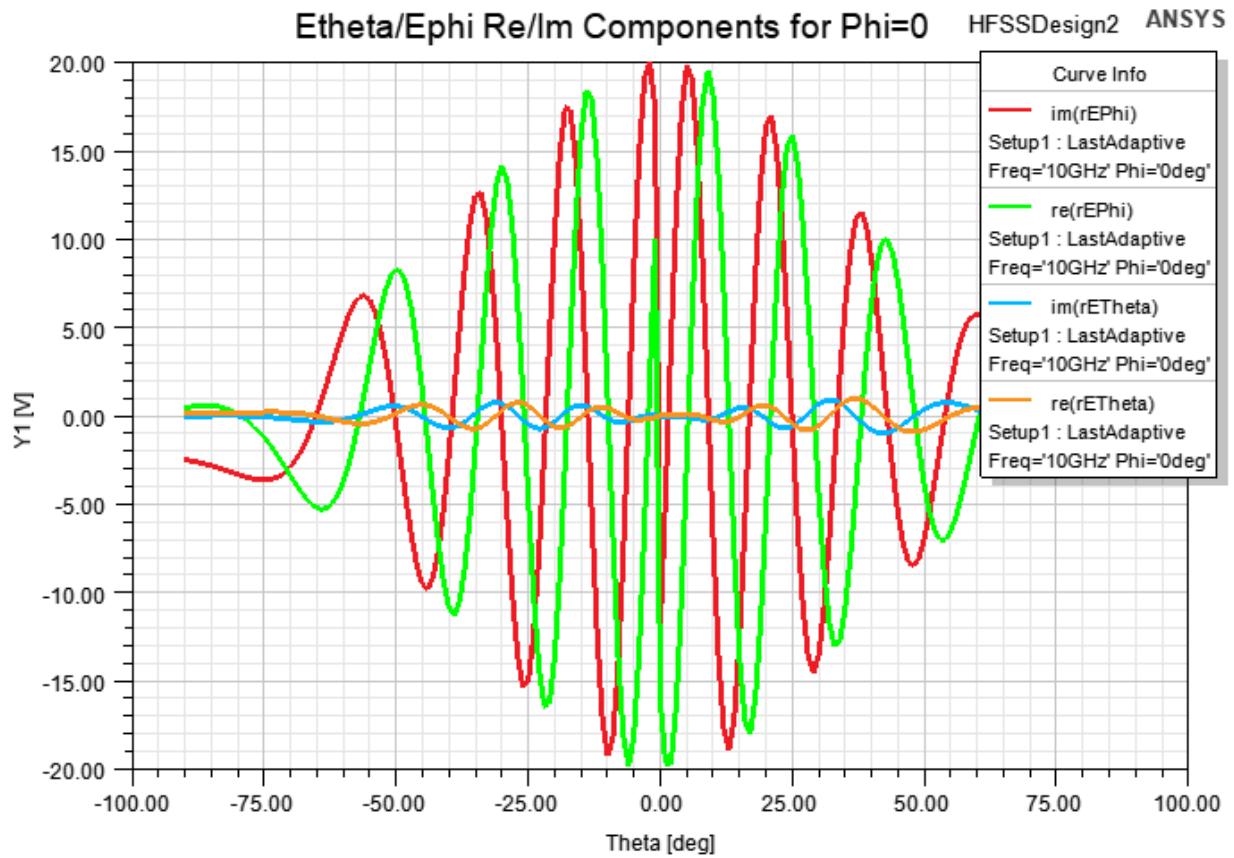
The last one asks for the array definition.

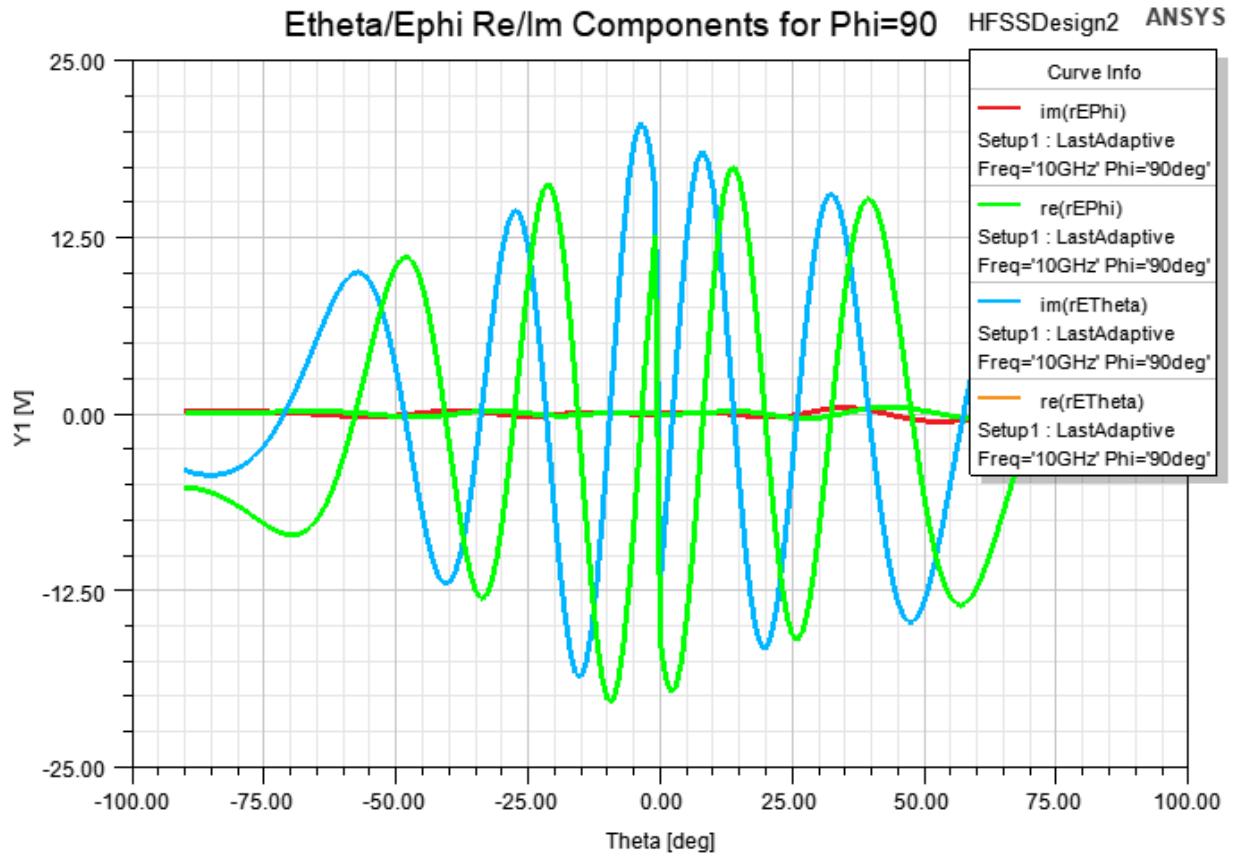


The following figure shows the waveguide array generated by the defaults.



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:Mode1",
        "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 radi-
ation 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)"]]
```

], [ ])

# Index

## 2

2.5D Via Commands 26-1

## 3

3D Modeler editor commands 10-1

AlignFaces 10-126

AssignMaterial 10-127

AssignSurfaceMaterial 10-256

Chamfer 10-130

ChangeProperty 5-12, 10-260,  
30-24

CleanUpModel 10-132

CloseAllWindows 10-263

Connect 10-133

Copy 10-105

CoverLines 10-134

CoverSurfaces 10-135

Create3DComponent 10-6

CreateBondwire 10-9

CreateBox 10-13

CreateCircle 10-16

CreateCone 10-19

CreateCutplane 10-21

CreateCylinder 10-23

CreateEllipse 10-26

CreateEntityList 10-136

CreateEquationCurve 10-29

CreateEquationSurface 10-32

CreateFaceCS 10-137, 10-143

CreateGroup 10-142

CreateHelix 10-35

CreateObjectFromedges 10-150

CreateObjectFromFace 10-152

CreateObjectFromFaces 10-154

CreatePoint 10-37

CreatePolyline 10-39

CreateRectangle 10-44

CreateRegion 10-47

CreateRegularPolygon 10-51

CreateRegularPolyhedron 10-54

CreateRelativeCS 10-155

CreateSphere 10-57

CreateSpiral 10-59

CreateTorus 10-61

CreateUserDefinedModel 10-64

CreateUserDefinedPart 10-66

Defeature 10-264

Delete 10-265

DeleteEmptyGroups 10-157

DeleteLastOperation	10-158	GetBodyNamesByPosition	10-270
DeleteOperation	10-159	GetCoordinateSystems	10-191
DeletePolylinePoint	10-105	GetEdgeByPosition	10-280
DetachEdges	10-161	GetEdgeIDFromNameForFirstOperation	
DetachFaces	10-162	10-282	
DuplicateAlongLine	10-107	GetEdgeIDsfromFace	10-282
DuplicateAroundAxi	10-109	GetEdgeIDsfromObject	10-283
DuplicateMirror	10-112	GetEdgeLength	10-284
EditEntityList	10-164	GetEdgePositionAtNormalizedParameter	
EditFaceCS	10-165	9-84	
EditObjectCS	10-171	GetEntityListIDByName	10-284
EditPolyline	10-79	GetFaceArea	10-286
EditRelativeCS	10-178	GetFaceByPosition	10-287
Export	10-180	GetFaceCenter	10-287
ExportModelImageToFile	10-182, 12-66	GetFaceIDFromNameForFirstOperation	
Fillet	10-186	10-289	
FlattenGroup	10-188	GetFacetIDs	10-290
GenerateAllUserDefinedModels		GetFacetIDsOfSheet	10-290
10-266		GetModelBoundingBox	10-292
GenerateHistory	10-189	GetPartsForUserDefinedModel	10-301
GenerateUserDefinedModel	10-267	GetPoints	10-302
Get3DComponentParameters		GetProperties	5-62, 7-67, 9-98
10-86		GetProperty	5-63, 7-68, 9-100,
Get3DComponentPartNames		10-304, 12-90, 29-121	
10-86		GetPropEvaluatedValue	10-303
GetActiveCoordinateSystem	10-190	GetPropSIValue	10-307
GetAct-		GetRelativeCoordinateSystems	10-309
iveCoordi-		GetPosition	10-311
ateSystemTransform	10-190	GetUser	
		GetVertexIDFromNameForFirstOperation	
		10-312	

---

GetVertexIDsFromEdge 10-312  
GetVertexIDsFromFace 10-313  
GetVertexIDsFromObject 10-  
314  
GetVertexPosition 10-314  
GetWireBodyNames 10-315  
HealObject 10-192  
Import 10-196  
ImportDXF 10-200, 30-121  
ImportFromClipboard 10-204  
ImportGDSII 10-205  
Imprint 10-208  
ImprintProjection 10-209  
Insert3DComponent 10-87  
InsertComponent 10-88  
Intersect 10-211  
Mirror 10-115  
Move 10-117  
MoveCSToEnd 10-213  
MoveEntityToGroup 10-214  
MoveFaces 10-215  
OffsetFaces 10-119  
PageSetup 10-316  
ProjectSheet 10-217  
RemoveBadEdges 10-318  
RemoveBadFaces 10-319  
RemoveBadVertices 10-320  
RenamePart 10-320  
Replacewith3DComponent 10-  
219  
Rotate 10-121  
Scale 10-122  
Section 10-222  
SeparateBody 10-224  
SetModelUnits 10-225  
SetPropValue Modeler 10-322  
SetTopDownViewDir-  
ectionForActiveView 10-323  
SetTopDownViewDir-  
ectionForAllViews 10-323  
SetWCS 10-226  
Simplify 10-228  
Split 10-231  
Stitch 10-233  
Subtract 10-235  
SweepAlongPath 10-92  
SweepAlongVector 10-94  
SweepAroundAxis 10-96  
SweepFacesAlongNormal 10-98,  
10-98, 10-236, 10-236  
Sweep-  
FacesAlongNormalWithAt-  
tributes 10-100  
ThickenSheet 10-238  
UncoverFaces 10-240  
Ungroup 10-242  
Unite 10-241  
UpdateComponentDefinition 10-  
103  
UpdatePriorityList 10-324  
UpgradeVersion 10-325

Validate3DComponent 10-326	AddCartesianLimitLine 12-6
WrapSheet 10-243	AddCartesianLimitLineFromCurve 12-8
WriteHistoryTreeLayoutForTest 10-327	AddCartesianLimitLineFromEquation 12- 10
3D Modeler editor object commands	AddCartesianXMarker 12-12
GetNumObjects 10-293	AddCartesianYMarker 12-12
GetObjectIDByName 10-294	AddCartesianYMarkerToStack 12-13
GetObjectName 10-294	AddCircuitRefPort 28-261, 30-8
GetObject NameByEdgeID 10- 295	AddDataset 5-4, 8-1, 9-7, 29-22
GetObject NameByFaceID 10- 296	AddDefinitionFromBlock 10-247, 29-25
GetObject NameByID 10-296	AddDefinitionFromLibFile 10-252
GetObject NameByVertexID 10- 297	AddDeltaMarker 12-14
GetObject ShapeType 10-298	AddDesignVariablesForDynamicLink 9-10
GetObject Volume 10-300	AddDiffPair 22-4
GetObjPath 10-301	AddDynamicLink 9-11
	AddDynamicNPortData 29-160
	AddHole 28-58
	Additional Property Scripting Example 7- 75
	AddLevel 30-146
	AddMarker 12-16
	AddMarker[Fields Reporter] 16-4
	AddMarkerToPlot 16-5
	AddMaterial 5-7, 29-30, 29-68
	AddMenuProp 7-28
	AddMenuProp2 7-30
	AddMessage 3-6
	AddModelingProperties 9-13
	AddNamedExpr 16-6, 17-3
<b>A</b>	
Aborting Scripts 1-11	
Activate 28-39, 30-7	
Activation and Deactivation Meth- ods 28-39	
Add 19-1, 19-8, 19-12, 19-18, 24-2, 25-2, 27-1, 29-3, 29-34, 29- 40, 29-141, 29-219, 29-270, 29-310, 29-342	
AddAddSolverOnDemandModel 29-169	
AddAllEyeMeasurements 12-6	
AddAntennaOverlay 18-11	
AddCableToBundle 10-330	

AddNote 12-17  
AddNPortData 29-163  
AddPoint 28-56  
AddPortsToAllNets 28-111, 28-263,  
28-263, 30-14, 30-14  
AddPortsToNet 28-111, 28-263, 30-  
14  
AddProp 7-31  
AddProp2 7-35  
AddRadFieldSourceGroup 18-13  
AddRefPort 25-2, 28-112, 28-264,  
30-14  
AddRefPortUsingEdges 25-4  
AddRuleSet 31-1  
AddRun 31-3  
AddSweep 24-7  
AddTraceCharacteristics 12-19  
AddTraces 12-20  
AlignFaces 10-126  
AlignHorizontal 30-147  
AlignObjects 28-287, 30-17  
AlignPorts 28-266, 30-18  
AlignVertical 30-148  
Analysis module commands  
    Analyze 9-13  
    AnalyzeAll 9-14  
    AnalyzeAllNominal 9-14  
    CopySetup 14-7, 14-82  
    EditSetup 14-9, 14-84  
    InsertSetup 14-41, 14-114  
Analyze 24-8, 24-8  
    Analysis module command 9-13  
AnalyzeDistributed 9-15  
AnalyzeSweep 24-9  
Ansoft Application Object  
    commands 2-1  
    GetAppDesktop 2-2  
ApplyMeshOps 9-16, 9-76  
ApplyReportTemplate 12-23  
Area 28-63  
AreMaterialPropertiesEqual 29-51  
AreSimulationsRunning 3-7  
AreSurfaceMaterialPropertiesEqual  
    29-52  
arithmetic operators 1-6  
array variables 1-4  
AssignCircuitRefPort 28-114, 28-266,  
28-269, 30-19, 30-21  
AssignDCThickness 9-17  
AssignMaterial 10-127  
AssignSurfaceMaterial 10-256

**B**

BBoxLL 28-60  
BBoxUR 28-60  
Boolean Operations on Primitives 28-  
66  
Boundary and Excitation Module  
    Script Commands 13-1

Boundary/Excitation module commands	CalculatorRead 16-10, 17-5
AssignDCThickness 9-17	CalculatorWrite 16-11, 17-7
CreateNportCircuitElements 30-53	CalcWrite 16-9, 17-8
CreatePML 13-2	Cascade 22-5
ModifyPMLGroup 13-5	Cavity Commands 27-1
PMLGroupCreated 13-7	Chamfer 10-130
PMLGroupModified 13-8	ChangeGeomSettings 16-12, 17-9
RecalculatePMLMaterials 13-10	ChangeOptions 28-128, 28-288, 30-212
BreakUDMConnection 10-259	ChangeProperty 5-12, 5-16, 9-19, 10-260, 12-23, 29-53, 30-24, 30-149
BringToFront 29-353, 30-148	CircleIntersectsPolygon 28-63
<b>C</b>	
Cable Modeling commands	ClcEval 16-13, 17-10
AddCableToBundle 10-330	ClcMaterial 16-14, 17-11
CreateCableBundle 10-331	ClcMaterialValue 16-14, 17-11
CreateCableHarness 10-332	CleanUpModel 10-132
CreateClockSource 10-335	ClearAllMarkers 12-27
CreatePWLSOURCE 10-337	ClearAllMarkers[Fields Reporter] 16-15
CreateStraightWireCable 10-338, 10-340	ClearAllNamedExpr 16-16, 17-12
ExportCableLibrary 10-342	ClearAllTraceCharacteristics 12-28
ImportCableLibrary 10-342	ClearDiffPairs 22-6
RemoveCables 10-343	ClearLayerMappings 28-245, 30-27
UpdateCableHarness 10-344	ClearMessages 3-7, 5-29
Cable Modeling Commands 10-328	ClearRefPort 28-143, 28-271, 30-27
CalcOp 16-8, 17-4	ClearRelative 28-73, 30-28
CalcRead (deprecated) 17-4	ClearSolutionCache 29-169
CalcStack 16-8, 17-6	ClipPlane 28-144, 28-294
	Clone 22-7
	CloneMaterial 5-30, 29-66
	CloneReportsFromDatasetSolution 12-29

Close 5-31, 22-8  
CloseAllWindows 3-8, 10-263  
CloseEditor 30-162, 30-218  
CloseProject 3-9  
CloseProjectNoForce 3-10  
ClosestPointOnLine 28-53  
Combine 22-9  
comment lines 1-12, 1-14  
comparison operators 1-7  
ComplInstance Functions 64  
ComplInstance Script Commands (multiple)  
Component Manager Script Commands 29-140  
conditional statements  
    If...Then... Else 1-8  
    Select Case 1-8  
    types of 1-8  
Connect 10-133, 28-32, 30-28  
ConstructVariationString 9-23  
conventions  
    scripting help 1-1  
converting data types 1-10  
ConvertPlaneToTrace 25-4, 30-28  
ConvertPrimitives 26-1, 28-144  
ConvertToDynamic 29-281  
ConvertToParametric 29-281  
ConvertTraceToPlane 25-5, 30-29  
Coordinate System Methods 28-69  
Copy 10-105, 28-32, 28-146, 30-29, 30-162  
CopyDesign 5-32  
CopyItemCommand 9-24  
CopyNamedExprToStack 16-16, 17-13  
CopyPlotSettings 12-29  
CopyRadFieldSetup 18-2  
CopyReportDefinition 12-30  
CopyReportsData 12-31  
Copyright and Trademark Information 2  
CopySetup  
    Analysis module command 14-7, 14-82  
    Optimetrics module command 14-7, 14-82  
CopyToPlanarEM 28-294, 30-29  
CopyTraceDefinitions 12-32  
CopyTracesData 12-32  
Core Global Script Context Commands 32-1  
Count 3-11  
CoupleEdgePorts 25-5  
CoverLines 10-134  
CoverSurfaces 10-135  
CPython 1-72  
Create Primitives 28-2  
Create Voids in Primitives 28-12  
Create3DComponent 10-6  
Create3DStructure 28-74, 30-30

CreateArc 30-163  
CreateBondwire 10-9  
CreateBox 10-13  
CreateCableBundle 10-331  
CreateCableHarness 10-332  
CreateCircle 10-16, 28-3, 28-150, 30-35, 30-164, 30-218  
CreateCircleVoid 28-13, 30-36, 30-219  
CreateCircuitPort 28-151, 28-272, 30-37  
CreateClockSource 10-335  
CreateComponent 28-18, 28-154, 30-40  
CreateCone 10-19  
CreateCS 28-70, 30-31  
CreateCutplane 10-21  
CreateCylinder 10-23  
CreateEdgePort 28-156, 28-274, 30-41, 30-220  
CreateEllipse 10-26  
CreateEntityList 10-136  
CreateEquationCurve 10-29  
CreateEquationSurface 10-32  
CreateFaceCS 10-137, 10-143  
CreateFieldPlot 16-17  
CreateGroundPortComponent 25-5, 30-43  
CreateGroup 10-142  
CreateGroupSelected 28-74, 30-44  
CreateHelix 10-35  
CreateHole 28-19, 30-44  
CreateImage 30-165  
CreateInterfaceGround 25-6, 30-46  
CreateInterfacePort 25-6, 30-46  
CreateInterfacePortComponent 25-7, 30-46  
CreateLine 28-4, 28-158, 30-47, 30-166, 30-222  
CreateLineFromPolygon 28-48, 28-161, 30-49  
CreateLineVoid 28-14, 30-50, 30-223  
CreateMeasure 28-21, 30-51, 30-224  
CreateNetClass 28-161, 28-253, 30-55  
CreateNPort 28-277  
CreateNportCircuitElements 30-53  
CreateObjectFromEdges 10-150  
CreateObjectFromFace 10-152  
CreateObjectFromFaces 10-154  
CreateObjectFromPolygon 28-47, 28-161, 30-56  
CreateOutputVariable 11-1  
CreatePin 28-23, 30-56, 30-167, 30-227  
CreatePinGroup 28-24, 30-57  
CreatePinGroupPort 28-24, 28-162, 30-57  
CreatePML 13-2  
CreatePoint 10-37  
CreatePolygon 28-8, 28-163, 30-58, 30-168, 30-228  
CreatePolygonVoid 28-15, 30-59, 30-226  
CreatePolyline 10-39

CreatePortInstancePorts 28-279,  
30-60

CreatePortsOnComponents 28-  
165, 28-279, 30-60

CreatePWLSource 10-337

CreateRectangle 10-44, 28-7, 28-  
167, 30-61, 30-169, 30-228

CreateRectangleVoid 28-16, 30-62

CreateRegion 10-47

CreateRegularPolygon 10-51

CreateRegularPolyhedron 10-54

CreateRelativeCS 10-155

CreateReport 9-24, 12-33

CreateReport [Designer] 12-48

CreateReportFromTemplate 12-54

CreateReportofAllQuantities 12-55

CreateSphere 10-57

CreateSpiral 10-59

CreateText 28-9, 30-64, 30-169,  
30-229

CreateTorus 10-61

CreateTrace 28-24, 30-67

CreateUserDefinedModel 10-64

CreateUserDefinedPart 10-66

CreateUserDefinedSolution 21-1

CreateVia 28-30, 30-72, 30-230

Cross 28-52

Cut 30-170

CutDesign 5-32

CutOutSubDesign 28-170, 28-295,  
30-73

**D**

dataset commands

AddDataset 5-4, 8-1, 9-7, 29-22

DeleteDataset 5-33, 8-4, 9-39, 29-  
67

EditDataset 5-36, 8-5, 29-95

ExportDataset 5-47, 8-7, 9-55, 29-  
114

HasDataset 8-8

ImportDataset 5-67, 8-9, 9-107,  
29-122

Dataset Script Commands 8-1

DeactivateOpen 28-40, 30-78

DeactivateShort 28-40, 30-79

DecoupleEdgePorts 25-7

Deembed 22-10

DeembedBack 22-11

DeembedFront 22-13

Defeature 10-264

DefeatureObjects 28-300, 30-79

Definition Editor Script  
Commands 30-1

Definition Manager commands

AddDefinitionFromBlock 10-247,  
29-25

AddDefinitionFromLibFile 10-252

GetExtendedDefinitionObject 10-  
285

GetProjectMaterialNames 29-120

UpdateDefFromBlock 29-135, 29-  
137

Definition Manager Script Commands 29-1  
Delete 10-265, 24-9, 25-7, 25-8, 26-1, 27-1, 28-33, 28-33, 28-41, 28-41, 28-175, 28-175, 30-81, 30-81, 30-171  
Delete [Interface Port] 25-7  
DeleteAllReports 12-57  
DeleteDataset 5-33, 8-4, 9-39, 29-67  
DeleteDesign 5-34  
DeleteEmptyGroups 10-157  
DeleteFarFieldSetup 18-3  
DeleteFieldPlot 16-49  
DeleteFieldVariation 9-40  
DeleteFullVariation 9-41  
DeleteImportData 15-1  
DeleteLastOperation 10-158  
DeleteLinkedDataVariation 9-42  
DeleteMarker 12-56  
DeleteMarker[Fields Reporter] 16-49  
DeleteNamedExpr 16-50, 17-13  
DeleteNearFieldSetup 18-3  
DeleteOperation 10-159  
DeleteOutputVariable 9-42, 9-44, 11-3  
DeletePinGroup 28-31, 28-176, 30-82  
DeletePolylinePoint 10-105  
DeleteProbePortAndVia 25-8  
DeleteProject 3-12

DeleteReport 12-57  
DeleteRuleSet 31-5  
DeleteRun 31-5  
DeleteSetup 18-4  
DeleteSetups  
Optimetrics module command 14-8, 14-83  
DeleteSolutionVariation 9-43, 15-2  
DeleteSource [Interface Source] 25-8  
DeleteSweep 24-9  
DeleteTraceCharacteristics 12-58  
DeleteTraces 12-59  
DeleteUneditablePlot 16-51  
DeleteUserDefinedSolutions 21-3  
DelNetClass 28-177, 28-254, 30-80  
Design object commands 12-77, 12-78, 12-80, 12-80, 12-81, 12-96  
AddCartesianLimitLine 12-6  
AddCartesianXMarker 12-12  
AddCartesianYMarker 12-12  
AddDeltaMarker 12-14  
AddDesignVariablesForDynamicLink 9-10  
AddDynamicLink 9-11  
AddMarker 12-16  
AddMarkerToPlot 16-5  
AddNote 12-17  
AddTraceCharacteristics 12-19  
AddTraces 12-20  
AnalyzeDistributed 9-15

ApplyMeshOps 9-16, 9-76  
CalculatorRead 16-10, 17-5  
ClearAllMarkers 12-27  
ClearAllTraceCharacteristics 12-28  
CloseAllWindows 3-8  
CopyReportDefinition 12-30  
CreateOutputVariable 11-1  
CreateReportFromTemplate 12-54  
DeleteAllReports 12-57  
DeleteOutputVariable 9-42, 9-44, 11-3  
DeleteReport 12-57  
DeleteTraces 12-59  
DistributeAnalysis 9-45  
DoesOutputVariableExist 11-4  
Edit Layout 9-50  
EditMarker 12-63  
EditOutputVariable 11-5  
ExportConvergence 9-54  
ExportMeshStats 9-64  
ExportModelMeshToFile 10-185, 12-69  
ExportPlot3DToFile 12-70  
ExportPlotImageToFile 16-71  
ExportPlotImageWithViewToFile 16-72  
ExportProfile 9-75  
ExportReport 12-71  
ExportToFile 12-74  
Fast Transformation 9-134, 9-135  
GeometryCheckAndAutofix 10-268, 30-103  
GetDesignValidationInfo 9-83  
GetDisplayType 12-85  
GetLibraryDirectory 3-28  
GetMatchedObjectName 10-291  
GetModelUnits 10-292  
GetModule 9-89, 9-91  
GetName 9-92, 12-89, 14-33, 14-107  
GetObjectsByMaterial 10-298  
GetObjectsInGroup 10-299  
GetObjPath 9-94, 12-89, 14-34, 14-108  
GetOutputVariableValue 9-95, 11-8  
GetProjectDirectory 3-34  
GetProperties 5-62, 7-67, 9-98  
GetPropertyValue 5-63, 7-68, 9-100, 10-304, 12-90, 29-121  
GetPropEvaluatedValue 10-303  
GetPropNames 9-101, 18-7  
GetPropSIValue 10-307  
GetRegistryInt 3-36, 3-79  
GetRegistryString 3-37, 3-80  
GetSelections 10-309  
GetSolutionType 9-103  
GetSolveInsideThreshold 9-104  
GetSubGroupsInGroup 10-310  
GetTempDirectory 3-41

GetVariables 5-66, 7-70, 9-105  
GetVariableValue 5-66, 7-70, 9-105  
GetVariationVariableValue 9-106  
GetVersion 3-42  
HasDataset 8-8  
ImportIntoReport 12-113  
IsFeatureEnabled 3-43  
PasteDesign 9-112  
PasteReports 12-118  
PasteReportsWithLegacyNames 12-119  
PasteTracesWithLegacyNames 12-120  
Redo 9-117  
RenameDesignInstance 9-119  
RenameReport 12-121  
RenameTraces 12-122  
RunToolkit 9-122  
SARSetup 9-123  
SavePlotSettingsAsDefault 12-123  
SetActiveEditor 9-123  
SetAllowMaterialOverride 9-125  
SetDesignSettings 9-127  
SetLibraryDirectory 3-67  
SetProjectDirectoryVBCommand> 3-67  
SetPropertyValue 5-85, 7-71, 9-139, 29-134  
SetSolutionType 9-141  
SetSolveInsideThreshold 9-143  
SetTempDirectory 3-72  
SetVariableValue 5-87, 7-73, 9-145  
Show Layout 9-140  
ShowWindow 10-227  
Solve 9-145, 9-147  
StopSimLink 9-146  
UpdateAllReports 12-127  
Design Object Script Commands 9-1  
Design Object SCripting commands  
ExportNetworkData 9-65  
Design Verification Script Commands 31-1  
Desktop Commands For Registry Values 3-77  
Desktop object commands  
AddMessage 3-6  
AreSimulationsRunning 3-7  
ClearMessages 3-7, 5-29  
CloseProject 3-9  
CloseProjectNoForce 3-10  
Count 3-11  
DeleteProject 3-12  
DownloadJobResults 3-13  
EnableAutosave 3-16  
ExportOptionsFiles 3-17  
GetActiveProject 3-17  
GetActiveScheduler 3-18  
GetActiveSchedulerInfo 3-19

GetAutosaveEnabled	3-20	PauseRecording	3-50
GetBuildTimeDateString	3-21	PauseScript	3-51
GetChildNames	5-51, 9-77, 18-5	Print	3-52
GetChildNames Modeler	10-271	QuitApplication	3-53
GetChildTypes	5-53, 9-79, 10-278, 18-6	RefreshJobMonitor	3-53
GetChildTypes Optimetrics	14-32, 14-107	ResetLogging	3-54
GetCustomMenuSet	3-21	RestoreWindow	3-56
GetDesigns	5-56	ResumeRecording	3-57
GetDesktopConfiguration	3-24	RunProgram	3-58
GetDistributedAnalysisMachines	3-25	RunScript	3-59
GetDis-		SelectScheduler	3-62
trib-		SetActiveProject	3-63
utedAna-		SetActiveProjectByPath	3-64
lysisMachinesForDesignType	3-26	SetCustomMenuSet	3-65
GetMonitorData	3-31	SetDesktopConfiguration	3-66
GetPPELicensingEnabled	3-33	SetSchematicEnvironment	3-71
GetProjectList	3-35	Sleep	3-73
GetProjects	3-30, 3-35	StopSimulations	3-74
GetPropNames	5-60, 14-36, 14-110	SubmitJob	3-75
GetSchematicEnvironment	3-39	TileWindows	3-76
KeepDesktopResponsive	3-45	Desktop Object Script Commands	3-1
LaunchJobMonitor	3-45	Desktop Scripting Conventions	1-85
NewProject	3-46	DetachEdges	10-161
OpenMultipleProjects	3-48	DetachFaces	10-162
OpenProject	3-48	DisableDiffPairs	22-14
PageSetup	3-50, 28-322, 30-127, 30-237	Disconnect	28-33, 30-82
		DisplayPinNames	30-171
		Distance	28-52
		DistanceFromLine	28-53

- DistributeAnalysis 9-45  
DistributedAnalyzeSetup, Optimetrics module command 14-9, 14-83  
DoesNamedExpressionExists 16-51, 17-14  
DoesOutputVariableExist 11-4  
DoesSupportTraceCharacteristics 12-60  
Draw Menu commands 10-4  
DumpAllReportsData 12-61  
Duplicate 28-178, 28-301, 30-83, 30-230  
DuplicateAcrossLyrs 28-179, 28-245, 30-83  
DuplicateAlongLine 10-107  
DuplicateAroundAxis 10-109  
DuplicateMirror 10-112  
DynamicMeshOverlays 24-9
- E**
- Edit 9-48, 19-5, 19-10, 19-15, 19-22, 24-10, 24-10, 25-8, 25-8, 26-2, 27-2, 28-34, 28-179, 29-69, 29-107, 29-169, 29-242, 29-282, 29-316, 29-353, 30-84  
Edit Menu Commands 10-104  
Edit3DComponent 10-74  
Edit3DComponentDefinition 10-76, 28-302, 30-85  
EditAntennaArraySetup 18-56  
EditAntennaOverlay 18-14  
EditBoxSetup 18-16
- EditCartesianXMarker 12-62  
EditCartesianYMarker 12-62  
EditCircuitExcitations 25-9  
EditCircuitPort [Interface Port] 25-9  
EditCoSimulationOptions 9-46  
EditDataset 5-36, 8-5, 29-95  
EditEntityList 10-164  
EditFaceCS 10-165  
EditFarFieldSphereSetup 18-18  
EditImportData 9-47  
EditInfiniteArray 9-48  
EditInfiniteSphereSetup 18-21  
EditLayoutForLayoutComponent 9-50  
EditLineSetup 18-23  
EditMarker 12-63  
EditMaterial 5-38, 29-98  
EditNearFieldBoxSetup 18-24  
EditNearFieldLineSetup 18-26  
EditNearFieldRectangleSetup 18-28  
EditNearFieldSphereSetup 18-29  
EditNotes 9-50, 9-51  
EditObjectCS 10-171  
EditOptions 9-52  
Editor object commands  
    SetPropertyValue 5-85, 7-71, 9-139, 29-134  
EditOutputVariable 11-5  
EditPolyline 10-79  
EditRadFieldSourceGroup 18-31

EditRectangleSetup	18-33	EnterScalar	16-62, 17-21
EditRelativeCS	10-178	EnterScalarFunc	16-63, 17-22
EditRuleSet	31-6	EnterSurf	16-64, 17-22
EditRun	31-7	EnterVector	16-64, 17-23
EditSetup	14-9, 14-84	EnterVectorFunc	16-65, 17-24
	optimization command 14-162	EnterVol	16-66, 17-24
	parametric command 14-154	EraseMeasurements	28-303, 30-87, 30-232
	sensitivity command 14-179	Event Callback Scripting	1-89
	statistical command 14-190	Example Scripts	33-1
EditSolverOnDemandModel	29-196	Excitations Commands	25-1
EditSources	15-3	ExecuteScript	7-37
EditSphereSetup	18-34	Export	10-180, 29-115, 29-117, 29-119, 29-208, 29-260, 29-293, 29-331, 29-365
EditSurfaceMeshSummaryData	16-52	ExportCableLibrary	10-342
EditSweep	24-10	ExportCitiFile	22-16
EditUserDefinedSolution	21-4	ExportConvergence	9-54
EditWithComps	29-197, 29-242, 29-282, 29-323, 29-353	ExportDataset	5-47, 8-7, 9-55, 29-114
EMDesignOptions	9-52	ExportDXConfigFile, Optimetrics module command	14-22, 14-97
EnableAutoSave	3-16	ExportDXF	28-303, 30-89
EnableDiffPairs	22-15	ExportEigenmodes	15-5
EnableSetup	14-22, 14-96	ExportEignemodes	15-5
EnterComplex	16-56, 17-15	ExportElementPatternToFile	18-62
EnterComplexVector	16-57, 17-16	ExportEyeMaskViolation	12-64
EnterCoord	16-58, 17-17	ExportFieldPlot	16-66
EnterEdge	16-59, 17-17	ExportFieldsToFile	18-63
EnterLine	16-59, 17-18	ExportFile	30-172
EnterOutputVar	16-60, 17-19	ExportForHSpice	9-56, 15-6
EnterPoint	16-61, 17-19		
EnterQty	16-61, 17-20		

- ExportForHSpice (Layout Editor) 9-57  
ExportForSpice (Layout Editor) 9-59  
ExportFullWaveSpice 22-18, 29-300  
ExportGDSII 28-306, 30-92  
ExportGerber 28-308, 30-94  
ExportImageToFile 12-65  
ExportMarkerTable 16-67  
ExportMaterial 5-48, 29-116  
ExportMatlab 22-19  
ExportMatrixData 9-61  
ExportMesh Stats 9-64  
ExportModelMeshToFile 10-185, 12-69  
ExportNCDrill 28-311, 30-97  
ExportNMFDATA 15-10  
ExportOptimetricsProfile, Optimetrics module command 14-23, 14-98  
ExportOptimetricsResults, Optimetrics module command 14-24, 14-99  
ExportOptionsFiles 3-17  
ExportOutputVariables 11-6  
ExportParametersToFile 18-65  
ExportParametricResults, Optimetrics module command 14-25, 14-100  
ExportPlot3DToFile 12-70  
ExportPlotImageToFile 16-71  
ExportPlotImageWithViewToFile 16-72  
ExportProfile 9-75  
ExportRadiationFieldsToFile 18-67  
ExportRadiationParametersToFile 18-69  
ExportReport 12-71  
ExportReportDataToFile 12-73  
ExportRespSurfaceMinMaxTable 14-27, 14-102  
ExportRespSurfaceRefinePoints 14-28, 14-102  
ExportRespSurfaceResponsePoints 14-29, 14-103  
ExportRespSurfaceVerificationPoints 14-30, 14-104  
ExportScript 29-118, 29-338  
ExportSpreadsheet 22-29  
ExportSurfaceMeshSummary 16-73  
ExportTableToFile 12-73  
ExportToFile 16-75  
ExportToHFSS 24-10, 30-99  
ExportToQ3D 24-11, 30-100  
ExportTouchstone 22-31  
ExportTouchstone2 22-34  
ExportTransientData 15-11  
ExportUniformPointsToFile 12-75  
Extract 22-36

**F**

FFTOnReport 15-12

Field Calculator commands	EnterScalar 16-62, 17-21
DoesNamedExpressionExists 16-51, 17-14	EnterScalarFunc 16-63, 17-22
Field Overlay module commands, GetFieldPlotNames 16-77	EnterSurf 16-64, 17-22
field overlay script commands 16-1	EnterVector 16-64, 17-23
Field Overlays module commands	EnterVectorFunc 16-65, 17-24
AddNamedExpr 16-6, 17-3	EnteVol 16-66, 17-24
CalcOp 16-8, 17-4	ExportFieldPlot 16-66
CalcStack 16-8, 17-6	ExportOnGrid 16-68
ChangeGeomSettings 16-12, 17-9	ExportSurfaceMeshSummary 16- 73
ClcMaterial 16-14, 17-11	GetFieldFolderNames 16-76
ClearAllNamedExpr 16-16, 17- 12	GetFieldPlotQuantityName 16-77
CopyNamedExprToStack 16-16, 17-13	GetMeshPlotNames 16-78, 24-11
CreateFieldPlot 16-17	HidePolarPlot 16-80
DeleteFieldPlot 16-49	HideRadiatedPlotOverlay 16-81
DeleteNamedExpr 16-50, 17-13	ModifyFieldPlot 16-83
DeleteUneditablePlot 16-51	ReassignFieldPlot 16-85
EditSurfaceMeshSummaryData 16-52	RenameFieldPlot 16-88
EnterComplex 16-56, 17-15	RenamePlotFolder 16-89
EnterComplexVector 16-57, 17- 16	SaveFieldsPlots 16-90
EnterCoord 16-58, 17-17	SetFieldPlotSettings 16-92
EnterEdge 16-59, 17-17	SetPlotFolderSettings 16-95
EnterLine 16-59, 17-18	UpdateAllFieldsPlots 16-101
EnterOutputVar 16-60, 17-19	UpdateQuantityFieldsPlots 16-101
EnterPoint 16-61, 17-19	Fields Calculator commands
EnterQty 16-61, 17-20	AddNamedExpr 16-6, 16-7, 17-2, 17-3
	CalcOp 16-8, 17-4
	CalcStack 16-8, 17-6

CalculatorWrite 16-11, 17-7  
CalcWrite 16-9, 17-8  
ChangeGeomSettings 16-12, 17-9  
ClcEval 16-13, 17-10  
ClcMaterial 16-14, 17-11  
ClcMaterialValue 16-14, 17-11  
ClearAllNamedExpr 16-16, 17-12  
CopyNamedExprToStack 16-16, 17-13  
DeleteNamedExpr 16-50, 17-13  
EnterComplex 16-56, 17-15  
EnterComplexVector 16-57, 17-16  
EnterCoord 16-58, 17-17  
EnterEdge 16-59, 17-17  
EnterLine 16-59, 17-18  
EnterOutputVar 16-60, 17-19  
EnterPoint 16-61, 17-19  
EnterQty 16-61, 17-20  
EnterScalar 16-62, 17-21  
EnterScalarFunc 16-63, 17-22  
EnterSurf 16-64, 17-22  
EnterVector 16-64, 17-23  
EnterVectorFunc 16-65, 17-24  
EnterVol 16-66, 17-24  
ExportOnGrid 17-25  
ExportToFile 17-28  
**Fields Calculator Script Commands** 17-1

Fields reporter module commands  
AddMarker[Fields Reporter] 16-4  
ClearAllMarkers[Fields Reporter] 16-15  
DeleteMarker[Fields Reporter] 16-49  
ExportMarkerTable 16-67  
Fillet 10-186  
FilterObjectList 28-44, 28-186, 30-100  
FindObjects 28-43, 28-187  
FindObjectsByPoint 28-47, 28-188, 30-101  
FindObjectsByPolygon 28-46, 28-189, 30-102  
FlattenGroup 10-188  
FlipHorizontal 28-37, 30-102, 30-173, 30-232  
FlipVertical 28-38, 30-103, 30-173, 30-233  
footprint  
editor scripts 30-191  
manager script commands 29-219  
For...Next loop 1-9  
functions  
VBScript procedures 1-10

**G**

GenerateAllUserDefinedModels 10-266  
GenerateHistory 10-189  
GenerateUserDefinedModel 10-267  
GenerateVariationData Parametric, parametric command 14-31, 14-105, 14-156

GeometryCheckAndAutofix	10-268, 30-103	GetAllReportNames	12-79
Get3DComponentDefinitionNames	10-83	GetAllSolutionNames	24-11
Get3DComponentInstanceNames	10-84	GetAllSourceMagnitudes	15-15
Get3DComponentMaterialNames	10-84	GetAllSourceModes	15-15
Get3DCom-		GetAllSourcePhases	15-16
ponentMaterialProperties	10-85	GetAllSources	15-17
Get3DComponentParameters	10-86	GetAntennaParameters	15-17
Get3DComponentPartNames	10-86	GetApplication	7-38
GetActiveCoordinateSystem	10-190	GetArrayVariables	5-50, 7-66
GetAct-		GetAutoSaveEnabled	3-20
iveCoordin-		GetAvailableDisplayTypes	12-80
ateSystemTransform	10-190	GetAvailableReportTypes	12-81
GetActiveDesign	5-49	GetAvailableSolutions	12-81
GetActiveEditor (Layout Editor)	9-77	GetAvailableVariations	15-18
GetActiveProject	3-17	GetBBox	28-46, 28-190, 30-105
GetActiveScheduler	3-18	GetBodyNamesByPosition	10-270
GetActiveSchedulerInfo	3-19	GetBoundingCircleCenter	28-62
GetAdaptiveFreq	15-13	GetBoundingCircleRadius	28-62
GetAdaptiveSettings	15-14	GetBuildTimeString	3-21
GetAllBoundariesList	25-12	GetCallback	7-38
GetAllCategories	12-77	GetChangedProperty	7-39
GetAllLayerNames	28-190, 28-245, 30-105, 30-233	GetChild Object (Report Setup), Report module command	12-82
GetAllPortsList	25-12	GetChildNames	5-51, 9-77, 18-5
GetAllQuantities	12-78	GetChildNames (Optimetrics), Report module command	12-82
		GetChildNames Modeler	10-271
		GetChildObject Design	5-51, 9-78
		GetChildObject Modeler	10-275

GetChildObject Radiation 18-6  
GetChildTypes 5-53, 9-79, 10-278, 18-6  
GetChildTypes Optimetrics 14-32, 14-107  
GetChildTypes ReportSetup 12-84  
GetClosestPoint 28-64  
GetClosestPoints 28-64  
GetComplInstanceFromRefDes 28-314, 30-106  
GetComponentInfo 28-313, 30-106  
GetComponentName 64  
GetComponentPinInfo 28-316, 30-107  
GetComponentPins 28-315, 30-108  
GetCoordinateSystems 10-191  
GetCSObjects  
    Coordinate Systems 28-45, 28-78, 28-191, 30-105  
GetCurvePropServerName 12-84  
GetData 29-209, 29-261, 29-265, 29-295, 29-332, 29-366  
GetDataExpressions 12-99  
GetDataUnits 12-100  
GetDefinitionManager 5-54  
GetDependentFiles 5-54  
GetDescription 7-40  
GetDesign 7-41, 7-41  
GetDesignID 9-81  
GetDesignName 9-81  
GetDesigns 5-56  
GetDesignType 9-82  
GetDesignValidationInfo 9-83  
GetDesignVariableNames 12-100  
GetDesignVariableUnits 12-101  
GetDesignVariableValue 12-102  
GetDesignVariationKey 12-103  
GetDisplayType 12-85  
GetDistributedAnalysisMachines 3-25  
GetDis-  
trib-  
utedAna-  
lysisMachinesForDesignType 3-26  
GetDocumentNames 20-8  
GetDynLinkIntrinsicVariables 12-86  
GetDynLinkQtyValueState 12-86  
GetDynLinkTraces 12-87  
GetDynLinkVariableValues 12-88  
GetEdgeByPosition 10-280  
GetEdgeIDFromNameForFirstOperation 10-282  
GetEdgeIDsFromFace 10-282  
GetEdgeIDsFromObject 10-283  
GetEdgeLength 10-284  
GetEdgePositionAtNormalizedParameter 9-84  
GetEditor 7-41, 9-85  
GetEditor (Layout Editor) 9-85  
GetEditorName [Layout] 28-191, 28-317, 30-109  
GetEntityListIDByName 10-284

---

GetEvaluatedText 7-42  
GetExeDir 3-27  
GetExtendedDefinitionObject 10-285  
GetFaceArea 10-286  
GetFaceByPosition 10-287  
GetFaceCenter 10-287  
GetFaceIDFromNameForFirstOperation 10-289  
GetFaceIDs 10-290  
GetFaceIDsOfSheet 10-290  
GetFieldFolderNames 16-76  
GetFieldPlotNames, Field Overlay module command 16-77  
GetFieldPlotQuantityName 16-77  
GetFieldType 15-19  
GetFileName 7-42  
GetFrequencies 22-37  
GetFrequencyCount 22-38  
GetGeo-  
    metryId-  
    sForAllNetLayerCombinations 9-87  
GetGeo-  
    metryId-  
    sForNetLayerCombinations 9-86  
GetHidden 7-43  
GetHoles 28-59  
GetImageDataValues 12-104  
GetIncludePortPostProcessing 15-20  
GetInstanceID 65  
GetInstanceName 66  
GetIntersectionType 28-63  
GetLayerInfo 28-191, 28-246, 30-233  
GetLibraryDirectory 3-28  
GetMatchedObjectName 10-291  
GetMaterialList 28-194, 28-317, 30-110  
GetMeshOperations 24-11  
GetMeshPlotNames 16-78  
GetModelBoundingBox 10-292  
GetModelUnits 10-292  
GetModule 9-89, 9-90, 9-91  
GetMonitorData 3-31  
GetMultipactionBreakdown 15-21  
GetName 5-58, 9-92, 9-93, 12-89, 14-33, 14-107, 22-39  
GetNames 29-210, 29-262, 29-266, 29-296, 29-333, 29-367  
GetNetClasses 28-194, 28-255, 30-110  
GetNetClassNets 28-195, 28-255, 30-110  
GetNetConnections 28-255, 30-111  
GetNetworkDataSolution 15-21  
GetNetworkDataSolutionDefinition 15-22  
GetNetworkPostprocSetup 15-23  
GetNPortData 29-214  
GetNumObjects 10-293  
GetObjectIDByName 10-294  
GetObjectName 10-294

GetObjectByNameByEdgeID 10-295  
GetObjectByNameByFacetID 10-296  
GetObjectByNameByID 10-296  
GetObjectByNameByVertexID 10-297  
GetObjectsByMaterial 10-298  
GetObjectShapeType 10-298  
GetObjectsInGroup 10-299  
GetObjectVolume 10-300  
GetObjPath 5-59, 9-94, 10-301, 12-89, 14-34, 14-108  
GetOptimetricsResults, Optimetrics module command 14-35  
GetOutputVariableValue 9-95, 11-8  
GetParentDesign 66  
GetPartsForUserDefinedModel 10-301  
GetPath 5-59  
GetPer-  
    Quant-  
        ityPrimarySweepValues 12-105  
GetPersonalLibDirectory 3-32  
GetPoints 10-302, 28-58  
GetPolygon 28-45, 28-196, 30-112  
GetPolygonDef 28-46, 28-196, 30-112  
GetPortCount 22-40  
GetPortInfo 28-197, 28-280, 30-112  
GetPortNumber 22-41  
GetPostProcSettings 22-42  
GetPPELicensingEnabled 3-33  
GetProcessID 3-33  
GetProgress 7-44, 7-44  
GetProjectDirectory 3-34  
GetProjectList 3-35  
GetProjectMaterialNames 29-120  
GetProjects 3-30, 3-35  
GetProperties 5-62, 7-67, 9-98, 28-200, 28-318, 29-268, 30-115, 30-174, 30-236  
GetPropertyValue 5-63, 7-68, 9-100, 10-304, 12-90, 28-200, 28-318, 29-121, 30-116, 30-174  
GetPropEvaluatedValue 10-303  
GetPropHost 7-44, 7-44, 9-98, 9-98, 67  
GetPropNames 5-60, 9-101, 12-92, 14-36, 14-110, 18-7  
GetPropNames Modeler 10-305  
GetPropServerName 67  
GetPropServers 7-45  
GetPropSIValue 10-307  
GetPropTabType 7-46  
GetPropValue Modeler 10-308  
GetPropValue Optimetrics 14-37, 14-110  
GetPropValue Project 5-61, 9-102, 18-7  
GetPropValue Reporter 12-92  
GetQtyExpressionsForSourceTrace 12-93  
GetReadOnly 7-46  
GetRealDataValues 12-106  
GetRegistryInt 3-36, 3-79  
GetRegistryString 3-37, 3-80

GetRelativeCoordinateSystems 10-309  
GetReportSummaryForRegressionTesting 12-94  
GetReportTraceNames 12-94  
GetRunStatus 7-47  
GetSelections 9-103, 10-309, 28-201, 28-318, 30-116  
GetSetupData 24-12  
GetSetupNames 18-5, 18-8  
GetSetupNames (Optimetrics), Optimetrics module command 14-31, 14-32, 14-38, 14-105, 14-106, 14-111  
GetSetupNamesByType (Optimetrics), Optimetrics module command 14-38, 14-112  
GetSolutionContexts 12-95  
GetSolutionDataPerVariation 12-96  
GetSolutionType 9-103  
GetSolutionVariation 22-43  
GetSolutionVersionID 15-23  
GetSolveInsideThreshold 9-104  
GetSolveRangeInfo 15-24  
GetSolverOnDemandData 29-214  
GetSolverOnDemandModelList 29-214  
GetSourceContexts 15-25  
GetSourceData 15-25  
GetStackupLayerNames 28-201, 28-248, 30-236  
GetSubGroupsInGroup 10-310  
GetSweepNames 12-107  
GetSweepUnits 12-108  
GetSweepValues 12-109  
GetSysLibDirectory 3-40  
GetTabTypeName 7-48  
GetTempDirectory 3-41  
GetTerminalExcitationType 15-26  
GetText 7-48  
GetTool 3-84  
GetTopDesignList 5-65  
GetTopEntryValue 16-79, 17-31  
GetTransientSolveTimes 15-26  
 GetUserDefinedSolutionNames 21-6  
 GetUserDefinedSolutionProperties 21-7  
 GetUserLibDirectory 3-42  
 GetUserPosition 10-311  
 GetValidISolutionList 15-27  
 GetValue 7-49  
 GetVariables 5-66, 7-70, 9-105  
 GetVariableValue 5-66, 7-70, 9-105  
 GetVariation 22-44  
 GetVariationVariableValue 9-106  
 GetVersion 3-42  
 GetVer-  
 texIDFromNameForFirstOperation 10-312  
 GetVertexIDsFromEdge 10-312  
 GetVertexIDsFromFace 10-313  
 GetVertexIDsFromObject 10-314

GetVertexPosition 10-314  
GetVisibleLevels 30-175  
GetWireBodyNames 10-315  
GetX (Layout Editor) 28-50  
GetY 28-50  
GridSetup 30-175  
Group 28-74, 30-116  
GroupPlotCurvesByGroup-  
ingStrategy 12-112

**H**

HasArcs 28-59  
HasDataset 8-8  
HasFields 15-28  
HasHoles 28-59  
HasMatrixData 15-28  
HasSameData 22-45  
HasSelfIntersections 28-60  
Heal (Layout Editor) 28-319, 30-  
117  
HealObject 10-192  
HideAntennaParametersOverlay  
16-80  
HidePolarPlot 16-80  
HideRadiatedPlotOverlay 16-81  
hierarchy of variables in HFSS 1-76  
HighlightNet 28-202, 28-256, 30-  
120

**I**  
Icepak boundary condition commands  
ImportIDX 3-98  
If...Then... Else statement 1-8  
Import (3D Modeler command) 10-196  
ImportANF 3-84, 3-86  
ImportAutoCAD 3-88  
ImportAWRMicrowaveOffice 3-88  
ImportCableLibrary 10-342  
ImportDataset 5-67, 8-9, 9-107, 29-122  
ImportDXF 10-200, 30-121  
ImportEDB 3-90  
ImportExport Tool 3-84  
ImportExtracta 3-91  
ImportFromClipboard 10-204  
ImportGDSII 3-92, 3-93, 10-205, 30-125  
ImportIDF 3-94, 3-97  
ImportIDX 3-98  
ImportIntoReport 12-113  
ImportIPC 3-101  
ImportODB 3-102  
ImportOutputVariables 11-9  
ImportReportDataIntoReport 12-114  
ImportSetup, Optimetrics module com-  
mand 14-39, 14-113  
ImportSolution 15-30  
ImportTable 15-31  
ImportXFL 3-103

Imprint 10-208  
ImprintProjection 10-209  
include files  
    scripts 1-11  
indentation, IronPython 1-21  
InputBox function 1-11  
Insert3DComponent 10-87  
InsertBoxSetup 18-37  
InsertComponent 10-88  
InsertDesign 5-69, 5-71, 9-109  
InsertDesignWithWorkflow 5-71  
InsertFarFieldSphereSetup 18-39  
InsertInfiniteSphereSetup 18-41  
InsertLineSetup 18-43  
InsertNearFieldBoxSetup 18-45  
InsertNearFieldLineSetup 18-47  
InsertNearFieldRectangleSetup 18-48  
InsertNearFieldSphereSetup 18-50  
InsertRectangleSetup 18-52  
InsertSetup 14-41, 14-114  
    Analysis module command 14-41, 14-114  
    optimization command 14-167  
    parametric command 14-80, 14-153, 14-156, 14-161  
    sensitivity command 14-186  
    statistical command 14-193  
InsertSphereSetup 18-53  
Intersect 10-211, 28-65, 28-65, 28-67, 28-67, 28-204, 28-204, 30-126, 30-126, 30-236  
IronPython 1-16  
    indentation in 1-21  
IsArc 28-51, 28-51, 28-61, 28-61  
IsBox 28-62  
IsCircle 28-62  
IsClosed 28-56  
IsConvex 28-61  
IsDataComplex 12-110  
IsEqual 28-51  
IsFeatureEnabled 3-43  
IsFieldAvailableAt 15-32  
IsParametric 28-60  
IsPerQuantityPrimarySweep 12-110  
IsPoint 28-61  
IsSegment 28-61  
IsUsed 29-215, 29-262, 29-268, 29-296, 29-333, 29-367  
IsValueConstant 7-50

**K**

KeepDesktopResponsive 3-45  
keywords, VBScript 1-12, 1-14

**L**

LaunchJobMonitor 3-45  
Layers Methods 28-227  
layout  
    script interrogation methods 28-41

Layout Interrogation 28-41  
Layout Scripting 1-89, 28-1, 28-79, 28-85, 28-104, 28-285  
Layout3DMeshOverlay 24-13  
LayoutMeshOverlay 24-13  
ListMatchingVariations 15-33  
ListValuesOfVariable 15-34  
ListVariations 15-35, 24-13  
LoadNamedExpressions 16-82, 17-32  
LoadSolution 22-47  
logical operators 1-8  
looping through code  
    Do ... Loop 1-9  
    For ... Next 1-9

**M**

Mag (Layout Editor) 28-52  
material commands  
    AddDefinitionFromBlock 10-247, 29-25  
    AddDefinitionFromLibFile 10-252  
    AddMaterial 5-7, 29-30, 29-68  
    AreMaterialPropertiesEqual 29-51  
    AreSurfaceMaterialPropertiesEqual 29-52  
    CloneMaterial 5-30, 29-66  
    EditMaterial 5-38, 29-98  
    ExportMaterial 5-48, 29-116

GetExtendedDefinitionObject 10-285  
RemoveMaterial 5-75, 29-128  
UpdateDefFromBlock 29-135  
UpdateDefFromBlockEx 29-137  
Materials Scripting Support 6-23  
Microsoft  
    VBScript user's guide 1-12  
Mirror 10-115  
MirrorX 28-58  
Model Manager Script Commands 29-270  
Modeler Menu Commands 10-124  
ModifyFieldPlot 16-83  
ModifyLibraries 29-339  
ModifyNetClass 28-205, 28-257, 30-127  
ModifyPMLGroup 13-5  
module script commands  
    field overlay 16-1  
    radiation 18-2  
ModuleHasMesh 15-29  
modules in HFSS scripting 1-76  
Move 10-117, 28-38, 28-38, 28-38, 28-52, 28-52, 28-52, 28-57, 28-57, 28-57, 28-57, 28-206, 28-206, 28-206, 30-127, 30-127, 30-127, 30-176, 30-237  
MoveCSToEnd 10-213  
MoveEntityToGroup 10-214  
MoveFaces 10-215  
MovePlotCurvestoGroup 12-114  
MovePlotCurvestoNewGroup 12-115  
MsgBox function 1-11

MultipleEdit 26-2

## N

Named Arguments 1-86

NdExplorer

    Script Commands 22-1

NetClass Methods 28-252

Network Data Explorer Manager  
    Commands 29-299

NewProject 3-46

Normalize 28-53

## O

oAnsoftApp object 1-76

Object Identifiers and Script Recording 28-2

Object Movement and Modification  
    Methods 28-31

Object Oriented Property  
    Scripting 6-1

oDesign object 1-76

oDesktop object 1-76

oEditor object 1-76

OffsetFaces 10-119

oModule object 1-76

Open 22-48

OpenAndConvertProject 3-47

OpenExternalEditor 10-316

OpenMultipleProjects 3-48

OpenProject 3-48, 30-237

OpenWindowForAllReports 12-116

OpenWindowForReports 12-117

operators

    arithmetic 1-6

    categories in VBScript 1-5

    comparison 1-7

    concatenation 1-7

    logical 1-8

    precedence of 1-5

oProject object 1-76

Optimetrics module commands

    DeleteSetups 14-8, 14-83

    DistributeAnalyzeSetup 14-9, 14-83

    EnableSetup 14-22, 14-96

    ExportDXConfigFile 14-22, 14-97

    ExportOptimetricsProfile 14-23,  
        14-98

    ExportOptimetricsResults 14-24,  
        14-99

    ExportParametricResults 14-25,  
        14-100

    ExportRespSurfaceMinMaxTable  
        14-27, 14-102

    ExportRespSurfaceRefinePoints  
        14-28, 14-102

    ExportRespSurfaceResponsePoints  
        14-29, 14-103

    ExportRespSurfaceVerificationPoints  
        14-30, 14-104

    GetChildNames 14-31, 14-32, 14-106, 14-106

    GetOptimetricResult 14-34, 14-

109	OverlayCurrents 9-111
GetOptimetricsResults 14-35	OverlayFarField 9-111
GetSetupNames 14-38, 14-111	OverlayMesh 9-112
GetSetupNamesByType 14-39, 14-112	OverlayNearField 9-112
ImportSetup 14-39, 14-113	<b>P</b>
RenameSetup 14-77, 14-150	Padstack Manager Script Commands 29- 309
SolveSetup 14-7, 14-78, 14-79, 14-82, 14-152, 14-152	PageSetup 10-316
Optimetrics Module Script Com- mands 14-1	PageSetup (Layout Editor) 3-50, 28-322, 30-127
optimization commands	Pan 30-177
EditSetup 14-162	parametric commands
InsertSetup 14-167	EditSetup 14-154
Optimization Script Commands 14- 162	GenerateVariationData Parametric 14- 31, 14-105, 14-156
Other Creation Methods 28-17	InsertSetup 14-80, 14-153, 14-156, 14- 161
Other oEditor Commands 10-244	Parametric Script Commands 14-154
output variable commands	Paste 5-73, 10-338, 10-340, 28-38, 28- 206, 30-178
CreateOutputVariable 11-1	Paste (Model Editor) 10-120
DeleteOutputVariable 9-42, 9- 44, 11-3	Paste (Project Object) 5-73
DoesOutputVariableExist 11-4	PasteDesign 9-112
EditOutputVariable 11-5	PasteItemCommand 9-114
GetOutputVariableValue 9-95, 11-8	PastePlotSettings 12-118
Output variable commands	PasteRadFieldSetup 18-8
ExportOutputVariables 11-6	PasteReports 12-118
ImportOutputVariables 11-9	PasteReportsWithLegacyNames 12-119
Output Variable Script Commands 11-1	

PasteSetup	CutDesign 5-32
Optimetrics module command 14-76, 14-149	DeleteDataset 5-33, 8-4, 9-39, 29-67
PasteTraces 12-120	DeleteDesign 5-34
PasteTracesWithLegacyNames 12-120	EditDataset 5-36, 5-47, 5-67, 8-5, 8-7, 8-9, 9-55, 9-107, 29-95, 29-114, 29-122
PauseRecording 3-50	EditMaterial 5-38, 29-98
PauseScript 3-51	ExportMaterial 5-48, 29-116
PMLGroupCreated 13-7	GetActiveDesign 5-49
PMLGroupModified 13-8	GetArrayVariables 5-50, 7-66
Point (Layout Editor) 28-42, 28-206, 30-128	GetChildObject Design 5-51, 9-78
Point Object 28-48	GetChildObject Modeler 10-275
PointInPolygon 28-63	GetChildObject Radiation 18-6
Polygon 28-42, 28-207, 30-129	GetDependentFiles 5-54
Polygon Object 28-54	GetDesign 7-41
Port Methods 28-260	GetDesignID 9-81
PositionRelative 28-75, 30-129	GetDesignName 9-81
Print 3-52	GetDesignType 9-82
Project object commands	GetName 5-58
AddDataset 5-4, 8-1, 9-7, 29-22	GetObjPath 5-59
AddMaterial 5-7, 29-30, 29-68	GetPath 5-59
AreMaterialPropertiesEqual 29-51	GetProjectMaterialNames 29-120
AreSurfaceMaterialPropertiesEqual 29-52	GetProperties 5-62, 7-67, 9-98, 28-200, 28-318, 30-115, 30-174, 30-236
ChangeProperty 5-16, 29-53, 30-149	GetPropertyValue 5-63, 7-68, 9-100, 10-304, 12-90, 28-200, 28-318, 29-121, 30-116, 30-174
CloneMaterial 5-30, 29-66	GetPropEvaluatedValue 10-303
Close 5-31	GetPropNames Modeler 10-305
CopyDesign 5-32	

GetPropSIValue 10-307  
GetPropvalue Modeler 10-308  
GetPropvalue Optimetrics 14-37, 14-110  
GetPropvalue Project 5-61, 9-102  
GetPropvalue Radiation 18-7  
GetPropvalue Reporter 12-92  
GetSolutionVariation 22-43  
GetTopDesignList 5-65  
GetTopEntryValue 16-79, 17-31  
GetVariables 5-66, 7-70, 9-105  
GetVariableValue 5-66, 7-70, 9-105  
InsertDesign 5-69, 9-109  
LoadSolution 22-47  
Paste 5-73, 5-73  
Redo 5-73  
RemoveAllUnusedDefinitions 5-74  
RemoveMaterial 5-75, 29-128  
RemoveUnusedDefinitions 5-76, 29-132  
Rename 5-77  
RestoreProjectArchive 3-55  
Save 5-78  
SaveAs 5-79  
SaveAsStandAloneProject 5-81  
SaveProjectArchive 5-82  
SetActiveDesign 5-84  
SetPropertyValue 5-85, 7-71, 9-139, 28-223, 28-327, 29-134, 30-241  
SetPropValue Design 9-138, 18-10  
SetPropValue Optimetrics 14-77, 14-151  
SetPropValue Project 5-84, 12-125  
SetVariableValue 5-87, 7-73, 9-145  
SimulateAll 5-11, 5-88  
Undo 5-89  
UpdateDefinitions 5-89, 29-139  
Project Object Script Commands 5-1  
ProjectSheet 10-217  
property commands  
    ChangeProperty 5-16, 29-53, 30-149  
    GetArrayVariables 5-50, 7-66  
    GetProjectMaterialNames 29-120  
    GetProperties 5-62, 7-67, 9-98, 28-200, 28-318, 30-115, 30-174, 30-236  
    GetPropertyValue 5-63, 7-68, 9-100, 10-304, 12-90, 28-200, 28-318, 29-121, 30-116, 30-174  
    GetPropEvaluatedValue 10-303  
    GetPropSIValue 10-307  
    GetTopEntryValue 16-79, 17-31  
    GetVariables 5-66, 7-70, 9-105  
    GetVariableValue 5-66, 7-70, 9-105  
    SetPropertyValue 5-85, 7-71, 9-139, 28-223, 28-327, 29-134, 30-241  
    SetVariableValue 5-87, 7-73, 9-145  
Property Script Commands 7-1  
Conventions used in this Chapter 7-13

---

Object Script Property Function Summary	7-3	EditInfiniteSphereSetup	18-21
PropertyExists	7-50	EditLineSetup	18-23
PropHost Functions	7-28	EditNearFieldBoxSetup	18-24
PurgeHistory	10-218	EditNearFieldLineSetup	18-26
PushExcitations	9-114, 28-323, 30-132	EditNearFieldRectangleSetup	18-28
Layout Editor	9-114, 28-323, 30-132	EditNearFieldSphereSetup	18-29
<b>Q</b>		EditRadFieldSourceGroup	18-31
QuitApplication	3-53	<b>EditRa-</b> diatedPowerCalculationMethod 18-5	
<b>R</b>		EditRectangleSetup	18-33
Radiation boundary		EditSphereSetup	18-34
Infinite Sphere	19-1, 19-5	ExportElementPatternToFile	18-62
Near Field Line	19-8, 19-10	ExportFieldsToFile	18-63
Near Field Plane	19-12, 19-15	ExportParametersToFile	18-65
Near Field Sphere	19-18, 19-22	ExportRadiationFieldsToFile	18-67
Radiation module commands		ExportRadiationParametersToFile	18-69
AddAntennaOverlay	18-11	GetSetupNames	18-8
AddRadFieldSourceGroup	18-13	InsertBoxSetup	18-37
CopyRadFieldSetup	18-2	InsertFarFieldSphereSetup	18-39
DeleteFarFieldSetup	18-3	InsertInfiniteSphereSetup	18-41
DeleteNearFieldSetup	18-3	InsertLineSetup	18-43
DeleteSetup	18-4	InsertNearFieldBoxSetup	18-45
EditAntennaArraySetup	18-56	InsertNearFieldLineSetup	18-47
EditAntennaOverlay	18-14	InsertNearFieldRectangleSetup	18-48
EditBoxSetup	18-16	InsertRectangleSetup	18-52
EditFarFieldSphereSetup	18-18		

InsertSphereSetup 18-50, 18-53  
PasteRadFieldSetup 18-8  
RenameSetup 18-9  
Radiation Module Script  
    Commands 18-1, 19-1  
radiation script commands 18-2  
ReassignFieldPlot 16-85  
RecalculatePMLMaterials 13-10  
Record Script and Edit  
    Properties 7-74  
Redo 9-118, 30-179  
    design-level command 9-117  
    project-level command 5-73  
references, for VBScript 1-12  
RefreshJobMonitor 3-53  
RefreshMeshOverlays 24-13  
ReleaseData 12-111  
Remove 29-125, 29-127, 29-129,  
    29-131, 29-215, 29-263, 29-  
    297, 29-334, 29-368  
RemoveAllUnusedDefinitions 5-74  
RemoveBadEdges 10-318  
RemoveBadFaces 10-319  
RemoveBadVertices 10-320  
RemoveCables 10-343  
RemoveImportData 9-118  
RemoveLayer 28-249, 30-135, 30-  
    238  
RemoveLevels 30-179  
RemoveMaterial 5-75, 29-128  
RemoveModelingProperties 9-119  
RemovePort 28-209, 30-180, 30-239  
RemovePortsFromAllNets 28-210, 28-  
    283, 30-136  
RemovePortsFromNet 28-283, 30-137  
RemovePortsOnComponents 28-210, 28-  
    284, 30-138  
RemoveProp 7-51  
RemoveRefPort 25-13  
RemoveScript 29-130, 29-340  
RemoveSolverOnDemandModel 29-216  
RemoveUnused 29-217, 29-264, 29-269,  
    29-298, 29-335, 29-369  
RemoveUnusedDefinitions 5-76, 29-132  
Rename 5-77, 22-49, 25-13, 26-2, 27-2  
Rename [Interface Port] 25-12  
RenameDesignInstance 9-119, 9-120  
RenameFieldPlot 16-88  
RenameImportData 9-120  
RenamePart 10-320  
RenamePlotFolder 16-89  
RenameReport 12-121  
RenameRuleSet 31-9  
RenameRun 31-10  
RenameSetup  
    Optimetrics module command 14-77,  
    14-150  
    Radiation module command 18-9  
RenameSource [Interface Source] 9-121,  
    25-12  
RenameSweep 24-13

RenameTraces 12-122	CloneReportsFromDatasetSolution 12-29
Renormalize 22-50	CopyPlotSettings 12-29
Reorder 22-51, 22-52	CopyReportDefinition 12-30
Repeating a Statement Until a Condition Becomes True 1-10	CopyReportsData 12-31
Repeating Statements While a Condition is True 1-9	CopyTraceDefinitions 12-32
ReplaceWith3DComponent 10-219	CopyTracesData 12-32
Report module commands	CreateReport 9-24, 12-33
GetChildNames 12-82	CreateReportFromTemplate 12-54
Reporter editor commands	CreateReportOfAllQuantities 12-55
AddAllEyeMeasurements 12-6	DeleteAllReports 12-57
AddCartesianLimitLine 12-6	DeleteReport 12-57
AddCartesianLimitLineFromCurve 12-8	DeleteTraceCharacteristics 12-58
AddCartesianLimitLineFromEquation 12-10	DeleteTraces 12-59
AddCartesianXMarker 12-12, 12-12	DoesSupportTraceCharacteristics 12-60
AddCartesianYMarkerToStack 12-13	DumpAllReportsData 12-61
AddDeltaMarker 12-14	EditCartesianXMarker 12-62
AddMarker 12-16	EditCartesianYMarker 12-62
AddMarkerToPlot 16-5	EditMarker 12-63
AddNote 12-17	ExportEyeMaskViolation 12-64
AddTraceCharacteristics 12-19	ExportImageToFile 12-65
AddTraces 12-20	ExportPlot3DToFile 12-70
ApplyReportTemplate 12-23	ExportPlotImageToFile 16-71, 16-72
ChangeProperty 9-19, 12-23	ExportReport 12-71
ClearAllMarkers 12-27	ExportReportDataToFile 12-73
ClearAllTraceCharacteristics 12-28	ExportTableToFile 12-73
	ExportToFile 12-74, 12-113

---

ExportUniformPointsToFile 12-75	ImportReportDataIntoReport 12-114
FFTOnReport 15-12	MovePlotCurvestoGroup 12-114
GetAllCategories 12-77	MovePlotCurvestoNewGroup 12-115
GetAllQuantities 12-78	OpenWindowForAllReports 12-116
GetAllReportNames 12-79	OpenWindowForReports 12-117
GetAvailableDisplayTypes 12-80	PastePlotSettings 12-118
GetAvailableReportTypes 12-81	PasteReports 12-118
GetAvailableSolutions 12-81	PasteReportsWithLegacyNames 12-119
GetChildTypes ReportSetup 12-84	PasteTraces 12-120
GetCurvePropServerName 12-84	PasteTracesWithLegacyNames 12-120
GetDisplayStyle 12-85	RenameReport 12-121
GetDynLinkIntrinsicVariables 12-86	RenameTraces 12-122
GetDynLinkQtyValueState 12-86	ResetPlotSettings 12-123
GetDynLinkTraces 12-87	SavePlotSettingsAsDefault 12-123
GetDynLinkVariableValues 12-88	SetLinkOutputTraces 12-124
GetPropNames 12-92	UnGroupPlotCurvesInGroup 12-126
GetQtyExpressionsForSourceTrace 12-93	UpdateAllReports 12-127
GetRe- portSum- maryForRegressionTesting 12-94	UpdateReports 12-127
GetReportTraceNames 12-94	UpdateTraces 12-128
GetSolutionContexts 12-95, 12-96	UpdateTracesContextAndSweeps 12-131
GroupPlotCurvesByGroup- ingStrategy 12-112	Reporter Editor Script Commands 12-1
	Reporter module commands
	GetChildObject 12-83
	ReportTemplates 9-121
	Reset 22-53
	ResetLogging 3-54
	ResetPlotSettings 12-123

---

RestoreWindow 3-56	Script and Library Scripts 29-336
ResumeRecording 3-57	script commands
Rotate 10-121, 28-39, 28-53, 28-57, 30-139, 30-181, 30-239	field overlay 16-1
RunAllDV 31-10	radiation 18-2
RunAllRuleSetDV 31-11	Script Commands for Creating and Modifying PMLs 13-2
RunDV 31-11	Script Commands for Creating and Modifying Radiation Setups 18- 11
Running Instance Manager Script Commands 4-1	Script Commands for Exporting Antenna Parameters and Max Field Parameters 18-61
RunProgram 3-58	Script Commands for Modifying Antenna Array Setups 18-55
RunScript 3-59	scripts
RunScriptWithArguments 3-60	CPython 1-72
RunToolkit 9-122	IronPython 1-16
<b>S</b>	
sample scripts	overview 1-1
simple HFSS 1-12	pausing 1-81
simple Q3D Extractor 1-14	recording 1-82
variable helix 33-16	running 1-80
waveguide 33-18	stopping 1-81
SARSetup 9-123	VBScript 1-2
Save 5-78, 30-181, 30-240	Scripts and Locked Layers 1-89
SaveAs 5-79	Section 10-222
SaveAsStandAloneProject 5-81	Select Case statement 1-8
SaveFieldsPlots 16-90	SelectAll 28-214, 28-326, 30-139, 30- 182
SaveNamedExpressions 16-91, 17- 33	SelectInLayout 25-13, 26-3, 27-2
SavePlotSettingsAsDefault 12-123	SelectScheduler 3-13, 3-61
Scale 10-122, 28-57	SendToBack 30-182
Scope and Lifetime of Variables 1-4	

sensitivity commands	
EditSetup 14-179	SetLayerMapping 28-250, 30-140
InsertSetup 14-186	SetLengthSettings 9-134
Sensitivity Script Commands 14-178	SetLibraryDirectory 3-67
SeparateBody 10-224	SetLinkOutputTraces 12-124
Set 28-49	SetModelUnits 10-225
SetActiveDefinitionEditor 5-83, 30-240	SetPlotFolderSettings 16-95
SetActiveDesign 5-84	SetPortDeembedDistance 22-56
SetActiveEditor 9-123, 9-124, 9-131	SetPortImpedance 22-57
SetActiveLevel 30-183	SetPostProcSettings 22-58
SetActiveProject 3-63	SetProjectDirectory 3-67
SetActiveProjectByPath 3-64	SetPropertyValue 5-85, 7-71, 9-139, 28-223, 28-327, 29-134, 30-185, 30-241
SetAllowMaterialOverride 9-125	SetPropValue Design 9-138, 18-10
SetAllPortImpedance 22-54	SetPropValue Modeler 10-322
SetArc (Layout Editor) 28-51	SetPropValue Optimetrics 14-77, 14-151
SetBackgroundMaterial 9-125	SetPropValue Project 5-84, 12-125
SetCallback 7-52	SetReadOnly 7-54
SetClosed 28-56	SetShowLayoutForLayoutComponent 9-140
SetCS 28-78, 28-218, 30-139	SetSolutionType 9-141
SetDescription 7-53	SetSolveInsideThreshold 9-143
SetDesignSettings 9-127	SetSourceContexts 9-144, 15-39
SetFastTransform- ationForLayoutComponent 9-134, 9-135	SetTempDirectory 3-72
SetFieldPlotSettings 16-92	SetText 7-55
SetHidden 7-53	Setting Numerical Values 1-88
SetInitialLevels 30-184	SetTopDownViewDirectionForActiveView 10-323
	SetTopDownViewDirectionForAllViews 10-323
	SetValue 7-55

---

SetVariableValue	5-87, 7-73, 9-145	ExportNMF	15-10
SetVisibleLevels	30-185	ExportTransientData	15-11
SetWCS	10-226	GetAdaptiveFreq	15-13
SetX	28-49	GetAdaptiveSettings	15-14
SetY	28-50	GetAllSourceMagnitudes	15-15
SGetAppDesktop	2-2	GetAllSourceModes	15-15
ShowAntennaParameterOverlay		GetAllSourcePhases	15-16
16-100		GetAllSources	15-17
ShowIWindow	10-227	GetAntennaParameters	15-17
simple and composite names	1-3	GetAvailableVariations	15-18
Simplify	10-228	GetFieldType	15-19
SimulateAll	5-11, 5-88	GetIncludePortPostProcessing	15-20
Simulation Setup Commands	24-1	GetMultipactionBreakdown	15-21
SimValueContext	11-10	GetNetworkDataSolution	15-21
Sleep	3-73	GetNetworkDataSolutionDefinition	15-22
Smooth	22-59	GetNetworkPostprocSetup	15-23
Solution module commands		GetSolutionVersionID	15-23
GetSourceContexts	15-25	GetSolveRangeInfo	15-24
SetSourceContexts	9-144, 15-39	GetTerminalExcitationType	15-26
Solutions module commands		GetTransientSolveTimes	15-26
ConstructVariationString	9-23	GetValidISolutionLlst	15-27
DeleteFieldVariation	9-40	HasFields	15-28
DeleteFullVariation	9-41	HasMatrixData	15-28
DeleteImportData	15-1	ImportSolution	15-30
DeleteLinkedDataVariation	9-42	ImportTable	15-31
DeleteSolutionVariation	9-43, 15-2	IsFieldAvailableAt	15-32
EditSources	15-3	ListMatchingVariations	15-33
ExportEigenmodes	15-5		
ExportForHSpice	9-56, 15-6		

- ListValuesOfVariable 15-34  
ListVariations 15-35  
ModuleHasMesh 15-29  
TDROnReport 15-40  
Solutions Module commands  
    ExportNetworkData 15-8  
Solutions Module Script Commands 15-1  
Solve 9-145  
SolveAllSetup, Optimetrics module command 14-78, 14-152  
SolveSetup, Optimetrics module command 14-79, 14-152  
Split 10-231  
SplitRegion 28-67  
StartAnalysis 9-146  
statistical commands  
    EditSetup 14-190  
    InsertSetup 14-193  
Statistical Script Commands 14-190  
Stitch 10-233  
StitchLines 28-327, 30-142  
StopSimLink 9-146  
StopSimulations 3-74  
Stretch 22-60  
string concatenation operator 1-7  
Sub procedures 1-12, 1-14  
Sub Procedures 1-10  
SubmitJob 3-74  
Subtract 10-235, 28-65, 28-65, 28-68, 28-68, 28-224, 28-224, 30-143, 30-143, 30-241  
SweepAlongPath 10-92  
SweepAlongVector 10-94  
SweepAroundAxis 10-96  
SweepFacesAlongNormal 10-98, 10-98, 10-236, 10-236  
SweepFacesAlongNormalWithAttributes 10-100  
Symbol Editor Scripts 30-145  
Symbol Manager Script Commands 29-341
- T**
- TDROnReport 15-40  
Terminate 22-61  
ThickenSheet 10-238  
ToggleLevel 30-186  
ToggleViaPin 28-225, 30-143, 30-187, 30-242
- U**
- UncoverFaces 10-240  
underscore ( \_ ) character 1-12, 1-14  
Undo 9-148, 30-187  
    project-level command 5-88  
Ungroup 10-242, 28-78, 28-226, 30-144  
UnGroupPlotCurvesInGroup 12-126  
Unite 10-241, 28-65, 28-65, 28-67, 28-67, 28-226, 28-226, 30-144, 30-144, 30-242

UnselectAll 28-227, 28-328, 30-144  
UpdateAllFieldsPlots 16-101  
UpdateAllReports 12-127  
UpdateCableHarness 10-344  
UpdateComponentDefinition 10-103  
UpdateDefFromBlock 29-135  
UpdateDefFromBlockEx 29-137  
UpdateDefinitions, project-level command 5-89, 29-139  
UpdateDynamicLink 29-218  
UpdatePriorityList 10-324  
UpdateQuantityFieldsPlots 16-101  
UpdateReports 12-127  
UpdateTraces 12-128  
UpdateTracesContextAndSweeps 12-131  
UpgradeVersion 10-325  
User defined documents module commands  
    GetDocumentNames 20-8  
User defined solution module commands  
    CreateUserDefinedSolution 21-1  
    DeleteUserDefinedSolutions 21-3  
    EditUserDefinedSolution 21-4  
     GetUserDefinedSolutionNames 21-6  
     GetUserDefinedSolutionProperties 21-7  
User Defined Solutions Commands 21-1  
Using a Do Loop 1-9

**V**

Validate3DComponent 10-326  
ValidateCircuit 9-148  
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 1-12  
    operators 1-5  
    references 1-12  
    Sub procedures 1-12, 1-14

VBScript Procedures 1-10

**W**

WrapSheet 10-243  
WriteHistoryTreeLayoutForTest 10-327

**X**

Xor 28-66

**Z**

ZoomArea 30-188

ZoomIn 30-189

ZoomOut 30-189

ZoomPrevious 30-190

ZoomToFit 28-328, 30-145, 30-  
190, 30-243