



POWERING INNOVATION THAT DRIVES HUMAN ADVANCEMENT

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

Mechanical Scripting Guide



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

Release 2025 R1
January 2025

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

Copyright and Trademark Information

© 1986-2025 ANSYS, Inc. Unauthorized use, distribution or duplication is prohibited.

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

Disclaimer Notice

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

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

U.S. Government Rights

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

Third-Party Software

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

Table of Contents

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

Sub Procedures	1-10
Converting Between Data Types	1-10
Including Scripts	1-10
Aborting Scripts	1-11
Interacting with a Script	1-11
Recommended VBScript References	1-11
Sample HFSS Script	1-12
Sample Q3D Extractor Script	1-14
Introduction to IronPython	1-15
Scope	1-16
Python compatibility	1-16
Advantages of IronPython	1-16
IronPython Mini Cookbook	1-17
Comments	1-17
Assigning/Creating Variables	1-18
Create Lists/Arrays	1-18
Create Dictionaries/Maps	1-19
Boolean Values	1-19
Converting Numbers to Strings and Vice Versa	1-19
String Formatting/Concatenation	1-20
Looping over Lists	1-20
Looping over a Range	1-20
Indentation in IronPython	1-21
Indenting Functions	1-21
Indenting If Conditions	1-21
Methods in IronPython	1-22
Finding Methods	1-22
Help	1-22
Translating Script Commands from VBScript to IronPython	1-22
Script Method Argument	1-22

Primitive Types	1-23
Named Arrays	1-23
Named Functions	1-23
VBScript Method Call Types	1-23
Converting VBScript Function calls to IronPython Syntax	1-24
Return Values	1-24
Primitive Method Arguments	1-24
Named Array Arguments	1-25
Named Array Values with All Key Value Pairs	1-25
Named Arrays with Nested Named Arrays	1-25
Function Blocks	1-26
Scripting Using Iron Python	1-27
Translating a script in VBScript to IronPython	1-27
Writing an IronPython script from scratch	1-27
IronPython Script Execution Environment	1-28
Script Argument for IronPython	1-28
Scripting using Embedded VBScript or JavaScript	1-29
Scripting with IronPython	1-32
Standalone IronPython	1-33
Running Standalone IronPython	1-33
Using a Recorded Script	1-33
Creating an External Script	1-33
Example Script	1-34
IronPython Samples	1-35
Change property	1-35
Create a Cone using IronPython	1-36
Creating User Defined Primitives and User Defined Models in Python Scripts	1-40
Advantages Compared to C++	1-40
Changes compared to C	1-41
Structures	1-41

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

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

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

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

Optimetrics Module Object:	2-37
Optimetrics Setup Object	2-38
ReportSetup(Results) Module Object:	2-39
ReportSetup(Results) Module Child Objects:	2-39
Radiation Module Object:	2-40
Radiation Module Child Objects:	2-41
Conventions Used in this Chapter	2-41
GetArrayVariables	2-44
GetProperties	2-45
GetPropertyValue	2-46
GetVariables	2-48
GetVariableValue	2-48
SetPropertyValue	2-49
SetVariableValue	2-51
Example Use of Record Script and Edit Properties	2-51
Additional Property Scripting Examples	2-52
3 - Application Object Script Commands	3-1
GetAppDesktop	3-2
4 - Desktop Object Script Commands	4-1
AddMessage	4-6
AreThereSimulationsRunning	4-7
ClearMessages	4-7
CloseAllWindows	4-11
CloseProject	4-12
CloseProjectNoForce	4-13
Count	4-14
DownloadJobResults	4-15
DeleteRegistryEntry	4-16
DoesRegistryValueExist	4-17
EnableAutoSave	4-18

ExportOptionsFiles	4-19
GetActiveProject	4-19
GetActiveScheduler	4-20
GetActiveSchedulerInfo	4-21
GetAutoSaveEnabled	4-22
GetBuildDateTimeString	4-23
GetCustomMenuSet	4-24
GetDefaultUnit	4-24
GetDesktopConfiguration	4-26
GetDistributedAnalysisMachines	4-27
GetDistributedAnalysisMachinesForDesignType	4-28
GetExeDir	4-29
GetGDIObjectCount	4-29
GetLibraryDirectory	4-30
GetLocalizationHelper	4-31
GetMessages	4-32
GetMonitorData	4-33
GetPersonalLibDirectory	4-34
GetProcessID	4-35
GetProjectDirectory	4-36
GetProjectList	4-36
GetProjects	4-37
GetRegistryInt	4-38
GetRegistryString	4-39
GetRunningInstancesMgr	4-39
GetSchematicEnvironment	4-40
GetScriptingToolsHelper	4-41
GetSysLibDirectory	4-42
GetTempDirectory	4-43
GetUserLibDirectory	4-43

GetVersion	4-44
IsFeatureEnabled	4-45
KeepDesktopResponsive	4-46
LaunchJobMonitor	4-47
OpenAndConvertProject	4-47
OpenMultipleProjects	4-48
OpenProject	4-49
OpenProjectWithConversion	4-50
PageSetup	4-50
PauseRecording	4-52
PauseScript	4-53
Print	4-54
QuitApplication	4-54
RefreshJobMonitor	4-55
ResetLogging	4-56
RestoreProjectArchive	4-57
RestoreWindow	4-58
ResumeRecording	4-58
RunACTWizardScript	4-59
RunProgram	4-59
RunScript	4-60
RunScriptWithArguments	4-62
SelectScheduler	4-63
SetActiveProject	4-65
SetActiveProjectByPath	4-65
SetCustomMenuSet	4-66
SetDesktopConfiguration	4-67
SetLibraryDirectory	4-68
SetProjectDirectory	4-69
SetRegistryFromFile	4-70

SetRegistryInt	4-70
SetRegistryString	4-71
SetSchematicEnvironment	4-72
SetTempDirectory	4-73
ShowDockingWindow	4-74
Sleep	4-74
StopSimulations	4-75
SubmitJob	4-76
TileWindows	4-77
Desktop Commands For Registry Values	4-78
DeleteRegistryEntry	4-79
DoesRegistryValueExist	4-79
GetRegistryInt	4-80
GetRegistryString	4-81
SetRegistryFromFile	4-82
SetRegistryInt	4-83
SetRegistryString	4-84
ImportExport Tool Commands	4-84
ImportANF	4-85
ImportANFv2	4-87
ImportAutoCAD	4-88
ImportAWRMicrowaveOffice	4-89
ImportEDB	4-91
ImportExtracta	4-92
ImportGDSII	4-93
ImportGerber	4-94
ImportIDF	4-95
ImportIDFandMerge	4-98
ImportIDX	4-99
ImportIPC	4-102

ImportODB	4-103
ImportXFL	4-104
5 - Running Instances Manager Script Commands	5-1
GetAllRunningInstances	5-1
GetRunningInstanceByProcessID	5-2
GetRunningInstancesMgr	5-2
6 - Project Object Script Commands	6-1
AddDataset	6-4
AddMaterial	6-7
AnalyzeAll [project]	6-11
ChangeProperty	6-12
ClearMessages	6-18
CloneMaterial	6-19
Close	6-19
CopyDesign	6-20
CutDesign	6-21
DeleteDataset	6-22
DeleteDesign	6-23
DeleteToolObject	6-23
EditDataset	6-24
EditMaterial	6-27
ExportDataset	6-36
ExportMaterial	6-36
GetActiveDesign	6-37
GetArrayVariables	6-38
GetChildNames [Project]	6-39
GetChildObject [Project]	6-40
GetChildTypes [Project]	6-41
GetConfigurableData (Project)	6-42
GetDefinitionManager	6-42

GetDependentFiles	6-43
GetDesign	6-43
GetDesigns	6-44
GetEDBHandle	6-45
GetLegacyName	6-46
GetName [Project]	6-46
GetObjPath [Project]	6-47
GetPath	6-48
GetPropEvaluatedValue	6-48
GetPropNames [Project]	6-49
GetPropSIValue	6-50
GetPropValue [Project]	6-51
GetProperties	6-52
GetPropertyValue	6-53
GetTopDesignList	6-55
GetVariableValue	6-55
GetVariables	6-56
ImportDataset	6-57
Note About File Types:	6-58
InsertDesignWithWorkflow	6-59
InsertToolObject	6-61
Paste (Project Object)	6-61
Redo [Project Level]	6-62
RemoveAllUnusedDefinitions	6-62
RemoveMaterial	6-63
RemoveUnusedDefinitions	6-64
Rename	6-65
Save	6-66
SaveAs	6-67
SaveAsStandAloneProject	6-69

SaveProjectArchive	6-70
SetActiveDefinitionEditor	6-71
SetActiveDesign	6-72
SetPropValue [Project]	6-72
SetPropertyValue	6-73
SetVariableValue	6-75
SimulateAll	6-76
Undo [Project]	6-76
UpdateDefinitions	6-77
7 - Design Object Script Commands	7-1
AddDataset	7-4
AddModelingProperties	7-7
AnalyzeAll [design]	7-7
AnalyzeAllNominal	7-8
ClearLinkedData (Design)	7-9
ConstructVariationString	7-10
CopyArray	7-10
CopyItemCommand	7-11
DeleteDataset	7-11
DeleteFieldVariation	7-12
DeleteFullVariation	7-13
DeleteLinkedDataVariation	7-14
DeleteOutputVariable	7-15
DeletePlotEntities	7-15
DeleteSolutionVariation	7-17
DeleteOutputVariable	7-18
EditCoSimulationOptions	7-18
EditInfiniteArray	7-19
EditLayoutForLayoutComponent	7-20
EMDesignOptions	7-20

ExportConvergence	7-23
ExportArray	7-23
ExportDataset	7-24
ExportProfile	7-25
GetChildNames [Design]	7-26
GetChildObject [Design]	7-26
GetChildTypes [Design]	7-28
GetConfigurableData	7-28
GetData	7-29
GetDesignID	7-29
GetDesignType	7-30
GetEdgePositionAtNormalizedParameter	7-31
GetManagedFilePath	7-32
GetModule	7-33
GetModule (RadiationSetupMgr)	7-34
GetName	7-35
GetNominalVariation	7-36
GetNoteText	7-36
GetObjPath [Design]	7-37
GetOutputVariableValue	7-38
GetOutputVariables	7-39
GetPostProcessingVariables	7-39
GetProperties	7-40
GetPropertyValue	7-41
GetPropNames [Design]	7-43
GetPropValue [Design]	7-44
GetSelections [Design]	7-45
GetSolutionType	7-45
GetSolveInsideThreshold	7-45
GetVariables	7-46

GetVariableValue	7-47
GetVariationVariableValue	7-48
ImportArray	7-48
ImportDataset	7-49
Note About File Types:	7-50
PasteArray	7-51
PasteDesign (Design Object)	7-52
Redo [Design]	7-53
RenameDesignInstance	7-54
RenameSource [Interface Source]	7-55
RevertAllToInitialCondition	7-55
SARSetup	7-56
SetActiveEditor	7-56
SetAllowMaterialOverride	7-57
SetBackgroundMaterial	7-58
SetDesignMode	7-58
SetDesignSettings (Mechanical)	7-59
SetFastTransformationForLayoutComponent	7-62
SetLengthSettings	7-63
SetObjectAttributesForLayoutComponent	7-63
SetPropValue [Design]	7-66
SetPropertyValue	7-67
SetShowLayoutForLayoutComponent	7-68
SetShowAllLayoutComponents	7-69
SetSolutionType	7-70
SetSolveInsideThreshold	7-72
SetSourceContexts	7-73
SetVariableValue	7-74
Solve	7-74
StopSimLink	7-75

Undo [Design]	7-76
ValidateLink	7-77
ValidateDesign	7-77
ValidateLink	7-78
8 - 3D Modeler Editor Script Commands	8-1
Conventions Used in this Chapter:	8-1
<AttributesArray>	8-1
<SelectionsArray>	8-2
Draw Menu Commands	8-4
Create3DComponent	8-6
CreateBondwire	8-11
CreateBox	8-15
CreateCircle	8-18
CreateCone	8-20
CreateCutplane	8-23
CreateCylinder	8-25
CreateEllipse	8-28
CreateEquationCurve	8-30
CreateEquationSurface	8-34
CreateHelix	8-37
CreatePoint	8-39
CreatePolyline	8-41
CreateRectangle	8-46
CreateRegularPolygon	8-49
CreateRegularPolyhedron	8-52
CreateSphere	8-55
CreateSpiral	8-57
CreateTorus	8-59
CreateUserDefinedModel	8-62
Python Example of Creating a UDM from a Discovery Model	8-64

CreateUserDefinedPart	8-71
Edit3DComponent	8-79
Edit3DComponentDefinition	8-81
EditNativeComponentDefinition [Beta – Layout Components in Mechanical]	8-83
EditPolyline	8-88
Get3DComponentDefinitionNames	8-92
Get3DComponentInstanceNames	8-93
Get3DComponentMaterialNames	8-94
Get3DComponentMaterialProperties	8-94
Get3DComponentParameters	8-95
Get3DComponentPartNames	8-96
Insert3DComponent	8-96
InsertComponent	8-97
InsertNativeComponent [Beta – Layout Component in Mechanical]	8-99
InsertPolylineSegment	8-108
SweepAlongPath	8-110
SweepAlongVector	8-112
SweepAroundAxis	8-114
SweepFacesAlongNormal	8-116
SweepFacesAlongNormalWithAttributes	8-118
UpdateComponentDefinition	8-121
Edit Menu Commands	8-122
Copy	8-123
DeletePolylinePoint	8-124
DuplicateAlongLine	8-125
DuplicateAroundAxis	8-127
DuplicateMirror	8-130
Mirror	8-133
Move	8-135
OffsetFaces	8-137

Paste (Model Editor)	8-138
Rotate	8-139
Scale	8-140
Modeler Menu Commands	8-142
AlignFaces	8-144
AssignMaterial	8-145
Chamfer	8-148
CleanUpModel	8-150
Connect	8-151
CoverLines	8-152
CoverSurfaces	8-153
CreateEntityList	8-154
CreateFaceCS	8-155
CreateGroup	8-160
CreateObjectCS	8-161
CreateObjectFromEdges	8-168
CreateObjectFromFace	8-170
CreateObjectFromFaces	8-172
CreateRelativeCS	8-173
DeleteEmptyGroups	8-175
DeleteLastOperation	8-176
DeleteOperation	8-177
DetachEdges	8-179
DetachFaces	8-180
EditEntityList	8-182
EditFaceCS	8-183
EditObjectCS	8-189
EditRelativeCS	8-196
Export	8-198
ExportModelImageToFile	8-200

ExportModelMeshToFile	8-204
Fillet	8-204
FlattenGroup	8-207
GenerateHistory	8-207
GetActiveCoordinateSystem	8-208
GetActiveCoordinateSystemTransform	8-209
GetCoordinateSystems	8-210
HealObject	8-210
Import	8-215
ImportDXF [Modeler]	8-218
ImportFromClipboard	8-223
ImportGDSII [Modeler]	8-223
Imprint	8-227
ImprintProjection	8-228
Intersect	8-230
MoveCStoEnd	8-231
MoveEntityToGroup	8-232
MoveFaces	8-233
ProjectSheet	8-236
PurgeHistory	8-237
ReplaceWith3DComponent	8-238
Section	8-241
SeparateBody	8-243
SetModelUnits	8-244
SetWCS	8-245
ShowWindow	8-246
Simplify	8-247
Split	8-249
Stitch	8-252
Subtract	8-253

SweepFacesAlongNormal	8-255
ThickenSheet	8-257
UncoverFaces	8-258
Unite	8-260
Ungroup	8-262
WrapSheet	8-262
Other oEditor Commands	8-264
AddDefinitionFromBlock	8-266
AddDefinitionFromLibFile	8-271
AssignSurfaceMaterial	8-275
BreakUDMConnection	8-278
ChangeProperty	8-279
CloseAllWindows[Editor]	8-285
Defeature	8-286
Delete	8-287
FitAll	8-287
GenerateAllUserDefinedModels	8-288
GenerateUserDefinedModel	8-289
GeometryCheckAndAutofix	8-290
GetBodyNamesByPosition	8-292
GetChildNames [Modeler]	8-293
GetChildObject [Modeler]	8-296
GetChildTypes [Modeler]	8-299
GetEdgeByPosition	8-301
GetEdgeIDFromNameForFirstOperation	8-303
GetEdgeIDsFromFace	8-304
GetEdgeIDsFromObject	8-304
GetEdgeLength	8-305
GetEntityListIDByName	8-306
GetExtendedDefinitionObject	8-307

GetFaceArea	8-308
GetFaceCenter	8-308
GetFaceByPosition	8-309
GetFaceIDFromNameForFirstOperation	8-311
GetFaceIDs	8-311
GetFaceIDsOfSheet	8-312
GetMatchedObjectName	8-313
GetModelBoundingBox	8-313
GetModelUnits	8-314
GetNumObjects	8-315
GetObjectIDByName	8-315
GetObjectName	8-316
GetObjectNameByEdgeID	8-317
GetObjectNameByFaceID	8-317
GetObjectNameByID	8-318
GetObjectNameByVertexID	8-319
GetObjectsByMaterial	8-319
GetObjectShapeType	8-320
GetObjectsInGroup	8-321
GetObjectVolume	8-322
GetObjPath [Editor]	8-323
GetPartsForUserDefinedModel	8-323
GetPoints [3D Modeler Editor]	8-324
GetPropEvaluatedValue	8-325
GetPropertyValue	8-325
GetPropNames [Modeler]	8-327
GetPropSIValue	8-328
GetPropValue [Modeler]	8-329
GetRelativeCoordinateSystems	8-330
GetSelections [Model Editor]	8-331

GetSubGroupsInGroup	8-331
GetUserPosition	8-332
GetVertexIDFromNameForFirstOperation	8-333
GetVertexIDsFromEdge	8-334
GetVertexIDsFromFace	8-335
GetVertexIDsFromObject	8-335
GetVertexPosition	8-336
GetWireBodyNames	8-337
PageSetup	8-337
RemoveBadEdges	8-339
RemoveBadFaces	8-340
RemoveBadVertices	8-340
RenamePart	8-341
SetPropValue [Modeler]	8-342
SetTopDownViewDirectionForActiveView	8-343
SetTopDownViewDirectionForAllViews	8-344
UpdatePriorityList	8-345
UpgradeVersion	8-345
Validate3DComponent	8-347
WriteHistoryTreeLayoutForTest	8-348
9 - Output Variable Script Commands	9-1
CreateOutputVariable	9-1
DeleteOutputVariable	9-2
DoesOutputVariableExist	9-3
EditOutputVariable	9-4
ExportOutputVariables	9-5
GetOutputVariables	9-6
GetOutputVariableValue	9-7
ImportOutputVariables	9-8
SimValueContext	9-9

10 - Reporter Editor Script Commands	10-1
AddAllEyeMeasurements	10-3
AddCartesianLimitLine	10-4
AddCartesianLimitLineFromCurve	10-6
AddCartesianLimitLineFromEquation	10-8
AddCartesianXMarker	10-10
AddCartesianYMarker	10-10
AddCartesianYMarkerToStack	10-11
AddDeltaMarker	10-12
AddMarker	10-13
AddNote	10-14
AddTraceCharacteristics	10-16
AddTraces	10-18
ApplyReportTemplate	10-20
ClearAllMarkers	10-21
ClearAllTraceCharacteristics	10-22
CloneReportsFromDatasetSolution	10-22
CopyPlotSettings	10-23
CopyReportDefinitions	10-24
CopyReportsData	10-25
CopyTraceDefinitions	10-25
CopyTracesData	10-26
CreateReport [Mechanical]	10-27
CreateReportFromFile	10-36
CreateReportFromTemplate	10-37
CreateReportOfAllQuantities	10-38
DeleteMarker	10-39
DeleteAllReports	10-40
DeleteReports	10-41
DeleteTraceCharacteristics	10-41

DeleteTraces	10-42
DoesSupportTraceCharacteristics	10-43
DumpAllReportsData	10-44
EditCartesianXMarker	10-45
EditCartesianYMarker	10-45
EditMarker	10-46
ExportEyeMaskViolation	10-47
ExportImageToFile [Reporter]	10-48
ExportModelImageToFile	10-52
ExportModelMeshToFile	10-56
ExportPlot3DToFile [Reporter]	10-57
ExportReport	10-58
ExportReportDataToFile	10-60
ExportTableToFile	10-60
ExportToFile [Reporter]	10-61
ExportUniformPointsToFile	10-62
GetAllCategories	10-64
GetAllQuantities	10-65
GetAllReportNames	10-66
GetAvailableDisplayTypes	10-67
GetAvailableReportTypes	10-68
GetAvailableSolutions	10-68
GetChildNames [Report Setup]	10-69
GetChildObject [Report Setup]	10-70
GetChildTypes [ReportSetup]	10-71
GetCurvePropServerName	10-71
GetDisplayType	10-72
GetDynLinkIntrinsicVariables	10-73
GetDynLinkQtyValueState	10-73
GetDynLinkTraces	10-74

GetDynLinkVariableValues	10-75
GetName	10-75
GetObjPath [Design]	10-76
GetPropertyValue	10-77
GetPropNames [Reporter]	10-78
GetPropValue [Report Setup]	10-79
GetQtyExpressionsForSourceTrace	10-80
GetReportTraceNames	10-80
GetReportSummaryForRegressionTesting	10-81
GetSolutionContexts	10-82
GetSolutionDataPerVariation	10-83
GetDataExpressions	10-85
GetDataUnits	10-86
GetDesignVariableNames	10-87
GetDesignVariableUnits	10-88
GetDesignVariableValue	10-89
GetDesignVariationKey	10-90
GetImagDataValues	10-90
GetPerQuantityPrimarySweepValues	10-91
GetRealDataValues	10-92
GetSweepNames	10-93
GetSweepUnits	10-94
GetSweepValues	10-95
IsDataComplex	10-96
IsPerQuantityPrimarySweep	10-97
Release Data	10-98
GroupPlotCurvesByGroupingStrategy	10-99
ImportIntoReport	10-100
ImportReportDataIntoReport	10-101
MovePlotCurvesToGroup	10-102

MovePlotCurvesToNewGroup	10-103
OpenWindowForAllReports	10-103
OpenWindowForReports	10-104
PastePlotSettings	10-105
PasteReports	10-106
PasteReportsWithLegacyNames	10-106
PasteTraces	10-107
PasteTracesWithLegacyNames	10-108
RenameReport	10-108
RenameTrace	10-109
ResetPlotSettings	10-110
SavePlotSettingsAsDefault	10-111
SetLinkOutputTraces	10-111
SetPropValue [Report Setup]	10-112
UnGroupPlotCurvesInGroup	10-113
UpdateAllReports	10-114
UpdateReports	10-115
UpdateTraces	10-115
UpdateTracesContextAndSweeps	10-118
11 - Boundary and Excitation Module Script Commands	11-1
General Commands Recognized by the Boundary/Excitations Module	11-2
AddAssignmentToBoundary	11-3
AutoidentifyLatticePair	11-4
AutoidentifyNets	11-4
AutoidentifyPorts	11-5
AutoidentifyTerminals	11-6
ChangeImpedanceMult	11-7
ConvertNportCircuitElementsToPorts	11-8
CreateNportCircuitElements	11-9
DeleteAllBoundaries	11-11

DeleteAllExcitations	11-12
DeleteBoundaries	11-12
EditNportCircuitElement	11-13
GetBoundaries	11-14
GetBoundariesOfType	11-15
GetBoundaryAssignment	11-16
GetDefaultBaseName	11-16
GetDiffPairs	11-17
GetExcitations	11-18
GetExcitationsOfType	11-18
GetHybridRegions	11-19
GetHybridRegionsOfType	11-20
GetNumBoundaries	11-20
GetNumBoundariesOfType	11-21
GetNumExcitations	11-22
GetNumExcitationsOfType	11-22
GetNumHybridRegions	11-23
GetNumHybridRegionsOfType	11-24
GetPortExcitationCounts	11-24
ReassignBoundary	11-25
RenameBoundary	11-26
ReprioritizeBoundaries	11-27
SetDefaultBaseName	11-28
Script Commands for Creating and Modifying Boundaries	11-29
AssignCircuitPort	11-32
AssignCurrent	11-34
AssignCylindricalWave	11-35
AssignDielectricCavity	11-38
AssignFEBl	11-39
AssignFiniteCond	11-40

AssignFloquet	11-43
AssignFresnel	11-50
AssignGaussianBeam	11-51
AssignGradientSurfaceRoughness	11-55
AssignHalfSpace	11-57
AssignHertzianDipoleWave	11-58
AssignHybridRegion	11-60
AssignIERegion	11-62
AssignInteriorNearFieldSource	11-62
AssignImpedance	11-63
AssignIncidentWave	11-65
AssignLatticePair	11-69
AssignLayeredImp	11-71
AssignLinearAntennaWave	11-74
AssignLinkedImpedance	11-78
AssignLinkedRegion	11-80
AssignLumpedPort	11-82
AssignLumpedRLC	11-85
AssignMagneticBias	11-88
AssignMultipactionChargeRegion	11-90
AssignMultipactionDCBias	11-92
AssignMultipactionSEE	11-96
AssignPerfectE	11-97
AssignPerfectH	11-99
AssignPlaneWave	11-100
AssignRadiation	11-104
AssignRFDischARGE DCBias	11-106
AssignScreeningImpedance	11-107
AssignSymmetry	11-111
AssignTerminal	11-112

AssignVoltage	11-112
AutoCreatePECCapforWavePort	11-114
CircuitPortToLumpedPort	11-115
EditAnisotropicImpedance	11-116
EditAperture	11-119
EditCircuitPort[HFSS]	11-120
EditDielectricCavity	11-122
EditDiffPairs	11-123
EditFEBI	11-125
EditFiniteCond	11-126
EditFloquetPort	11-127
EditFresnel	11-134
EditGlobalMatEnv	11-135
EditGradientSurfaceRoughness	11-136
EditHalfSpace	11-137
EditHybridRegion	11-139
EditImpedance	11-140
EditIncidentWave	11-141
EditInteriorNearFieldSource	11-144
EditLatticePair	11-145
EditLayeredImp	11-146
EditLinkedImpedance	11-149
EditLinkedRegion	11-151
EditLumpedPort	11-153
EditLumpedRLC	11-156
EditMagneticBias	11-158
EditMultipactionChargeRegion	11-160
EditMultipactionDCBias	11-161
EditMultipactionSEE	11-165
EditRFDischargeDCBias	11-167

EditPerfectE	11-168
EditPerfectH	11-169
EditRadiation	11-170
EditSymmetry	11-172
EditTerminal	11-173
EditVoltage	11-174
EditVoltageDrop	11-176
EditWavePort	11-176
LumpedPortToCircuitPort	11-180
SetAllHybridRegionsToOneWayCoupled	11-181
SetAllHybridRegionsToTwoWayCoupled	11-181
SetHybridRegionCoupledGroup	11-182
SetHybridRegionsCoupling	11-185
SetScatteredFieldFormulation	11-186
SetSBRCreepingWaveSettings	11-186
SetSBRTxRxSettings	11-188
SetTerminalReferenceImpedances	11-190
SetTotalFieldFormulation	11-191
SwapCircuitPortDirection	11-191
SwapLatticePair	11-192
UnassignIERegions	11-193
Mechanical Boundary, Contact, & Excitation Commands	11-194
AssignContact and EditContact	11-195
AssignContact	11-195
EditContact	11-198
AssignConvection and EditConvection	11-199
AssignConvection	11-199
EditConvection	11-202
AssignCylindricalSupport and EditCylindricalSupport	11-203
AssignCylindricalSupport	11-203

EditCylindricalSupport	11-204
AssignDisplacement and EditDisplacement	11-205
AssignDisplacement	11-205
EditDisplacement	11-207
AssignEMLoss and EditEMLoss	11-208
AssignEMLoss	11-208
EditEMLoss	11-210
AssignFixedSupport and EditFixedSupport	11-211
AssignFixedSupport	11-211
EditFixedSupport	11-212
AssignForce and EditForce	11-213
AssignForce	11-213
EditForce	11-216
AssignFrictionlessSupport and EditFrictionlessSupport	11-217
AssignFrictionlessSupport	11-217
EditFrictionlessSupport	11-218
AssignHeatFlux and EditHeatFlux	11-219
AssignHeatFlux	11-219
AssignHeatGeneration and EditHeatGeneration	11-222
AssignHeatGeneration	11-222
EditHeatGeneration	11-223
AssignInitialTemperature and EditInitialTemperature	11-224
AssignInitialTemperature	11-224
EditInitialTemperature	11-225
AssignPressure and EditPressure	11-226
AssignPressure	11-226
EditPressure	11-229
AssignRotatingFluid and EditRotatingFluid	11-230
AssignRotatingFluid	11-230
EditRotatingFluid	11-231

AssignTemperature and EditTemperature	11-232
AssignTemperature	11-232
EditTemperature	11-233
AssignThermalCondition and EditThermalCondition	11-234
AssignThermalCondition	11-234
EditThermalCondition	11-237
12 - Thermal Monitor Script Commands	12-1
AssignPointMonitor	12-1
ReassignPointMonitor	12-3
13 - Analysis Setup Module Script Commands	13-1
AddTwoWayCoupling	13-1
ClearLinkedData (Module)	13-3
CopySetup	13-3
CopyDrivenSetup	13-4
CopyEigenSetup	13-5
DeleteSetups	13-6
DeleteTwoWayCoupling	13-6
EditSetup	13-7
EditTwoWayCoupling	13-20
GetSetupCount	13-21
GetSetups	13-22
GetSweeps	13-23
InsertSetup [Mechanical]	13-23
Parameters For Modal Solutions Only	13-25
Modal InsertSetup Script Examples	13-27
Parameters For Thermal Solutions Only	13-29
Thermal InsertSetup Script Examples	13-34
Parameters For Structural Solutions Only	13-39
Structural InsertSetup Script Examples	13-40
PasteDrivenSetup	13-42

PasteEigenSetup	13-43
PasteSetup	13-43
RenameSetup	13-44
ResetToTimeZero	13-44
RevertAllToInitial	13-45
RevertAllToZeroDisplacement	13-46
RevertCoupledSolution	13-46
RevertSetupToInitial	13-47
RevertSetupToInitialCondition	13-48
RevertSetupToZeroDisplacement	13-49
SolveSetup	13-49
14 - Optimetrics Module Script Commands	14-1
CopySetup	14-6
DeleteSetups [Optimetrics]	14-7
DistributedAnalyzeSetup	14-7
EditSetup	14-8
EnableSetup	14-21
ExportDXConfigFile	14-22
ExportOptimetricsProfile	14-23
ExportOptimetricsResult	14-24
ExportParametricResults	14-25
ExportParametricSetupTable	14-26
ExportRespSurfaceMinMaxTable	14-27
ExportRespSurfaceRefinePoints	14-27
ExportRespSurfaceResponsePoints	14-28
ExportRespSurfaceVerificationPoints	14-29
GenerateVariationData [Parametric]	14-30
GetChildNames [Optimetrics]	14-31
GetChildObject [Optimetrics]	14-31
GetChildTypes [Optimetrics]	14-32

GetName	14-33
GetObjPath [Design]	14-33
GetOptimetricResult	14-34
GetOptimetricsResult	14-35
GetPropNames [Optimetrics]	14-36
GetPropValue [Optimetrics]	14-37
GetSetupNames [Optimetrics]	14-37
GetSetupNamesByType [Optimetrics]	14-38
ImportSetup	14-39
PasteSetup [Optimetrics]	14-40
RenameSetup [Optimetrics]	14-41
SetPropValue [Optimetrics]	14-42
SolveAllSetup	14-43
SolveSetup [Optimetrics]	14-44
General Commands Recognized by the Optimetrics Module	14-44
CopySetup	14-45
DeleteSetups [Optimetrics]	14-46
DistributedAnalyzeSetup	14-47
EditSetup	14-48
EnableSetup	14-61
ExportDXConfigFile	14-61
ExportOptimetricsProfile	14-62
ExportOptimetricsResult	14-63
ExportParametricResults	14-64
ExportParametricSetupTable	14-65
ExportRespSurfaceMinMaxTable	14-66
ExportRespSurfaceRefinePoints	14-67
ExportRespSurfaceResponsePoints	14-68
ExportRespSurfaceVerificationPoints	14-69
ExportDOEResponseCurve	14-69

ExportDOEResponseCurveSlices	14-70
ExportDOEResponseSurface	14-71
ExportDOELocalSensitivity	14-72
ExportDOELocalSensitivityCurve	14-73
GenerateVariationData [Parametric]	14-74
GetChildNames [Optimetrics]	14-75
GetChildObject [Optimetrics]	14-75
GetChildTypes [Optimetrics]	14-76
GetName	14-76
GetObjPath [Design]	14-77
GetOptimetricResult	14-78
GetPropNames [Optimetrics]	14-79
GetPropValue [Optimetrics]	14-79
GetSetupNames [Optimetrics]	14-80
GetSetupNamesByType [Optimetrics]	14-81
ImportSetup	14-82
PasteSetup [Optimetrics]	14-83
RenameSetup [Optimetrics]	14-84
SetPropValue [Optimetrics]	14-85
SolveAllSetup	14-85
SolveSetup [Optimetrics]	14-86
Parametric Script Commands	14-87
EditSetup [Parametric]	14-87
ExportParametricSetupTable	14-88
GenerateVariationData [Parametric]	14-89
InsertSetup [Parametric]	14-89
Optimization Script Commands	14-96
EditSetup [Optimization]	14-96
InsertSetup [Optimization]	14-101
Sensitivity Script Commands	14-109

EditSetup [Sensitivity]	14-109
InsertSetup [Sensitivity]	14-116
Statistical Script Commands	14-120
EditSetup [Statistical]	14-120
InsertSetup [Statistical]	14-123
15 - Field Overlays Module Script Commands	15-1
AddMarker[Fields Reporter]	15-1
AddMarkerToPlot	15-2
AddNamedExpression	15-4
CalcOp	15-5
CalcStack	15-5
CalculatorRead	15-6
CalculatorWrite	15-7
ChangeProperty	15-8
ChangeGeomSettings	15-13
ClcEval	15-13
ClcMaterial	15-14
ClcMaterialValue	15-15
ClearAllMarkers[Fields Reporter]	15-16
ClearAllNamedExpr	15-16
CopyNamedExprToStack	15-17
CreateFieldPlot	15-18
DeleteFieldPlot	15-30
DeleteMarker[Fields Reporter]	15-31
DeleteNamedExpr	15-31
DeleteUneditablePlot	15-32
DoesNamedExpressionExists	15-33
EnterComplex	15-34
EnterComplexVector	15-34
EnterCoord	15-35

EnterEdge	15-36
EnterLine	15-37
EnterOutputVar	15-37
EnterPoint	15-38
EnterQty	15-39
EnterScalar	15-40
EnterScalarFunc	15-40
EnterSurf	15-41
EnterVector	15-42
EnterVectorFunc	15-42
EnterVol	15-43
ExportFieldPlot	15-44
ExportMarkerTable	15-45
ExportOnGrid [Field Overlays]	15-45
ExportPlotImageToFile	15-48
ExportPlotImageWithViewToFile [Reporter]	15-49
ExportToFile	15-51
GetFieldFolderNames	15-52
GetFieldPlotNames	15-53
GetFieldPlotQuantityName	15-53
GetMeshPlotNames	15-54
GetTopEntryValue	15-55
LoadNamedExpressions	15-56
ModifyFieldPlot	15-57
ReassignFieldPlot	15-59
RenameFieldPlot	15-62
RenamePlotFolder	15-63
SaveFieldsPlots	15-64
SaveNamedExpressions	15-65
SetFieldPlotSettings	15-66

SetPlotFolderSettings	15-68
UpdateAllFieldsPlots	15-74
16 - Fields Summary Script Commands	16-1
EditFieldsSummarySetting	16-1
ExportFieldsSummary	16-2
17 - User Defined Document Script Commands	17-1
AddDocument	17-1
DeleteAllDocuments	17-4
DeleteDocument	17-5
EditDocument	17-5
GetDocumentDefinitionNames	17-7
GetDocumentNames	17-8
RenameDocument	17-8
SaveHtmlDocumentAs	17-9
SavePdfDocumentAs	17-10
UpdateAllDocuments	17-10
UpdateDocument	17-11
ViewHtmlDocument	17-12
ViewPdfDocument	17-13
Explication of a Sample UDD Script	17-13
Example Python Script: Defining a Document	17-15
18 - User Defined Solutions Commands	18-1
CreateUserDefinedSolution	18-1
DeleteUserDefinedSolutions	18-3
EditUserDefinedSolution	18-4
GetUserDefinedSolutionNames	18-6
GetUserDefinedSolutionProperties	18-6
19 - Network Data Explorer Script Commands	19-1
AddDiffPair	19-4
Cascade (SPISim)	19-5

ClearDiffPairs	19-6
Clone	19-7
Close	19-8
Combine (SPISim)	19-9
Deembed (SPISim)	19-10
DeembedBack (SPISim)	19-11
DeembedFront (SPISim)	19-13
DisableDiffPairs	19-14
EnableDiffPairs	19-15
ExportCitiFile	19-16
ExportFullWaveSpice	19-18
ExportMatlab	19-19
ExportNMFData	19-21
ExportNetworkData	19-21
ExportSpreadsheet	19-24
ExportTouchstone	19-26
ExportTouchstone2	19-28
Extract (SPISim)	19-31
GetFrequencies	19-32
GetFrequencyCount	19-33
GetName	19-34
GetPortCount	19-34
GetPortNumber	19-35
GetPostProcSettings	19-37
GetSolutionVariation	19-38
GetVariation	19-39
HasSameData	19-40
LoadSolution	19-41
Open	19-43
Rename (SPISim)	19-43

Renormalize (SPISim)	19-45
Reorder	19-46
Reorder (SPISim)	19-47
Reset	19-48
SetAllPortImpedances	19-49
SetPortDeembedDistance	19-50
SetPortImpedance	19-51
SetPostProcSettings	19-52
Smooth	19-53
Stretch (SPISim)	19-55
Terminate	19-56
20 - ComInstance Script Commands	(multiple)
Callback Scripting Using ComInstance Object	(multiple)
ComInstance Functions	60
GetComponentName	60
GetInstanceID [Component Instance]	61
GetInstanceName [Component Instance]	62
GetParentDesign	62
GetPropServerName	63
21 - Definition Manager Script Commands	21-1
Add [component manager]	21-2
AddDataset	21-20
AddDefinitionFromBlock	21-23
AddMaterial	21-28
Add [padstack manager]	21-32
Add [symbol manager]	21-39
AreMaterialPropertiesEqual	21-49
AreSurfaceMaterialPropertiesEqual	21-50
CloneMaterial	21-51
DeleteDataset	21-52

DoesMaterialExist	21-53
Edit [component manager]	21-54
EditDataset	21-81
EditMaterial	21-83
Edit [padstack manager]	21-92
ExportDataset	21-99
Export [footprint manager]	21-100
ExportMaterial	21-101
Export [padstack manager]	21-102
ExportScript	21-103
Export [symbol manager]	21-103
GetProjextMaterialNames	21-105
GetPropertyValue	21-105
ImportDataset	21-107
Note About File Types:	21-108
Remove [component manager]	21-109
Remove [footprint manager]	21-111
RemoveMaterial	21-112
Remove [padstack manager]	21-113
RemoveScript	21-114
Remove [symbol manager]	21-115
RemoveUnusedDefinitions	21-116
SetPropertyValue	21-118
UpdateDefFromBlock	21-119
UpdateDefFromBlockEx	21-121
UpdateDefinitions	21-124
Component Manager Script Commands	21-125
Add [component manager]	21-126
AddNPortData [component manager]	21-144
AddSolverOnDemandModel	21-150

ClearSolutionCache [component manager]	21-150
Edit [component manager]	21-151
EditSolverOnDemandModel	21-178
EditWithComps [component manager]	21-178
Export [component manager]	21-190
GetData [component manager]	21-191
GetNames [component manager]	21-192
GetNPortData [component manager]	21-193
GetSolverOnDemandData	21-196
GetSolverOnDemandModelList	21-196
IsUsed [component manager]	21-196
Remove [component manager]	21-197
RemoveSolverOnDemandModel	21-198
RemoveUnused [component manager]	21-198
Update Dynamic Link [component manager]	21-199
Add [footprint manager]	21-200
Edit [footprint manager]	21-223
EditWithComps [footprint manager]	21-223
Export [footprint manager]	21-241
GetData [footprint manager]	21-242
GetNames [footprint manager]	21-243
IsUsed [footprint manager]	21-243
Remove [footprint manager]	21-244
RemoveUnused [footprint manager]	21-245
Material Manager Script Commands	21-246
GetData [material manager]	21-246
GetNames [material manager]	21-247
GetProperties [material manager]	21-248
IsUsed [material manager]	21-249
RemoveUnused [material manager]	21-250

Model Manager Script Commands	21-250
Add [model manager]	21-251
ConvertToDynamic	21-262
ConvertToParametric	21-262
Edit [deprecated]	21-262
EditWithComps [model manager]	21-263
Export [model manager]	21-274
GetData [model manager]	21-275
GetNames [model manager]	21-276
IsUsed [model manager]	21-277
Remove [model manager]	21-278
RemoveUnused [model manager]	21-279
Network Data Explorer Manager Script Commands	21-280
ExportFullWaveSpice	21-281
ExportNetworkData	21-282
ExportNMFData	21-285
Add [padstack manager]	21-285
Edit [padstack manager]	21-291
EditWithComps [padstack manager]	21-298
Export [padstack manager]	21-305
GetData [padstack manager]	21-306
GetNames [padstack manager]	21-307
IsUsed [padstack manager]	21-308
Remove [padstack manager]	21-308
RemoveUnused [padstack manager]	21-309
Script and Library Scripts	21-310
AddScript	21-310
EditScript	21-311
ExportScript	21-313
ModifyLibraries	21-313

RemoveScript	21-315
Symbol Manager Script Commands	21-316
Add [symbol manager]	21-317
BringToFront [symbol manager]	21-327
Edit [deprecated]	21-328
EditWithComps [symbol manager]	21-328
Export [symbol manager]	21-339
GetData [symbol manager]	21-340
GetNames [symbol manager]	21-341
IsUsed [symbol manager]	21-342
Remove [symbol manager]	21-342
RemoveUnused [symbol manager]	21-343
22 - Core Global Script Context Commands	22-1
AddErrorMessage	22-1
AddFatalMessage	22-2
AddInfoMessage	22-2
AddWarningMessage	22-3
LogDebug	22-3
LogError	22-4
23 - Example Scripts	23-1
VBScript Example Scripts	23-1
Data Export Script	23-1
Variable Helix Script	23-4
GetPropNames and GetPropValues for Layered Impedance Boundary Script	23-8
IronPython Example Scripts	23-10
BH Coordinates Python Script	23-11
Script Contents	23-11
Posco_BH_Curve.tab Contents	23-13
Equation Based Curve Python Script	23-16
HFSS Waveguide Array Python Script	23-18

Index Index-1

1 - Introduction to Scripting

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

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

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

The following sections contain more information about scripting:

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

Scripting Help Conventions

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

[ScriptName]

[Description of script use.]

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

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

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

Variable Types

The following data types are used throughout the help:

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

Introduction to VBScript

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

This chapter provides an overview of key VBScript components.

[Simple and Composite Names](#)

[VBScript Variables](#)

[VBScript Operators](#)

[Controlling Program Execution](#)

[Looping Through Code](#)

[VBScript Procedures](#)

[Converting Between Data Types](#)

[Including Scripts](#)

[Aborting Scripts](#)

[Interacting with a Script](#)

[Recommended VBScript References](#)

[Sample HFSS Script](#)

[Sample Circuit Script](#)

[Sample Q3D Script](#)

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

Simple and Composite Names

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

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

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

VBScript Variables

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

Declaring Variables

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

```
Dim box_xsize
```

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

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

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

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

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

Declaring Variables in Python

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

You can directly assign information to it. For example:

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

Variable Naming Conventions

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

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

Scope and Lifetime of Variables

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

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

Array Variables

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

```
Dim Primitives(2)
```

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

```
Primitives(0) = "Box1"
```

```
Primitives(1) = "Cone1"
```

```
Primitives(2) = "Cylinder1"
```

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

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

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

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

Note:

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

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

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

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

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

VBScript Operators

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

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

Operator Precedence

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

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

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

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

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

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

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

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

Arithmetic Operators

Following is a list of VBScript's arithmetic operators:

Symbol	Description
^	Exponentiation
-	Unary negation
*	Multiplication
/	Division
\	Integer division
Mod	Modulus arithmetic
+	Addition
-	Subtraction

Note:

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

String Concatenation Operator

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

Example:

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

produces the following resultant string:

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

Comparison Operators

Following is a list of VBScript's comparison operators:

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

Note:

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

Logical Operators

Following is a list of VBScript's logical operators:

Symbol	Description
Not	Logical negation
And	Logical conjunction
Or	Logical disjunction
Xor	Logical exclusion
Eqv	Logical equivalence
Imp	Logical implication

Note:

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

Controlling Program Execution

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

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

Using If...Then...Else

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

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

Using Select Case

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

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

Looping Through Code

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

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

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

Using a For...Next Loop

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

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

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

Using a Do Loop

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

Repeating Statements While a Condition is True

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

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

Repeating a Statement Until a Condition Becomes True

Following is the syntax:

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

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

VBScript Procedures

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

Function Procedures

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

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

Sub Procedures

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

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

Converting Between Data Types

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

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

Including Scripts

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

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

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

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

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

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

Aborting Scripts

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

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

Interacting with a Script

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

the `InputBox` function and the `MsgBox` function.

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

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

The last two arguments to the function are optional.

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

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

Recommended VBScript References

Microsoft Corporation. *VBScript User's Guide*.

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

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

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

Sample HFSS Script

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

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

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

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

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

```
Dim oDesign
```

```
Dim oEditor
```

```
Dim oModule
```

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

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

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

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

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

```
oDesktop.NewProject
```

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

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

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

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

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

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

' the Sub procedure InsertDesign of the object oProject.

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

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

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

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

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

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

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

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

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

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

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

' character:

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

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

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

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

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

Sample Q3D Extractor Script

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

```
Dim oAnsoftApp
```

```
Dim oDesktop
```

```
Dim oProject
```

```
Dim oDesign
```

```
Dim oEditor
```

```
Dim oModule
```

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

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

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

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

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

```
oDesktop.NewProject
```

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

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

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

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

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

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

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

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

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

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

Introduction to IronPython

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

Scope

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

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

Python compatibility

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

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

Advantages of IronPython

The advantages that IronPython use provides are significant:

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

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



```

IronPython Command Window

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

```

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

[Introduction to IronPython](#)

[IronPython Mini Cookbook](#)

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

[Scripting Using Iron Python](#)

[Standalone IronPython and Desktop IronPython](#)

[IronPython Examples](#)

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

IronPython Mini Cookbook

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

Comments

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

Assigning/Creating Variables

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

Create Lists/Arrays

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

Create Dictionaries/Maps

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

Boolean Values

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

Converting Numbers to Strings and Vice Versa

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

String Formatting/Concatenation

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

Looping over Lists

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

Looping over a Range

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

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

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

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

Note:

Python recommends the use of spaces over tabs.

Indenting Functions

Define a function that starts at 0 indentation:

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

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

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

```
    return a
```

Indenting If Conditions

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

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

```
If a%2 == 0:
```

```
    return (a * b) + 100
```

```
else:
```

```
    return (a * b) + 1000
```

Methods in IronPython

Finding Methods

To list all methods available in the [string module](#), import the module:

```
import string
```

Then get the directory listing:

```
dir(string)
```

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

Help

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

Related Topics

[Introduction to IronPython](#)

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

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

[IronPython Samples](#)

Translating Script Commands from VBScript to IronPython

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

Related Topics

[Script Method Argument](#)

[VBscript Method Call Types](#)

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

[Introduction to IronPython](#)

[IronPython Mini Cookbook](#)

[Scripting Using Iron Python](#)

[IronPython Samples](#)

Script Method Argument

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

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

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

- Primitive Types
- Named Arrays
- Named Functions

Primitive Types

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

Named Arrays

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

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

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

Named Functions

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

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

Related Topics

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

VBScript Method Call Types

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

VBScript Functions

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

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

```
Set oProject = oDesktop.NewProject
```

Note:

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

VBScript Sub-Routines

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

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

Related Topics

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

Converting VBScript Function calls to IronPython Syntax

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

So:

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

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

Return Values

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

Primitive Method Arguments

Replace each VBScript primitive with an equivalent IronPython primitive.

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

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

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

Named Array Arguments

Formatting helps readability immensely but is not required.

All that *must* be done is:

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

Named Array Values with All Key Value Pairs

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

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

Named Arrays with Nested Named Arrays

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

["NAME:name",

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

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

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

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

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

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

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

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

Function Blocks

Function blocks in VBScript argument syntax are represented as arrays without the "NAME:..." field. However, functions are always introduced by a key in a parent structure. Function blocks

can therefore never exist as a top-level argument. They are only found as the value pairs inside a named array or inside another function block.

Important:

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

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

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

Scripting Using Iron Python

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

Translating a script in VBScript to IronPython

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

Writing an IronPython script from scratch

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

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

[IronPython Script Execution Environment](#)

[Scripting using Embedded VBScript or JavaScript](#)

[Scripting with IronPython](#)

[Standalone IronPython and Desktop IronPython](#)

Related Topics

[Introduction to IronPython](#)

[IronPython Mini-cookbook](#)

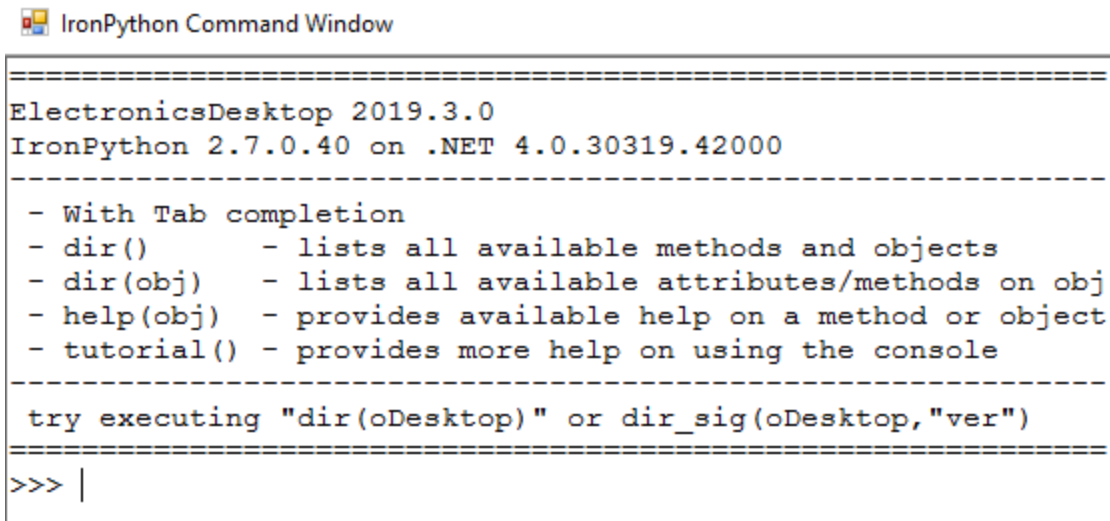
[Translating Script commands from VBScript to IronPython](#)

[Appendix: IronPython Samples](#)

IronPython Script Execution Environment

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

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



```
IronPython Command Window

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

- **Tools > Run Script** menu item, select "IronPython" from the file type drop-down list.
- Launch the product with a script argument.
- Register an IronPython script as an external tool using the **Tools > External Tools** menu item.

When desktop executes a script, it does so in an execution environment setup with predefined variables and functions. These predefined variables and functions are how the script communicates with the desktop, and they come in four flavors addressed in the following subtopics:

[Script Argument for IronPython](#)

Script Argument for IronPython

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

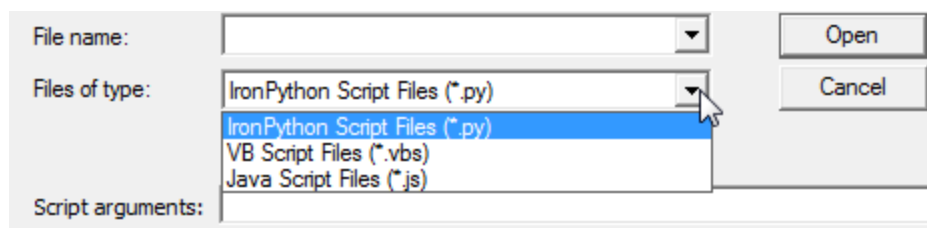


Figure 1: Run Script dialog and script arguments

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

Related Topics

[IronPython Script Execution Environment](#)

Scripting using Embedded VBScript or JavaScript

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

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

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

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

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

Set oAnsoftApp = CreateObject("Ansoft.ElectronicsDesktop")
Set oDesktop = oAnsoftApp.GetAppDesktop()
oDesktop.RestoreWindow

Set oProject = oDesktop.GetActiveProject()
oProject.InsertDesign "HFSS", "HFSSPyTestDesign", "DrivenModal", ""
Set oDesign = oProject.SetActiveDesign("HFSSPyTestDesign")
Set oEditor = oDesign.SetActiveEditor("3D Modeler")
oEditor.CreateCone Array("NAME:ConeParameters", _
    "XCenter:=", "0mm", "YCenter:=", "0mm", "ZCenter:=", "0mm", _
    "WhichAxis:=", "Z", "Height:=", "2mm", _
```

```
"BottomRadius:=", "3mm", _
"TopRadius:=", "0mm"), Array("NAME:Attributes", "Name:=", _
"Cone1", "Flags:=", "", "Color:=", "(132 132 193)", "Trans-
parency:=", 0, _
"PartCoordinateSystem:=", "Global", "UDMId:=", "", "Mater-
ialValue:=", _
"" & Chr(34) & "vacuum" & Chr(34) & "", "SolveInside:=", _
true)
""
```

```
SetScriptingLanguageToVBScript()
```

```
RunScriptCommand(coneScript)
```

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

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

```
#import the CLR references
import clr
clr.AddReference("System.Windows.Forms")

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

# the GUI form to show some text
# the class below derives from Form (System.Windows.Forms.Form)
# imported above from the .NET assembly.
class ShowPropertiesForm(Form):
    def __init__(self, name, text):
        self.Name = name
        self._label = Label()
```

```
self._label.Text = text
self._label.Dock = DockStyle.Fill

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

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

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

#-----
# Main script code
#-----
#display the ScriptArgument variable as the text label
# in the form.
gui = ShowPropertiesForm("Sample Form", ScriptArgument)

# This makes it a modal dialog.
gui.ShowDialog()

# the following will make it a non-modal dialog
#Application.Run(gui)
```

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

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

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

Related Topics

[IronPython Script Execution Environment](#)

Scripting with IronPython

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

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

Note:

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

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

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

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

Related Topics

[IronPython Script Execution Environment](#)

[Standalone IronPython and Desktop IronPython](#)

Standalone IronPython

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

Running Standalone IronPython

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

It is located in:

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

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

```
"C:\Program Files\ANSYS Inc\v251\AnsysEM\common\IronPython\ipy64.exe"  
"<filePath>\myScript.py"
```

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

Using a Recorded Script

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

Creating an External Script

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

- `import sys`

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

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

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

- `import ScriptEnv`

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

or `ScriptEnv.InitializeNew(NonGraphical=True)`

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

You must end the script with:

- `ScriptEnv.Shutdown()`

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

Example Script

```
import sys

sys.path.append(r"C:\Program Files\ANSYS Inc\v251\AnsysEM")

sys.path.append(r"C:\Program Files\ANSYS Inc\v251\An-
sysEM\PythonFiles\DesktopPlugin")

import ScriptEnv

ScriptEnv.Initialize("Ansoft.ElectronicsDesktop")

oDesktop.RestoreWindow()

oProject = oDesktop.NewProject()

oProject.InsertDesign("HFSS", "HFSSDesign1", "DrivenModal", "")

oDesign = oProject.SetActiveDesign("HFSSDesign1")

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

oEditor.CreateRectangle(

[

    "NAME:RectangleParameters",

    "IsCovered:= ", True,

    "XStart:= ", "-0.2mm",
```

```

        "YStart:= ", "-3mm",
        "ZStart:= ", "0mm",
        "Width:= ", "0.8mm",
        "Height:= ", "1.2mm",
        "WhichAxis:= ", "Z"
    ],
    [
        "NAME:Attributes",
        "Name:= ", "Rectangle1",
        "Flags:= ", "",
        "Color:= ", "(132 132 193)",
        "Transparency:= ", 0,
        "PartCoordinateSystem:= ", "Global",
        "UDMId:= ", "",
        "MaterialValue:= ", "\"vacuum\"",
        "SolveInside:= ", True
    ])
oDesign.SetDesignSettings(['NAME:Design Settings Data', 'Allow Material Override:=', True, 'Calculate Lossy Dielectrics:=', True])
oEditor.SetModelUnits(['NAME:Units Parameter', 'Units:=', 'mil', 'Rescale:=', False ])
ScriptEnv.Shutdown()

```

IronPython Samples

Change property

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

```

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

```

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

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

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

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

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

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

Create a Cone using IronPython

Most scripting tasks using IronPython are expected to be formatted as the following example. One starts with the predefined **oDesktop** object and drills down to the design, editors, modules

etc and issues any required commands on the object while formatting the script command arguments in natural python syntax.

```
oProject = oDesktop.GetActiveProject()
oDesign = oProject.InsertDesign("HFSS", "Random", "DrivenModal", "")
oEditor = oDesign.SetActiveEditor("3D Modeler")
oEditor.CreateCone(
{
    "NAME" : "ConeParameters",
    "XCenter" : "0mm",
    "YCenter" : "0mm",
    "ZCenter" : "0mm",
    "WhichAxis" : "Z",
    "Height" : "2mm",
    "BottomRadius" : "1.56204993518133mm",
    "TopRadius" : "0mm"
},
{
    "NAME" : "Attributes",
    "Name" : "Cone1",
    "Flags" : "",
    "Color" : "(132 132 193)",
    "Transparency" : 0,
    "PartCoordinateSystem": "Global",
    "UDMId" : "",
    "MaterialValue" : "\"vacuum\"",
    "SolveInside" : True
}
)
```

Sample Script 16: IronPython script to create a cone

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

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

```
oProject = oDesktop.GetActiveProject()

oDesign = oProject.InsertDesign("HFSS", "Hersheys 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 [UDMFunctionLibrary](#) listed further in this document. A separate chapter includes a [UDP IronPython example of fillet and chamfer](#).

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

- functions names in UDPFunctionLibrary and UDMFunctionLibrary are capitalized
- arrays become a python list of objects
- `void * callback parameter` is dropped from the parameter list
- output parameters (pointer types that are filled during the function call) usually become

return values

- 'list size' parameter usually will be omitted as redundant

Output Parameters

The rule for the output parameters is as follows:

- If the function has one output parameter variable and no return value, the variable will become function's return value. The same will happen if the return value is a 'success/failure' boolean ('None' will be returned on failure and parameter variable - on success).
- If the function has one output parameter and a return value, the function will return a Python tuple where function return value will be the first one in the tuple.
- If there is more than one out variable, the function will return a Python tuple with all output parameters in the specified order. If function has a return value, it must always be the first in the tuple.

one output parameter; return value is ignored

```
udmDefinition = udmFunctionLibrary.GetDefinition()
```

**# one output parameter; return value must be preserved. return
and output values are packed into the return tupe, in order**

```
(lRet, partIdsList) = udpFunctionLibrary.DetachFaces(nPartIds, 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); where udmDefinition is an output parameter</pre>	<pre>udmDefinition = udmFunctionLibrary.GetDefinition() (Note: callbackData is omitted in py interface)</pre>
<pre>long detachIFaces(int nFacesAndPartIds,</pre>	<pre>(bRet, partIds) = udmFunctionLibrary.DetachIFaces (nFacesAndPartIds, faceIds)</pre>

C	Python
<pre>long* facIds, long* partIds, void* callbackData); where partIds is an output parameter</pre>	(Note: callbackData is omitted in py interface)

'List Size' Parameters

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

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

Example:

input list, input list size

```
lret = udpFunctionLibrary.Unite(objectIds)
```

output list, output list size

```
faceIdList = udmFunctionLibrary.GetAllFaces(PartId)
```

output list, input list size

```
(lret, partIdList) = udpFunctionLibrary.DetachFaces(listSize,
faceIdList)
```

Comparison with C function:

C	Python
<pre>bool getAllFaces(long partId, long* numFaces, long** facIds, void* callbackData);</pre>	<pre>facIds = udmFunctionLibrary.GetAllFaces(partId)</pre> (ignore numFaces as redundant: fol-

C	Python
where numFaces and facelds are output parameters and numFaces is the size of faceld.	ded into facelds, return value is omitted: folded into the facelds is None check callbackData is omitted)
long unite(long numObjects, long* objectIds, void* callbackData); where numObjects and objectIds are input parameters and numObjects is the size of objectIds.	lret = udpFunctionLibrary.Unite(objectIds) (ignore numObjects as redundant: folded into objectIds callbackData is omitted)
long detachFaces(long nSize, long* facelds, long* partIds, void* callbackData); where partIds is and output list and nSize is an input parameters and nSize is the size of partIds.	(lret, partIdList) = udpFunctionLibrary.DetachFaces(nSize, facelds) (nSize is not ignored, callbackData is omitted)

Added Parameters

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

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

Example:

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

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

Here cloneIdsSize is a new integer parameter.

Comparison with C function:

C	Python
long duplicateAlongLine(	(lret, cloneIds) = udmFunctionLibrary.DuplicateAlongLine(partId,

C	Python
<pre>long partId, UDPVector transVector, int nClones, long* nClones, void* callbackData);</pre>	<pre>transVec, nClones, clonesSize) (callbackData is omitted clonesSize is a new parameter)</pre>
<pre>long duplicateAroundAxis(long partId, UDPCoordinateSystemAxis axis, double angle, int nClones, long* nClones, void* callbackData);</pre>	<pre>(lret, clones) = udmFunctionLibrary.DuplicateAroundAxis (partId, axis, angle, nClones, clonesSize) (callbackData is omitted clonesSize is a new parameter)</pre>

Developing a UDM/UDP

Creation

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

Location

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

Draw > User Defined Primitives for UDP or **Draw > User Defined Model for UDM**.

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

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

Organize

"Lib" sub-directory is a special directory. The contents of this directory is not shown in the menu. In the "Lib" directory you can create Python scripts with base classes and utilities to be used in UDP/UDM Python scripts. All the Lib directories upstream of a script (till the UserDefinedModels

or UserDefinedPrimitives) are included in the Python search path and this allows for easy import of helper modules in such directories.

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

For UDM:extension:

```
from UDM import *
```

For UDP:extension:

```
from UDP import *
```

Edit/Reload

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

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

UDPEExtension

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: UDPEExtension

You must write a class derived from IUDPEExtension with a mandatory name UDPEExtension:

```
class UDPEExtension(IUDPEExtension):
```

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

IUDPEExtension methods

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

Mandatory methods.

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

GetLengthParameterUnits()

- returns string.

GetPrimitiveTypeInfo()

- returns UDPPrimitiveTypeInfo.

GetPrimitiveParametersDefinition2()

- returns a list of UDPPrimitiveParameterDefinition2 or None on failure

AreParameterValuesValid2(errorMsg, udpParams)

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

CreatePrimitive2(funcLib, udpParams)

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

Optional methods

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

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

MapParametersDefinitionVersions2(oldVersion, oldUDPParams)

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

GetOldPrimitiveParametersDefinition2(version)

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

Example UDP

```
import sys

class UDPExtension(IUDPExtension):

    def GetLengthParameterUnits(self):
        return "mm"

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

        return typeInfo

...

...
```


UDMExtension

Import

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

However you have to add import statements to your helper scripts in your Lib sub-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, udmInParams, updatedParams, refreshModifiedPartsOnly, nonEditedPartRefs)

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 udmlnParams to updatedParams.
- refreshModifiedPartsOnly is a boolean

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

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

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

returns True on success, False on failure.

ReleaseInstance(instanceId)

- instanceId 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 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 the name of the attribute containing entity IDs.
- For example, it can return a unique string such as "ATTRIB_XACIS_ID" (Can be same as GetAttribNameForEntityId())
- Python UDMs should implement this method.

Optional methods

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

DialogForDefinitionOptionsAndParams(self, defData, optData, params):

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

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

DialogForDefinitionAndOptions(self, defData, optData) [Deprecated]

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

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

GetInstanceSourceInfo(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 = **GetInstanceld()**
7. **string:** str = **GetPartRefld(int: partId)**
8. **bool:** bret = **SetPartRefld(int: partId, string: refld)**
9. **List_of_int:** facelds = **GetAllFaces(int: partId)**
10. **List_of_int:** edgelds = **GetAllEdges(int: partId)**
11. **List_of_int:** vertexlds = **GetAllVertices(int: partId)**
12. **bool:** bret = **SetFaceAttribs(List_of_int: facelds, List_of_string: attribs)**
13. **bool:** bret = **SetEdgeAttribs(List_of_int: edgelds, List_of_string: attribs)**
14. **bool:** bret = **SetVertexAttribs(List_of_int: vertexlds, List_of_string: attribs)**
15. **string:** str = **GetModelerUnit()**
16. **string:** str = **GetCacheFileForUDMResume()**
17. **bool:** bret = **SetPartColor(int: partId, int: nColor)**
18. **bool:** bret = **SetPartFlags(int: partId, int: nFlags)**
19. (**bool:** bret, **UDPPosition:** low, **UDPPosition:** high) = **GetBoundingBox(int: partId, bool: exact)**
20. **bool:** bret = **IsParametricUpdate()**
21. **bool:** bret = **SetMaterialNameByRefld(string: partRefld, string: matName)**
22. **bool:** bret = **SetPartNameByRefld(string: partRefld, string: partName)**
23. **bool:** bret = **SetPartColorByRefld(string: partRefld, int: nColor)**
24. **bool:** bret = **SetPartFlagsByRefld(string: partRefld, int: nFlags)**

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

Example:

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

UDM/UDP Functions

Return Values for Each UDM and UDP Function

ID – ID of created Object

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

Functions list:

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

UDP/UDM Structures and Constants

The following sections describe:

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

UDP/UDM Structures

Differences compared to C API

- **UDMDefinition**
- **UDMOptions**
- **UDMPParameters**

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 UDMPParameter list):

```
udmParamList = [  
    UDMPParameter(  
        "cubeSizeName", UnitType.LengthUnit,  
        UDPParam(ParamDataType.Double, cubeSize),  
        ParamPropType.Value,  
        ParamPropFlag.MustBeReal),  
    UDMPParameter(  
        "cubeDistanceName", UnitType.LengthUnit,  
        UDPParam(ParamDataType.Double, cubeDistance),  
        ParamPropType.Value,  
        ParamPropFlag.MustBeReal),  
    UDMPParameter("numCubesName", UnitType.LengthUnit,  
        UDPParam(ParamDataType.Int, numCubes),  
        ParamPropType.Number,  
        ParamPropFlag.MustBeInt]
```

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

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

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

Example:

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

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

- **UDP3x3Matrix**

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

Example:

```
rotationMatrix = [0,0,1, 1,0,0, 0,0,1]
udpFunctionLib.Transform(partId, rotationMatrix, translationVector)
```

List of structures

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

Example:

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

Structure	Construction	Members
UDPPrimitiveTypeInfo	UDPPrimitiveTypeInfo(string name, string purpose, string company, string date, string version)	string Name string Purpose string Company string Date string Version
UDPPrimitiveParameterDefinition	UDPPrimitiveParameterDefinition(string name, string purpose, string company, string date, string version)	string Name string Description

Structure	Construction	Members
	string name, string description, UnitType unitType, double defaultValue)	UnitType UnitType double DefaultValue
UDPParam	UDPParam() UDPParam(ParamDataType dataType, object data) object can be int, double, string, bool or UDPPosition methods: setInt(int val) setBool(bool val) setString(string val) setDouble(double val) setPosition(UDPPosition val)	ParamDataType DataType object Data object can be int, double , string, bool or UDPPosition
UDPPrim- itiveParameterDefinition2	UDPPrim- itiveParameterDefinition2(string name, string description, UnitType unitType, ParamPropType propType, ParamPropFlag propFlag, UDPParam defaultValue)	string Name string Description UnitType UnitType ParamPropType PropType ParamPropFlag PropFlag UDPParam DefaultValue
UDPPosition	UDPPosition(double x, double y,	double X double Y double Z

Structure	Construction	Members
	double z)	
UDPVector	UDPVector(double x, double y, double z)	double X double Y double Z
UDPSweepOptions	UDPSweepOptions(SweepDraftType draftType, double draftAngle, double twistAngle)	SweepDraftType DraftType double DraftAngle double TwistAngle
UDPPolylineSegmentDefinition	UDPPolylineSegmentDefinition(PolylineSegmentType segmentType, int segmentStartIndex, int numberOfPoints, double angle, UDPPosition centerPoint, CoordinateSystemPlane arcPlane)	PolylineSegmentType SegmentType int segmentStartIndex, int numberOfPoints, double angle, UDPPosition centerPoint, CoordinateSystemPlane arcPlane)
UDPPolylineDefinition	UDPPolylineDefinition() UDPPolylineDefinition(List_of_UDPPosition positions, List_of_UDPPolylineSegmentDefinition segDefs, int closed, int covered)	int IsClosed int IsCovered List_of_UDPPosition ArrayOfPosition List_of_UDPPolylineSegmentDefinition ArrayOfSegmentDefinition
UDPEquationBasedCurveDefinition	UDPEquationBasedCurveDefinition(string functionXt, string functionYt, string functionZt,	string FunctionXt string FunctionYt string FunctionZt double TStart double TEnd

Structure	Construction	Members
	double tStart, double tEnd, int numOfPointsOnCurve)	int NumOfPointsOnCurve
UDPEqua- tionBasedSurfaceDefinition	UDPEqua- tionBasedSurfaceDefinition(string functionXuv, string functionYuv, string functionZuv, double uStart, double uEnd, double vStart, double vEnd int reserved1 int reserved2)	string FunctionXuv string FunctionYuv string FunctionZuv double UStart double UEnd double VStart double VEnd two integer arguments that are reserved for future use. They need to be provided, for example as 0. For example: theSurfaceDefinition = UDPEqua- tionBasedSur- faceDefinition ("u", "v", "1", 0, 1, 0, 1, 0, 0)
UDPHelixDefinition	UDPHelixDefinition(int profileID, UDPPosition ptOnAxis, UDPPosition axisDir, double noOfTurns, bool isRightHanded, double radi- usChangePerTurn, double pitch)	int ProfileID UDPPosition PtOnAxis UDPPosition AxisDir double NoOfTurns bool IsRightHanded double RadiusChangePerTurn double Pitch
UDPSpiralDefinition	UDPSpiralDefinition(int profileID, UDPPosition ptOnAxis, UDPPosition axisDir, double noOfTurns,	int ProfileID UDPPosition PtOnAxis UDPPosition AxisDir double NoOfTurns bool IsRightHanded

Structure	Construction	Members
	bool isRightHanded, double radiusChangePerTurn)	double RadiusChangePerTurn
UDPBLNDElements	UDPBLNDElements(int partID, int noOfEdges; int* listOfEdges;) UDPBLNDElements(int partID, int noOfVertices; int* listOfVertices;)	UDPBLNDElements can hold either edges or vertices, but not both at the same time. Edges should be applied to solids, and vertices should be applied to sheets. int PartID /* part to be blended i.e. filleted/chamfered */ int noOfEdges; int* listOfEdges; /* edges to be blended */ int noOfVertices; int* listOfVertices; /* vertices to be blended */
UDPBLNDFilletOptions	UDPBLNDFilletOptions(bool suppressFillet, BLNDFilletRadiusLaw filletRadiusLaw, double filletStartRadius, double filletEndRadius, bool followSmoothEdgeSequence, BLNDFilletType filletType, double setbackDistance, double bulgeFactor)	bool SuppressFillet /* Reserved for future */ BLNDFilletRadiusLaw FilletRadiusLaw double FilletStartRadius double FilletEndRadius bool FollowSmoothEdgeSequence /* Reserved for future */ BLNDFilletType FilletType double SetbackDistance double BulgeFactor /* Reserved for future */
UDPBLNDChamferOptions	UDPBLNDChamferOptions(bool suppressChamfer, BLNDChamferRangeLaw chamferRangeLaw,	bool SuppressChamfer BLNDChamferRangeLaw ChamferRangeLaw double ChamferLeftRange

Structure	Construction	Members
	double chamferLeftRange, double chamferRightRange)	double ChamferRightRange
UDPProgress	UDPProgress(int prog, int subProg, string mesg, string subMesg)	int Prog int SubProg string Mesg string SubMesg
UDMInfo	UDMInfo(string name, string purpose, string company, string date, string version)	string Name string Purpose string Company string Date string Version
UDMDefinition	UDMDefinition() UDMDefinition(string name, UDParam value, ParamPropType propType, ParamPropFlag propFlag)	string DefName UDPParam DefValue ParamPropType PropType ParamPropFlag PropFlag
UDMOption	UDMOption() UDMOption(string name, UDParam value, ParamPropType propType, ParamPropFlag propFlag)	string OptName UDPParam OptValue ParamPropType PropType ParamPropFlag PropFlag
UDMParameter	UDMParameter() UDMParameter(	string ParamName UDPParam ParamValue UnitType UnitType

Structure	Construction	Members
	string name, UDParam value, UnitType unitType, ParamPropType propType, ParamPropFlag propFlag)	ParamPropType PropType ParamPropFlag PropFlag

UDP/UDM Constants

Full names of enum constants must be used in scripts.

Example:

```
unitType = UnitType.LengthUnit
```

```
dataType = ParamDataType.Int
```

Enum constants:

enum Constant	Parameters
UnitType	NoUnit LengthUnit AngleUnit
ParamDataType	Int Double String Bool Position Unknown
ParamPropType	Text Menu Number Value FileName Checkbox Position Unknown

enum Constant	Parameters
ParamPropFlag	NoFlag ReadOnly MustBeInt MustBeReal Hidden Unknown
CoordinateSystemAxis	XAxis YAxis ZAxis
CoordinateSystemPlane	XYPlane YZPlane ZXPlane
SweepDraftType	ExtendedDraft RoundDraft NaturalDraft MixedDraft
SplitWhichSideToKeep	SplitKeepBoth SplitKeepPositiveOnly SplitKeepNegativeOnly
PolylineSegmentType	LineSegment ArcSegment SplineSegment AngularArcSegment
MessageSeverity	WarningMessage ErrorMessage InfoMessage IncompleteMessage FatalMessage
BLNDFilletRadiusLaw	BLNDConstantRadius BLNDVariableRadius

enum Constant	Parameters
BLNDFilletType	BLNDRound /* The outward surface of the fillet is curved.*/ BLNDMitered /* The outward surface of the fillet is flat and cut at an angle.*/
BLNDChamferRangeLaw	BLNDConstantRange BLNDVariableRange
PartPropertyFlags	PropNonModel PropDisplayWireFrame PropReadOnly PostprocessingGeometry PropInvisible PropShowDirection PropDummy

UDP Python Example

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

```
import sys

primitive_info = UDPPrimitiveTypeInfo(
    name="Fillet_Chamfer",
    purpose="Fillet Chamfer Example",
    company="Ansys",
    date="09/11/2020",
    version="1.0")

primitive_param_definitions = [
    UDPPrimitiveParameterDefinition2(
        "x_size",
        "",
        UnitType.LengthUnit,
        ParamPropType.Value,
        ParamPropFlag.MustBeReal,
        UDPParam(ParamDataType.Double, 10)),
```

```

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

length_units = "mm"

#####
# Class Implementation
#####

class UDPExtension(IUDPExtension):
    def CreatePrimitive2(self, func_lib, param_values):
        """
        Inbuilt function that is called to generate a UDP after suc-
        cessful validation

        :param func_lib: drawing inbuilt class, see in Help: UDMFunc-
        tionLibrary

        :param param_values: list of udp parameter values (user input)
        generated by UDP Core

        :return: None
        """

```

```
param_dict = self.get_param_dict(param_values)

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

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

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

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

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

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

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

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

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

def GetPrimitiveTypeInfo(self):
    return primitive_info

def GetLengthParameterUnits(self):
    return length_units

def GetPrimitiveParametersDefinition2(self):
    return primitive_param_definitions

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

# Custom Functions
def get_param_value_by_name(self, param_values, param_name):
    """
```

```
Function to get a value of a single parameter accessing it by
name

:param param_values: list of udp parameter values (user input)
generated by UDP Core

:param param_name: name of the parameter as specified in defin-
ition list

:return: Value of the parameter or None if parameter does not
exist
"""

param_dict = self.get_param_dict(param_values)
value = param_dict.get(param_name, None)
return value

def get_param_dict(self, param_values):
    """
    Function to return a dictionary of UDP parameter name and value
    (key: value) pairs

    :param param_values: list of udp parameter values (user input)
    generated by UDP Core

    :return: dict of parameter name and values
    """

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

Introduction to CPython

An Ansys Electronics Desktop 2024R1 feature allows CPython to be used for standalone scripting, as an alternative to IronPython, to:

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

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

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

Important:

(Supported on both Windows and Linux in 24R1)

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

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

Creating an External Script

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

```
import sys

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

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

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

Those lines are followed by:

```
import ScriptEnv

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

Follow that with:

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

- Closing command, as described below.

Start ansysedt as GRPC server

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

-grpcsrv Flag

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

ansysedt -grpcsrv

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

ansysedt -grpcsrv portNumber

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

ansysedt -grpcsrv FirstPortNumber:LastPortNumber

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

ansysedt -grpcsrv FirstPortNumber:NumberOfPorts

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

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

Connect Functions:

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

Example:

```
import sys

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

import ScriptEnv
```



```
ScriptEnv.Connect(r"c:\myProjects\Project1.aedt")
oProject = oDesktop.GetActiveProject()
print(oProject.GetName())
```

Launching Electronics Desktop

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

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

Where:

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

Note:

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

Connecting with a Running Instance of Electronics Desktop

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

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

Where:

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

Closing Electronics Desktop/Ending the Script

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

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

-grpcsrv Flag

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

On Windows:

```
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](#)

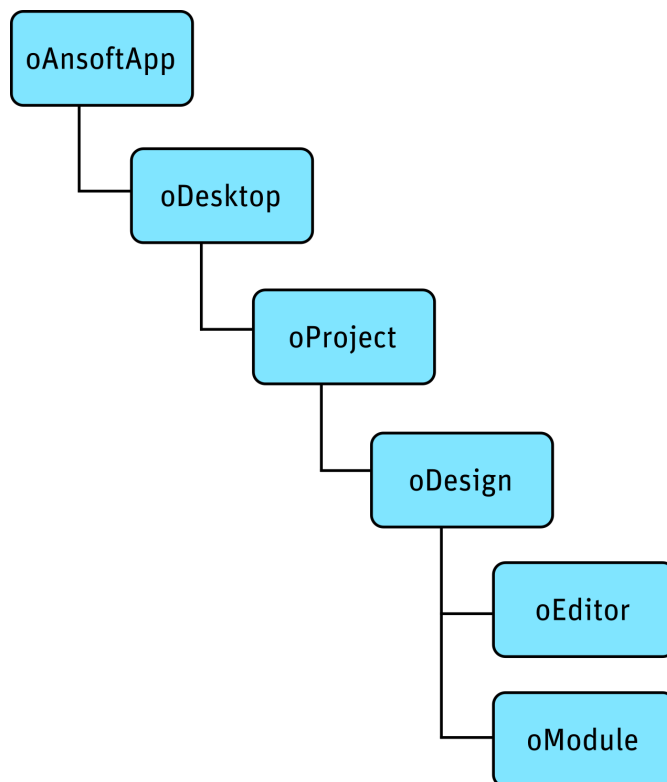
[Working with Project Scripts](#)

[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()
```

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

- [Desktop Object Script Commands](#).

oProject

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

In Iron Python, for example:

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

In VBscript, for example:

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

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

- [Project Object Script Commands](#)

oDesign

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

In Iron Python, for example:

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

In VBscript, for example:

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

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

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

oEditor

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

In Iron Python, for example:

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

In VBscript, for example:

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

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

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

Important:

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

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

oModule

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

In IronPython, for example:

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

In VBscript, for example:

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

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

- Analysis Module Script Commands
- Boundary and Excitation Module Script Commands
- Field Overlays Module Script Commands
- Mesh Operations Module Script Commands

- Optimetrics Module Script Commands
- Radiation Module Script Commands
- Reduce Matrix Module Script Commands
- Solutions Module Script Commands

Example Script Opening

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

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

In VBScript, this would be:

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

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

GetActiveProject and GetActiveDesign for Wider Use

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

In IronPython:

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

In VBScript:

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

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

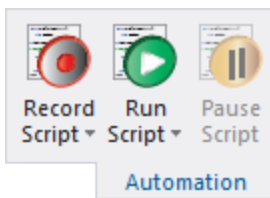
Running a Script

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

Within Electronics Desktop

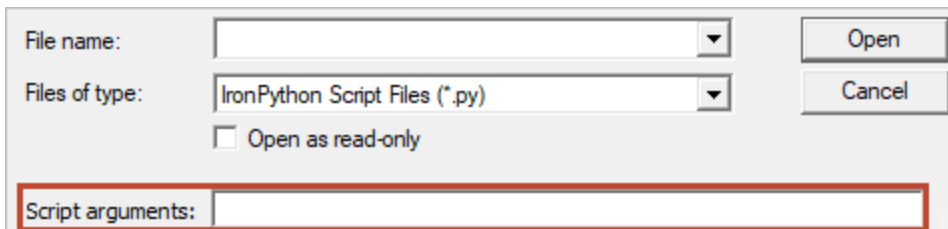
To run scripts in Electronics Desktop:

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



The **Run Script** file browser appears.

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



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

4. Click **Open**.

Electronics Desktop executes the script.

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

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

From the Command Line

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

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

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

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

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

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

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

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

Direct Launch

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

```
on error resume next
dim args
Set args = AnsoftScript.arguments
if(IsEmpty(args)) then
Set args = WSH.arguments
End if
on error goto 0

// At this point, args has the arguments regardless of collection
msgbox "Count is " & args.Count
for i = 0 to args.Count - 1
msgbox args(i)
next
```

Recording a Script

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

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

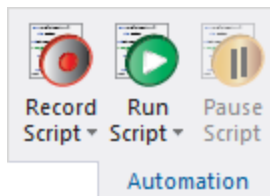
Important:

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

Recording a Script to File

To record a script to file:

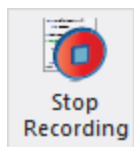
1. Click **Tools > Record Script to File**, or select the **Automation** tab and click the **Record Script** icon:



A **Save As** file browser appears.

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

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



6. Perform the steps you want to record.

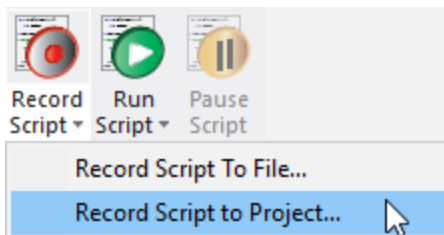
7. When you have finished recording the script, click **Stop Recording**, or select **Tools > Stop Script Recording**.

The recorded script is saved to the folder you specified.

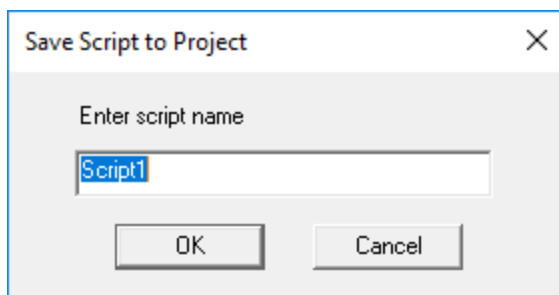
Recording a Script to a Project

To record a script to a project:

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



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



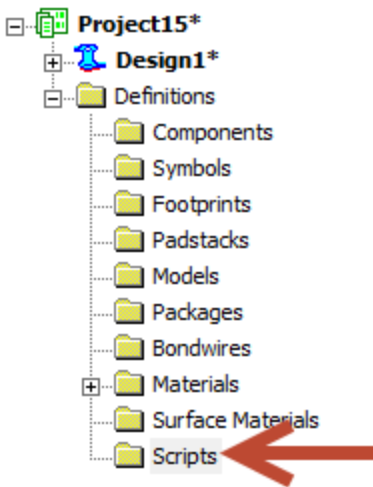
2. Enter a name for the script in the text box, then click **OK**.
3. Perform the steps you want to record.
4. When you have finished, click **Stop Recording**, or select **Tools > Stop Script Recording**.

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

Working with Project Scripts

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

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



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



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

Note:

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

Executing a Script from Within a Script

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

In IronPython:

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

In VBScript:

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

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

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

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

Electronics Desktop Scripting Conventions

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

Consult the following topics:

- [Named Arguments](#)
- [Setting Numerical Values](#)
- [Event Callback Scripting](#)

Named Arguments

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

1. Named data, where name precedes data.

For example:

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

2. Named Array, where name precedes array.

For example:

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

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

For example:

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

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

The names are used both to identify what the data means to you and to inform Electronics Desktop which data is being given. The names must be included or the script will not play back correctly. However, if you are writing a script, you do not need to pass in every piece of data that

the command can take. For example, if you are modifying a boundary, the script will be recorded to include every piece of data needed for the boundary, whether or not it was modified. If you are writing a script by hand, you can add only the data that changed and omit anything that you do not want to change. Electronics Desktop will use the names to determine which data you provided.

VBscript Example

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

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

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

```
oModule.EditImpedance "Imped1", Array("NAME:Imped1",  
    "Resistance:=", "100")
```

IronPython Example

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

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

```
"IsGapSource:=", true,  
"UpperProbe:=", false,  
"LayoutObject:=", "Port1",  
"Pin:=", "",  
"ReferencePort:=", ""  
]  
)
```

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

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

Setting Numerical Values

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

- Pass in the number directly.

In IronPython:

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

In VBScript:

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

- Pass in a string containing the number with units.

In IronPython:

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

In VBScript:

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

- Pass in a variable name.

In IronPython:

```
var = 3.5

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

In VBScript:

```
dim var

var = "3.5V"

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

Layout Scripts and the Active Layer

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

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

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

Scripts and Locked Layers

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

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

Event Callback Scripting

Event Callback scripting allows you to define custom JavaScript and VBScript routines that will run automatically after detecting a triggering event, such as placing a component or running a

simulation. When defining an Event Callback script, specify one or more scripts that will be run after a particular event is detected.

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

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

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

The function, **ExecuteAnsoftScript(<ScriptName>)**, searches the configured Ansys Electronics Desktop script libraries by name for the script passed to it, and invokes the found `ScriptName`. The invoked script will run with the same set of visible script items as the originally

called script. That is, **ComplInstance** is visible from the invoked sub-script, ScriptName, and ComplInstance's functions can be called from ScriptName.

PyAEDT (Beta)

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

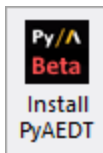
Documentation for PyAEDT can be found online at: <https://aedt.docs.pyansys.com/version/stable/>

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

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

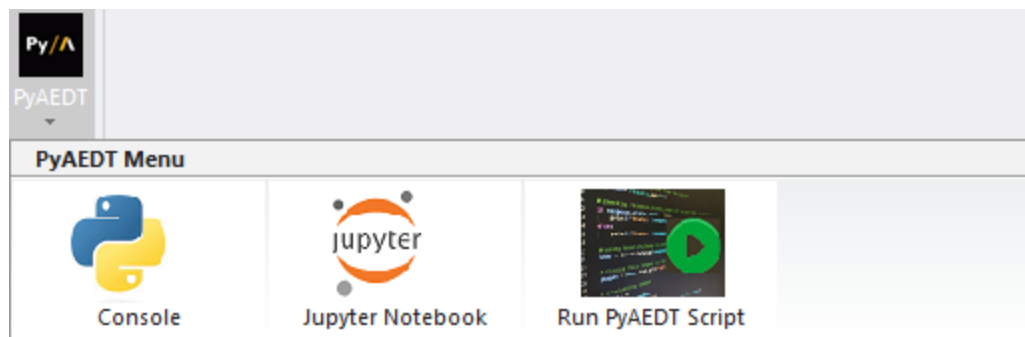
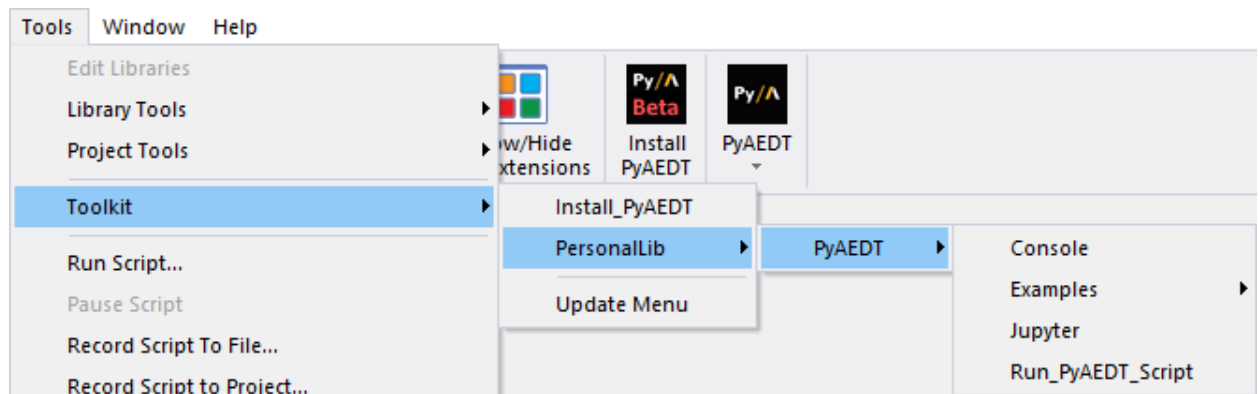
To install PyAEDT:

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



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

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



2 - Object-Oriented Property Scripting

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

This topic covers the following:

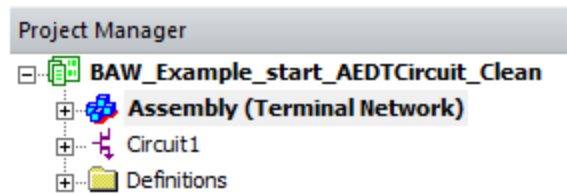
- Logic/syntax
- Basic attributes of project and design properties
- Example Scripts

Object-Oriented Scripting

There are five basic functions in object-oriented scripting for retrieving and setting properties:

1. GetChildNames()
2. GetChildObjects()
3. GetPropNames()
4. GetPropValue()
5. SetPropValue()

At a high level, use GetChildNames() to determine what object instances exist for a given object. An example is shown below to demonstrate for an AEDT Project shown that has two Designs.



If you open the Command Window that allows for executing python code, you first define the Project Object, oProject, and the Design Object, oDesign, as shown:

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

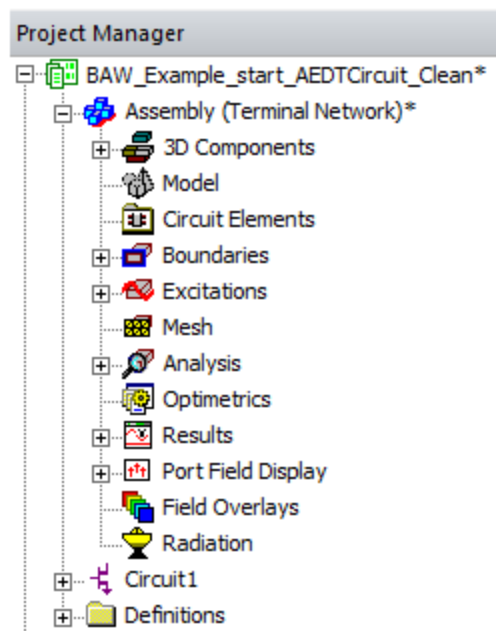
Once the objects have been defined, you can use GetChildNames() to learn what object instances exist for each. As an example, observe the Child Names of oProject, and you see a list of the Designs in the AEDT Project, per the GUI.

```
>>> oProject.GetChildNames()  
['Assembly', 'Circuit1']
```

As another example, retrieve the names of the Object Instances available in oDesign, to see the various objects associated with a Design setup:

```
>>> oDesign.GetChildNames()  
['Boundaries', 'Excitations', 'Circuit Elements', 'Model', 'Mesh', 'Analysis', 'Optimetrics', 'Port  
Field Display', 'Field Overlays', 'Radiation', 'Results', '3D Modeler']
```

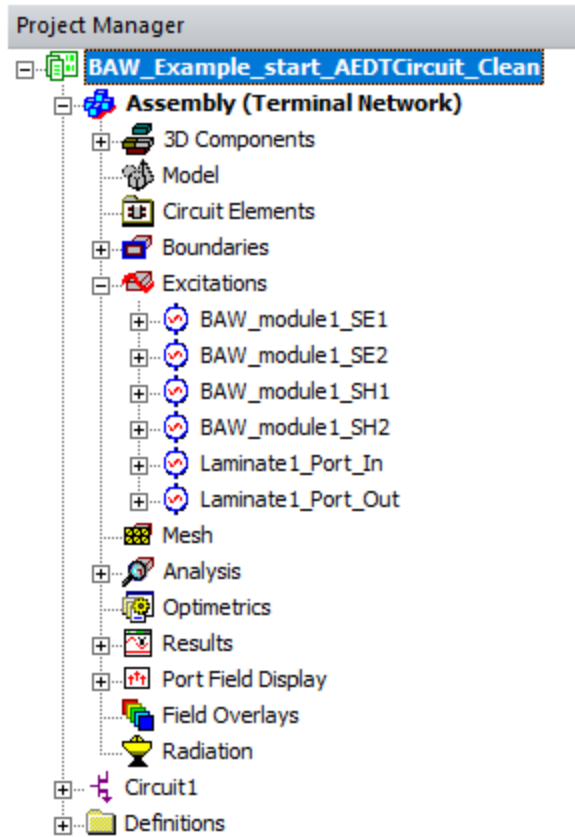
These names are what you would expect based on the Project Manager Layout:



In the **Project Manager** above, any object that has the '+' symbol is populated with children that you can query. Once you know the name of the object you want, you can instance it via the `GetChildObject()` command. This defines the instance to the desired object. As an example, set an instance for the Excitations:

```
>>> oExcitations = oDesign.GetChildObject('Excitations')
```

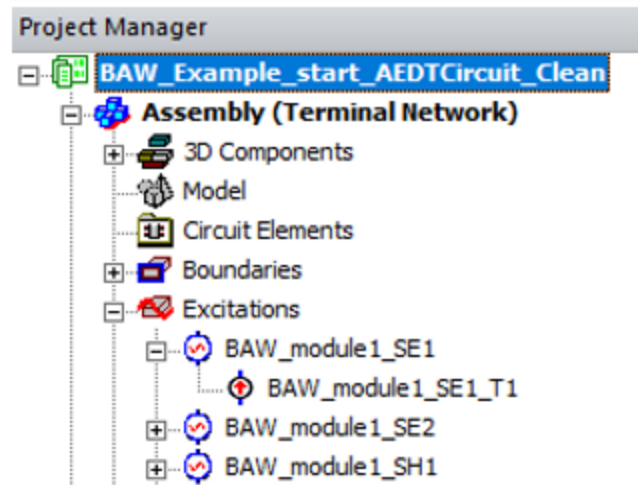
You now have an object, oExcitations defined to be the oDesign Child Object 'Excitations'. What does this mean? If you expand the Excitations dialogue in the Project Manager, you expect that the Child Names of this object would be the names of the Excitations as defined, in this case six ports:



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


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

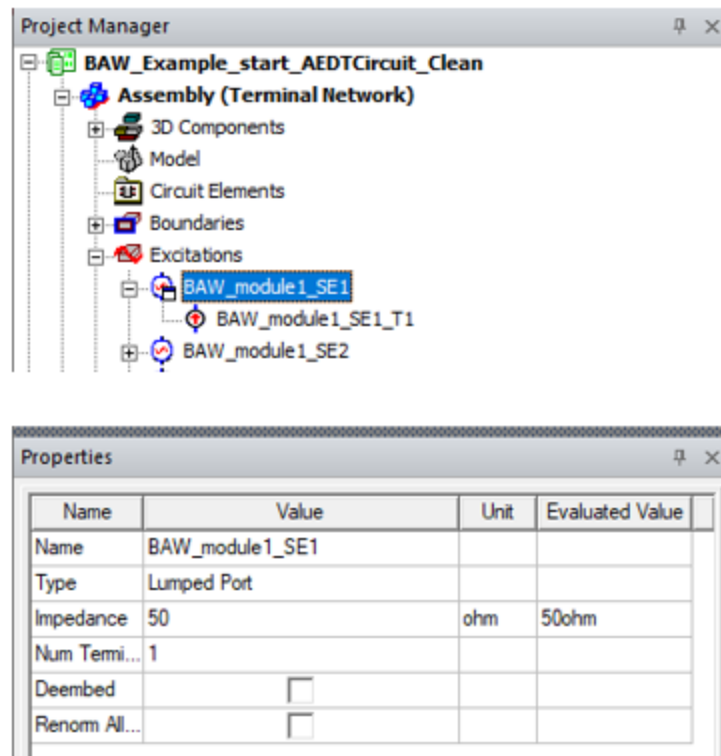
Expand the first port to see its expected Child Object, its Terminal, in the **Project Manager** Window:



Through scripting, first define the Port Object (in this example, oPort) using the 'GetChildObject()' command for the first port, 'BAW_module1_SE1.' Then determine its Object Child Name, the terminal definition:

```
>>> oPort = oExcitations.GetChildObject('BAW_module1_SE1')
>>> oPort.GetChildNames()
['BAW_module1_SE1_T1']
```

As the use and logic for `GetChildNames()` and `GetChildObject()` have been demonstrated, you can now explore the properties of each of these objects, if they exist. The function to determine what properties exist is `GetPropNames()`. Use this to determine what properties exist to be retrieved or modified for a given object. The properties available are readily identifiable in the **Property** window, by default located beneath the **Project Manager** window. For example, if you select a given port object, 'BAW_module1_SE1' the Property window populates as shown:



If you execute the `GetPropNames()` function on the previously defined object, `oPort`, you see the same Property Names as available in the **Properties** window:

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

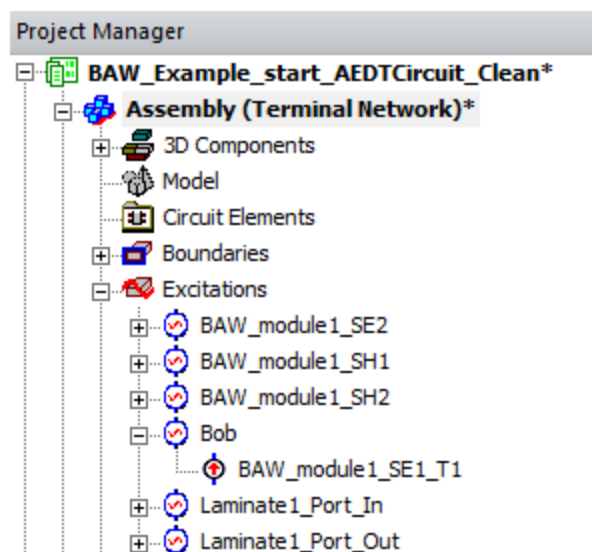
Once you identify the desired object and you know the desired property, you can access the value via `GetPropValue()`. For example, if you want to retrieve the name of the object `oPort`:

```
>>> oPort.GetPropValue('Name')  
'BAW_module1_SE1'
```

To change the value of the property, use the `SetPropValue()` function. The arguments for this function are (*Property Name*, *New Value*). For example, to change the name of the port to 'Bob':

```
>>> oPort.SetPropValue('Name', 'Bob')  
True
```

This function returns a Boolean 'True' if successful. The Project Manager window updates accordingly:



This approach for retrieving and setting properties is general and can be used for many aspects of an Ansys Electronics Desktop simulation. This Object-Oriented method of property identification and modification operates only on existing objects. Object-Oriented scripting cannot create new instances; you must revert to the functions in a given Module to do that. Not all Children of a given object may be accessible via the GetChildNames() command just yet. An example is given for Material property modification later in this App Note. However, if you need specific objects you can reference details in the Scripting Help or reach out to an Application Engineer.

Material Properties and Examples

This section discusses the material properties and how to access and modify them. Because materials are globally defined, the objects are children of the Project, oProject, as shown below:

```
>>> oProject = oDesktop.GetActiveProject()
>>> oMaterials = oProject.GetChildObject('Materials')
>>> oMaterials.GetChildNames()
['vacuum', 'Cap_Mat', 'Outline_Mat', 'SolderMask_Mat', 'copper', 'pec']
```

All materials with a Project Definition, or assigned to an object, in the Project are accessible. For example, assume you want to see the conductivity of 'copper.' Follow the same flow as in the previous section:

```
>>> oCopper = oMaterials.GetChildObject('copper')
>>> oCopper.GetPropNames()
['Coordinate System Type', 'Coordinate System Type/Choices', 'Relative Permittivity Type', 'Relative Permittivity Type/Choices', 'Relative Permeability Type', 'Relative Permeability Type/Choices', 'Relative Permeability', 'Bulk Conductivity Type', 'Bulk Conductivity Type/Choices', 'Bulk Conductivity', 'Dielectric Loss Tangent Type', 'Dielectric Loss Tangent Type/Choices', 'Dielectric Loss Tangent', 'Magnetic Loss Tangent Type', 'Magnetic Loss Tangent Type/Choices', 'Magnetic Loss Tangent', 'Electric Coercivity Type', 'Electric Coercivity Magnitude', 'Magnetic Coercivity Type', 'Magnetic Coercivity Magnitude', 'Thermal Conductivity Type', 'Thermal Conductivity Type/Choices', 'Thermal Conductivity', 'Magnetic Saturation Type', 'Magnetic Saturation', 'Lande G Factor Type', 'Lande G Factor', 'Delta H Type', 'Delta H', '- Measured Frequency Type', '- Measured Frequency', 'Core Loss Model', 'Core Loss Model/Choices', 'Mass Density Type', 'Mass Density', 'Composition', 'Composition/Choices', 'Specific Heat Type', 'Specific Heat', 'Young's Modulus Type', 'Young's Modulus Type/Choices', 'Young's Modulus', 'Poisson's Ratio Type', 'Poisson's Ratio Type/Choices', 'Poisson's Ratio', 'Thermal Expansion Coefficient Type', 'Thermal Expansion Coefficient Type/Choices', 'Thermal Expansion Coefficient', 'Magnetostriiction Type', 'Inverse Magnetostriiction Type', 'Thermal Material Type', 'Thermal Material Type/Choices', 'Solar Behavior Type', 'Solar Behavior Type/Choices', 'Solar Behavior']
```

The Material Property of interest is "Bulk Conductivity." So you create a "Cond" object to store the value and use the GetPropValue function to obtain it. Then name the Cond object to see the value:

```
>>> Cond = oCopper.GetPropValue('Bulk Conductivity')
>>> Cond
'58000000'
```

To change the conductivity, use the `SetPropValue()` function as shown below:

```
>>> oCopper.SetPropValue('Bulk Conductivity', '100')
True
>>> NewCond = oCopper.GetPropValue('Bulk Conductivity')
>>> NewCond
'100'
```

Body Properties and Modification

The following example shows how to retrieve the properties of a Body in the model, in this case a Region object. Once you identify the desired property, you can modify it as needed.

```
>>> oModel = oDesign.GetChildObject('3D Modeler')
>>> oModel.GetChildNames()
['RadBox_Region_1']
>>> oRegion = oModel.GetChildObject('RadBox_Region_1')
>>> oRegion.GetPropNames()
['Name', 'Material', 'Solve Inside', 'Orientation', 'Orientation/Choices', 'Model', 'Group', 'Display
Wireframe', 'Material Appearance', 'Color', 'Color/Red', 'Color/Green', 'Color/Blue', 'Transparent']
```

Retrieving Variables

Retrieving defined variables in a Design or Project is a common effort for automation. There are two types of variables, Design and Project. Project variables are preceded with a '\$' symbol and are retrieved in the Project object as it is globally defined to all Designs. Design variables do not have any preceding symbols and are retrieved in the Design object as their scope is limited to a given Design. The following example demonstrates the retrieval of Project Variable names and values, and then Design variable names and values.:

```
>>> oProjVar = oProject.GetChildObject("Variables")
>>> oProjVar.GetPropNames()
['$test']
>>> oProjVar.GetPropValue('$test')
'0'
>>> oDesVar = oDesign.GetChildObject("Variables")
>>> oDesVar.GetPropNames()
['test']
>>> oDesVar.GetPropValue('test')
'0'
```

Retrieve Datasets and Values

The GetChildObject, GetChildTypes and GetChildNames functions operate on the oDesktop objects. This allows you to retrieve and view datasets and values. The dataset script wrapper store all values internally in SI units, and converts them back to user-supplied units when you request non-SI property values. For example, if you assigned a dataset to the example OptimTee project in HFSS, you could use these functions in the command window:

```
>>>oDesktop.GetChildTypes()

['Projects']

>>>oDesktop.GetChildNames()

['OptimTee']

>>>arrProjectNames = oDesktop.GetChildNames()

>>>tp = oDesktop.GetChildObject('OptimTee')

>>>tp.GetChildTypes()
```

```
['Design', 'Project Data']
>>>tp.GetChildNames('Project Data')
['Variables', 'Materials', 'Surface Materials', 'Datasets']
>>>ds = tp.GetChildObject('datasets')
>>>ds.GetChildNames()
['$ds1']
>>>ds1=ds.GetChildObject('$ds1')
>>>ds1.GetPropValue('[:,:]')
[[1.0, 4.0], [2.0, 5.0], [3.0, 6.0]]
>>>ds1.GetPropSIValue()
[[1.0, 4.0], [2.0, 5.0], [3.0, 6.0]]
>>>ds1.DimUnits
>>>ds1.DimUnits = ['mm','mm']
>>>ds1.DimUnits
['mm', 'mm']
>>>ds1.GetPropSIValue()
[[0.001, 0.0040000000000000001], [0.002, 0.0050000000000000001], [0.0030000000000000001,
0.0060000000000000001]]
>>>ds1.GetPropValue('[:,:]')
[[1.0, 4.0], [2.0, 5.0], [3.0, 6.0]]
```


GetSolutionData API

Many users want to use scripts to extract solution data from Ansys Electronics Desktop for custom Post Processing. Scripting includes a new method to do this without having to export data to a file and then re-import it for use in a script. The new function is accessible via the “ReportSetup” Module. The function call is “GetSolutionDataPerVariation()”. A code snippet to extract Terminal S Parameter data is shown:

```
8  oModule=.oDesign.GetModule("ReportSetup")
9  Results=.oModule.GetSolutionDataPerVariation("Terminal.Solution.Data", "Setup1::Sweep1:",
10  .....[
11  .....    "Domain:=", "Sweep"
12  .....],
13  .....[
14  .....    "Freq:=", ["All"]
15  .....],
16  .....[
17  .....    "dB(St(Terminal_1,Terminal_1))"
18  .....]
19  ##.Get.Dependent.and.Independent.Variable.data.for.Nominal.Variation
20  NominalData=.Results[0]
21  ##.Get.Independent.Variable.Data
22  ##.For.second.argument,.if.pass.True.then.data.is.in.SI.Units
23  ##.if.pass.False.then.data.is.in.default.scale.units.instead.of.SI
24  SweepValues=.NominalData.GetSweepValues("Freq", True)
25  ##.Get.Dependent.Variable.DataValues
26  ##.Note:.Can.pass.any.'Y.Component'.Name
27  DataValues=.NominalData.GetRealDataValues("dB(St(Terminal_1))")
```

The above code shows how you can extract the Dependent and Independent data to variables for easy manipulation. For more information on other functions available for this, see [GetSolutionDataPerVariation](#).

Summary

Scripting has been advancing in Ansys Electronics Desktop to better allow you to customize and automate their repetitive or complex simulations. The ability to easily retrieve and set property values via the Object-Oriented scripting allows for ease of both writing and

reading. The ability to extract solution data within a script execution is a new functionality that markedly enables more advanced post processing.

Object oriented property scripting presents an easy to use, intuitive and object oriented representation of the data model. The framework supports query of objects and their properties including the edits of the data model in an object oriented fashion. With the new scripting framework, data exposure is intuitive and provides maximum coverage.

Each exposed script object supports the following COM functions:

- **GetName**
 - Return name of the object as text string
 - e.g. name of a design, solve setup, boundary, etc
- **GetChildTypes**
 - An object can have different types of children.
 - Return array of text string. Can be empty if the object's children are NOT categorized into different types.

For example, a design object has 3 children types. The following examples show how the commands run in the **Tools > Open Command Window** for IronPython.

```
>>> design.GetChildTypes()  
['Module', 'Editor', 'Design Data']
```

- **GetChildNames**
 - Input: [String – Type]. Default = “Module” and “Editor” for design script object. ‘All’ for other script objects.
 - Return an array of immediate children's names, of a given type if specified

For example, a Mechanical design object has these children.

```
>>> design.GetChildNames()  
['Boundaries', 'Excitations', 'Optimetrics', 'Results', '3D Modeler']
```

Four of the children are of “Module” type

```
>>> design.GetChildNames("module")
```

```
['Boundaries', 'Excitations', 'Optimetrics', 'Results']
```

- **GetChildObject**

- Input: String -- Object path. The path may include multiple generations.
- Return the child object if found

For example,

```
>>> d = project.GetChildObject("hfss")
>>> d.GetChildObject("3d modeler").GetChildNames()
['Box1', 'Box1_1', 'Box1_1_1']
>>> project.GetChildObject("hfss/3d modeler").GetChildNames()
['Box1', 'Box1_1', 'Box1_1_1']
```

- **GetPropNames**

- Input: [BOOL - IncludeReadOnly] -- default to true
- Return an array of the object's properties

For example,

```
>>> geom = project.GetChildObject("hfss/3d modeler").GetChildObject("Box1")
>>> geom.GetPropNames()
['Name', 'Material', 'Material/SIValue', 'Material/EvaluatedValue', 'Solve Inside', 'Orientation', 'Orientation/Choices', 'Model', 'Group', 'Display Wireframe', 'Material Appearance', 'Color', 'Color/Red', 'Color/Green', 'Color/Blue', 'Transparent']
```

- **GetPropValue**

- Input: String – Property Path. The path may include multiple generations.
- Return the property value as VARIANT

For example,

```
>>> geom.GetPropValue("material")
' "vacuum" '

>>> geom.GetPropValue("xsize")
' 3mm '

>>> op.GetPropValue("attach to original object")
False
```

- **SetPropValue**

- Input: String – Property Path. The path may include multiple generations.
- Input: String – Data. New value of the property.
- Return -- True if property data is updated successfully. False if failed to assign the new value.

For example,

```
>>> geom.SetPropValue("model", False)

>>> boxcmd.SetPropValue("ysize", "4mm")
```

- **GetPropEvaluatedValue (<PropName>)**

For example,

```
oVar = oDesign.GetChildObject(" Variables/var")
oVar.GetPropEvaluatedValue()
```

- **GetPropSIValue (<PropName>)**

For example,

```
oCreateBox = oDesign.GetChildObject("3D Modeler/Box1/CreateBox:1")
oCreateBox.GetPropValue("xSize")
    return "length / 2"
oCreateBox.GetPropEvaluatedValue("xSize")
    return '0.4mm'
oCreateBox.GetPropSIValue("xSize")
    return 0.0004
```

Additional Details Specific to AEDT Solvers

“3D Modeler” of 3D products and “Machine” of Rmxprt are exposed as “Editor” type children of a design script object.

“Variables” and “Design Settings” are exposed as “Design Data” type children of a design script object.

The following “Module” types are exposed as “Module” type children of a design script object.

HFSS

- Boundaries, Excitations, Circuit Elements, Hybrid Regions, Analysis, Radiation, Field Overlays, Optimetrics, Results

HFSS 3D Layout

- Boundaries, Excitations, Circuit Elements, Analysis, Radiation, Field Overlays, Optimetrics, Results

Maxwell 3D/2D

- Boundaries, Excitations, Analysis, Field Overlays, Optimetrics, Results

Rmxprt

- Analysis, Field Overlays, Optimetrics, Results

Q3D

- Boundaries, Nets, Analysis, Optimetrics,
Q2D
- Boundaries, Conductors, Analysis, Field Overlays, Optimetrics,
Icepak
- Thermal, Monitor, Mesh, Analysis, Field Overlays, Optimetrics, Results
Mechanical
- Boundaries, Excitations, Analysis, Field Overlays, Optimetrics, Results
Circuit
- Optimetrics, Results
Circuit Netlist
- Results
EMIT
- Coupling
Simplorer/Twin Builder
- Analysis, Optimetrics, Results

Additional details on Boundaries/Excitations

Each design type presents its boundaries/excitations data in the project tree as different groups. For example, an HFSS design has Boundaries, Excitations, Circuit Elements and Hybrid Regions while a Icepak design has just a “Thermal” project tree folder.

These module script objects do not have properties

```
>>> project.GetChildObject("icepak/thermal").GetPropNames()  
  
[]
```

GetChildTypes of these module script objects returns the types of its immediate children

```
>>> d = p.GetChildObject("q2d")
>>> d.GetChildObject("conductors").GetChildTypes()
['NonIdealGround', 'SignalLine']
```

GetChildNames of these module object returns its immediate children

```
>>> p.GetChildObject("icepak/thermal").GetChildNames()
['Source1', 'Resistance1', 'ConductingPlate1', 'Source2', 'Resistance2', 'ConductingPlate2',
'Source3', 'Resistance3', 'ConductingPlate3']
```

GetChildNames can be invoked with a “type” and the returns will be filtered by that given type.

```
>>> p.GetChildObject("icepak/thermal").GetChildNames("resistance")
['Resistance1', 'Resistance2', 'Resistance3']
```

Children of a module object are scriptable objects and have properties.

```
>>> port = p.GetChildObject("hfss/excitations/1")
>>> port.GetPropNames(False)
['Name', 'Deembed', 'Deembed Dist', 'Renorm All Terminals']
```

You can query/edit these properties

```
>>> port.GetPropValue("deembed")
False
>>> port.SetPropValue("deembed", True)
True
>>> port.GetPropValue("deembed")
```

True

A boundary/excitation script object can also have children. For example, HFSS terminal is a child of its port. Q3D source/sink can be children of a net.

```
>>> port.GetChildNames()
['Box1_T1']
>>> port.GetChildTypes()
['Terminal']
>>> p.GetChildObject("q3d/nets/s2").GetChildNames()
['Source2', 'Sink2']
>>> p.GetChildObject("q3d/nets/s2").GetChildTypes()
['Sink', 'Source']
```

3D component encapsulation

These script interfaces are compliant with encapsulation. For example,

- Design.GetChildObject("boundaries").GetChildNames() will not return component boundaries
- SetPropValues of component excitations can only be used to edit post processing settings such as 'Deembed', 'Deembed Dist' of a HFSS port.

Additional details on Solve setup

All solve setups are children of the "Analysis" script object. This parent script object is also of the type "Module".

```
>>> d = oDesktop.GetActiveProject().GetChildObject("hfss")
>>> d.GetChildNames()
['Boundaries', 'Excitations', 'Hybrid Regions', 'Circuit Elements', 'Analysis', 'Optimetrics',
'RadField', 'Results', '3D Modeler']
```



```
>>> setups = d.GetChildObject("analysis")
```

This module script object has no property

```
>>> setups.GetPropNames()  
  
[]
```

The children of this module script object in a 3D design is not categorized into different types because the solve setup type is one-to-one to the solution type of a 3D design.

```
>>> setups.GetChildTypes()  
  
[]
```

The children of this module script object in a 3DLayout and Simplorer/TwinBuilder design is categorized into different solve setup types, such as “Transient”, “AC” and “DC” in a Simplorer/TwinBuilder design and “HFSS”, “PlanarEM”, “SIwave” in a 3D layout design.

A solve setup script object can also have children. Children are typically frequency sweeps.

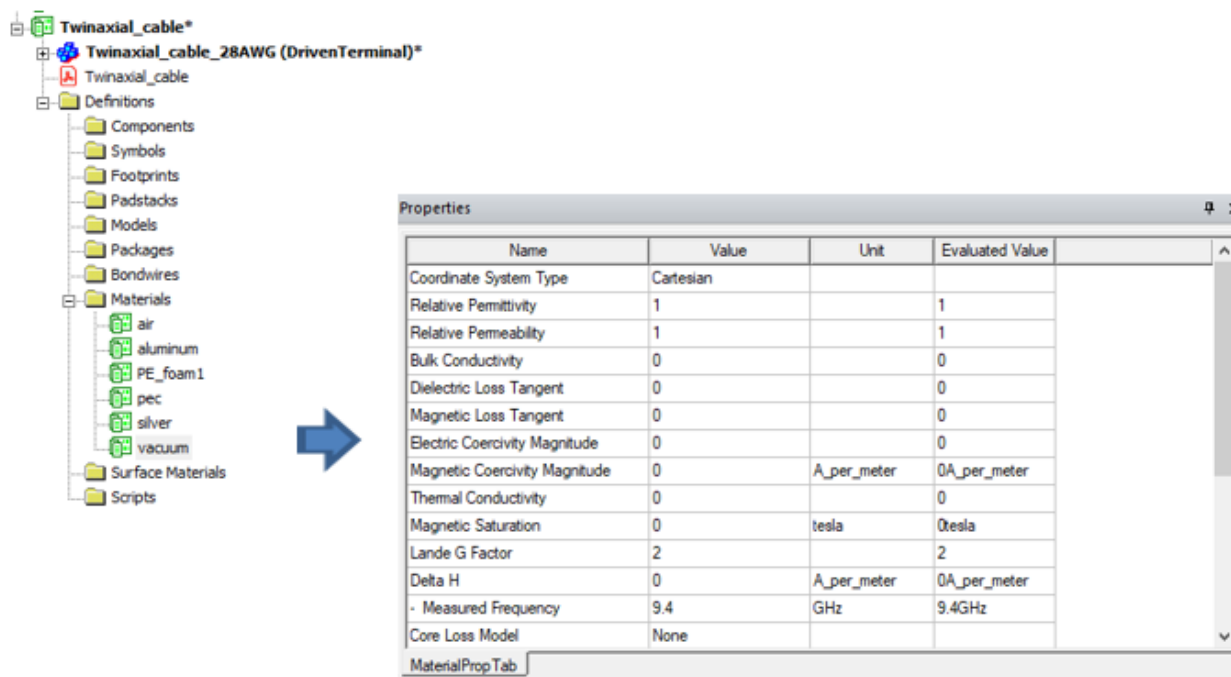
```
>>> setup.GetChildNames()  
  
['Sweep', 'Sweep1', 'Sweep2']  
  
>>> setup.GetChildTypes()  
  
['Discrete', 'Interpolating']  
  
>>> sweep1 = setup.GetChildObject("sweep")
```

Related Topics

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

Materials Scripting Support

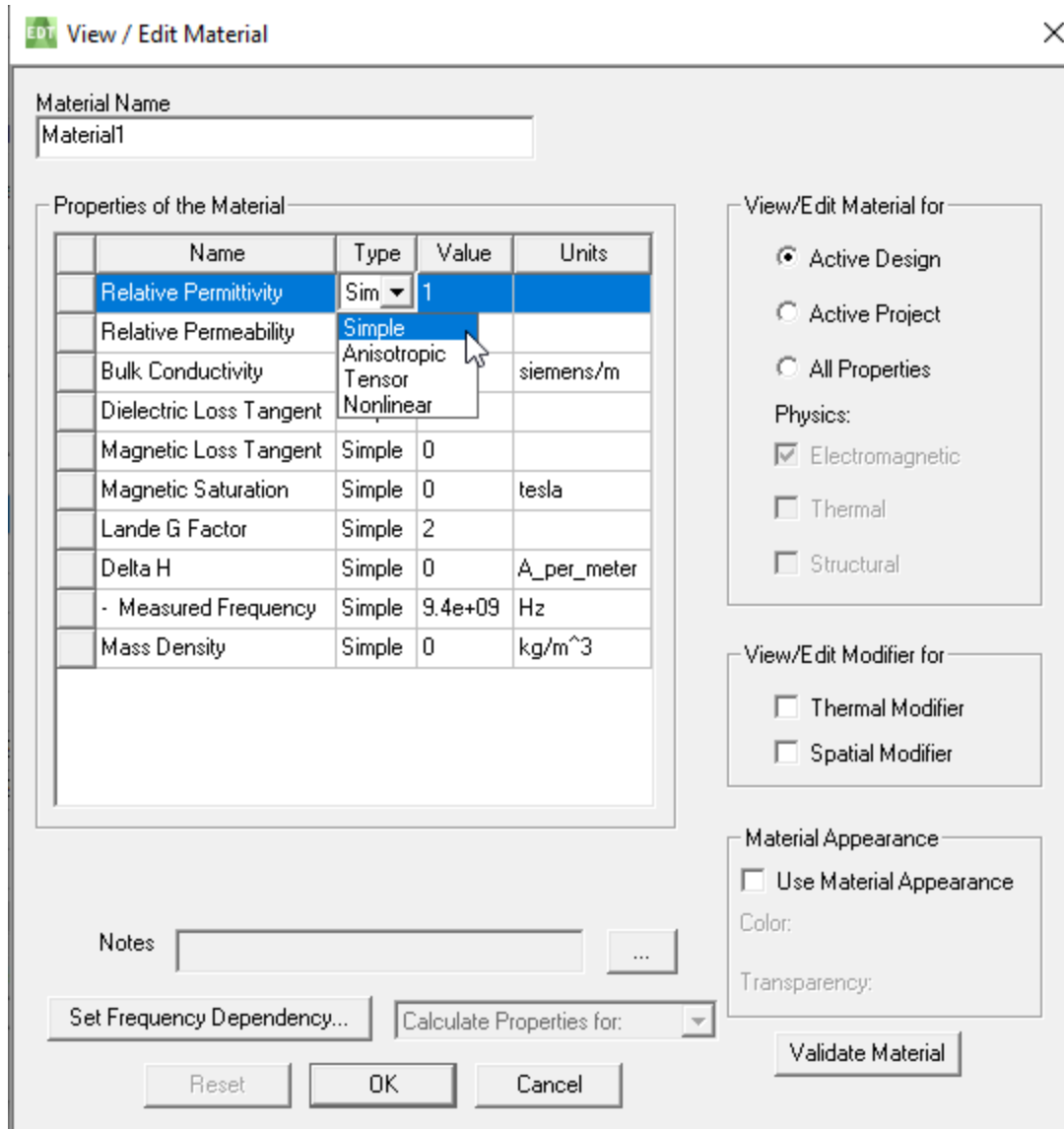
Supported material properties are shown in a Property window for the material item.



The Properties window displays the following material properties:

Name	Value	Unit	Evaluated Value
Coordinate System Type	Cartesian		
Relative Permittivity	1		1
Relative Permeability	1		1
Bulk Conductivity	0		0
Dielectric Loss Tangent	0		0
Magnetic Loss Tangent	0		0
Electric Coercivity Magnitude	0		0
Magnetic Coercivity Magnitude	0	A_per_meter	0A_per_meter
Thermal Conductivity	0		0
Magnetic Saturation	0	tesla	0tesla
Lande G Factor	2		2
Delta H	0	A_per_meter	0A_per_meter
- Measured Frequency	9.4	GHz	9.4GHz
Core Loss Model	None		

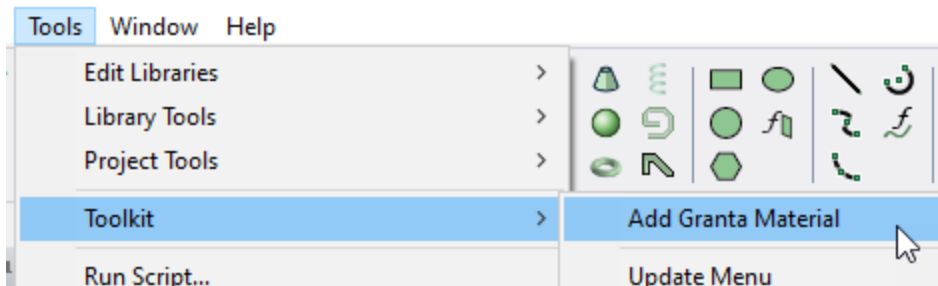
Some Material Properties like Relatively Permittivity may have values assigned as BH Curves or Tensors, as discussed in the Assigning Materials chapter of the online help.

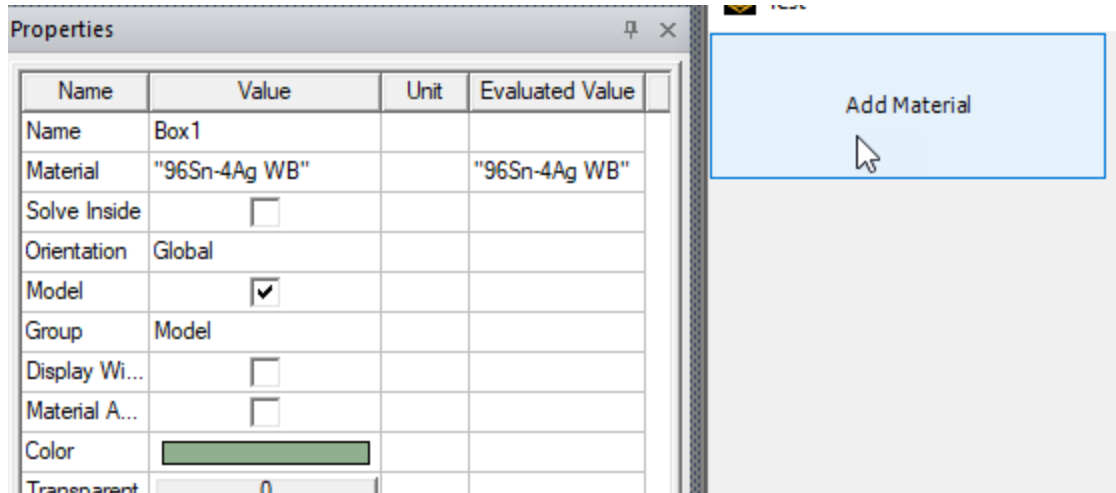


With this feature enabled you can then:

- Get/Set Simple material property
- Get/Set Anisotropic material property
- Get/Set Nonlinear material property
- Get/Set Vector material property
 - Components hide/shown as needed
- Get/Set Tensor material property
- Get/Set Choice material property
- [AddDefinitionFromBlock](#)
- [AddDefiniitonFromLibFile](#)
- [GetExtendedDefinitionObect](#)

A new Toolkit allows you to select materials from the Granta materials gateway, such that project materials will automatically be added when you select a material from the gateway, and that the gateway itself is easily accessed from the materials.





What is not supported:

- Change of property type
- Custom material property, due to its complexity

Object Oriented Scripting for Materials

Materials are Child objects of the Active Project. In the IronPython command window, you can execute `GetPropNames()` for a specified material as follows:

```
>>> omats = oDesktop.GetActiveProject().GetChildObject("Materials")
>>> omat = omats.GetChildObject("vacuum")
>>> omat.GetPropNames()
```

```
['Coordinate System Type', 'Coordinate System Type/Choices', 'Relative Permeability Type',  
'Relative Permeability Type/Choices', 'Relative Permeability', 'Relative Per-  
meability/SIValue', 'Relative Permeability/EvaluatedValue', 'Bulk Conductivity Type', 'Bulk  
Conductivity Type/Choices', 'Bulk Conductivity', 'Bulk Conductivity/SIValue', 'Bulk Con-  
ductivity/EvaluatedValue', 'Magnetic Coercivity Type', 'Magnetic Coercivity Magnitude', 'Mag-  
netic Coercivity Magnitude/SIValue', 'Magnetic Coercivity Magnitude/EvaluatedValue',  
'Composition', 'Composition/Choices', 'Young's Modulus Type', 'Young's Modulus Type/Choices',  
'Young's Modulus', 'Young's Modulus/SIValue', 'Young's Modulus/EvaluatedValue', 'Poisson's  
Ratio Type', 'Poisson's Ratio Type/Choices', 'Poisson's Ratio', 'Poisson's Ratio/SIValue',  
'Poisson's Ratio/EvaluatedValue']
```

Examples showing change to material property type:

```
>>> omat.GetPropValue("Relative Permeability Type/Choices")  
  
['Simple', 'Anisotropic', 'Tensor', 'Nonlinear']  
  
>>> omat.GetPropValue("Relative Permeability Type")  
  
'Nonlinear'  
  
>>> omat.SetPropValue("Relative Permeability Type", "Simple")  
  
True  
  
>>> omat.GetPropValue("Relative Permeability Type")  
  
'Simple'  
  
>>> omat.SetPropValue("Relative Permeability", 10)  
  
True
```

Examples showing change to a vector component value

```
>>> omat.GetPropValue("Magnetic Coercivity Magnitude")  
  
'0A_per_meter'  
  
>>> omat.SetPropValue("Magnetic Coercivity Magnitude", "-1A_per_meter")
```

```
True
```

```
>>> omat.GetPropNames()
```

```
['Coordinate System Type', 'Coordinate System Type/Choices', 'Relative Permeability Type',  
'Relative Permeability Type/Choices', 'Relative Permeability', 'Relative Per-  
meability/SIValue', 'Relative Permeability/EvaluatedValue', 'Bulk Conductivity Type', 'Bulk  
Conductivity Type/Choices', 'Bulk Conductivity', 'Bulk Conductivity/SIValue', 'Bulk Con-  
ductivity/EvaluatedValue', 'Magnetic Coercivity Type', 'Magnetic Coercivity Magnitude', 'Mag-  
netic Coercivity Magnitude/SIValue', 'Magnetic Coercivity Magnitude/EvaluatedValue', 'Magnetic  
Coercivity Components', 'Magnetic Coercivity Components/Component1', 'Magnetic Coercivity Com-  
ponents/Component2', 'Magnetic Coercivity Components/Component3', 'Composition', 'Com-  
position/Choices', '- Stacking Factor Type', '- Stacking Factor', '- Stacking Factor/SIValue',  
'- Stacking Factor/EvaluatedValue', '- Stacking Direction', '- Stacking Direction/Choices',  
"Young's Modulus Type", "Young's Modulus Type/Choices", "Young's Modulus", "Young's Mod-  
ulus/SIValue", "Young's Modulus/EvaluatedValue", "Poisson's Ratio Type", "Poisson's Ratio  
Type/Choices", "Poisson's Ratio", "Poisson's Ratio/SIValue", "Poisson's Ratio/EvaluatedValue"]
```

```
>>> omat.SetPropValue("Magnetic Coercivity Components/Component2", 2)
```

```
True
```

```
>>> omat.GetPropValue("Magnetic Coercivity Components")
```

```
['Component1:=', '2', 'Component2:=', '2', 'Component3:=', '0']
```

Change choice property value

```
>>> omat.GetPropValue("Composition/Choices")
```

```
['Solid', 'Lamination', 'Litz Wire']
```

```
>>> omat.SetPropValue("Composition", "Lamination")
```

```
True
```

```
>>> omat.GetPropNames()
```

```
[ 'Coordinate System Type', 'Coordinate System Type/Choices', 'Relative Permeability Type',
'Relative Permeability Type/Choices', 'Relative Permeability', 'Relative Per-
meability/SIValue', 'Relative Permeability/EvaluatedValue', 'Bulk Conductivity Type', 'Bulk
Conductivity Type/Choices', 'Bulk Conductivity', 'Bulk Conductivity/SIValue', 'Bulk Con-
ductivity/EvaluatedValue', 'Magnetic Coercivity Type', 'Magnetic Coercivity Magnitude', 'Mag-
netic Coercivity Magnitude/SIValue', 'Magnetic Coercivity Magnitude/EvaluatedValue', 'Magnetic
Coercivity Components', 'Magnetic Coercivity Components/Component1', 'Magnetic Coercivity Com-
ponents/Component2', 'Magnetic Coercivity Components/Component3', 'Composition', 'Com-
position/Choices', '- Stacking Factor Type', '- Stacking Factor', '- Stacking Factor/SIValue',
'- Stacking Factor/EvaluatedValue', '- Stacking Direction', '- Stacking Direction/Choices',
"Young's Modulus Type", "Young's Modulus Type/Choices", "Young's Modulus", "Young's Mod-
ulus/SIValue", "Young's Modulus/EvaluatedValue", "Poisson's Ratio Type", "Poisson's Ratio
Type/Choices", "Poisson's Ratio", "Poisson's Ratio/SIValue", "Poisson's Ratio/EvaluatedValue"]
```

```
>>> omat.SetPropValue("Magnetic Coercivity Components/Component2", 2)
```

```
True
```

```
>>> omat.GetPropValue("Magnetic Coercivity Components")
```

```
['Component1:=', '2', 'Component2:=', '2', 'Component3:=', '0']
```

Change choice property value

```
>>> omat.GetPropValue("Composition/Choices")
```

```
['Solid', 'Lamination', 'Litz Wire']
```

```
>>> omat.SetPropValue("Composition", "Lamination")
```

```
True
```

```
>>> omat.GetPropNames()
```

```
[ 'Coordinate System Type', 'Coordinate System Type/Choices', 'Relative Permeability Type',
'Relative Permeability Type/Choices', 'Relative Permeability', 'Relative Per-
meability/SIValue', 'Relative Permeability/EvaluatedValue', 'Bulk Conductivity Type', 'Bulk
Conductivity Type/Choices', 'Bulk Conductivity', 'Bulk Conductivity/SIValue', 'Bulk
```



```
Conductivity/EvaluatedValue', 'Magnetic Coercivity Type', 'Magnetic Coercivity Magnitude',  
'Magnetic Coercivity Magnitude/SIValue', 'Magnetic Coercivity Magnitude/EvaluatedValue', 'Mag-  
netic Coercivity Components', 'Magnetic Coercivity Components/Component1', 'Magnetic Coer-  
civity Components/Component2', 'Magnetic Coercivity Components/Component3', 'Composition',  
'Composition/Choices', '- Stacking Factor Type', '- Stacking Factor', '- Stacking Fact-  
or/SIValue', '- Stacking Factor/EvaluatedValue', '- Stacking Direction', '- Stacking Dir-  
ection/Choices', 'Young's Modulus Type', 'Young's Modulus Type/Choices', 'Young's Modulus',  
'Young's Modulus/SIValue', 'Young's Modulus/EvaluatedValue', 'Poisson's Ratio Type', 'Pois-  
son's Ratio Type/Choices', 'Poisson's Ratio', 'Poisson's Ratio/SIValue', 'Poisson's Ratio/E-  
valuatedValue']
```

```
>>>
```

Property Script Commands

Property script commands allow you to navigate through all objects and properties in a project. You can get and set all properties for all objects in the Project tree with simple data types.

Property Object is the base class defined for all script objects that support the properties Get and Set.

`GetName()`

- Returns the name of the object.

`GetChildTypes()`

- An object may have different types of children. For example, a design may have variables, modules, and editors.
- Returns an array of text strings; may be empty if the children are not divided into different types.

`GetChildNames(<type>)`

- <type> – Child type name. By default, returns all children names for all types.
- Returns an array of immediate children names, belonging to a type if specified.

`GetChildObject(<objPath>)`

- <objPath> – A child object path; can contain multiple generations (for example, designObject/moduleObject/SetupObject).
- Returns a child property object if the object is found.

GetPropNames (<bIncludeReadOnly>)

- <bIncludeReadOnly> – Optional; defaults to true. True includes read-only properties; False excludes read-only properties.
- Returns an array of the object's property names.

GetPropValue (<propertyPath>)

- <propertyPath> – The property's path; may be a child object's path appended with a property name (for example, TeeModel/Offset/SIValue).
- Returns the property value if found. Otherwise causes script error.

SetPropValue (<propertyPath>, <data>)

- <propertyPath> – The property's path; may be a child object's path appended with a property name (for example, TeeModel/Offset/SIValue).
- <data> – New data; type depends on property type.
- Returns True if updated successfully; False if new data is invalid.

For a detailed summary of how Property script commands are used in a range of contexts, including Variable objects, see: [Object Script Property Function Summary](#). Additional examples for these commands are listed under [Project Objects](#), [Design Objects](#), [3D Modeler](#), [Optimetrics](#), Radiation Module and [Reporter](#).

Note:

Older property commands should be executed by the oProject object.

```
Set oProject = oDesktop.SetActiveProject("Project1")
```

```
oProject.CommandName <args>
```

Some of the topics covered in this chapter are as follows:

[Object Script Property Function Summary](#)

[Conventions Used in this Chapter](#)

[GetArrayVariables](#)

[GetProperties](#)

[GetPropertyValue](#)

[GetVariables](#)

[GetVariableValue](#)

[SetPropertyValue](#)

[SetVariableValue](#)

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

[Additional Property Scripting Example](#)

Object Script Property Function Summary

Object Path

The Object path can be used to navigate through objects and properties in an Ansys EM project.

- An Object path consisted of one or multiple Object-ID-Nodes separated by "/" .
- Object-ID-Node; may exist in the following forms:
 - A simple object name or property name.
 - Type[Name] for object; Tab[name] for property.
 - Name[attr1="v1", attr2 = "v2", ...]. When more than one child object have the same name, use attributes to specify the difference.

- `ArrayName[index]`. For example, in an Optimetric setup with multiple calculations, "Calculation[0]" could be used to identify the first calculation.
- Name beginning with '@' character denoted as a property name, when an object has a child and property with the same name.

Property Object

The Property Object is the base class defined for all script object that support property Get & Set.

- `GetName()`
 - Returns the name of the object.
- `GetChildTypes()`
 - An object may have different type of children. For example, a design may have variables, modules, and editors.
 - Returns array of text strings; may be empty if the children are *not* divided to different types.
- `GetChildNames(<type>)`
 - `<type>` – children type name; default returns all children names for all types.
 - Returns an array of immediate children names, belonging to the type if specified.
- `GetChildObject(<objPath>)`
 - `<objPath>` – A child object path. The path may include multiple generations, such as (designObject/moduleObj/SetupObject).
 - Returns a child property object if the object found.
- `GetPropEvaluatedValue(<propName>)`
 - Return the Evaluated-Value for Value-Property and Variable.
 - Return the Property-value as text string for other property types.
- `GetPropSIValue(<propName>)`
 - Return the SI-Value for Value-Property and Variable.
 - Return NAN for other property type if its value is cannot be converted to a double-floating point value.

- `GetPropNames(<blIncludeReadOnly>);`
 - `<blIncludeReadOnly>` – optional, default to true; True will include read-only properties, False will exclude read-only properties.
 - Returns an array of the object's property names.
- `GetPropValue(<propertyPath>)`
 - `<propertyPath>` – the property's full path. A property name or child object's path appended with a property name, like "TeeModel/Offset/SIValue"
 - Returns the property value if the property is found; otherwise causes script error.
- `SetPropValue(<propertyPath>, <data>)`
 - `<propertyPath>` – the property's full path. A property name or child object's path appended with a property name, like "TeeModel/Offset/Value"
 - `<data>` – new data, type is dependent on property type.
 - Returns True if property data is updated successfully; False if the new data is invalid.

Project Object

Project Object inherited all functions defined in the Property Object. But it doesn't have property, `GetPropValue()` & `SetPropValue()` function can be used to set its child object's property.

- `GetChildTypes()` always return ["Design", "Variable"].
- `GetChildNames(type)`
`GetChildNames()` & `GetChildName("Design")` will return all Design names of the project.
`GetChildNames("Variable")` return all project variable names.
- `GetChildObject(objPath)`

`oDesign = oProject("TeeModel")`

`oVariable = oProject.GetChildObject("VariableName")`

`oReport = oProject.GetChildObject("TeeModel/Results/S Parameter Plot 1")`

- `GetPropNames(bIncludeReadOnly)` always return empty array since the project has no property.
- `GetPropValue(propertyPath)`

```
oProject.GetPropValue("TeeModel/offset") //get the offset variable value in the TeeModel Design
oProject.GetPropValue("TeeModel/Results/S Parameter Plot 1/Display Type") // Get the report display type.
```

- `SetPropValue(propertyPath, newValue)`

```
oProject.SetPropValue("TeeModel/offset", "2mm") //Set the offset variable value to "2mm" in the TeeModel Design
oProject.SetPropValue("TeeModel/Results/S Parameter Plot 1/Display Type", "Data Table") // Set the report display type to data table.
```

Design Object

Design Object inherited all functions defined in the Property Object. But it doesn't have property, `GetPropValue()` & `SetPropValue()` function can be used to set its child object's property..

- `GetChildTypes()` always return ['Module', 'Editor', 'Variable'].
- `GetChildNames(type)`
`GetChildNames()` will return modules & editor child names.
`GetChildNames("Variable")` will return all variable names
`GetChildNames("Module")` will return all module names that support property-object-script like ['Optimetrics', 'RadField', 'Results']
`GetChildNames("Editor")` will return a 3D editor name for all 3D Designs
- `GetChildObject()`
`oVariable = oDesign.GetChildObject("VariableName")`
`oReport = oDesign.GetChildObject("Results/S Parameter Plot 1")`
`oRptModule = oDesign.GetChildObject("ReportSetup")`
- `GetPropNames(bIncludeReadOnly)` always return empty array since the design has no property.
- `GetPropValue()`
`oDesign.GetPropValue("offset/SIValue")` //get the offset variable SI value in the Design
`oDesign.GetPropValue("offset/SIValue")` //get the offset variable SI value in the Design
`oDesign.GetPropValue("Results/S Parameter Plot 1/Display Type")` // Get the report display type

- SetPropValue()
oDesign.SetPropValue("offset", "2mm") //Set the offset variable value to "2mm" in the Design
oDesign.SetPropValue("Results/S Parameter Plot 1/Display Type", "Data Table") // Set the report display type to data table.

3D Modeler Object

GetChild commands returns the appropriate properties for modeler objects. For 3D Components and UDMs, these commands do not return parts, coordinate systems, plans, as top-level modeler children.

```
oModeler = oDesktop.GetActiveProject().GetActiveDesign().GetChildObject("3D Modeler")
oModeler.GetChildNames()
oModeler.GetChildNames("ModelParts")
oModeler.GetChildNames("AllParts")
oModeler.GetChildNames("NonModelParts")
oModeler.GetChildNames("Planes")
oModeler.GetChildNames("CoordinateSystems")
```

Variable Object

Is a Property Object that has no child. It also provides quick function call to get/set it properties by adding functions with property name appended to Get_ & Set_ prefix. To find what functions it provided enter dir(oVar) the command window. It can accessed by the project or design object's GetChildObject(VariableName) function.

```
oProjVar = oProject.GetChildObject("$VarName")
oVar = oProject.GetChildObject("DesignName/VarName")
oVar = oDesign.GetChildObject("variableName")
oProject..GetChildNames("Variable") will return all project variable names.
oDesign.GetChildNames(Variable) will return all Design Variable names.
```

- GetChildTypes() always return empty array.
- GetChildNames() always return empty array , since variable has no child.
- GetChildObject(objPath) it has no child.

- `GetPropNames(bIncludeReadOnly)` ['EvaluatedValue', 'SIValue'] are read-only properties
 - Independent variable :['Value', 'EvaluatedValue', 'SIValue', 'Description', 'ReadOnly', 'Hidden', 'Sweep', 'Optimization/Included', 'Optimization/Min', 'Optimization/Max', 'Sensitivity/Included', 'Sensitivity/Min', 'Sensitivity/Max', 'Sensitivity/IDisp', 'Statistical', 'Statistical/Included', 'Tuning/Included', 'Tuning/Step', 'Tuning/Min', 'Tuning/Max'].
 - Dependent variable ['Value', 'EvaluatedValue', 'SIValue', 'Description', 'ReadOnly', 'Hidden', 'Sweep']
- `GetPropValue(propName)`
`oVar.GetPropValue()` return the variable value as text string.
`oVar.GetPropValue("Value")` return the variable value as text string.
`oVar.GetPropValue("SIValue")` return the SI-value of variable as number.
`oVar.Get_SIValue()` also return the SI value.
- `SetPropValue(propName, newValue)`
`oVar.SetPropValue("Value", 888)`
`oVar.SetPropValue("Sensitivity/Included", True)`
`oVar.SetPropValue("Sensitivity/Max", '1.8pF')`
`oVar.Set_Sensitivity_Max( '1.8pF')` also works as last call.
`oVar.SetPropValue("Sensitivity", ['Min:=' , '0.8pF', 'Max:=' , '1.8pF'])`
//set multiple attributes at one call:
`oVar.SetPropValue("@", ['Value:=' , 288, 'Sensitivity', ['Included', True, 'Min', '0.0']])`
`oVar.SetPropValue("", ['Value:=' , 288, 'Sensitivity', ['Included', True, 'Min', '0.0']])`

Optimetrics Module Object:

Optimetrics Module Object inherited all functions defined in the Property Object. But it doesn't have property, `GetPropValue()` & `SetPropValue()` function can be used to set its child object's property..

- `GetChildTypes()` there are six type of children, they are ['OptiParametric', 'OptiOptimization', 'OptiSensitivity', 'OptiStatistical', 'OptiDesignExplorer', 'OptiDXDOE']. But the return array only included those that have setup defined, so it may be an empty array if no optimetrics setup is defined. The `GetChildNames(type)` function also recognized the type name without the prefix "Opti".
- `GetChildNames(type)`
`GetChildNames()` will return all setup for all types.
`GetChildNames("OptiOptimization")` & `GetChildNames("Optimization")` will return all Optimization setup.

- `GetChildObject()`
`oParamSetup = oOptModule.GetChildObject('ParametricSetup1')` get the
`oOptSetup = oOptModule.GetChildObject('OptimizationSetup1')`
- `GetPropNames(bIncludeReadOnly)` always return empty array since the it has no property.
- `GetPropValue(propPath)` may be used to get its child's property value
`oOptModule.GetPropValue("OptimizationSetup1\Optimizer")` get the optimizer name for OptimizationSetup1
- `SetPropValue(propPath, newValue)` may be used to set its child's property value
`oOptModule.SetPropValue(ParametricSetup1\Enabled", False)` //disable ParametricSetup1

Optimetrics Setup Object

This is a new Object inherited all functions defined in the Property Object. But it doesn't have child. It is accessible through its parents.

```
oOptSetup = oOptModule.GetChildObject('OptimizationSetup1')
```

```
oOptSetup = oDesign.GetChildObject('Optimetrics\OptimizationSetup1')
```

```
oOptSetup = oProject.GetChildObject('TeeModel\Optimetrics\OptimizationSetup1')
```

- `GetChildTypes()` always return empty array.
- `GetChildNames(type)` always return empty array
- `GetChildObject()`
- `GetPropNames(bIncludeReadOnly)` will return the property names listed in the property window when the setup is selected.
- `GetPropValue(propName)`
`oOptSetup.GetPropValue("Optimizer")` return the selected optimizer name.
`oOptSetup.GetPropValue("Optimizer/Choices")` return all optimizer names.
- `SetPropValue(propName, newValue)`
`oOptSetup.SetPropValue("Optimizer", "NotAnOptimizerName");` will return false.
`oOptSetup.SetPropValue("Optimizer", "Quasi Newton");` return true, since "Quasi Newton" is one of the optimizer name returned as the Optimizer Choices.
- `HasResult()` return true if the setup is solved. Otherwise return false.

- Validate() return true if the setup is valid for analyze. Otherwise return false. Calling the SetPropValue() function to change the property may invalid the setup.

ReportSetup(Results) Module Object:

ReportSetup module Object inherited all functions defined in the Property Object. But it doesn't have property, GetPropValue() & SetPropValue() function can be used to get/set its child object's property..

- GetChildTypes() always empty array.
- GetChildNames(type)
GetChildNames() return all report names
- GetChildObject(objPath)
oRpt = oRptModule.GetChildObject("S Parameter Plot 1") return the report property object
oTrace = oRptModule.GetChildObject("S Parameter Plot 1/dB(S(Port1,Port1))") return the trace property object
oAxisX = oRptModule.GetChildObject("S Parameter Plot 1/AxisX") return the axis X property object
- GetPropNames(bIncludeReadOnly) always return empty array since the it \has no property.
- GetPropValue()
oRptModule.GetPropValue("S Parameter Plot 1/Display Type")
- SetPropValue()
oRptModule.SetPropValue("S Parameter Plot 1/Display Type", "DataTable")

ReportSetup(Results) Module Child Objects:

These are Property Objects. Its first level of child object is report. Report has trace, axis, header, Legend, and more children. Trace has curve as child etc.

Those child objects can be accessed by calling all levels of parent object's GetChildObject(path) function.

```
oRpt = oRptModule.GetChildObject(reportName)
```

```
oRpt = oDesign.GetChildObject("Results/reportName")
```

```
oTrace = oRpt.GetChildObject(traceName)
```

```
oTrace = oRptModule.GetChildObject(ReportName/TraceName)
```

- `GetChildTypes()` always return empty array.
- `GetChildNames()` get the object's child names. What will be returned will depend on the object instance.
- `GetChildObject(objPath)`
- `GetPropNames(bIncludeReadOnly)` will return the property names listed in the property window when the object is selected.
- `GetPropValue(propName)`
 - `oRpt.GetPropValue("Display Type")` return the report's display type.
 - `oOptSetup.GetPropValue("Display Type/Choices")` return all optimizer names.
 - `oTrace.GetPropValue("X Component")`
- `SetPropValue(propName, newValue)`
 - `oTrace.SetPropValue("Primary sweep", "Freq")`

Radiation Module Object:

This inherited all functions defined in the Property Object. But it doesn't have property, `GetPropValue()` & `SetPropValue()` function can be used to set its child object's property.

- `GetChildTypes()` always return empty array, now its children
- `GetChildNames(type)`
 - `GetChildNames()` return all setup names.
- `GetChildObject(setupName)` return the setup object as Property object.
 - `oOverlay = oRadModule.GetChildObject('Antenna Parameter Overlay1')`
 - `oSphere = oRadModule.GetChildObject('Infinite Sphere1')`
- `GetPropNames()` return empty array; it has no property.
- `GetPropValue()`
 - `oRadModule.GetPropValue('Line1/Num Points')` //Get the the Line1 setups' "Num Points" property value.
- `SetPropValue()`
 - `oRadModule.SetPropValue('Line1/Num Points', 100)` ; Set the Line1 setups' "Num Points" property to 100.

Radiation Module Child Objects:

These are Property Objects. It also provides quick function call to get/set its properties by adding functions with property name appended to Get_ & Set_ prefix. To find what functions it provides enter dir(oVar) the command window.

Those child objects can be access by call all levels of parent object's GetChildObject(path) function.

```
oRadSetup = oRadModule.GetChildObject(setupName)
```

```
oRadSetup = oDesign.GetChildObject(RadField/setupName)
```

- GetChildTypes() always return empty array.
- GetChildNames() always return empty array , since Radiation setup has no child.
- GetChildObject(objPath) it has no child.
- GetPropNames(bIncludeReadOnly) will return the property names listed in the property window when the setup is selected.
- GetPropValue(propName)
oRadSetup.GetPropValue("Num Points") return the line setup's "Num Points" property value.
oRadSetup.Get_NumPoints() will also get the same value.
- SetPropValue(propName, newValue)
oRadSetup.SetPropValue('Num Points', 888)
oRadSetup.Set_NumPoints(888)

Conventions Used in this Chapter

General Definitions:

Property	A single item that can be modified in the Properties window or in the modal Properties pop-up window.
<PropServer>	The item whose properties are being modified. This is usually a compound name, giving all information needed by the editor, design, or project in order to locate the item.
<PropTab>	Corresponds to one tab in the Properties window, the one under which properties are being edited.
<PropName>	The name of a single property.

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

For Project Variables:

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

For Local Variables:

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

For Passed Parameters:

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

For Definition Parameters:

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

For Modules and Editors:

<PropServer>	<ModuleName>:<ItemName> where <ItemName> is the boundary name, solution setup name, etc. For example, "BoundarySetup:PerfE1"
<PropTab>	Boundary Module: "HfssTab" Mesh Operations Module: "MeshSetupTab" Analysis Module: "HfssTab" Optimetrics Module: "OptimetricsTab" Solutions Module: <i>Does not support properties.</i> Field Overlays Module: "FieldsPostProcessorTab" Radiation Module: "RadFieldSetupTab"

	Circuit Module: "CCircuitTab" System Module: "SystemTab" HFSS 3D Layout Module: "HFSS 3D LayoutTab" Nexxim Module: "NexximTab" Layout elements: "BaseElementTab" Schematic elements: "ComponentTab" Optimetrics Module: "OptimetricsTab"

For 3D Model Editor objects:

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

For 3D Model Editor operations:

<PropServer>	<ObjName>:<OperationName>:<int> where <int> is the operation's history index. For example, "Box2:CreateBox:2" refers to the second "CreateBox" operation in Box2's history.
<PropTab>	"Geometry3DCmdTab"

For Reporter operations on Report properties:

<PropServer>	<ReportSetup>
<ChangeProperty>	Array. For example, to set the company name in a plot header to "My Company": <pre>Set oModule = oDesign.GetModule("ReportSetup") oModule.ChangeProperty Array("NAME:AllTabs", _ Array("NAME:Header", _ Array ("NAME:PropServers", _ "XY Plot1:Header"), Array("NAME:ChangedProps", _ Array("NAME:Company Name", "Value:=", "My Company"))))</pre>

Note:

For scripted property changes in the various modules and editors, refer to the chapters on the System, HFSS 3D Layout, and Nexxim tools, as well as the Layout and Schematic editors.

GetArrayVariables

Returns a list of array variables. To get a list of indexed project variables, execute with oProject. To get a list of indexed local variables, use oDesign.

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

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

VB Syntax	GetArrayVariables
VB Example	<pre>oProject.GetArrayVariables oDesign.GetArrayVariables</pre>

GetProperties

Gets a list of all the properties belonging to a specific `<PropServer>` and `<PropTab>`. This can be executed by the `oProject`, `oDesign`, or `oEditor` variables.

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

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

VB Syntax	GetProperties <PropTab>, <PropServer>
VB Example	oEditor.GetProperties "PassedParameterTab", "k"

GetPropertyValue

Returns the value of a single property belonging to a specific <PropServer> and <PropTab>. This function is available with the Project, Design or Editor objects, including definition editors.

Tip: Use the script recording feature and edit a property, and then view the resulting script to see the format for that property.

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

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

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

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

GetVariables

Returns a list of all defined variables. To get a list of project variables, execute this command using `oProject`. To get a list of local variables, use `oDesign`.

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

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

VB Syntax	<code>GetVariables</code>
VB Example	<code>oProject.GetVariables</code> <code>oDesign.GetVariables</code>

GetVariableValue

Gets the value of a single specified variable. To get the value of project variables, execute this command using `oProject`. To get the value of local variables, use `oDesign`.

UI Access	N/A

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

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

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

SetPropertyValue

Sets the value of a single property belonging to a specific PropServer and PropTab. This function is available with the Project, Design or Editor objects, including definition editors. This is not supported for properties of the following types: ButtonProp, PointProp, V3DPointProp, and VPointProp. Only the ChangeProperty command can be used to modify these properties.

Use the script recording feature and edit a property, and then view the resulting script entry or use GetPropertyValue for the desired property to see the expected format.

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

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

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

VB Syntax	SetPropertyValue <propTab>, <propServer>, <propName>, <propValue>
VB Example	oEditor.SetPropertyValue "PassedParameterTab", "k", "R", "2200"

SetVariableValue

Sets the value of a variable. To set the value of a project variable, execute this command using `oProject`. To set the value of a local variable, use `oDesign`.

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

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

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

Example Use of Record Script and Edit Properties

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

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

```
Dim oEditor
Dim oModule

Set oAnsoftApp = CreateObject("Ansoft.ElectronicsDesktop")
Set oDesktop = oAnsoftApp.GetAppDesktop()
oDesktop.RestoreWindow

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

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

Additional Property Scripting Examples

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

```
Dim all_props
Dim prop

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

For Each prop In all_props
```

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

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

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



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

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

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

3 - Application Object Script Commands

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

`oAnsoftApp.<CommandName> <args>`

General Application Script Commands

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

- [GetAppDesktop](#)

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

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

GetAppDesktop

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

UI Access	NA		
Parameters	Name	Type	Description
	None		
Return Value	Object		

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

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

4 - Desktop Object Script Commands

Desktop commands should be executed by the oDesktop object. Some new commands permit you to query objects when you do not know the names. See: [Object Oriented Property Scripting](#).

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

[AddMessage](#)

[AreThereSimulationsRunning](#)

[ClearMessages](#)

[CloseAllWindows](#)

[CloseProject](#)

[CloseProjectNoForce](#)

DeleteProject

[DeleteRegistryEntry](#)

[DoesRegistryValueExist](#)

[DownloadJobResults](#)

[EnableAutoSave](#)

[ExportOptionsFiles](#)

[GetActiveProject](#)

[GetActiveScheduler](#)

[GetActiveSchedulerInfo](#)

[GetAutoSaveEnabled](#)

[GetBuildDateTimeString](#)

[GetCustomMenuSet](#)

[GetDefaultUnit](#)

[GetDesktopConfiguration](#)

[GetDistributedAnalysisMachines](#)

[GetDistributedAnalysisMachinesForDesignType](#)

[GetExeDir](#)

[GetGDIObjectCount](#)

[GetLibraryDirectory](#)

[GetLocalizationHelper](#)

[GetMessages](#)

[GetMonitorData](#)

[GetPersonalLibDirectory](#)

[GetProcessID](#)

[GetProjectDirectory](#)

[GetProjectList](#)

[GetProjects](#)

[GetRegistryInt](#)

[GetRegistryString](#)

[GetRunningInstancesMgr](#)

[GetSchematicEnvironment](#)

[GetScriptingToolsHelper](#)

[GetSysLibDirectory](#)

[GetTempDirectory](#)

[GetUserLibDirectory](#)

[GetVersion](#)

[IsFeatureEnabled](#)

[LaunchJobMonitor](#)

[NewProject](#)

[OpenAndConvertProject](#)

[OpenMultipleProjects](#)

[OpenProject](#)

[OpenProjectWithConversion](#)

[PageSetup](#)

[PauseRecording](#)

[PauseScript](#)

[Print](#)

[QuitApplication](#)

[RefreshJobMonitor](#)

[ResetLogging](#)

[RestoreProjectArchive](#)

[RestoreWindow](#)

[ResumeRecording](#)

[RunACTWizardScript](#)

[RunProgram](#)

[RunScript](#)

[RunScriptWithArguments](#)

[SelectScheduler](#)

[SetActiveProject](#)

[SetActiveProjectByPath](#)

[SetCustomMenuSet](#)

[SetDesktopConfiguration](#)

[SetLibraryDirectory](#)

[SetProjectDirectory](#)

[SetRegistryFromFile](#)

[SetRegistryInt](#)

[SetRegistryString](#)

[SetSchematicEnvironment](#)

[SetTempDirectory](#)

[ShowDockingWindow](#)

[Sleep](#)

[StopSimulations](#)

[SubmitJob](#)

[TileWindows](#)

Related Topics:

[Desktop Commands For Registry Values](#)

[ImportExport Tool Commands](#)

AddMessage

Add a message with severity and context to message window.

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

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

VB Syntax	AddMessage <projectName>, <designName>, <severity>, <msg>, <category>
VB Example	oDesktop.AddMessage "Project1", "Mechanical1", 0, "This is a test message", ""

AreThereSimulationsRunning

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

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

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

ClearMessages

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

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

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

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

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

```
2)

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

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

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

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

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

CloseAllWindows

Closes all MDI child windows on the desktop.

UI Access	From main menu, Window > CloseAll .

Parameters	None.
Return Value	None.

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

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

CloseProject

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

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

Python Syntax	CloseProject (<ProjectName>)

Python Example	<code>oDesktop.CloseProject ("MyProject")</code>

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

CloseProjectNoForce

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

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

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

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

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

Count

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

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

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

	<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</pre>

DownloadJobResults

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

UI Access	Select Scheduler		
Parameters	Name	Type	Description
	<jobID>	String	Provide the job ID of the target job. The job ID must be able to be found in current .q (or .q.completed) file, or inside the “batchinfo” folder in projectPath. If the job ID is empty, the job ID in current .q (or .q.completed) file will be used.

	<projectPath>	String	A string of path to locate the project folder. The project file may be not necessary, but .q (or .q.completed) file or “batchinfo” folder with valid .jobid files are required.
	<resultPath>	String	A string giving the folder path for the download to save to.
	<Filters> (optional)	String	A string containing filters to download. The delimiter of file types is “;”. If no filter specified, the default filter “*” will be applied, which requests all files for download.
Return Value	A Boolean result about download complete or not		

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

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

DeleteRegistryEntry

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

UI Access	N/A		
Parameters	Name	Type	Description
	<pathToRegistry>	String	Path to Registry entry.

Return Value	Boolean: • True – Key has been deleted. • False – Key does not exist.

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

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

DoesRegistryValueExist

Determines whether a registry value exists.

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

Python Syntax	DoesRegistryValueExist(<KeyName>)
Python Example	<code>Exist = oDesktop.DoesRegistryValueExist('Desktop/ActiveDSOConfigurations/Mechanical')</code>

VB Syntax	DoesRegistryValueExist(<KeyName>)
VB Example	<code>bExist = oDesktop.DoesRegistryValueExist("Desktop/ActiveDSOConfigurations/Mechanical")</code>

EnableAutoSave

Enable or disable autosave feature.

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

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

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

ExportOptionsFiles

Copies the options config files to the *DestinationDirectory*.

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

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

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

GetActiveProject

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

Note:

GetActiveProject returns normally if there are no active objects.

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

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

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

GetActiveScheduler

Obtains the name of the scheduler active in the Desktop.

Note:

GetActiveScheduler returns normally if there are no active objects.

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

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

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

GetActiveSchedulerInfo

Obtains info for the scheduler active in the Desktop.

Note:

GetActiveSchedulerInfo returns normally if there are no active objects.

UI Access	N/A
Parameters	None.
Return Value	Info for the scheduler that is currently active in the desktop or a remote scheduler. Gets results such as "RSM", "Remote RSM", "Ansys Cloud", "SLURM", "LSF", "PBS", "SGE", "Windows HPC", as well as server info used to

	select a server. For a local scheduler, the return format resembles '{"Selected Scheduler":"RSM","Scheduler Name":"RSM","Server":""}'. For a remote scheduler, the return format resembles '{"Selected Scheduler":"Remote RSM","Scheduler Name":"SLURM","Server":"<server>"}'

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

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

GetAutoSaveEnabled

Checks whether the autosave feature is enabled.

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

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

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

GetBuildDateTimeString

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

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

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

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

GetCustomMenuSet

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

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

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

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

GetDefaultUnit

Returns the default unit for a physical quantity.

UI Access	Tools > Options > General Options > Default Units . Note that this menu only displays units that can be changed, while the script can be used to view additional default units.		
Parameters	Name	Type	Description
	<type>	String	String containing a type of measurement.

			Valid strings are (case insensitive): • "Acceleration" • "Angle" • "AngularAcceleration" • "AngularDamping" • "AngularSpeed" • "Capacitance" • "Conductance" • "Current" • "CurrentChangeRate" • "DataRate" • "DeltaH" (Magnetic Field Strength) • "Density" • "Flux" • "Force" • "Frequency" • "Inductance" • "Length" • "MagneticReluctance" • "Mass" • "MassFlowRate" • "MomentInertia" • "Power" • "Pressure" • "PressureCoefficient"

			• "Resistance" • "SaturateMagnetization" (Magnetic Inductance) • "Speed" • "Temperature" • "Time" • "Torque" • "Voltage" • "VoltageChangeRate" • "Volume" • "VolumeFlowPerPressureRoot" • "VolumeFlowRate"
Return Value	String containing the default unit (for example, "mm").		

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

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

GetDesktopConfiguration

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

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

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

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

GetDistributedAnalysisMachines

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

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

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

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

GetDistributedAnalysisMachinesForDesignType

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

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

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

VB Syntax	GetDistributedAnalysisMachinesForDesignType <designTypeName>

VB Example	<pre>Set machineNames =oDesktop. GetDistributedAnalysisMachinesForDesignType ("Mechanical")</pre>

GetExeDir

Returns the path where the executable is located.

UI Access	N/A
Parameters	None.
Return Value	String path where executable is located. Example: "C:\Program Files\ANSYS Inc\v251\AnsysEM"

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

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

GetGDIObjectCount

Note:

This command is for internal Ansys use only.

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

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

GetLibraryDirectory

Get the path to the `SysLib` directory.

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

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

VB Syntax	GetLibraryDirectory
VB Example	MsgBox oDesktop.GetLibraryDirectory

VB Example:

message box returns the path in this example

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

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

GetLocalizationHelper

Note:
This command is for internal Ansys use only.

Returns the object for the localization helper.

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

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

VB Syntax	GetLocalizationHelper
VB Example	oDesktop.GetLocalizationHelper()

GetMessages

Get the messages from a specified project and design.

UI Access	NA		
Parameters	Name	Type	Description
	<ProjectName>	String	Name of the project for which to collect messages. An incorrect project name results in no messages (design is ignored). An empty project name

			results in all messages (design is ignored)
	<DesignName>	String	Name of the design in the named project for which to collect messages. An incorrect design name results in no messages for the named project. An empty design name results in all messages for the named project
	<Severity>	Integer	Severity is 0-3, and is tied in to info/warning/error/fatal types as follows: • 0 – info and above • 1 – warning and above • 2 – error and fatal • 3 – fatal only (rarely used)
Return Value	Array of string messages.		

Python Syntax	GetMessages (<ProjectName>, <DesignName>, <Severity>)
Python Example	Messages = oDesktop.GetMessages("MyProject", "Mechanical1", 1)

VB Syntax	GetMessages <ProjectName>, <DesignName>, <Severity>
VB Examples	Messages = oDesktop.GetMessages "MyProject", "Mechanical1", 1

GetMonitorData

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

UI Access	NA

Parameters	Name	Type	Description
	<Request>	String	a json string describing what monitor data is desired. This is a list of triples separated by spaces of the form: <code>type startposition maxnumber</code>. The <code>type</code> must be one of <code>convergence</code> <code>profile</code> <code>sweptvar</code> <code>progress</code> <code>variations</code>, <code>displaytype</code>, or a guide that was previously returned in a <code>displaytype</code> block. The <code>startposition</code> is an integer that specifies where to start in the sequence of monitor data of that type. Each of these types is available in a list, ordered by time. For example, as the simulation goes on, data points are added to the internal list of convergence items. If the caller needs to incrementally update a display, it can repeatedly call this method with the <code>startposition</code> for each of the types given as the size of the list it has previously obtained and displayed, which ensures there will be no duplication and the returned data will be new each time. The <code>maxnumber</code>, if greater than 0, is the limit on the number of new items returned for each type. If the caller is able to handle any number of returned values, it can be left at 0.
Return Value	a json string with the results.		

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

GetPersonalLibDirectory

Get the path to the `PersonalLib` directory.

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

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

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

GetProcessID

Returns the process ID of `ansysedt.exe`.

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

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

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

GetProjectDirectory

Gets the path to the Project directory.

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

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

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

GetProjectList

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

UI Access	N/A

Parameters	None.
Return Value	Array of strings containing the names of all open projects in Electronics Desktop.

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

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

GetProjects

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

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

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

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

GetRegistryInt

Obtains registry key integer value.

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

Python Syntax	GetRegistryInt(<KeyName>)
Python Example	<code>num = oDesktop.GetRegistryInt('Desktop/Settings/ProjectOptions/Mechanical/UpdateReportsDynamicallyOnEdits')</code>

VB Syntax	GetRegistryInt(<KeyName>)

VB Example	<pre>num = oDesktop.GetRegistryInt("Desktop/Settings/ProjectOptions/Mechanical/UpdateReportsDynamicallyOnEdits")</pre>

GetRegistryString

Obtains registry key string value.

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

Python Syntax	<code>GetRegistryString(<KeyName>)</code>
Python Example	<pre>activeDSO = oDesktop.GetRegistryString('Desktop/ActiveDSOConfigurations/Mechanical')</pre>

VB Syntax	<code>GetRegistryString(<KeyName>)</code>
VB Example	<pre>activeDSO = oDesktop.GetRegistryString("Desktop/ActiveDSOConfigurations/Mechanical")</pre>

GetRunningInstancesMgr

Returns the object of the Running Instances Manager.

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

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

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

GetSchematicEnvironment

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

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

	• 1 = Twin Builder • 2 = Maxwell Circuit

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

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

GetScriptingToolsHelper

Note:

This command is for internal Ansys use only.

Returns the object for the scripting tools helper.

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

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

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

GetSysLibDirectory

Get the path to the `SysLib` directory.

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

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

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

GetTempDirectory

Gets the path to the Temp directory.

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

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

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

GetUserLibDirectory

Gets the path to the UserLib directory.

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

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

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

GetVersion

Returns a string representing the version.

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

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

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

IsFeatureEnabled

Returns a Boolean for whether a queried feature is enabled.

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

Pytho- n Syn- tax	IsFeatureEnabled()
Pytho- n Exam- ple	<pre> import ScriptEnv ScriptEnv.Initialize("Ansoft.ElectronicsDesktop") oDesktop.RestoreWindow() feature_strs = ["SF3519", "F195709_EXPORT_TO_EMIT", "F362235_EMIT_RESULTS_WINDOW_ IMPROVEMENTS",_ "F136736_SBR_Rough_Surface", "F353006_VOLUMETRIC_SBR", "F359673_SBR_MULTISTATE_ARRAY", "F393115_SWE_ANTENNA",_ "S196592_SBR_Directivity", "F432541_VSBR_IMPROVEMENTS", "F353007_VRT_FILTERS_ENHANCE", "S540337_SBR_REGION_LOSS_DIRECTIVITY",_ "F11941_VRT_CURRENT_DENSITY", "S544593_SBR_3D_COMPONENT_ARRAY", "F539850_SBR_GO_ BLOCKAGE", "F540275_SBR_RAYSTATS"] results = [oDesktop.IsFeatureEnabled(str) for str in feature_strs] result_txt = open("C:/Users/MyResults/Downloads/results.txt", "w") </pre>

```
for i in range(len(feature_strs)):
    result_txt.write('%s : %s\n' % (feature_strs[i], results[i]))
result_txt.close()
```

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

KeepDesktopResponsive

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

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

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

VB Syntax	KeepDesktopResponsive <TimeInMilliseconds>

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

LaunchJobMonitor

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

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

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

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

OpenAndConvertProject

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

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

			1: rename (changes extension to .aedt, the original file and results are renamed) 2: copy (creates new file with .aedt extension, and the original file and results remain available)
Return Value	An object reference to the newly opened project which has the .aedt extension		

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

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

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

OpenMultipleProjects

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

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

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

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

OpenProject

Opens a specified project.

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

Python Syntax	OpenProject(<FileName>)		

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

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

OpenProjectWithConversion

Note:
This command is for internal Ansys use only.

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

PageSetup

Specifies page settings for printing.

UI Access	File > Page Setup.		
Parameters	Name	Type	Description
	<Parameters>	Array	Structured array. [

			<pre>"NAME:PageSetupData", "margins:=", <SetupArray>]</pre>
	<SetupArray>	Array	Structured Array <pre>["left:=", <string>, "right:=", <string>, "top:=", <string>, "bottom:=", <string>)]</pre>
Return Value	None.		

Python Syntax	PageSetup(<Parameters>)
Python Example	<pre>oEditor.PageSetup(["NAME:PageSetupData", "margins:=", ["left:=", "10mm", "right:=", "10mm", "top:=", "10mm", "bottom:=", "10mm"]</pre>

	l)
VB Syntax	PageSetup <Parameters>
VB Example	<pre>oEditor.PageSetup Array("NAME:PageSetupData", "margins:=", Array("left:=", "10mm", "right:=", "10mm", "top:=", "10mm", "bottom:=", "10mm"))</pre>

PauseRecording

Temporarily stop script recording.

UI Access	NA		
Parameters	Name	Type	Description
	None		
Return Value	None		

Python Syntax	PauseRecording()

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

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

PauseScript

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

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

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

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

Print

Prints the contents of the active view window.

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

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

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

QuitApplication

Exits the desktop.

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

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

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

RefreshJobMonitor

For use in monitoring a job.

UI Access	Tools > Job Management > Monitor Jobs.
Parameters	None.
Return Value	A string specifying the job state. The result can be any of the following strings: • "Monitor Not Visible" • "Queued" • "Running" • "Shutting Down" • "Unknown" • "Completed" • "Not Monitoring" • "Starting Monitoring"

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

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

ResetLogging

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

UI Access	N/A		
Parameters	Name	Type	Description
	<code><logFile></code>	String	Path to log file.
	<code><logLevel></code>	Integer	Specified log level.
Return Value	None.		

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

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

RestoreProjectArchive

Restores a previously archived project to a specified path.

UI Access	File > Restore Archive.		
Parameters	Name	Type	Description
	<ArchiveFilePath>	String	Path to archived file
	<ProjectFilePath>	String	Path to restore location
	<OverwriteExistingFiles>	Boolean	True to overwrite an existing file of the same name; else False.
	<OpenProjectAfterRestore>	Boolean	True to open the project after it is restored; else False.
Return Value	None.		

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

VB Syntax	RestoreProjectArchive <Archivefilepath>, <ProjectFilePath>, <OverwriteExistingFiles>, <OpenProjectAfterRestore>
VB Example	oDesktop.RestoreProjectArchive "C:\Users\jdoe\Documents\OptimTee.aedtz", _

	<code>"C:\Documents\OptimTee.aedt", false, true</code>

RestoreWindow

Restores a minimized Desktop window.

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

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

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

ResumeRecording

Resume recording a script.

UI Access	N/A
Parameters	None.

Return Value	None

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

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

RunACTWizardScript

Note:

This command is for internal Ansys use only.

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

RunProgram

Runs an external program.

UI Access	NA		
Parameters	Name	Type	Description
	<ProgName>	String	Name of the program to run.
	<ProgPath>	String	Location of the program. Pass in an empty string to use the system path.
	<WorkPath>	String	Working directory in which program will start.
	<ArgArray>	Array of Strings	Arguments to pass to the program. If no arguments, pass in <code>None</code>
Return Value	None		

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

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

RunScript

Launches another script from within the script currently being executed.

UI Access	Tools>Run Script		
Parameters	Name	Type	Description
	<ScriptPath>	String	Name or full path of the script to execute. If the full path to the script is not specified, Twin Builder searches for the specified script in the following locations, in this order: 1. Personal library directory. This is the PersonalLib subdirectory in the project directory. The project directory can be specified in the General Options dialog box (click >Tools > Options > General Options to open this dialog box) under the Project Options tab. 2. User library directory. This is the userlib subdirectory in the library directory. The library directory can be specified General Options dialog in the box (click Tools > Options > General Options to to open this dialog box) under the Project Options tab. 3. System library directory. This is the syslib subdirectory in the library directory. The library directory can be specified in the General Options dialog box (click Tools > Options > General Options to open this dialog box) under the Project Options tab. 4. HFSS installation directory.
Return Value	Long the return code for the script method.		

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

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

RunScriptWithArguments

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

UI Access	NA		
Parameters	Name	Type	Description
	<ScriptPath>	String	The name or full path of the script to execute. If the full path to the script is not specified, the software looks for the script in the following locations: • Personal library directory: "PersonalLib" The PersonalLib directory can be specified in Tools > Options > General Options on the 'Project Options' tab. • User library directory: "userlib" The UserLib directory can be specified in Tools > Options > General Options on the 'Project Options' tab. • System library directory: "syslib" The SysLib directory can be specified in Tools > Options > General

	Options on the 'Project Options' tab.		
	Software installation directory		
	<Arguments>	String	The arguments to supply to the specified script.
Return Value	Long the return code for the script method.		

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

VB Syntax	RunScriptWithArguments <ScriptPath>, <Arguments>
VB Example	<pre>oDesktop.RunScriptWithArguments "C:/Project/test2.vbs", "foo"</pre>

SelectScheduler

Selects the scheduler used for batch job submission. It tries non-graphical selection of the scheduler, attempting to get version information from the scheduler in order to check for successful selection. If unable to get the information, it displays the **Select Scheduler** window and waits for the user to complete the settings.

UI Access	Tools > Job Management > Select Scheduler.		
Parameters	Name	Type	Description
	<option>	String	One of the following options (not case sensitive):

			• Empty string for remote RSM service • "RSM" for local RSM • "Windows HPC" for Windows HPC • "LSF" for Load-Sharing Facility • "SGE" for Grid Engine (GE, OGE, SGE, UGE, etc.) • "PBS" for Portable Batch Scheduler/System (PBSPro, Torque, Maui, etc.) • "Ansys Cloud" for Ansys Cloud
	<address (optional)>	String	String specifying the IP address or hostname of the head node or for the remote host running the RSM service.
	<username (optional)>	String	Username string to use for remote RSM service (or blank to use username stored in current submission host user settings). If the (non-blank) username doesn't match the username stored in current submission host user settings, then the Select Scheduler dialog is displayed to allow for password entry prior to job submission.
	<forcePasswordEntry (optional)>	String	Boolean used to force display of the Select Scheduler GUI to allow for password entry prior to job submission.
Return Value	The selected scheduler (if selection was successful, this string should match the input option string, although it could differ in upper/lowercase).		

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

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

SetActiveProject

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

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

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

VB Syntax	SetActiveProject <i><ProjectName></i>
VB Example	<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 in selected). This command permits unambiguous specification of the active project.

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

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

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

SetCustomMenuSet

Sets the custom menu set for Electronics Desktop.

UI Access	Navigate to Tools > Options > General Options > General > Desktop Configuration . Select a configuration using the Custom Menu Set drop-down menu.		
Parameters	Name	Type	Description
	<customMenuSet>	String	Name of desired menu set. Can be one of: 'Default' 'EM'

			• 'RF' • 'RF.0' • 'SI' • 'SI1.0' • 'SI2.0' • 'Twin Builder'
Return Value	None		

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

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

SetDesktopConfiguration

Sets the desktop configuration.

UI Access	Navigate to Tools > Options > General Options > General > Desktop Configuration . Select a configuration using the Set targeted configuration drop-down menu.		
Parameters	Name	Type	Description
	<configName>	String	Name of desired Desktop configuration. Can be one of: • 'All'

	<table><tr><td></td><td></td><td> • 'EM' • 'RF' • 'SI' • 'Twin Builder' </td></tr></table>			• 'EM' • 'RF' • 'SI' • 'Twin Builder'
		• 'EM' • 'RF' • 'SI' • 'Twin Builder'		
Return Value	None			

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

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

SetLibraryDirectory

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

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

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

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

SetProjectDirectory

Sets the project directory path.

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

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

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

SetRegistryFromFile

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

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

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

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

SetRegistryInt

Sets registry key to an integer value.

UI Access	N/A		
Parameters	Name	Type	Description

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

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

VB Syntax	SetRegistryInt <KeyName> <int>
VB Example	<pre>oDesktop.SetRegistryInt "Desktop/Settings/ProjectOptions/Mechanical/UpdateReportsDynamicallyOnEdits", 0</pre>

SetRegistryString

Sets registry key to a string value.

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

	value.

Python Syntax	SetRegistryString(<KeyName>, <value>)
Python Example	<code>oDesktop.SetRegistryString('Desktop/ActiveDSOConfigurations/Mechanical', 'Local')</code>

VB Syntax	SetRegistryString <KeyName>, <value>
VB Example	<code>oDesktop.SetRegistryString "Desktop/ActiveDSOConfigurations/Mechanical", "Local"</code>

SetSchematicEnvironment

Sets the schematic environment for Electronics Desktop.

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

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

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

SetTempDirectory

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

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

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

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

ShowDockingWindow

Shows or hides a docking window.

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

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

VB Syntax	ShowDockingWindow <windowName> <show>
VB Example	<pre>oDesktop.ShowDockingWindow "Message Manager" False</pre>

Sleep

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

UI Access	NA

Parameters	Name	Type	Description
	< <i>TimeInMilliseconds</i> >	Integer	The time that the execution should be suspended in milliseconds
Return Value	None		

Python Syntax	Sleep (< <i>TimeInMilliseconds</i> >)
Python Example	<code>oDesktop.Sleep(1000)</code>

VB Syntax	Sleep < <i>TimeInMilliseconds</i> >
VB Example	<code>oDesktop.Sleep 1000</code>

StopSimulations

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

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

Python Syntax	StopSimulations (<bool>, clean)

Python Example	<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.		
Parameters	Name	Type	Description
	<code><settingsPath></code>	String	Path to the settings file (exported from the Submit Job GUI) to use for submission.
	<code><projectPath></code>	String	Path to the project file to use in the batch job. This could be an archive (.aedtz file) or an un-archived project.
	<code><design (optional)></code>	String	Name of the design to use for batch solve.
	<code><setup (optional)></code>	String	Name of the specific setup to solve.
Return Value	Array of job ID strings (empty if no jobs submitted).		

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

	"Design1", "Setup1")

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

TileWindows

Arrange all open windows in a tiled format.

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

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

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

Desktop Commands For Registry Values

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

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

1. Start Mechanical.
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/Mechanical", "test")
```

See the following subtopics:

[DeleteRegistryEntry](#)

[DoesRegistryValueExist](#)

[GetRegistryInt](#)

[GetRegistryString](#)

[SetRegistryFromFile](#)

[SetRegistryInt](#)

[SetRegistryString](#)

DeleteRegistryEntry

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

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

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

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

DoesRegistryValueExist

Determines whether a registry value exists.

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

Python Syntax	DoesRegistryValueExist(<KeyName>)
Python Example	<pre>Exist = oDesktop.DoesRegistryValueExist('Desktop/ActiveDSOConfigurations/Mechanical')</pre>

VB Syntax	DoesRegistryValueExist(<KeyName>)
VB Example	<pre>bExist = oDesktop.DoesRegistryValueExist("Desktop/ActiveDSOConfigurations/Mechanical")</pre>

GetRegistryInt

Obtains registry key integer value.

UI Access	N/A

Parameters	Name	Type	Description
	<KeyName>	String	Full name of registry key, including path.
Return Value	Integer if the integer value is found. Return as Bad-Argument-Value if registry key does not exist or it is not an integer value.		

Python Syntax	GetRegistryInt(<KeyName>)
Python Example	num = oDesktop.GetRegistryInt('Desktop/Settings/ProjectOptions/Mechanical/Up-dateReportsDynamicallyOnEdits')

VB Syntax	GetRegistryInt(<KeyName>)
VB Example	num = oDesktop.GetRegistryInt("Desktop/Settings/ProjectOptions/Mechanical/Up-dateReportsDynamicallyOnEdits")

GetRegistryString

Obtains registry key string value.

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

Python Syntax	GetRegistryString(<KeyName>)
Python Example	<code>activeDSO = oDesktop.GetRegistryString('Desktop/ActiveDSOConfigurations/Mechanical')</code>

VB Syntax	GetRegistryString(<KeyName>)
VB Example	<code>activeDSO = oDesktop.GetRegistryString("Desktop/ActiveDSOConfigurations/Mechanical")</code>

SetRegistryFromFile

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

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

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

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

SetRegistryInt

Sets registry key to an integer value.

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

Python Syntax	SetRegistryInt(<KeyName>, <int>)
Python Example	oDesktop.SetRegistryInt('Desktop/Settings/ProjectOptions/Mechanical/UpdateReportsDynamicallyOnEdits', 0)

VB Syntax	SetRegistryInt <KeyName> <int>
VB Example	oDesktop.SetRegistryInt "Desktop/Settings/ProjectOptions/Mechanical/UpdateReportsDynamicallyOnEdits", 0

e	

SetRegistryString

Sets registry key to a string value.

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

Python Syntax	SetRegistryString(<KeyName>, <value>)
Python Example	<code>oDesktop.SetRegistryString('Desktop/ActiveDSOConfigurations/Mechanical', 'Local')</code>

VB Syntax	SetRegistryString <KeyName>, <value>
VB Example	<code>oDesktop.SetRegistryString "Desktop/ActiveDSOConfigurations/Mechanical", "Local"</code>

ImportExport Tool Commands

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

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

Scripts run via the ImportExport Tool include:

[ImportANF](#)

[ImportANFV2](#)

[ImportAutoCAD](#)

[ImportAWRMicrowaveOffice](#)

[ImportEDB](#)

[ImportExtracta](#)

[ImportGDSII](#)

[ImportGerber](#)

[ImportIDF](#)

[ImportIDFandMerge](#)

[ImportIPC](#)

[ImportODB](#)

[ImportXFL](#)

ImportANF

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

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

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

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

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

ImportANFv2

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

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

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

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

	<pre>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
	<dxfFileName>	String	Full path of DXF file.

	<outputPathFileName>	String	Full path of EDB file to create during import.
	<controlFileName>	String	Full path of XML control file. Pass empty string if none.
Return Value	None.		

Python Syntax	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.

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

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

VB Syntax	<code>ImportAWRMicrowaveOffice <xmlFileName>, <outputPathName>, <logFileName></code>
VB Example	<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.ImportAWRMicrowaveOffice "C:/MyFiles/package4.xml", "C:/MyFiles/package4.aedb",
-
"C:/MyFiles/package4.log"

```

ImportEDB

Imports an EDB file into a new project.

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

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

VB Syntax	ImportEDB <edbFileName>

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

ImportExtracta

Imports a Cadence Extracta file into a new project.

Note:

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

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

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

VB Syntax	<code>ImportExtracta <extractaFileName>, <outputPathName>, <controlFileName></code>
VB Example	<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.

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

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

	<code>'C:/Files/test.ircx', 'C:/Files/test.xml')</code>

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

ImportGerber

Imports a GERBER file into a new project.

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

Python Syntax	<code>ImportGerber(<gerberFileName>, <outputPathName>, <controlFileName>)</code>
Python Example	<pre>oDesktop.RestoreWindow() Set oTool = oDesktop.GetTool('ImportExport')</pre>

	<pre>oTool.ImportGERBER('C:/Files/test.gbr', 'C:/Files/test.aedb.edb', 'C:/Files/test.xml')</pre>

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

ImportIDF

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

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

	<NumInternalLayer>	int	Number of internal layers
	<HighSurfaceCopper>	int	High side surface coverage percentage
	<LowSurfaceCopper>	int	Low side surface coverage percentage
	<InternalLayerCopper>	int	Internal layer coverage percentage
	<TraceMaterial>	string	Trace material name
	<SubstrateMaterial>	int	Substrate material name
	<CreateBoard>	bool	True or False
	<ModelBoardAsRect>	bool	True or False
	<ModelDeviceAsRect>	bool	True or False
	<Cutoff>	bool	True or False
	<IncludeDrilledHoles>	bool	True or False
	<HoleDiameterCutoff>	string	Hole diameter size and unit
	<ReplaceDevices>	bool	True or False
	<CreatePointsOnBoardForInstances>	bool	True or False
Return Value	None.		

Python Syntax	ImportIDF_1 (<NAME>, <Board>, <Library>, <Control>, <Filters>, <CreateFilteredAsNonModel>, <NAME>, <NAME>, <HighSurfThickness>, <LowSurfThickness>, <InternalLayerThickness>, <NumInternalLayer>, <HighSurfaceCopper>, <LowSurfaceCopper>, <InternalLayerCopper>, <TraceMaterial>, <SubstrateMaterial>, <CreateBoard>, <ModelBoardAsRect>, <ModelDeviceAsRect>, <Cutoff>, <IncludeDrilledHoles>, <ReplaceDevices>)
Python Example	<pre>oDesign.ImportIDF(["NAME:Settings",</pre>

```

"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,
"FootPrint:="            , "0.1mm2",
[
    "NAME:definitionOverridesMap"
],
[
    "NAME:instanceOverridesMap"
],
"HighSurfThickness:="    , "0.07mm",
"LowSurfThickness:="     , "0.07mm",
"InternalLayerThickness:=", "0.07mm",
"NumInternalLayer:="     , 2,
"HighSurfaceCopper:="    , 30,
"LowSurfaceCopper:="     , 30,
"InternalLayerCopper:="  , 30,
"TraceMaterial:="        , "Cu-Pure",
"SubstrateMaterial:="    , "FR-4",
"CreateBoard:="          , True,
"ModelBoardAsRect:="     , True,
"ModelDeviceAsRect:="    , False,
"Cutoff:="               , False,
"IncludeDrilledHoles:="  , True,
"HoleDiameterCutoff:="   , "0.1mm",
"ReplaceDevices:="       , False,
"CreatePointsOnBoardForInstances:=", False
] )

```

VB Syntax	ImportIDF <idfFileName>, <outputPathName>, <controlFileName>, <libFileName>
VB Example	<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](#).

UI Access	File > Import > IDF. Enable Merge with existing and select a project/design.		
Parameters	Name	Type	Description

	<idfFileName>	String	Full path of GERBER file.
	<libFileName>	String	Optional. Full path of library file. Pass empty string if none.
	<controlFileName>	String	Optional. Full path of XML control file. Pass empty string if none.
	<project>	String	Name of project to merge with.
	<design>	String	Name of design to merge with.
Return Value	None.		

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

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

ImportIDX

Imports and IDX model into a new Icepak project.

UI Access	Icepak > Import IDX

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

Python Syntax	ImportIDX (<NAME>, <Board>, <Library>, <Control>, <Filters>, <CreateFilteredAsNonModel>, <NAME>, <NAME>, <HighSurfThickness>, <LowSurfThickness>, <InternalLayerThickness>, <NumInternalLayer>,

	<code><HighSurfaceCopper>, <LowSurfaceCopper>, <InternalLayerCopper>, <TraceMaterial>, <SubstrateMaterial>, <CreateBoard>, <ModelBoardAsRect>, <ModelDeviceAsRect>, <Cutoff>, <ReplaceDevices>)</code>
Python Example	<pre> 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]) </pre>

VB Syntax	ImportIDX (<NAME>, <Board>, <Library>, <Control>, <Filters>, <CreateFilteredAsNonModel>, <NAME>, <NAME>, <HighSurfThickness>, <LowSurfThickness>, <InternalLayerThickness>, <NumInternalLayer>, <HighSurfaceCopper>, <LowSurfaceCopper>, <InternalLayerCopper>, <TraceMaterial>, <SubstrateMaterial>, <CreateBoard>, <ModelBoardAsRect>, <ModelDeviceAsRect>, <Cutoff>, <ReplaceDevices>)
VB Example	<pre>oDesign.ImportIDX Array("NAME:Settings", "Board:=", _"C:\Users\Model files" _ "\PCB-00278_A.idx", "Library:=", "", "Control:=", "", "Filters:=", Array(_"HeightExclude2D"), "CreateFilteredAsNonModel:=", false, Array("NAME:definitionOverridesMap"), Array("NAME:instanceOverridesMap"), "HighSurfThickness:=", _0.07mm", "LowSurfThickness:=", "0.07mm", "InternalLayerThickness:=", "0.07mm", "NumInternalLayer:=", _2, "HighSur- faceCopper:=", 30, "LowSurfaceCopper:=", 30, "InternalLayerCopper:=", _30, "TraceMaterial:=", "Cu-Pure", "SubstrateMaterial:=", "FR-4", "CreateBoard:=", _true, "ModelBoardAsRect:=", false, "ModelDeviceAsRect:=", false, "Cutoff:=", _false, "ReplaceDevices:=", false)</pre>

ImportIPC

Imports an IPC2581 file into a new project.

UI Access	File > Import > IPC2581.		
Parameters	Name	Type	Description
	<ipcFileName>	String	Full path of IPC2581 file.
	<outputPathName>	String	Full path of EDB file to create during import.
	<controlFileName>	String	Optional. Full path of XML control file. Pass empty string if none.

Return Value	None.

Python Syntax	<code>ImportIPC(<ipcFileName>, <outputPathName>, <controlFileName>)</code>
Python Example	<pre>oDesktop.RestoreWindow() Set oTool = oDesktop.GetTool('ImportExport') oTool.ImportIPC('C:/Files/test.cvg', 'C:/Files/test.aedb', 'C:/Files/test.xml', 'C:/Files/test.txt')</pre>

VB Syntax	<code>ImportIPC <ipcFileName>, <outputPathName>, <controlFileName></code>
VB Example	<pre>Set oTool = oDesktop.GetTool("ImportExport") oTool.ImportIPC "C:/Files/test.cvg", "C:/Files/test.aedb", _ "C:/Files/test.xml"</pre>

ImportODB

Imports an ODB++ file into a new project.

UI Access	File > Import > ODB++.		
Parameters	Name	Type	Description
	<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.
Return Value	None.		

Python Syntax	<code>ImportODB(<odbFileName>, <outputPathName>, <controlFileName>)</code>
Python Example	<pre>oDesktop.RestoreWindow() Set oTool = oDesktop.GetTool('ImportExport') oTool.ImportODB('C:/Files/test.odb', 'C:/Files/test.aedb', 'C:/Files/test.xml')</pre>

VB Syntax	<code>ImportODB <odbFileName>, <outputPathName>, <controlFileName></code>
VB Example	<pre>Set oTool = oDesktop.GetTool("ImportExport") oTool.ImportODB "C:/Files/test.odb", "C:/Files/test.aedb", _ "C:/Files/test.xml"</pre>

ImportXFL

Imports an XFL file into a new project.

UI Access	File > Import > XFL.		
Parameters	Name	Type	Description
	<code><xflFileName></code>	String	Full path of XFL file.
	<code><outputPathName></code>	String	Full path of EDB file to create during import.
	<code><controlFileName></code>	String	Optional. Full path of XML control file. Pass empty string if none.

Return Value	None.

Python Syntax	<code>ImportXFL(<xflFileName>, <outputPathName>, <controlFileName>)</code>
Python Example	<pre>oDesktop.RestoreWindow() Set oTool = oDesktop.GetTool('ImportExport') oTool.ImportXFL('C:/Files/test.xfl', 'C:/Files/test.aedb', 'C:/Files/test.xml')</pre>

VB Syntax	<code>ImportXFL <xflFileName>, <outputPathName>, <controlFileName></code>
VB Example	<pre>Set oTool = oDesktop.GetTool("ImportExport") oTool.ImportXFL "C:/Files/test.xfl", "C:/Files/test.aedb", _ "C:/Files/test.xml"</pre>

This page intentionally
left blank.

5 - Running Instances Manager Script Commands

The Running Instances Manager is a scripting object that lets you identify and connect to all running instances of Electronics Desktop. oDesktop objects that are returned provide full scripting functionality. Running Instances Manager commands should be executed by the oDesktop object. For example:

```
Set oRunningInstances = oDesktop.GetRunningInstancesMgr()
```

[GetAllRunningInstances](#)

[GetRunningInstanceByProcessID](#)

[GetRunningInstanceByProject](#)

GetAllRunningInstances

Returns a list of running instances of Ansys Electronics Desktop.

UI Access	N/A
Parameters	None.
Return Value	Array containing list of Ansys Electronics Desktop instances.

Python Syntax	GetAllRunningInstances()
Python Example	<code>obj = oRunningInstances.GetAllRunningInstances()</code>

VB Syntax	GetAllRunningInstances
VB Example	<code>set obj = oRunningInstances.GetAllRunningInstances()</code>

GetRunningInstanceByProcessID

Returns the instance of Ansys Electronics Desktop that is running a specified process.

UI Access	N/A		
Parameters	Name	Type	Description
	<processID>	Integer	Process ID
Return Value	String of the returned instance.		

Python Syntax	GetRunningInstanceByProcessID(<processID>)
Python Example	<code>obj = oRunningInstances.GetRunningInstanceByProcessID(12345)</code>

VB Syntax	GetRunningInstanceByProcessID <processID>
VB Example	<code>set obj = oRunningInstances.GetRunningInstanceByProcessID(12345)</code>

GetRunningInstancesMgr

Returns the object of the Running Instances Manager.

UI Access	N/A		
Parameters	Name	Type	Description
	None		

Return Value	Object Running instances manager object

Python Syntax	<code>GetRunningInstancesMgr()</code>
Python Example	<code>oRunningInstances = oDesktop.GetRunningInstancesMgr()</code>

VB Syntax	<code>GetRunningInstancesMgr()</code>
VB Example	<code>Set oRunningInstances = oDesktop.GetRunningInstancesMgr()</code>

This page intentionally
left blank.

6 - Project Object Script Commands

Project commands should be executed by the oProject object.

One example of accessing this object is:

```
Set oProject = oDesktop.GetActiveProject()
```

Dataset Script Commands:

[AddDataset](#)

[DeleteDataset](#)

[EditDataset](#)

[ExportDataset](#)

[HasDataset](#)

[ImportDataset](#)

Other Project Object Script Commands:

[AnalyzeAll](#)

[ChangeProperty](#)

[ClearMessages](#)

[CloneMaterial](#)

[Close](#)

[CopyDesign](#)

[CutDesign](#)

[DeleteDesign](#)

[DeleteToolObject](#)

[ExportMaterial](#)

[GetActiveDesign](#)

[GetArrayVariables](#)

[GetChildNames \[Project\]](#)

[GetChildObject \[Project\]](#)

[GetChildTypes \[Project\]](#)

[GetConfigurableData](#)

[GetDefinitionManager](#)

[GetDependentFiles](#)

[GetDesign](#)

[GetDesigns](#)

[GetEDBHandle](#)

[GetLegacyName](#)

[GetName \[Project\]](#)

[GetObjPath \[Project\]](#)

[GetPath](#)

[GetPropEvaluatedValue](#)

[GetPropNames \[Project\]](#)

[GetPropSIValue](#)

[GetPropValue \[Project\]](#)

[GetProperties](#)

[GetPropertyValue](#)

[GetTopDesignList](#)

[GetVariableValue](#)

[GetVariables](#)

[InsertDesign](#)

[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
	<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

```
oProject.AddDataset(  
[  
  "NAME:$ds1",  
  [  
    "NAME:Coordinates",  
    [  
      "NAME:Coordinate",  
      "X:=", 2,  
      "Y:=", 4  
    ],  
    [  
      "NAME:Coordinate",  
      "X:=", 6,  
      "Y:=", 8  
    ]  
  ]  
]  
)  
oDesign.AddDataset(  
[  
  "NAME:$ds1",
```


	<pre>["NAME:Coordinates", ["NAME:Coordinate", "X:=", 2, "Y:=", 4], ["NAME:Coordinate", "X:=", 6, "Y:=", 8]]</pre>

VB Syntax	AddDataset <DatasetDataArray>
VB Example	<pre>oProject.AddDatasetArray("NAME:ds1", Array("NAME:Coordinates",</pre>

	<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))) oDesign.AddDatasetArray("NAME:ds1", Array("NAME:Coordinates", Array("NAME:Coordinate", "X:=", 1, "Y:=", 2, Array("NAME:Coordinate", "X:=", 3, "Y:=", 4), Array("NAME:Coordinate", "X:=", 5, "Y:=", 7), Array("NAME:Coordinate", "X:=", 6, "Y:=", 20)))</pre>

AddMaterial

Adds a local material.

UI Access	Add Material in the material editor.		
Parameters	Name	Type	Description
	<MaterialParams>	Array	["NAME: <name of the material to be added>", <MatProperty>, <MatProperty>, ...]
	<MatProperty>	Array	For simple material:

			"<PropertyName>:=", <value> For anisotropic material: ["NAME:<PropertyName>", "property_type:=", "AnisoProperty", "unit:=", <Unit>", "component1:=", <value>, "component2:=", <value>, "component3:=", <value>))]
	<PropertyName>	String	Should be one of the following (depending on the material, design, and solution types): Electromagnetic (Maxwell-exclusive material properties omitted, see Maxwell Scripting help): "permittivity", "permeability", "conductivity", "dielectric_loss_tangent", "magnetic_loss_tangent", "electric_coercivity", "magnetic_coercivity", "saturation_mag", "lande_g_factor", "delta_H", "delta_h_freq", "mass_density" Thermal (including solids, Icepak fluid flow, and Mechanical rotating fluid modeling):

			"thermal_conductivity", "mass_density", "specific_heat", "thermal_expansion_coefficient", "thermal_material_type", "viscosity", "diffusivity", "molecular_mass", "clarity_type" Structural: "mass_density", "youngs_modulus", "poissons_ratio", "thermal_expansion_coefficient"
	<Unit>	String	Possible values (Maxwell-exclusive properties omitted, see Maxwell Scripting Help; other missing entries are unitless): conductivity: "siemens/m" saturation_mag: "uTesla", "mTesla", "tesla", "kTesla", "uGauss", "mGauss", "gauss", "kGauss" delta_H: "A_per_meter", "kA_per_meter", "Oe", "kOe" delta_h_frequency: "Hz", "kHz", "MHz", "GHz", "THz", "rps", "per_sec" mass_density: "kg/m^3" thermal_conductivity: "W/m-C" specific_heat: "J/kg-C" youngs_modulus: "N/m^2" thermal_expansion_coefficient: "1/C"
Return Value	None		

Python Syntax	AddMaterial(["NAME:<MaterialName>", <MatProperty>, <MatProperty>, ...])
Python Example	<pre>oDefinitionManager.AddMaterial(["permittivity:=", "2.2", "0.002"]) oDefinitionManager.AddMaterial [("NAME:Material2", _ "dielectric_loss_tangent:=", "44", Array("NAME:saturation_mag", _ "property_type:=", "AnisoProperty", _ "unit:=", "Gauss", _ "component1:=", "11", _ "component2:=", "22", _ "component3:=", "33"), _ "delta_H:=", "440e")]</pre>

VB Syntax	AddMaterial Array("NAME:<MaterialName>", <MatProperty>, <MatProperty>, ...)
VB Example	<pre>oDefinitionManager.AddMaterial Array(</pre>

	<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")</pre>

AnalyzeAll [project]

Runs the project-level script command from the script, which simulates all solution setups and Optimetrics setups for all design instances in the project. The UI waits until simulation is finished before continuing with the script.

UI Access	Project > Analyze All.
Parameters	None.
Return Value	None.

Python Syntax	AnalyzeAll()
Python Example	<code>oProject.AnalyzeAll()</code>

VB Syntax	AnalyzeAll
VB Example	<code>oProject.AnalyzeAll</code>

ChangeProperty

Changes the properties of an object in the history tree.

UI Access	Right-click an object in the History Tree and select Properties .		
Parameters	Name	Type	Description
	<code><propertyArgs></code>	Array	Structured array. The properties vary depending on the object. Due to the number of potential configurations, it is recommended that you generate this script using the UI's Automation tab.
Return Value	None.		

Python Syntax	<code>ChangeProperty(<propertyArgs>)</code>
Python Example	Example: Changing the Position of a Box and the Reference Temperature (for Structural Solutions only) <pre>oEditor.ChangeProperty(["NAME:AllTabs", ["NAME:Geometry3DCmdTab",</pre>

```

        [
            "NAME:PropServers" ,
            "Box1:CreateBox:1"
        ],
        [
            "NAME:ChangedProps",
            [
                "NAME:Position",
                "X:="          , "0.35in",
                "Y:="          , "0.55in",
                "Z:="          , "0in"
            ],
            [
                "NAME:Reference Temperature", # Applicable only to Mechanical-
                "Value:="          , "27cel"   # Structural solutions
            ]
        ]
    ]
])

```

Example: Changing a Box's Material and Wireframe Display

```

oEditor.ChangeProperty(
[
    "NAME:AllTabs",
    [
        "NAME:Geometry3DAttributeTab",
        [
            "NAME:PropServers" ,
            "Box1"
        ],
        [
            "NAME:ChangedProps",
            [
                "NAME:Material",

```



```

        "Value:="      , "\"vacuum\""
    ],
    [
        "NAME:Display Wireframe",
        "Value:="      , True
    ]
]
]
])
Example: Change Default Handle (reference coordinate system)
oEditor.ChangeProperty(
    [
        "NAME:AllTabs",
        [
            "NAME:General",
            [
                "NAME:PropServers",
                "HFSSDesign1_1"
            ],
            [
                "NAME:ChangedProps",
                [
                    "NAME:Handle",
                    "Value:="      , " NotchOutY"
                ]
            ]
        ]
    ]
)

```

```

oEditor.ChangeProperty(
[
  "NAME:AllTabs",
  [
    "NAME:Component Data",
    [
      "NAME:PropServers",
      "BoundarySetup:LC1_1_Port1"
    ],
    [
      "NAME:ChangedProps",
      [
        "NAME:Voltage",
        "Value:=" , "10mV"
      ]
    ]
  ]
])
...

```

Example: Change Material Import for a Discovery Model

```

oEditor = oDesign.SetActiveEditor("3D Modeler")
oEditor.ChangeProperty(
[
  "NAME:AllTabs",
  [
    "NAME:Options",
    [
      "NAME:PropServers",
      "Discovery1"
    ],
    [
      "NAME:ChangedProps",
      [
        "NAME:Materials",
        "Value:=" , "Assignments and Properties"
      ]
    ]
  ]
])
...

```

	<pre>]]])</pre>

VB Syntax	ChangeProperty <propertyArgs>
VB Example	Example: Changing the Position of a Box and the Reference Temperature (for Structural Solutions only) <pre>oEditor.ChangeProperty Array("NAME:AllTabs", Array("NAME:Geometry3DCmdTab", Array("NAME:PropServers" , "Box1:CreateBox:1"), Array("NAME:ChangedProps", Array("NAME:Position", "X:=" , "0.35in", "Y:=" , "0.55in", "Z:=" , "0in"), Array("NAME:Reference Temperature", ' Applicable only to Mechanical- "Value:=", "27cel" ' Structural solutions))))</pre> Example: Changing a Box's Material and Wireframe Display

```

oEditor.ChangeProperty(
Array("NAME:AllTabs",
    Array("NAME:Geometry3DAttributeTab",
        Array("NAME:PropServers",
            "Box1"
        ),
        Array("NAME:ChangedProps",
            Array("NAME:Material",
                "Value:=", "\"vacuum\""
            ),
            Array("NAME:Display Wireframe",
                "Value:=", True
            )
        )
    )
)

```

Example: Change Default Handle (reference coordinate system)

```

oEditor.ChangeProperty
Array("NAME:AllTabs",
    Array("NAME:General",
        Array("NAME:PropServers",
            "HFSSDesign1_1"
        ),
        Array("NAME:ChangedProps",
            Array("NAME:Handle",
                "Value:=", " NotchOutY"
            )
        )
    )
)

```

	)
	)

ClearMessages

Clears information in the **Messages** window.

Note:

Additional options are available when using the Desktop-level [ClearMessages](#) command.

UI Access	N/A
Parameters	None.
Return Value	None.

Python Syntax	ClearMessages()
Python Example	<code>oProject.ClearMessages()</code>

VB Syntax	ClearMessages
VB Example	<code>oProject.ClearMessages</code>

CloneMaterial

Clones a local material.

UI Access	N/A		
Parameters	Name	Type	Description
	<matName>	String	Name of existing material.
	<newName>	String	Name for newly cloned material.
Return Value	Boolean: • 1 - Material is cloned. • 0 - Existing material not found or a conflict with the new material name.		

Python Syntax	CloneMaterial (<matName>, <newName>)
Python Example	oDefinitionManager.CloneMaterial ("copper1", "copper3")

VB Syntax	CloneMaterial <matName>, <newName>
VB Example	oDefinitionManager.CloneMaterial "copper1", "copper3"

Close

Closes the active project.

Warning:

Unsaved changes will be lost.

UI Access	N/A
Parameters	None.
Return Value	None.

Python Syntax	Close()
Python Example	<code>oProject.Close()</code>

VB Syntax	Close
VB Example	<code>oProject.Close</code>

CopyDesign

Copies a specified design.

UI Access	Edit > Copy.		
Parameters	Name	Type	Description
	<DesignName>	String	Name of the design to copy from.

Return Value	None.

Python Syntax	<code>CopyDesign (<DesignName>)</code>
Python Example	<code>oProject.CopyDesign ("HFSSDesign1")</code>

VB Syntax	<code>CopyDesign <DesignName></code>
VB Example	<code>oProject.CopyDesign "HFSSDesign1"</code>

CutDesign

Cuts a design from the active project. The design is stored in memory and can be pasted.

Warning:

This is a legacy command that is no longer supported and should not be used as it may have unintended effects on solved designs.

UI Access	Edit > Cut.		
Parameters	Name	Type	Description
	<DesignName>	String	Name of the design.
Return Value	None.		

Python Syntax	CutDesign (<DesignName>)
Python Example	<code>oProject.CutDesign("SimplorerDesign1")</code>

VB Syntax	CutDesign <DesignName>
VB Example	<code>oProject.CutDesign "SimplorerDesign1"</code>

DeleteDataset

Deletes a specified dataset. This can be executed by the oProject, or oDesign variables.

UI Access	Project > Datasets > Remove.		
Parameters	Name	Type	Description
	<DatasetName>	String	Name of the dataset found in the project.
Return Value	None.		

Python Syntax	DeleteDataset (<DatasetName>)
Python Example	<code>oProject.DeleteDataset('\$ds1')</code>
	<code>oDesign.DeleteDataset('\$ds1')</code>

VB Syntax	DeleteDataset <DatasetName>

VB Example	<pre>oProject.DeleteDataset "\$ds1" oDesign.DeleteDataset "\$ds1"</pre>

DeleteDesign

Deletes a specified design in the project.

UI Access	Edit > Delete , or Delete in the ribbon.		
Parameters	Name	Type	Description
	<DesignName>	String	Name of the design.
Return Value	None.		

Python Syntax	DeleteDesign (<DesignName>)
Python Example	<pre>oProject.DeleteDesign ("MechanicalDesign2")</pre>

VB Syntax	DeleteDesign <DesignName>
VB Example	<pre>oProject.DeleteDesign "MechanicalDesign2"</pre>

DeleteToolObject

Note:

This command is for internal Ansys use only.

UI Access	N/A		
Parameters	Name	Type	Description
	<ObjectName>	String	Name of the tool object.
Return Value	None.		

Python Syntax	DeleteToolObject(<ObjectName>)		
Python Example	<code>oProject.DeleteToolObject("object1")</code>		

VB Syntax	DeleteToolObject <ObjectName>		
VB Example	<code>oProject.DeleteToolObject "object1"</code>		

EditDataset

Modifies a dataset. This can be executed by the oProject, or oDesign variables.

UI Access	Project > Datasets > Edit.		
Parameters	Name	Type	Description
	<OriginalName>	String	Name of the original dataset.
	<DatasetDataArray>	Array	Data for the modified dataset.
Return Value	None.		

Python Syntax	EditDataset (<OriginalName> <DatasetDataArray>)
Python Example	<pre>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</pre>

	<pre>], ["NAME:Coordinate", "X:=", 3, "Y:=", 4]]]) </pre>

VB Syntax	EditDataset <OriginalName> <DatasetDataArray>
VB Example	<pre> oProject.EditDataset "ds1" Array("NAME:ds2",_ Array("NAME:Coordinates",Array("NAME:Coordinate", "X:=", 1, "Y:=", 2), Array("NAME:Coordinate", "X:=", 3, "Y:=", 4))) oDesign.EditDataset "ds1" Array("NAME:ds2",_ Array("NAME:Coordinates",Array("NAME:Coordinate", "X:=", 1, "Y:=", 2), Array("NAME:Coordinate", "X:=", 3, "Y:=", 4))) </pre>

EditMaterial

Modifies an existing material.

UI Access	View/Edit Materials command in the material editor		
Parameters	Name	Type	Description
	<OriginalName>	String	Name of the material before editing.
	<MatProperties>	Array	Structured array containing material properties: ["NAME:<New material name>", "CoordinateSystemType:=", <string>, "BulkOrSurfaceType:=" , <integer>, ["NAME:PhysicsTypes", "set:=" , <array containing string physics types>], <Optional ModifierDataArray>, "permeability:=" , <string containing value>, "conductivity:=" , <string containing value>, "thermal_conductivity:=", <string containing value>, "mass_density:=" , <string containing value>, "specific_heat:=" , <string containing value>, "youngs_modulus:=" , <string containing value>,]

			<pre>"poissons_ratio:=" , <string containing value>, "thermal_expansion_coefficient:=" , <string containing value>]</pre>
	<code><ModifierDataArray></code>	Array	Optional structured array containing thermal or spatial modifiers: <pre>["NAME:ModifierData", ["NAME:<ThermalModifierData or SpatialModifierData>", "modifier_data:=" , <"thermal_modifier_data" or "spa- tial_modifier_data">, ["NAME:<all_thermal_modifiers or all_spatial_mod- ifiers>", ["NAME:<modifierName>", "Property::=" , <string property being mod- ified>, "Index::=" , <integer>, "prop_modifier:=" , <"thermal_modifier" or "spa- tial_modifier">, "use_free_form:=" , <Boolean>, </pre>

	<pre> "free_form_value:=" , <string modifier value>,]]]</pre>
Return Value	None.

Python Syntax	EditMaterial (<OriginalName>, <MatProperties>)
Python Example	Without Modifiers: <pre>oDefinitionManager.EditMaterial("alumina_92pct", ["NAME:alumina_92pct", "CoordinateSystemType:=" , "Cartesian", "BulkOrSurfaceType:=" , 1, ["NAME:PhysicsTypes", "set:=" , ["Electromagnetic","Thermal","Structural"]], "permittivity:=" , "9.3", "dielectric_loss_tangent:=" , "0.008",</pre>


```
"thermal_conductivity:=" , "26",  
"mass_density:=" , "3720",  
"specific_heat:=" , "790",  
"youngs_modulus:=" , "267000000000",  
"poissons_ratio:=" , "0.26",  
"thermal_expansion_coefficient:=" , "7.2e-006"  
]  
)
```

With Thermal Modifier:

```
oDefinitionManager.EditMaterial("copper",  
[  
  "NAME:copper",  
  "CoordinateSystemType:=" , "Cartesian",  
  "BulkOrSurfaceType:=" , 1,  
  [  
    "NAME:PhysicsTypes",  
    "set:=" , ["Electromagnetic","Thermal","Structural"]  
  ],  
[  
  "NAME:ModifierData",
```

```
[
  "NAME:ThermalModifierData",
  "modifier_data:=" , "thermal_modifier_data",
  [
    "NAME:all_thermal_modifiers",
    [
      "NAME:one_thermal_modifier",
      "Property:=" , "permittivity",
      "Index:=" , 0,
      "prop_modifier:=" , "thermal_modifier",
      "use_free_form:=" , True,
      "free_form_value:=" , "if(Temp > 1000cel, 1, if(Temp < -273.15cel, 1, 1))"
    ]
  ]
],
"permeability:=" , "0.999991",
"conductivity:=" , "58000000",
"thermal_conductivity:=" , "400",
"mass_density:=" , "8933",
"specific_heat:=" , "385",
```

```
"youngs_modulus:=" , "120000000000",
"poissons_ratio:=" , "0.38",
"thermal_expansion_coefficient:=" , "1.77e-05"
])
```

Transient Solve, Non-linear Drude Data Plasma

```
import ScriptEnv
ScriptEnv.Initialize("Ansoft.ElectronicsDesktop")
oDesktop.RestoreWindow()
oProject = oDesktop.SetActiveProject("Drude_plasma_parameters_r231")
oDefinitionManager = oProject.GetDefinitionManager()
oDefinitionManager.EditMaterial("Drude",
[
    "NAME:Drude",
    "CoordinateSystemType:=", "Cartesian",
    "BulkOrSurfaceType:=" , 1,
    [
        "NAME:PhysicsTypes",
        "set:=" , ["Electromagnetic"]
    ],
    [
```

	<pre> "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]]]) </pre>

VB Syntax	EditMaterial <OriginalName>, <MatProperties>
VB Example	Without Modifiers: <pre> oDefinitionManager.EditMaterial "alumina_92pct", Array("NAME:alumina_92pct", </pre>

```

"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",
"specific_heat:=", "790",
"youngs_modulus:=", "267000000000",
"poissons_ratio:=", "0.26",
"thermal_expansion_coefficient:=", "7.2e-006"
)

With Thermal Modifier:

oDefinitionManager.EditMaterial "copper",
    Array("NAME:copper",
        "CoordinateSystemType:=", "Cartesian",
        "BulkOrSurfaceType:=" , 1,
        Array("NAME:PhysicsTypes",
            "set:=" , Array("Electromagnetic","Thermal","Structural")),

```

```
Array("NAME:ModifierData",
      Array("NAME:ThermalModifierData",
            "modifier_data:=" , "thermal_modifier_data",
            Array("NAME:all_thermal_modifiers",
                  Array("NAME:one_thermal_modifier",
                        "Property:=" , "permittivity",
                        "Index:=" , 0,
                        "prop_modifier:=" , "thermal_modifier",
                        "use_free_form:=" , True,
                        "free_form_value:=" , "if(Temp > 1000cel, 1, if(Temp < -273.15cel, 1, 1))"
                      )
                    )
              )
    ),
    "permeability:=" , "0.999991",
    "conductivity:=" , "580000000",
    "thermal_conductivity:=" , "400",
    "mass_density:=" , "8933",
    "specific_heat:=" , "385",
    "youngs_modulus:=" , "1200000000000",
    "poissons_ratio:=" , "0.38",
```

	<pre>"thermal_expansion_coefficient:=" , "1.77e-05")</pre>

ExportDataset

Exports a dataset to a named file. This can be executed by the oProject, or oDesign variables.

UI Access	Project > Datasets > Export.		
Parameters	Name	Type	Description
	<datasetFilePath>	String	The full path to the file.
Return Value	None.		

Python Syntax	ExportDataset (<datasetFilePath>)
Python Example	<pre>oProject.ExportDataset('e:/tmp/dsdata.txt')</pre>
	<pre>oDesign.ExportDataset('e:/tmp/dsdata.txt')</pre>

VB Syntax	ExportDataset <datasetFilePath>
VB Example	<pre>oProject.ExportDataset "e:/tmp/dsdata.txt"</pre>
	<pre>oDesign.ExportDataset "e:/tmp/dsdata.txt"</pre>

ExportMaterial

Exports a local material to a library.

UI Access	Export to Library command in the material editor.		
Parameters	Name	Type	Description
	<ExportData>	Array	["NAME:<LibraryName>", <MaterialName>, <MaterialName>, ...]
	<LibraryName>	String	Name of the exported library.
	<MaterialName>	String	Name of the material to be exported.
	<LibraryLocation>	String	Location to save the library. Only "PersonalLib" and "UserLib" are allowed.
Return Value	None.		

Python Syntax	ExportMaterial (<ExportData>, <LibraryLocation>)
Python Example	<pre>oDefinitionManager.ExportMaterial (["NAME:mylib",_ "Material1", "Material2", "Material3"], "PersonalLib")</pre>

VB Syntax	ExportMaterial <ExportData>, <LibraryLocation>
VB Example	<pre>oDefinitionManager.ExportMaterial Array("NAME:mylib",_ "Material1", "Material2", "Material3"), "PersonalLib"</pre>

GetActiveDesign

Returns the design in the active project

Note: GetActiveDesign will return normally if there are no active objects.

UI Access	N/A
Parameters	None.
Return Value	Object of the active design.
Python Syntax	GetActiveDesign()
Python Example	<code>oDesign = oProject.GetActiveDesign()</code>

VB Syntax	GetActiveDesign
VB Example	<code>Set oDesign = oProject.GetActiveDesign</code>

GetArrayVariables

Returns a list of array variables. To get a list of indexed project variables, execute with oProject. To get a list of indexed local variables, use oDesign.

UI Access	N/A
Parameters	None.
Return Value	Array of strings containing names of variables.

Python Syntax	GetArrayVariables()
Python Example	<code>oProject.GetArrayVariables()</code> <code>oDesign.GetArrayVariables()</code>

VB Syntax	GetArrayVariables
VB Example	<pre>oProject.GetArrayVariables oDesign.GetArrayVariables</pre>

GetChildNames [Project]

Returns the names of the project's child objects.

UI Access	N/A		
Parameters	Name	Type	Description
	<type>	String	(Optional) Default is "design"; Any value returned by GetChildTypes() can be used.
Return Value	Array of names of children for the queried object.		

Python Syntax	GetChildNames (<type>)
Python Example	<pre>arrDesignNames = oProject.GetChildNames() arrVarbleNames = oProject.GetChildNames("Variable")</pre>

VB Syntax	GetChildNames (<type>)
VB Example	<pre>arrDesignNames = oProject.GetChildNames() arrVarbleNames = oProject.GetChildNames ("Variable")</pre>

GetChildObject [Project]

Returns the project's child objects.

Note:

GetChildObject will return normally if there are no active objects.

UI Access	N/A		
Parameters	Name	Type	Description
	<path>	String	The path may include multiple generations (for example, "designObject/moduleObj/SetupObject"). See Object Path .
Return Value	Object of a found child.		

Python Syntax	GetChildObject(<path>)
Python Example	<pre>oProject = oDeskop.GetActiveProject() oDesign = oProject.GetChildObject("TeeModel") oVariable = oProject.GetChildObject("VariableName") oReport = oProject.GetChildObject("TeeModel/Results/S Parameter Plot 1")</pre>

VB Syntax	GetChildObject(<path>)

VB Example	<pre> Set oProject = oDesktop.GetActiveproject() Set oDesign = oProject.GetChildObject("TeeModel") Set oVariable = oProject.GetChildObject("VariableName") Set oReport = oProject.GetChildObject("TeeModel/Results/S Parameter Plot 1") </pre>

GetChildTypes [Project]

Returns the types of the project's child objects.

UI Access	N/A
Parameters	None.
Return Value	Array of string represents types of the child objects.

Python Syntax	GetChildTypes()
Python Example	<code>oProject.GetChildTypes()</code>

VB Syntax	GetChildTypes
VB Example	<code>oProject.GetChildTypes</code>

GetConfigurableData (Project)

Note:

This command is for internal Ansys use only.

Python Syntax	GetConfigurableData()
Python Example	<code>oProject.GetConfigurableData()</code>

GetDefinitionManager

Gets the `DefinitionManager` object.

UI Access	N/A
Parameters	None.
Return Value	<code>DefinitionManager</code> object.

Python Syntax	GetDefinitionManager()
Python Example	<code>oDefinitionManager = oProject.GetDefinitionManager()</code>

VB Syntax	GetDefinitionManager

VB Example	<code>Set oDefinitionManager = oProject.GetDefinitionManager</code>

Note: For more information on commands for the DefinitionManager, see [Definition Manager Script Commands](#).

GetDependentFiles

Provides a list of the external files referenced in the project, including characteristic (for example, MDX) and coupled project files.

UI Access	N/A
Parameters	None.
Return Value	List of referenced files.

Python Syntax	<code>GetDependentFiles()</code>
Python Example	<code>files = oProject.GetDependentFiles()</code>

VB Syntax	<code>GetDependentFiles</code>
VB Example	<code>files = oProject.GetDependentFiles</code>

GetDesign

Returns the interface to a specific design in a given project.

UI Access	N/A

Parameters	Name	Type	Description
	<designName>	String	Name of the design.
Return Value	Object of the specified design.		

Python Syntax	GetDesign (<designName>)
Python Example	<code>oProject.GetDesign ("HFSSDesign1")</code>

VB Syntax	GetDesign <designName>
VB Example	<code>oProject.GetDesign "HFSSDesign1"</code>

GetDesigns

Obtains all designs in the current project.

UI Access	N/A
Parameters	None.
Return Value	List of objects for all designs in the project.

Python Syntax	GetDesigns()
Python Example	<code>oProject.GetDesigns()</code>

VB Syntax	GetDesigns
VB Example	<code>oProject.GetDesigns</code>

GetEDBHandle

Returns the EDB handle for the project.

Important:

This script is for internal Ansys use only.

UI Access	N/A
Parameters	None.
Return Value	String indicating the EDB handle for the project.

Python Syntax	GetEDBHandle()
Python Example	<code>oProject.GetEDBHandle()</code>

VB Syntax	GetEDBHandle

VB Example	<code>oProject.GetEDBHandle</code>

GetLegacyName

Obtains the legacy name of a project.

Note:

This command is for internal Ansys use only.

UI Access	N/A
Parameters	None.
Return Value	String containing the legacy project name.

Python Syntax	<code>GetLegacyName()</code>
Python Example	<code>oProject.GetLegacyName()</code>

VB Syntax	<code>GetLegacyName</code>
VB Example	<code>oProject.GetLegacyName</code>

GetName [Project]

Obtains the project name

UI Access	N/A
Parameters	None.
Return Value	String containing the project name, not including the path or extension.

Python Syntax	GetName()
Python Example	<code>oProject.GetName()</code>

VB Syntax	GetName
VB Example	<code>oProject.GetName</code>

GetObjPath [Project]

Obtains the project name from the full path.

UI Access	N/A
Parameters	None.
Return Value	String containing only the project name.

Python Syntax	GetObjPath()
Python Example	<code>oProject.GetObjPath()</code>

VB Syntax	GetObjPath
VB Example	<code>oProject.GetObjPath</code>

GetPath

Returns the location of the project on disk.

UI Access	N/A
Parameters	None.
Return Value	String containing the path to the project, not including the project name.

Python Syntax	GetPath()
Python Example	<code>oProject.GetPath()</code>

VB Syntax	GetPath
VB Example	<code>oProject.GetPath</code>

GetPropEvaluatedValue

Returns the Evaluated-Value for Value-Property and Variable. Returns the Property-value as text string for other property types

Note:

This command is not supported by the EMIT and Circuit design types.

UI Access	N/A		
Parameters	Name	Type	Description
	<PropName>	String	Name of the property.
Return Value	String value of the evaluated value.		

Python Syntax	GetPropEvaluatedValue (<PropName>)		
Python Example	<pre>oVar = oDesign.GetChildObject(" Variables/var") oVar.GetPropEvaluatedValue()</pre>		

GetPropNames [Project]

Obtains the property name of the object. At the project level, GetPropNames always returns empty because the project is not associated with any property.

UI Access	N/A		
Parameters	Name	Type	Description
	<includeReadOnly>	Boolean	(Optional) • True – Include read only props.

	<table><tr><td></td><td></td><td>• False – Do not include read only props.</td></tr></table>			• False – Do not include read only props.
		• False – Do not include read only props.		
Return Value	Empty array.			

Python Syntax	GetPropNames ()
Python Example	<code>oProject.GetPropNames ()</code>

VB Syntax	GetPropNames
VB Example	<code>oProject.GetPropNames</code>

GetPropSIValue

Returns the SI-Value for Value-Property and Variable. Return NAN for other property type if its value is not able to convert to be a double-floating point value.

Note:

This command is not supported by the EMIT and Circuit design types.

UI Access	N/A		
Parameters	Name	Type	Description
	<PropName>	String	Name of the property.
Return Value	Property value as a double floating value, or NAN if the property value cannot be converted to double floating		

	point.

Python Syntax	<code>GetPropSIValue (<PropName>)</code>
Python Example	<pre>oCreateBox = oDesign.GetChildObject("3D Modeler/Box1/CreateBox:1") oCreateBox.GetPropValue("xSize") return "length / 2" oCreateBox.GetPropEvaluatedValue("xSize") return '0.4mm' oCreateBox.GetPropSIValue("xSize") return 0.0004</pre>

GetPropValue [Project]

Returns the property value for the active project object, or specified property values.

UI Access	N/A		
Parameters	Name	Type	Description
	<propPath>	String	A child object's property path. See: Property Function Summary .
Return Value	String of property value.		

Python Syntax	<code>GetPropValue(<propPath>)</code>
Python Example	<code>oProject.GetPropValue("TeeModel/offset")</code>

VB Syntax	GetPropValue <propPath>
VB Example	oProject.GetPropValue "TeeModel/offset"

GetProperties

Gets a list of all the properties belonging to a specific <PropServer> and <PropTab>. This can be executed by the oProject, oDesign, or oEditor variables.

UI Access	N/A		
Parameters	Name	Type	Description
	<PropTab>	String	One of the following, where tab titles are shown in parentheses: • PassedParameterTab ("Parameter Values") • DefinitionParameterTab (Parameter Defaults") • LocalVariableTab ("Variables" or "Local Variables") • ProjectVariableTab ("Project variables") • ConstantsTab ("Constants") • BaseElementTab ("Symbol" or "Footprint") • ComponentTab ("General") • Component("Component") • CustomTab ("Intrinsic Variables") • Quantities ("Quantities") • Signals ("Signals")
	<PropServer>	String	An object identifier, generally returned from another script method, such as CompInst@R;2;3

Return Value	Array of strings containing the names of the appropriate properties.

Python Syntax	<code>GetProperties(<PropTab>, <PropServer>)</code>
Python Example	<code>oEditor.GetProperties('PassedParameterTab', 'k')</code>

VB Syntax	<code>GetProperties <PropTab>, <PropServer></code>
VB Example	<code>oEditor.GetProperties "PassedParameterTab", "k"</code>

GetPropertyValue

Returns the value of a single property belonging to a specific *<PropServer>* and *<PropTab>*. This function is available with the Project, Design or Editor objects, including definition editors.

Tip: Use the script recording feature and edit a property, and then view the resulting script to see the format for that property.

UI Access	N/A		
Parameters	Name	Type	Description
	<i><PropTab></i>	String	One of the following, where tab titles are shown in parentheses: - PassedParameterTab ("Parameter Values") - DefinitionParameterTab (Parameter Defaults") - LocalVariableTab ("Variables" or "Local Variables") - ProjectVariableTab ("Project variables") - ConstantsTab ("Constants") - BaseElementTab ("Symbol" or "Footprint")

			• ComponentTab ("General") • Component("Component") • CustomTab ("Intrinsic Variables") • Quantities ("Quantities") • Signals ("Signals")
	<i><PropServer></i>	String	An object identifier, generally returned from another script method, such as <code>CompInst@R;2;3</code>
	<i><PropName></i>	String	Name of the property.
Return Value	String value of the property.		

Python Syntax	<code>GetPropertyValue (<PropTab>, <PropServer>, <PropName>)</code>
Python Example	<pre>selectionArray = oEditor.GetSelections() for k in selectionArray: val = oEditor.GetPropertyValue("PassedParameterTab", k, "R") ...</pre>

VB Syntax	<code>GetPropertyValue (<PropTab>, <PropServer>, <PropName>)</code>
VB Example	<pre>selectionArray = oEditor.GetSelections for k in selectionArray: val = oEditor.GetPropertyValue("PassedParameterTab", k, "R")</pre>

	...

GetTopDesignList

Returns a list of top-level design names.

UI Access	N/A
Parameters	None.
Return Value	List of strings containing name of top-level designs.

Python Syntax	GetTopDesignList()
Python Example	<code>oProject.GetTopDesignList()</code>

VB Syntax	GetTopDesignList
VB Example	<code>oProject.GetTopDesignList</code>

GetValue

Gets the value of a single specified variable. To get the value of project variables, execute this command using `oProject`. To get the value of local variables, use `oDesign`.

UI Access	N/A

Parameters	Name	Type	Description
	<VarName>	String	Name of the variable to access.
Return Value	String represents the value of the variable.		

Python Syntax	GetVariableValue(<VarName>)
Python Example	<code>oProject.GetVariableValue("var_name")</code>

VB Syntax	GetVariableValue <VarName>
VB Example	<code>oProject.GetVariableValue "var_name"</code>

GetVariables

Returns a list of all defined variables. To get a list of project variables, execute this command using `oProject`. To get a list of local variables, use `oDesign`.

UI Access	N/A
Parameters	None.
Return Value	Array of strings containing the variables.

Python Syntax	GetVariables ()

Python Example	<pre>oProject.GetVariables() oDesign.GetVariables()</pre>

VB Syntax	GetVariables
VB Example	<pre>oProject.GetVariables oDesign.GetVariables</pre>

ImportDataset

Imports a dataset from a named file. This can be executed by the oProject, or oDesign variables. The name of the dataset is file-name+index number (e.g., dsdata1) unless the filename ends with a trailing number. When there is a trailing number at the end, we will remove the number and use first unused index. Alternatively, the name of the dataset can be explicitly defined by providing a string as an optional second argument.

UI Access	Project > Datasets > Import.		
Parameters	Name	Type	Description
	<datasetFilePath>	String	The full path to the file containing the dataset values. *.tab files recommended (see note below).
	<optionalDatasetName>	String	Optional. User-defined dataset name.
Return Value	None.		

Python Syntax	ImportDataset (<datasetFilePath>,<optionalDatasetName>)
Python Example	<pre>oProject.ImportDataset('e:\tmp\dsdata.tab') oDesign.ImportDataset('e:\tmp\dsdata.tab')</pre>

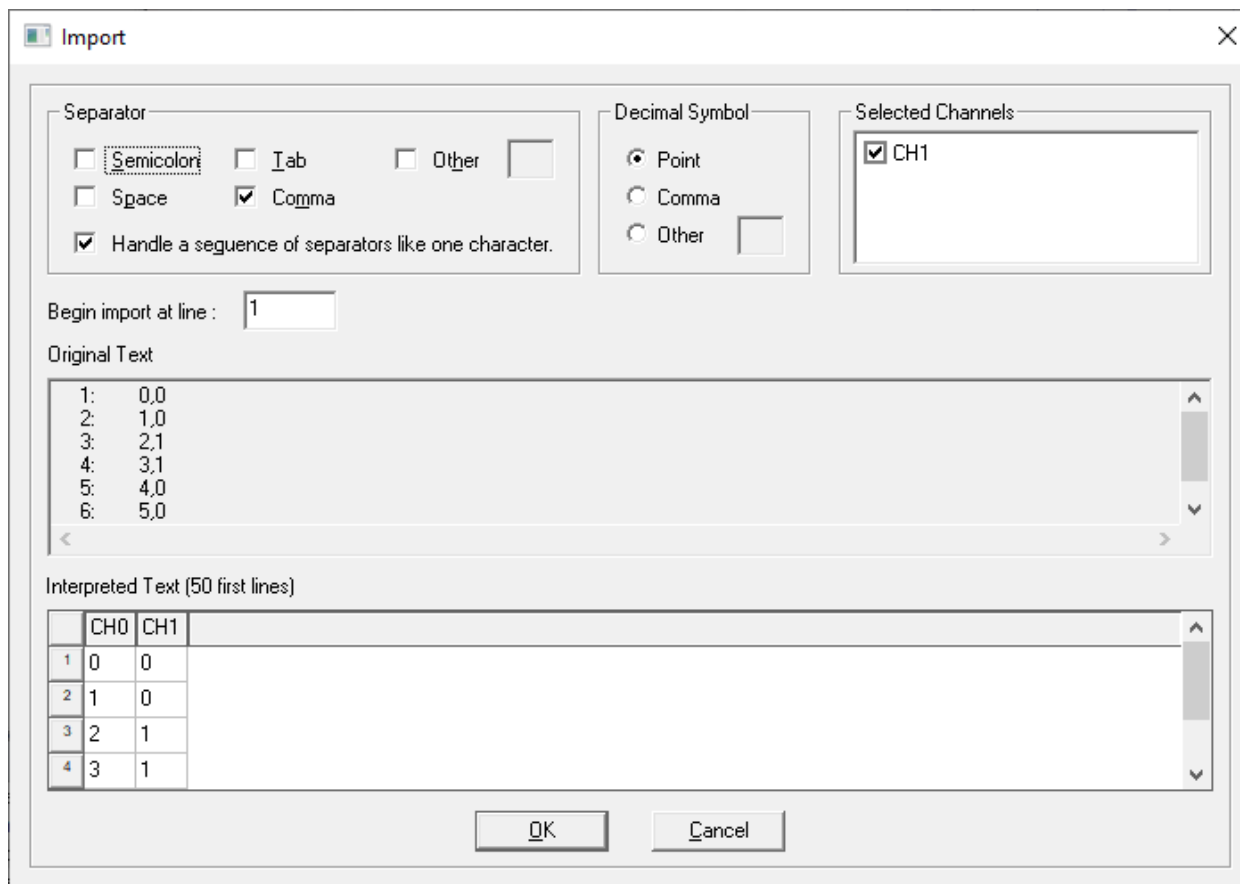
	<pre>oProject.ImportDataset('e:\tmp\dsdata.tab', 'MyDatasetName') oDesign.ImportDataset('e:\tmp\dsdata.tab', 'MyDatasetName')</pre>

VB Syntax	ImportDataset <datasetFilePath>, <optionalDatasetName>
VB Example	<pre>oProject.ImportDataset "e:\tmp\dsdata.tab" oDesign.ImportDataset "e:\tmp\dsdata.tab" oProject.ImportDataset "e:\tmp\dsdata.tab", "MyDatasetName" oDesign.ImportDataset "e:\tmp\dsdata.tab", "MyDatasetName"</pre>

Note About File Types:

Tab-delimited or space-delimited files with the extension *.tab are the recommended file type. When using ImportDataset at the Design level, *.tab is the only file type supported.

At the Project level, other file types are supported (for example, *.csv). However, after calling the command, you must configure the file import format manually through the Electronics Desktop GUI by selecting **Project > Datasets** and clicking **Import**.



InsertDesignWithWorkflow

Inserts a design with a named workflow and returns an IDispatch string.

UI Access	N/A		
Parameters	Name	Type	Description
	<type>	String	Type of design.
	<workflowName>	String	Name of the workflow.
	<specName>	String	Name of the spec.
	<fileName>	String	Name of the file.
	<libLoc>	String	Type of library, such as SysLib.
	<stationaryPath>	String	Path.
Return Value	IDispatch string, such as 'IDispatch(IAltraSimScript)'		

Python Syntax	InsertDesignWithWorkflow(<type>, <workflowName>, <specName>, <fileName>, <libLoc>, <stationaryPath>)
Python Example	<pre>oProject.InsertDesignWithWorkflow ("Circuit Design", "Serial Design", "PCIE3 Stressed", "LongChannel", "SysLib", "C:\\Program Files\\ANSYS Inc\\v251\\AnsysEM\\syslib\\MS" - RT_duroid 6010 (Er=10.2) 0.010 inch, 0.5 oz copper.asty")</pre>

VB Syntax	InsertDesignWithWorkflow <type>, <workflowName>, <specName>, <fileName>, <libLoc>, <stationaryPath>
VB Example	<pre>oProject.InsertDesignWithWorkflow "Circuit Design", "Serial Design", _ "PCIE3 Stressed", "LongChannel", "SysLib", _ "C:\\Program Files\\ANSYS Inc\\v251\\AnsysEM\\syslib\\MS - RT_duroid 6010" & _</pre>

	" (Er=10.2) 0.010 inch, 0.5 oz copper.asty"

InsertToolObject

Note:

This command is for internal Ansys use only.

Python Syntax	InsertToolObject()
Python Example	<code>oProject.InsertToolObject()</code>

Paste (Project Object)

Pastes a design in the active project.

UI Access	Edit > Paste.
Parameters	None.
Return Value	None

Python Syntax	Paste()
Python Example	<code>oProject.Paste()</code>

VB Syntax	Paste

VB Example	<code>oProject.Paste</code>

Redo [Project Level]

Reapplies the last project-level command.

UI Access	Edit > Redo.
Parameters	None.
Return Value	None.

Python Syntax	<code>Redo()</code>
Python Example	<code>oProject.Redo()</code>

VB Syntax	<code>Redo</code>
VB Example	<code>oProject.Redo</code>

RemoveAllUnusedDefinitions

Removes all unused project definitions.

UI Access	N/A
Parameters	None.
Return Value	None.

Python Syntax	<code>RemoveAllUnusedDefinitions()</code>
Python Example	<code>oProject.RemoveAllUnusedDefinitions()</code>

VB Syntax	<code>RemoveAllUnusedDefinitions</code>
VB Example	<code>oProject.RemoveAllUnusedDefinitions</code>

RemoveMaterial

Removes a material from a library.

UI Access	Remove Material(s) command in the material editor		
Parameters	Name	Type	Description
	<code><MaterialName></code>	String	Name of the material to be removed.
	<code><IsProjectMaterial></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><LibraryName></code>	String	Name of the user or personal library where the material resides.
	<code><LibraryLocation></code>	String	Location of library. Valid options: "UserLib" or "PersonalLib".
Return Value	None.		

Python Syntax	<code>RemoveMaterial (<MaterialName>, <IsProjectMaterial>, <LibraryName>, <LibraryLocation>)</code>

Python Example	<pre>oDefinitionManager.RemoveMaterial (["Material1", false, "mo0907", "UserLib"])</pre>

VB Syntax	<code>RemoveMaterial <MaterialName>, <IsProjectMaterial>, <LibraryName>, <LibraryLocation></code>
VB Example	<pre>oDefinitionManager.RemoveMaterial "Material1", false, "mo0907", "UserLib"</pre>

RemoveUnusedDefinitions

Removes any unused project definitions.

UI Access	Tools > Project Tools > Remove Unused Definitions.		
Parameters	Name	Type	Description
	<Definitions>	Array	Definitions to be removed, such as materials and surface materials.
Return Value	None.		

Python Syntax	<code>RemoveUnusedDefinitions(<Definitions>)</code>
Python Example	<pre>oProject.RemoveUnusedDefinitions([["Material1", false, "mo0907", "UserLib"]]])</pre>

	<pre>"NAME:Materials", "Al-Extruded"], ["NAME:SurfaceMaterials", "Steel-oxidised-surface"]])</pre>

VB Syntax	RemoveUnusedDefinitions <Definitions>
VB Example	<pre>oProject.RemoveUnusedDefinitions Array(Array("NAME:Materials", "Al-Extruded"), Array ("NAME:SurfaceMaterials", "Steel-oxidised-surface"))</pre>

Rename

Renames the project and saves it. Similar to [SaveAs\(\)](#).

UI Access	Edit > Rename.		
Parameters	Name	Type	Description
	<NewName>	String	Desired name of the project. The path is optional.
	<OverWriteOk>	Boolean	• True - overwrite the file on disk if it exists. • False - prevent overwrite.

Return Value	None.

Python Syntax	<code>Rename(<NewName>,<OverWriteOK>)</code>
Python Example	<code>oProject.Rename("c:\projects\MyProject.aedt", True)</code>

VB Syntax	<code>Rename <NewName>,<OverWriteOK></code>
VB Example	<code>oProject.Rename "c:\projects\MyProject.aedt", true</code>

Save

Saves the active project.

UI Access	File > Save.
Parameters	None.
Return Value	None.

Python Syntax	<code>Save()</code>
Python Example	<code>oProject.Save()</code>

VB Syntax	Save
VB Example	<code>oProject.Save</code>

SaveAs

Saves the project under a new name. Requires a full path.

Note:

This script takes two parameters for non-schematic/layout designs and four parameters for schematic/layout designs.

UI Access	File > Save As.		
Parameters	Name	Type	Description
	<NewName>	String	The desired name of the project, with directory and extension.
	<OverWriteOK>	Boolean	True to overwrite the file of the same name, if it exists. False to prevent overwrite.
	<DefaultAction>	String	For Schematic/Layout projects only. Otherwise omit. See note below. Valid actions: ef_overwrite , ef_copy_no_overwrite, ef_make_path_absolute, or empty string.
	<OverwriteActions>	Array	For Schematic/Layout projects only. Otherwise omit. See note below. Structured array: Array("Name: <Action>", <FileName>, <FileName>, ...) Valid actions: ef_overwrite , ef_copy_no_overwrite, ef_make_path_absolute, or empty string.
Return Value	None.		

Pytho-	For non-Schematic/Layout project: <code>SaveAs (<NewName> <OverWriteOK>)</code>

n Syn- tax	For Schematic/Layout project: SaveAs (<NewName> <OverWriteOK> <DefaultAction> <OverrideActions>)
Pytho- n Exam- ple	<pre>oProject.SaveAs('D:/projects/project1.aedt', True) ---- oProject.SaveAs('D:/Projects/Project1.aedt', True, 'ef_overwrite', ['NAME:Over- rideActions', ['NAME:ef_copy_no_overwrite', ['NAME:Files', '\$PROJECTDIR/circuit_models.inc']], ['NAME:ef_make_path_absolute', ['NAME:Files', '\$PROJECTDIR/SL_6s.sp']]])</pre>

VB Syn- tax	For non-Schematic/Layout project: SaveAs <NewName> <OverWriteOK> For Schematic/Layout project: SaveAs <NewName> <OverWriteOK> <DefaultAction> <OverrideActions>
VB Examp- les	<pre>oProject.SaveAs "D:/projects/project1.aedt", true ---- oProject.SaveAs "F:\Ansys Projects\TA33097\HighSpeedChannel.aedt", true, "ef_overwrite", Array ("NAME:OverrideActions", Array("NAME:ef_copy_no_overwrite", Array("NAME:Files", "\$PROJECTDIR/circuit_mod- els.inc")), Array("NAME:ef_make_path_absolute", Array("NAME:Files", "\$PROJECTDIR\SL_6s.sp")))</pre>

Important:

The DefaultAction and OverrideActions strings correspond to the following actions:

- **ef_overwrite** – Copy file to new project directory and overwrite.
- **ef_copy_no_overwrite** – Copy file to new project directory and don't overwrite.
- **ef_make_path_absolute** – Change reference to point to file in old project directory.
- **Empty String** – Do nothing.

The DefaultAction is applied to all files that are NOT explicitly listed in the OverrideActions array. Those in the OverrideActions array are separate arrays for actions that are different from the default action; those actions are applied to the files listed in the same array:

- If OverrideActions are not specified, DefaultAction is applied to ALL files in project directory.

SaveAsStandAloneProject

Saves the project as a standalone copy.

Note:

This script is not supported when the application is being controlled by Ansys Workbench.

UI Access	N/A		
Parameters	Name	Type	Description
	<projectName>	String	The desired name of the project, with directory and extension.
Return Value	None.		

Python Syntax	SaveAsStandAloneProject(<projectName>)
Python Example	<code>oProject.SaveAsStandAloneProject('D:/projects/project1.aedt')</code>

VB Syntax	SaveAsStandAloneProject <projectName>
VB Examples	<code>oProject.SaveAsStandAloneProject "D:/projects/project1.aedt"</code>

SaveProjectArchive

Saves the active project as an archive to the specified file path.

UI Access	File > Archive.		
Parameters	Name	Type	Description
	<archiveFilePath>	String	Path to archived file.
	<IncludeExternalFiles>	Boolean	True to include external files; False to exclude.
	<IncludeResultsFiles>	Boolean	True to include simulation files associated with the project; False to exclude.
	<AdditionalFiles>	Array	Additional specified files to include.
	<ArchiveNotes>	String	String describe the archive.
Return Value	None.		

Python Syntax	SaveProjectArchive(<archivefilepath>, <IncludeExternalFiles>, <IncludeResultsFiles>, <AdditionalFiles>, <ArchiveNotes>)
Python	<code>oProject.SaveProjectArchive("C:\\Users\\Documents\\Ansoft\\Project27.aedt", True,</code>

Example	<code>False, [], ""</code>

VB Syntax	<code>SaveProjectArchive <archivefilepath>, <IncludeExternalFiles>, <IncludeResultsFiles>, <AdditionalFiles>, <ArchiveNotes></code>
VB Example	<code>oProject.SaveProjectArchive "C:\Documents\OptimTee.aedt", true, false, Array(), "My notes"</code>

SetActiveDefinitionEditor

Obtains a specified definition editor.

UI Access	N/A		
Parameters	Name	Type	Description
	<code><EditorName></code>	String	Name of the definition editor to set active, one of "SymbolEditor", "FootprintEditor".
	<code><DefinitionName></code>	String	The combination name for the symbol or footprint, <code><libname>:<def-name></code>
Return Value	Object for the definition to be edited.		

Python Syntax	<code>SetActiveDefinitionEditor(<EditorName>, <DefinitionName>)</code>
Python Example	<code>oProject.SetActiveDefinitionEditor("SymbolEditor", "Simplorer Elements\Basic Elements\Circuit\Passive Elements:R")</code>

VB Syntax	SetActiveDefinitionEditor <EditorName>, <DefinitionName>
VB Example	<pre>oProject.SetActiveDefinitionEditor "SymbolEditor",_ "Simplorer Elements\Basic Elements\Circuit\Passive Elements:R"</pre>

SetActiveDesign

Sets a design to be the active design.

UI Access	N/A		
Parameters	Name	Type	Description
	<DesignName>	String	Name of the design to set as the active design.
Return Value	None.		

Python Syntax	SetActiveDesign (<DesignName>)
Python Example	<pre>oDesign = oProject.SetActiveDesign("SimplorerDesign2")</pre>

VB Syntax	SetActiveDesign <DesignName>
VB Example	<pre>Set oDesign = oProject.SetActiveDesign "SimplorerDesign2"</pre>

SetPropValue [Project]

Sets a property value for an active project's child object.

UI Access	Edit Properties on ProjectTree objects.		
Parameters	Name	Type	Description
	<propPath>	String	A child object's property path. See: Property Function .
	<newValue>	String	New property value.
Return Value	Boolean: • True – property found. • False – property not found.		

Python Syntax	SetPropValue(<propPath>, <newValue>)
Python Example	<pre>oProject.SetPropValue("TeeModel/offset", "2mm") oProject.SetPropValue("TeeModel/Results/S Parameter Plot 1/Display Type", "Data Table")</pre>

VB Syntax	SetPropValue <propPath>, <newValue>
VB Example	<pre>oProject.SetPropValue "TeeModel/offset", "2mm" oProject.SetPropValue "TeeModel/Results/S Parameter Plot 1/Display Type", "Data Table"</pre>

SetPropertyValue

Sets the value of a single property belonging to a specific PropServer and PropTab. This function is available with the Project, Design or Editor objects, including definition editors. This is not supported for properties of the following types: ButtonProp, PointProp, V3DPointProp, and VPointProp. Only the ChangeProperty command can be used to modify these properties.

Use the script recording feature and edit a property, and then view the resulting script entry or use `GetPropertyValue` for the desired property to see the expected format.

UI Access	N/A		
Parameters	Name	Type	Description
	<propTab>	String	One of the following, where tab titles are shown in parentheses: - PassedParameterTab ("Parameter Values") - DefinitionParameterTab (Parameter Defaults") - LocalVariableTab ("Variables" or "Local Variables") - ProjectVariableTab ("Project variables") - ConstantsTab ("Constants") - BaseElementTab ("Symbol" or "Footprint") - ComponentTab ("General") - Component("Component") - CustomTab ("Intrinsic Variables") - Quantities ("Quantities") - Signals ("Signals")
	<propServer>	String	An object identifier, generally returned from another script method, such as <code>CompInst@R;2;3</code>
	<propName>	String	Name of the property.
	<propValue>	String	The value for the property
Return Value	None.		

Python Syntax	SetPropertyValue(<propTab>, <propServer>, <propName>, <propValue>)
Python Example	oEditor.SetPropertyValue("PassedParameterTab", "k", "R", "2200")

VB Syntax	SetPropertyValue <propTab>, <propServer>, <propName>, <propValue>
VB Example	oEditor.SetPropertyValue "PassedParameterTab", "k", "R", "2200"

SetVariableValue

Sets the value of a variable. To set the value of a project variable, execute this command using oProject. To set the value of a local variable, use oDesign.

UI Access	N/A		
Parameters	Name	Type	Description
	<VarName>	String	Variable name.
	<VarValue>	Value	New value for the variable.
Return Value	None.		

Python Syntax	SetVariableValue (<VarName>, <VarValue>)
Python Example	oProject.SetVariableValue('\$Var1', '3mm')

VB Syntax	SetVariableValue <VarName>, <VarValue>

VB Example	<code>oProject.SetVariableValue "\$Var1", "3mm"</code>

SimulateAll

Simulates all solution setups and Optimetrics setups for all design instances in the project. Script processing only continues when all analyses are finished.

UI Access	N/A
Parameters	None.
Return Value	None.

Python Syntax	<code>SimulateAll()</code>
Python Example	<code>oProject.SimulateAll()</code>

VB Syntax	<code>SimulateAll</code>
VB Example	<code>oProject.SimulateAll</code>

Undo [Project]

Cancels the last project-level command.

UI Access	Edit > Undo.

Parameters	None.
Return Value	None.

Python Syntax	Undo()
Python Example	<code>oProject.Undo()</code>

VB Syntax	Undo
VB Example	<code>oProject.Undo</code>

UpdateDefinitions

Updates all definitions. The **Messages** window reports when definitions are updated, or warns when definitions cannot be found.

UI Access	Tools > Project Tools > Update Definitions. Click Select All , then Update .
Parameters	None.
Return Value	None.

Python Syntax	UpdateDefinitions()
Python Example	<code>oProject.UpdateDefinitions()</code>

VB Syntax	UpdateDefinitions
VB Example	<code>oProject.UpdateDefinitions</code>

7 - Design Object Script Commands

Design object commands should be executed by the oDesign object.

`oDesign.CommandName <args>`

For example:

Conventions Used in this Chapter

<ModuleName> is a placeholder for any one of the following modules:

- [Analysis Module](#) – "AnalysisSetup"
- [Boundary Module](#) – "BoundarySetup"
- [Field Overlays Module](#) – "FieldsReporter"
- Mesh Module – "MeshSetup"
- [Optimetrics Module](#) – "Optimetrics"
- Radiation Module – "RadField"
- Reduce Matrix Module – "ReduceMatrix"
- [Reporter Module](#) – "ReportSetup"
- Simulation Setup Module – "SimSetup"
- Solutions Module – "Solutions"

[ApplyMeshOps](#)

[ClearLinkedData](#)

[ConstructVariationString](#)

[CopyArray](#)

[DeleteFieldVariation](#)

[DeleteFullVariation](#)

[DeleteLinkedDataVariation](#)

EditDesignSettings

EditNotes

[ExportConvergence](#)

ExportLayoutViaCurrentDensity

ExportMatrixData

ExportMeshStats

[ExportProfile](#)

[ExportReport](#)

GenerateMesh

GetActiveEditor

[GetAllPorts](#)

GetBoundingBox

[GetChildNames \[Design\]](#)

[GetChildObject \[Design\]](#)

[GetChildTypes \[Design\]](#)

[GetConfigurableData](#)

[GetData](#)

[GetDesignType](#)

[GetManagedFilePath](#)

[GetModule](#)

[GetName](#)

[GetNominalVariation](#)

[GetNoteText](#)

[GetProject](#)

[GetPostProcessingVariables](#)

[GetPropNames \[Design\]](#)

[GetPropValue \[Design\]](#)

[GetSelections](#)

[PasteDesign](#)

[Redo](#)

[RenameDesignInstance](#)

[RenameSource](#)

[RevertAllToInitialCondition](#)

[RunToolkit](#)

[SetActiveEditor](#)

[SetPropValue \[Design\]](#)

[SetPropertyValue](#)

[SetShowLayoutForLayoutComponent](#)

[Solve](#)

[StartAnalysis](#)

[StopSimLink](#)

[Undo](#)

ValidateCircuit

[ValidateDesign](#)

[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([</pre>

```
"NAME:$ds1",  
  [  
    "NAME:Coordinates",  
    [  
      "NAME:Coordinate",  
      "X:=", 2,  
      "Y:=", 4  
    ],  
    [  
      "NAME:Coordinate",  
      "X:=", 6,  
      "Y:=", 8  
    ]  
  ]  
]  
)  
oDesign.AddDataset(  
  [  
    "NAME:$ds1",  
    [  
      "NAME:Coordinates",
```

	<pre>["NAME:Coordinate", "X:=", 2, "Y:=", 4], ["NAME:Coordinate", "X:=", 6, "Y:=", 8]])</pre>

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),</pre>

	<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)))</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>

AnalyzeAll [design]

Runs all solution setups and Optimetrics setups for the current design instance.

UI Access	N/A		
Parameters	Name	Type	Description

	<table><tr><td>isBlocking</td><td>Boolean</td><td>An optional arg that defaults to <code>true</code>. If it's <code>false</code>, the command immediately returns while the analysis or analyses run in the background. The status of the analyses can be checked with the <code>AreThereSimulationsRunning</code> command.</td></tr></table>	isBlocking	Boolean	An optional arg that defaults to <code>true</code> . If it's <code>false</code> , the command immediately returns while the analysis or analyses run in the background. The status of the analyses can be checked with the <code>AreThereSimulationsRunning</code> command.
isBlocking	Boolean	An optional arg that defaults to <code>true</code> . If it's <code>false</code> , the command immediately returns while the analysis or analyses run in the background. The status of the analyses can be checked with the <code>AreThereSimulationsRunning</code> command.		
Return Value	None			
Python Syntax	AnalyzeAll()			
Python Example	<code>oDesign.AnalyzeAll(isBlocking)</code>			

VB Syntax	AnalyzeAll
VB Example	<code>oDesign.AnalyzeAll</code>

AnalyzeAllNominal

Runs all defined setups.

UI Access	Right-click Analysis in the project tree, and select Analyze All .		
Parameters	Name	Type	Description
	isBlocking	Boolean	An optional arg that defaults to <code>true</code> . If it's <code>false</code> , the command immediately returns while the analysis or analyses run in the background. The status of the analyses can be checked with the <code>AreThereSimulationsRunning</code> command.
Return Value	None		

Python Syntax	AnalyzeAllNominal()
Python Example	<code>oDesign.AnalyzeAllNominal()</code>

VB Syntax	AnalyzeAllNominal
VB Example	<code>oDesign.AnalyzeAllNominal</code>

ClearLinkedData (Design)

Clear the linked data of all solution setups in the design. (This command is similar to the ClearLinkedData command for the module level, which clears link data for one or more specific solution setups.)

UI Access	Mechanical > Analysis > Clear Linked Data (with no solution setup selected in the Project Manager)
Parameters	None.
Return Value	None.

Python Syntax	ClearLinkedData()
Python Example	<code>oDesign.ClearLinkedData()</code>

VB Syntax	ClearLinkedData
VB Example	<code>oDesign.ClearLinkedData</code>

ConstructVariationString

Lists and orders the variables and values associated with a design variation.

UI Access	N/A		
Parameters	Name	Type	Description
	<ArrayOfVariableNames>	Array of Strings	List of variable names.
	<ArrayOfVariableValuesIncludingUnits>	Array of Strings	List of variable values, including units, in the same order as the list of names.
Return Value	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 ConstructVariationString can be used, see the Python example.		

Python Syntax	ConstructVariationString(<ArrayOfVariableNames>, <ArrayOfVariableValuesIncludingUnits>)
Python Example	<pre>varStr = oDesign.ConstructVariationString(["xx", "yy"], ["2mm", "1mm"]) oDesign.ExportProfile("Setup1", varStr, "C:\profile.prof")</pre>

VB Syntax	ConstructVariationString <ArrayOfVariableNames>, <ArrayOfVariableValuesIncludingUnits>
VB Example	<pre>oDesign.ConstructVariationString Array("x_size", "y_size"), Array("2mm", "1mm")</pre>

CopyArray

Copies a specified array.

UI Access	Edit > Copy.		
Parameters	Name	Type	Description
	<ArrayName>	String	Name of the array to copy.
Return Value	None.		

Python Syntax	Copy (<ArrayName>)		
Python Example	oModule = oDesign.GetModule("ModelSetup")		
	oModule.CopyArray("Array")		
	oModule.PasteArray() # returns the new array name		

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")

DeleteDataset

Deletes a specified dataset. This can be executed by the oProject, or oDesign variables.

UI Access	Project > Datasets > Remove.		
Parameters	Name	Type	Description
	<DatasetName>	String	Name of the dataset found in the project.
Return Value	None.		

Python Syntax	DeleteDataset (<DatasetName>)		
Python Example	oProject.DeleteDataset ('\$ds1')		
	oDesign.DeleteDataset ('\$ds1')		

VB Syntax	DeleteDataset <DatasetName>		
VB Example	oProject.DeleteDataset "\$ds1"		
	oDesign.DeleteDataset "\$ds1"		

DeleteFieldVariation

Deletes field variations, fields and meshes, or fields and meshes for specified variations.

UI Access	[Solver] > Results > Clean Up Solutions > [Fields Only / Fields and Meshes].		
Parameters	Name	Type	Description
	<variationKeys>	Array	Array containing either "All" string to select all variations, or strings of variations to include.
	<deleteMesh>	Boolean	When true, deletes mesh data.

	<code><deleteLinkedData></code>	Boolean	When true, deletes linked data.
Return Value	None.		

Python Syntax	<code>DeleteFieldVariation(<variationKeys>, <deleteMesh>, <deleteLinkedData></code>
Python Example	<pre>oProject = oDesktop.GetActiveProject() oDesign = oProject.GetActiveDesign() Design.DeleteFieldVariation(['All'], True, False)</pre>

VB Syntax	<code>DeleteFieldVariation <variationKeys>, <deleteMesh>, <deleteLinkedData></code>
VB Example	<pre>Set oProject = oDesktop.SetActiveProject "OptimTee" Set oDesign = oProject.SetActiveDesign "TeeModel" oDesign.DeleteFieldVariation Array("offset=" & Chr(39) & "0.0947046688081817in" & Chr(39) & ""), true, false</pre>

DeleteFullVariation

Deletes either all solution data, or selected variation data.

UI Access	[Mechanical] > Results > Clean Up Solutions.		
Parameters	Name	Type	Description
	<code><variationKeys></code>	Array	Array containing either "All" string to select all variations, or strings of variations to include.
	<code><deleteLinkedData></code>	Boolean	When true, deletes linked data as well.
Return Value	None.		

Python Syntax	DeleteFullVariation(<variationKeys>, <deleteLinkedData>)
Python Example	<code>oDesign.DeleteFullVariation(['All'], false)</code>

VB Syntax	DeleteFullVariation <variationKeys>, <deleteLinkedData>
VB Example	<code>oDesign.DeleteFullVariation Array('All'), false</code>

DeleteLinkedDataVariation

Deletes the linked data of specified variations.

UI Access	[Mechanical] > Results > Clean Up Solutions.		
Parameters	Name	Type	Description
	<DesignVariationKeys>	Array	Array containing strings of the variations keys whose linked data are going to be deleted.
Return Value	None.		

Python Syntax	DeleteLinkedDataVariation(<DesignVariationKeys>)
Python Example	<code>oDesign.DeleteLinkedDataVariation(["current=\'0.9mA\'", "current=\'1.0mA\'"])</code>

VB Syntax	DeleteLinkedDataVariation <DesignVariationKeys>

VB Example	<code>oDesign.DeleteLinkedDataVariation Array("current=\'0.9mA\'", "current=\'1.0mA\'")</code>

DeleteOutputVariable

Deletes an existing output variable. The variable can only be deleted if it is not in use by any traces.

UI Access	Mechanical > Results > Output Variables. In the Output Variables window, click Delete .		
Parameters	Name	Type	Description
	<OutputVarName>	String	Name of the output variable.
Return Value	None.		

Python Syntax	<code>DeleteOutputVariable (<OutputVarName>)</code>
Python Example	<pre>oModule = oDesign.GetModule("OutputVariable") oModule.DeleteOutputVariable ("testNew")</pre>

VB Syntax	<code>DeleteOutputVariable <OutputVarName></code>
VB Example	<pre>Set oModule = oDesign.GetModule("OutputVariable") oModule.DeleteOutputVariable "testNew"</pre>

DeletePlotEntities

Delete plot entities including limit line, marker, X-marker, Y-marker and note.

UI Access	[Edit] > Delete.

Parameters	Name	Type	Description
	<plotname>	string	Name of the plot containing entities to delete.
	< "NAME:PlotEntityNames" ,"<PlotEntityName>"	string	limit line, marker, X-marker, Y-marker and not
Return Value	None.		

Python Syntax	DeletePlotEntities(<plotName>, ["NAME:PlotEntityNames", "<PlotEntityName>"])]
Python Example	<pre>oProject = oDesktop.SetActiveProject("Tee-23R2-01") oDesign = oProject.SetActiveDesign("TeeModel") oModule = oDesign.GetModule("ReportSetup") oModule.DeletePlotEntities("Z Parameter Plot 1-01", [["NAME:PlotEntityNames", "LimitLine1"]])</pre>

DeleteSolutionVariation

Deletes all solution data for specific solutions and design variations. This is obsolete and is supported only for backward compatibility. You should use DeleteFullVariation.

UI Access	Right-click on Results , select Browse Solutions... , click Delete button in the dialog.		
Parameters	Name	Type	Description
	<SoluParams>	Array	Structured array. Array(<DataSpecifierArray>, ...)
	<DataSpecifierArray>	Array	Structured array. Array(<DesignVariationKey>, <SetupName>, <SolutionName>)
Return Value	None.		

Python Syntax	DeleteSolutionVariation(<SoluParams>)
Python Example	<pre>oModule.DeleteSolutionVariation([["width='2in'", "Setup1", "Adaptive_1"], ["width='2in'", "Setup1", "Sweep1"]])</pre>

VB Syntax	DeleteSolutionVariation <SoluParams>
VB Example	<pre>oModule.DeleteSolutionVariation Array(_</pre>

	<pre>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.

UI Access	Mechanical > Results > Output Variables. In the Output Variables window, click Delete .		
Parameters	Name	Type	Description
	<OutputVarName>	String	Name of the output variable.
Return Value	None.		

Python Syntax	DeleteOutputVariable (<OutputVarName>)
Python Example	<pre>oModule = oDesign.GetModule("OutputVariable") oModule.DeleteOutputVariable ("testNew")</pre>

VB Syntax	DeleteOutputVariable <OutputVarName>
VB Example	<pre>Set oModule = oDesign.GetModule("OutputVariable") oModule.DeleteOutputVariable "testNew"</pre>

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")
```

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>

EditLayoutForLayoutComponent

Opens the definition component for editing in HFSS 3D Layout for a Layout Component in an HFSS 3D design.

UI Access	Edit Layout...		
Parameters	Name	Type	Description
	<ComponentDefinitionName>	String	Name of the Layout Component Definition.
Return Value	None.		

Python Syntax	EditLayoutForLayoutComponent ([Name:LayoutComponent EDB", Definitions:=", [<ComponentDefinitionName>])
Python Example	<pre>oDesign.EditLayoutForLayoutComponent (["NAME:Layout Component EDB", "Definitions:=" , ["Diff_Via_diffViaNominal"]])</pre>

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.
Return Value	None.		

Python Syntax	
Python Example	

VB Syntax	
VB Example	

ExportArray

Exports an array definition to a csv file.

UI Access	Export Array to CSV File

Parameters	<filepath>.
Return Value	None.

Python Syntax	ExportArray()
Python Example	<pre>oModule = oDesign.GetModule("ModelSetup") oModule.ExportArray("Array", "C:\\Users\\MyFolder\\Export Array3.csv")</pre>

VB Syntax	ImportArray
VB Example	<pre>oModule = oDesign.GetModule("ModelSetup") oModule.ExportArray("Array", "C:\\Users\\MyFolder\\Export Array3.csv")</pre>

ExportDataset

Exports a dataset to a named file. This can be executed by the oProject, or oDesign variables.

UI Access	Project > Datasets > Export.		
Parameters	Name	Type	Description
	<datasetFilePath>	String	The full path to the file.
Return Value	None.		

Python Syntax	<code>ExportDataset (<datasetFilePath>)</code>
Python Example	<pre>oProject.ExportDataset('e:/tmp/dsdata.txt') oDesign.ExportDataset('e:/tmp/dsdata.txt')</pre>

VB Syntax	<code>ExportDataset <datasetFilePath></code>
VB Example	<pre>oProject.ExportDataset "e:/tmp/dsdata.txt" oDesign.ExportDataset "e:/tmp/dsdata.txt"</pre>

ExportProfile

Exports a solution profile to file.

UI Access	N/A		
Parameters	Name	Type	Description
	<SetupName>	String	Text of the design's notes.
	<VariationString>	String	Variation name. Pass empty string for the current nominal variation.
	<filePath>	String	Full file path, including extension *.prof.
Return Value	None.		

Python Syntax	
Python Example	

VB Syntax	
VB Example	

GetChildNames [Design]

Returns the names of the design's child objects.

UI Access	N/A		
Parameters	Name	Type	Description
	<type>	String	Optional. Valid options are "Module", "Editor", "Variable". Default returns Module names.
Return Value	Names of children for the queried object).		
Python Syntax	GetChildNames (<type>)		
Python Example	oDesign.GetChildNames ()		

VB Syntax	GetChildNames <type>
VB Example	oDesign.GetChildNames

GetChildObject [Design]

Returns the design's child objects.

UI Access	N/A		
Parameters	Name	Type	Description
	<path>	String	The path may include multiple generations (for example, "designOb-

	ject/moduleObj/SetupObject"). See: Object Path .
Return Value	A child object if one is found. Otherwise script error.

Python Syntax	GetChildObject(<path>)
Python Example	<pre>oDesign = oProject.GetActiveDesign() oOptimModule = oDesign.GetChildObject("Optimetrics") oEditor = oDesign.GetChildObject("3D Model") oBox = oDesign.GetChildObject("3D Model/Box1") oRpt = oDesign.GetChildObject("Results/S Parameter Plot 1") oVariable= oDesign.GetChildObject("Offset")</pre>

VB Syntax	GetChildObject <path>
VB Example	<pre>oDesign = oProject.GetActiveDesign oOptimModule = oDesign.GetChildObject "Optimetrics" oEditor = oDesign.GetChildObject "3D Model" oBox = oDesign.GetChildObject "3D Model/Box1" oRpt = oDesign.GetChildObject "Results/S Parameter Plot 1" oVariable= oDesign.GetChildObject "Offset"</pre>

GetChildTypes [Design]

Returns the design's child object types.

UI Access	N/A
Parameters	None.
Return Value	String: "Module", "Editor", or "Variable"

Python Syntax	GetChildTypes()
Python Example	<code>oDesign.GetChildTypes()</code>

VB Syntax	GetChildTypes
VB Example	<code>oDesign.GetChildTypes</code>

GetConfigurableData

Obtains configurable data of a specified type

Note:

This command is for internal Ansys use only.

UI Access	N/A

Parameters	Name	Type	Description
	<type>	String	Specified type.
Return Value	None.		

Python Syntax	GetConfigurableData(<type>)
Python Example	<code>oDesign.GetConfigurableData ("model")</code>

VB Syntax	GetConfigurableData <type>
VB Example	<code>oDesign.GetConfigurableData "model"</code>

GetData

Note:

This command is for internal Ansys use only.

Python Syntax	GetData()
Python Example	<code>oDesign.GetData()</code>

GetDesignID

Returns the unique identification number of the active design.

UI Access	N/A
Parameters	None.
Return Value	String indicating the unique identification number of the active design.

Python Syntax	GetDesignID()
Python Example	<pre>oDesign = oProject.GetActiveDesign() oDesign.GetDesignID()</pre>

VB Syntax	GetDesignType
VB Example	<pre>Set oDesign = oProject.GetActiveDesign oDesign.GetDesignID</pre>

GetDesignType

Returns the design type of the active design.

UI Access	N/A
Parameters	None.
Return Value	String indicating the design type of the active design ("Circuit Design", "Circuit Netlist", "EMIT", "HFSS 3D Layout Design", "HFSS", "HFSS-IE", "Icepak", "Maxwell 2D", "Maxwell 3D", "Q2D Extractor", "Q3D Extractor", "RMxpvt", or "Twin Builder").

Python Syntax	GetDesignType()
Python Example	<pre>oDesign = oProject.GetActiveDesign() oDesign.GetDesignType()</pre>

VB Syntax	GetDesignType
VB Example	<pre>Set oDesign = oProject.GetActiveDesign oDesign.GetDesignType</pre>

GetEdgePositionAtNormalizedParameter

Gets the position on an edge with normalized parameter.

UI Access	N/A		
Parameters	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.
Return Value	Array of string containing the x, y and z coordinate values.		

Python Syntax	GetEdgePositionAtNormalizedParameter(<EdgeID>, <NormParam>)
Python Example	<pre>oEditor.GetEdgePositionAtNormalizedParameter(5, 0) oEditor.GetEdgePositionAtNormalizedParameter(5, 1) oEditor.GetEdgePositionAtNormalizedParameter(5, 0.5)</pre>

VB Syntax	<code>GetEdgePositionAtNormalizedParameter <EdgeID>, <NormParam></code>
VB Example	<code>oEditor.GetEdgePositionAtNormalizedParameter 5, 0</code> <code>oEditor.GetEdgePositionAtNormalizedParameter 5, 1</code> <code>oEditor.GetEdgePositionAtNormalizedParameter 5, 0.5</code>

GetManagedFilePath

Get the path to the project's results folder.

UI Access	N/A
Parameters	None.
Return Value	String containing path where project results are located.

Python Syntax	<code>oDesign.GetManagedFilePath()</code>
Python Example	<code>oDesign.GetManagedFilePath()</code>

VB Syntax	<code>oDesign.GetManagedFilePath</code>
VB Example	<code>oDesign.GetManagedFilePath</code>

GetModule

Returns the IDispatch for the specified module.

UI Access	N/A		
Parameters	Name	Type	Description
	<modulename>	String	One of the following: • Analysis Module – "AnalysisSetup" • Boundary Module – "BoundarySetup" • Field Overlays Module – "FieldsReporter" • Mesh Module – "MeshSetup" • Optimetrics Module – "Optimetrics" • Radiation Module – "RadField" • Reduce Matrix Module – "ReduceMatrix" • Reporter Module – "ReportSetup" • Simulation Setup Module – "SimSetup" • Solutions Module – "Solutions"
Return Value	Module IDispatch		

Python Syntax	GetModule (<modulename>)
Python Example	oModule = oDesign.GetModule("SimSetup")

VB Syntax	GetModule <modulename>

VB Example

```
set oModule = oDesign.GetModule "SimSetup"
```

GetModule (RadiationSetupMgr)

Returns the IDispatch for the specified module.

UI Access	N/A		
Parameters	Name	Type	Description
	<modulename>	String	One of the following: • Analysis Module – "AnalysisSetup" • Boundary Module – "BoundarySetup" • Field Overlays Module – "FieldsReporter" • Mesh Module – "MeshSetup" • Optimetrics Module – "Optimetrics" • Radiation Module – "RadField" • Reduce Matrix Module – "ReduceMatrix" • Reporter Module – "ReportSetup" • Simulation Setup Module – "SimSetup" • Solutions Module – "Solutions"
Return Value	Module IDispatch		

Python Syntax

```
GetModule (<modulename>)
```

Python Example	<pre>oModule = oDesign.GetModule("SimSetup")</pre>

VB Syntax	<code>GetModule <modulename></code>
VB Example	<pre>set oModule = oDesign.GetModule("SimSetup")</pre>

GetName

Returns the design name of the active design, in that order separated by a semicolon.

UI Access	N/A
Parameters	None.
Return Value	String indicating the name of the active design.

Python Syntax	<code>GetName()</code>
Python Example	<pre>design_name = oDesign.GetName()</pre>

VB Syntax	<code>GetName</code>
VB Example	<pre>design_name = oDesign.GetName</pre>

GetNominalVariation

Returns the current nominal variation.

UI Access	N/A
Parameters	None.
Return Value	String containing current nominal variation.

Python Syntax	GetNominalVariation()
Python Example	<code>oDesign.GetNominalVariation()</code>

VB Syntax	GetNominalVariation
VB Example	<code>oDesign.GetNominalVariation</code>

GetNoteText

Returns the text of the note attached to a design.

UI Access	N/A
Parameters	None.
Return Value	String: text of the design note.

Python Syntax	GetNoteText()
Python Example	<code>oDesign.GetNoteText()</code>

VB Syntax	GetNoteText
VB Example	<code>oDesign.GetNoteText</code>

GetObjPath [Design]

Obtains the path to the design.

UI Access	N/A
Parameters	None.
Return Value	String containing the path to the design.

Python Syntax	GetObjPath()
Python Example	<code>oDesign.GetObjPath()</code>

VB Syntax	GetObjPath
VB Example	<code>oDesign.GetObjPath</code>

GetOutputVariableValue

Returns the double value of an output variable. Only expressions that return a double value are supported. The expression is evaluated only for a single point.

UI Access	N/A		
Parameters	Name	Type	Description
	<OutputVarName>	String	Name of the output variable.
	<IntrinsicVariation>	String	A set of intrinsic variable value pairs to use when evaluating the output expression.
	<SolutionName>	String	Name of the solution as listed in the output variable UI. For example, "Setup1 : Last Adaptive".
	<ReportType>	String	The name of the report type as seen in the output variable UI.
	<ContextArray>	Array	Structured array containing context for which the output variable expression is being evaluated. Can be empty. Array("Context:=", <string>)
Return Value	Double value of the output variable.		

Python Syntax	GetOutputVariableValue(<OutputVarName>, <IntrinsicVariation>, <SolutionName>, <ReportTypeName>, <ContextArray>)
Python Example	Val = oDesign.GetOutputVariableValue("test", "Freq = '20Ghz' Theta='20deg' Phi='30deg'", "TR", "Standard", [])

VB Syntax	GetOutputVariableValue <OutputVarName>, <IntrinsicVariation>, <SolutionName>, <ReportTypeName>, <Con-

	<i>textArray</i> >
VB Example	<pre>Val = oDesign.GetOutputVariableValue "test", "Freq = '20Ghz' Theta='20deg' Phi='30deg'", "TR", "Standard", Array()</pre>

GetOutputVariables

Returns the list of output variables.

UI Access	N/A
Parameters	None.
Return Value	Array containing all output variables.

Python Syntax	GetOutputVariables()
Python Example	<code>oDesign.GetOutputVariables()</code>

VB Syntax	GetOutputVariables
VB Example	<code>oDesign.GetOutputVariables</code>

GetPostProcessingVariables

Returns the list of post-processing variables.

UI Access	N/A

Parameters	None.
Return Value	Returns array containing variables.

Python Syntax	GetPostProcessingVariables()
Python Example	<code>oDesign.GetPostProcessingVariables()</code>

VB Syntax	GetPostProcessingVariables
VB Example	<code>oDesign.GetPostProcessingVariables</code>

GetProperties

Gets a list of all the properties belonging to a specific <PropServer> and <PropTab>. This can be executed by the oProject, oDesign, or oEditor variables.

UI Access	N/A		
Parameters	Name	Type	Description
	<PropTab>	String	One of the following, where tab titles are shown in parentheses: - PassedParameterTab ("Parameter Values") - DefinitionParameterTab (Parameter Defaults") - LocalVariableTab ("Variables" or "Local Variables") - ProjectVariableTab ("Project variables") - ConstantsTab ("Constants")

			• BaseElementTab ("Symbol" or "Footprint") • ComponentTab ("General") • Component("Component") • CustomTab ("Intrinsic Variables") • Quantities ("Quantities") • Signals ("Signals")
	<PropServer>	String	An object identifier, generally returned from another script method, such as Complnst@R;2;3
Return Value	Array of strings containing the names of the appropriate properties.		

Python Syntax	GetProperties(<PropTab>, <PropServer>)
Python Example	oEditor.GetProperties('PassedParameterTab', 'k')

VB Syntax	GetProperties <PropTab>, <PropServer>
VB Example	oEditor.GetProperties "PassedParameterTab", "k"

GetPropertyValue

Returns the value of a single property belonging to a specific <PropServer> and <PropTab>. This function is available with the Project, Design or Editor objects, including definition editors.

Tip: Use the script recording feature and edit a property, and then view the resulting script to see the format for that property.

UI Access	N/A		
Parameters	Name	Type	Description
	<PropTab>	String	One of the following, where tab titles are shown in parentheses: - PassedParameterTab ("Parameter Values") - DefinitionParameterTab (Parameter Defaults") - LocalVariableTab ("Variables" or "Local Variables") - ProjectVariableTab ("Project variables") - ConstantsTab ("Constants") - BaseElementTab ("Symbol" or "Footprint") - ComponentTab ("General") - Component("Component") - CustomTab ("Intrinsic Variables") - Quantities ("Quantities") - Signals ("Signals")
	<PropServer>	String	An object identifier, generally returned from another script method, such as <code>CompInst@R;2;3</code>
	<PropName>	String	Name of the property.
Return Value	String value of the property.		

Python Syntax	GetPropertyValue (<PropTab>, <PropServer>, <PropName>)
Python Example	<pre>selectionArray = oEditor.GetSelections() for k in selectionArray:</pre>

	<pre>val = oEditor.GetPropertyValue("PassedParameterTab", k, "R") ...</pre>

VB Syntax	<code>GetPropertyValue (<PropTab>, <PropServer>, <PropName>)</code>
VB Example	<pre>selectionArray = oEditor.GetSelections for k in selectionArray: val = oEditor.GetPropertyValue("PassedParameterTab", k, "R") ...</pre>

GetPropNames [Design]

Returns array containing names of property. For designs, always returns an empty array because the design has no property.

UI Access	N/A		
Parameters	Name	Type	Description
	<code><includeReadOnly></code>	Boolean	(Optional). • True - include read only property. • False - do not include read only property.
Return Value	Empty array.		
Python Syntax	<code>GetPropNames(<includeReadOnly>)</code>		
Python Example	<code>oDesign.GetPropNames()</code>		

VB Syntax	GetPropNames <includeReadOnly>
VB Example	<code>oDesign.GetPropNames</code>

GetPropValue [Design]

Returns the property value for the active design object, or specified property values.

UI Access	N/A		
Parameters	Name	Type	Description
	<propPath>	String	A child object's property path. See: Object Property Script Function Summary .
Return Value	The property value (integer, string, or array of strings) of the specified child object.		

Python Syntax	GetPropValue(<propPath>)
Python Example	<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>

VB Syntax	GetPropValue(<propPath>)
VB Example	<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.

UI Access	N/A
Parameters	None.
Return Value	String containing the solution type. Possible values are: "SBR+", "HFSS [Modal Terminal] [Network Composite]", "Transient [Network Composite]", "Eigenmode", or "Characteristic".

Python Syntax	GetSolutionType()
Python Example	<code>oDesign.GetSolutionType()</code>

VB Syntax	GetSolutionType
VB Example	<code>oDesign.GetSolutionType</code>

GetSolveInsideThreshold

Returns the solve inside threshold. This command does not apply to HFSS-IE.

UI Access	N/A

Parameters	None.
Return Value	Double representing the solve inside threshold.

Python Syntax	<code>GetSolveInsideThreshold()</code>
Python Example	<code>oDesign.GetSolveInsideThreshold()</code>

VB Syntax	<code>GetSolveInsideThreshold</code>
VB Example	<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`.

UI Access	N/A
Parameters	None.
Return Value	Array of strings containing the variables.

Python Syntax	<code>GetVariables()</code>
Python Example	<code>oProject.GetVariables()</code>

	<code>oDesign.GetVariables()</code>

VB Syntax	<code>GetVariables</code>
VB Example	<code>oProject.GetVariables</code> <code>oDesign.GetVariables</code>

GetVariableValue

Gets the value of a single specified variable. To get the value of project variables, execute this command using `oProject`. To get the value of local variables, use `oDesign`.

UI Access	N/A		
Parameters	Name	Type	Description
	<code><VarName></code>	String	Name of the variable to access.
Return Value	String represents the value of the variable.		

Python Syntax	<code>GetVariableValue(<VarName>)</code>
Python Example	<code>oProject.GetVariableValue("var_name")</code>

VB Syntax	<code>GetVariableValue <VarName></code>
VB Example	<code>oProject.GetVariableValue "var_name"</code>

GetVariationVariableValue

Returns the value for a specified variation's variable.

UI Access	N/A		
Parameters	Name	Type	Description
	<VariationString>	String	The name of the design variation.
	<VariableName>	String	The name of the variable.
Return Value	Returns a double precision value in SI units, interpreted to mean the value of the variable contained in the variation string.		

Python Syntax	GetVariationVariableValue(<VariationString>, <VariableName>)
Python Example	<pre>oDesign.GetVariationVariableValue('x_size = 2mm y_size = 1mm', 'y_size')</pre>

VB Syntax	GetVariationVariableValue <VariationString>, <VariableName>
VB Example	<pre>varval = oDesign.GetVariationVariableValue "x_size = 2mm y_size = 1mm", "y_size"</pre>

ImportArray

Imports an array definition using a csv file.

UI Access	Create Array Through CSV
Parameters	<filepath>.
Return Value	None.

Python Syntax	ImportArray()
Python Example	<pre>oModule = oDesign.GetModule("ModelSetup") oModule.ImportArray("Array", "C:\\Users\\MyFolder\\Export Array3.csv")</pre>

VB Syntax	ImportArray
VB Example	<pre>oModule = oDesign.GetModule("ModelSetup") oModule.ImportArray("Array", "C:\\Users\\MyFolder\\Export Array3.csv")</pre>

ImportDataset

Imports a dataset from a named file. This can be executed by the oProject, or oDesign variables. The name of the dataset is file-name+index number (e.g., dsdata1) unless the filename ends with a trailing number. When there is a trailing number at the end, we will remove the number and use first unused index. Alternatively, the name of the dataset can be explicitly defined by providing a string as an optional second argument.

UI Access	Project > Datasets > Import.		
Parameters	Name	Type	Description
	<datasetFilePath>	String	The full path to the file containing the dataset values. *.tab files recommended (see note below).
	<optionalDatasetName>	String	Optional. User-defined dataset name.

Return Value	None.

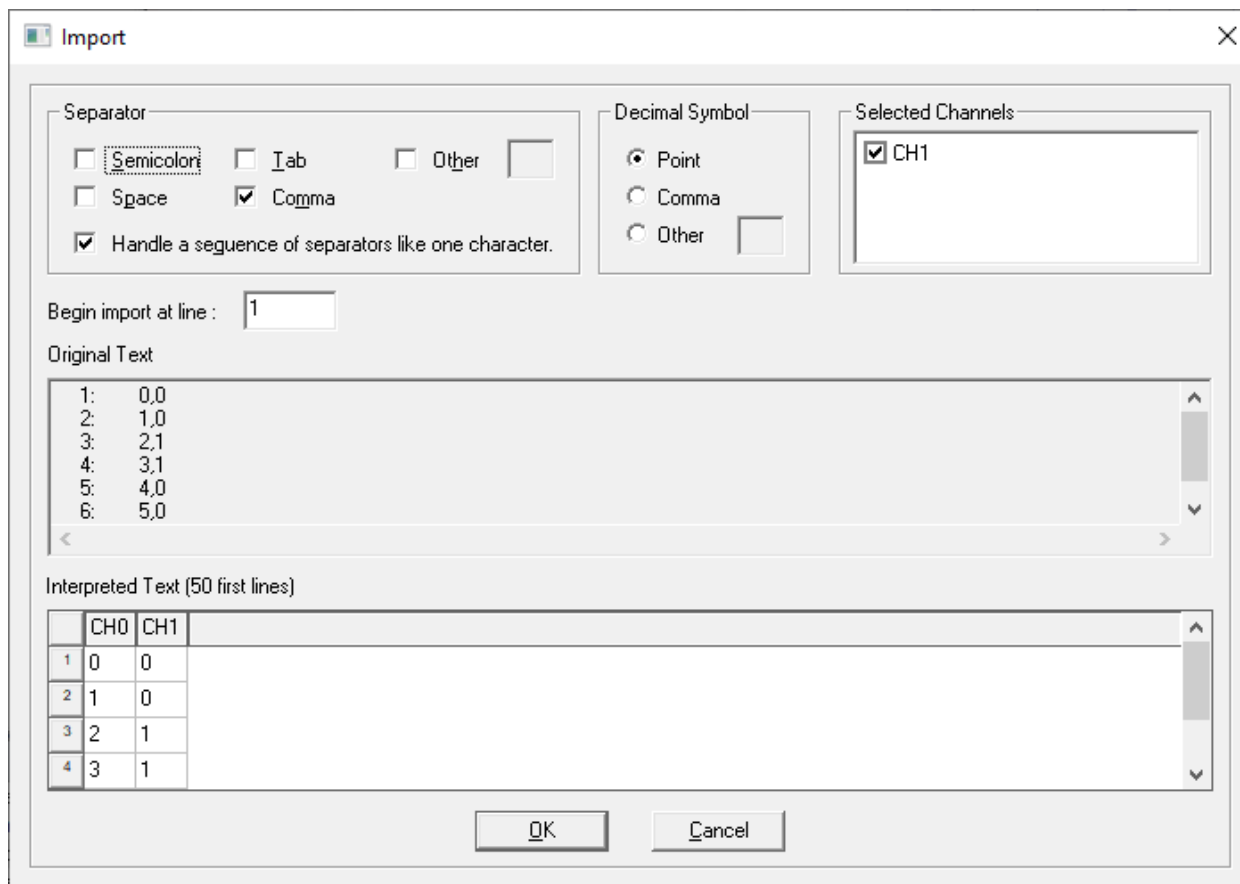
Python Syntax	<code>ImportDataset (<datasetFilePath>,<optionalDatasetName>)</code>
Python Example	<pre>oProject.ImportDataset('e:\tmp\dsdata.tab') oDesign.ImportDataset('e:\tmp\dsdata.tab') oProject.ImportDataset('e:\tmp\dsdata.tab', 'MyDatasetName') oDesign.ImportDataset('e:\tmp\dsdata.tab', 'MyDatasetName')</pre>

VB Syntax	<code>ImportDataset <datasetFilePath>, <optionalDatasetName></code>
VB Example	<pre>oProject.ImportDataset "e:\tmp\dsdata.tab" oDesign.ImportDataset "e:\tmp\dsdata.tab" oProject.ImportDataset "e:\tmp\dsdata.tab", "MyDatasetName" oDesign.ImportDataset "e:\tmp\dsdata.tab", "MyDatasetName"</pre>

Note About File Types:

Tab-delimited or space-delimited files with the extension *.tab are the recommended file type. When using ImportDataset at the Design level, *.tab is the only file type supported.

At the Project level, other file types are supported (for example, *.csv). However, after calling the command, you must configure the file import format manually through the Electronics Desktop GUI by selecting **Project > Datasets** and clicking **Import**.



PasteArray

Copies a specified array.

UI Access	Edit > Paste.
Parameters	None. You must previously use CopyArray.
Return Value	New array name.

Python Syntax	Paste ()
Python Example	<pre>oModule = oDesign.GetModule("ModelSetup") oModule.CopyArray("Array") oModule.PasteArray() # returns the new array name</pre>

PasteDesign (Design Object)

Pastes a design that has already been copied to the clipboard into another design.

UI Access	Edit > Paste.		
Parameters	Name	Type	Description
	<pasteOption>	Integer	One of the following: • 0 – Link to the existing design that was copied • 1 – Create a new copy of the design and keep the original layers of the design being copied • 2 – Create a new copy of the design and merge the layers of the design being copied.

Return Value	None.

Python Syntax	<code>PasteDesign(<pasteOption>)</code>
Python Example	<pre>oProject.CopyDesign('DesignB') oDesign = oProject.SetActiveDesign('DesignA') oDesign.PasteDesign(1)</pre>

VB Syntax	<code>PasteDesign <pasteOption></code>
VB Example	<pre>oProject.CopyDesign "B" Set oDesign = oProject.SetActiveDesign("A") oDesign.PasteDesign 1</pre>

Redo [Design]

Reapplies the last design-level command.

UI Access	Edit > Redo.
Parameters	None.
Return Value	None.

Python Syntax	<code>Redo()</code>
Python Example	<code>oDesign.Redo()</code>

VB Syntax	Redo
VB Example	<code>oDesign.Redo</code>

RenameDesignInstance

Renames a design instance.

UI Access	Right-click a design instance in the project tree, and then click Rename on the shortcut menu.		
Parameters	Name	Type	Description
	<OldName>	String	The current name of the design, which must be the design on which this command is invoked.
	<NewName>	String	The new name for the design.
Return Value	None.		

Python Syntax	<code>RenameDesignInstance (<OldName>, <NewName>)</code>
Python Example	<code>oDesign.RenameDesignInstance ("Design1", "Design2")</code>

VB Syntax	<code>RenameDesignInstance <OldName>, <NewName></code>
VB Example	<code>oDesign.RenameDesignInstance "Design1", "Design2"</code>

RenameSource [Interface Source]

Use: Renames an interface source

Syntax: RenameSource (STRING: Interface Source name)

VB Example: `oModule.RenameSource "OldName" "NewName"`

RevertAllToInitialCondition

Reverts all setups to their initial condition. Used for linked thermal designs to revert the target solution (clearing restart, thermal monitor, and field results).

UI Access	Project Manager > right-click Analysis > Revert to Initial Condition . or, with either <i>nothing</i> or with Analysis selected in the Project Manager: Mechanical > Analysis > Revert to Initial Condition (from the menu bar)
Parameters	None.
Return Value	None

Python Syntax	<code>RevertAllToInitialCondition()</code>
Python Example	<code>oModule.RevertAllToInitialCondition()</code>

VB Syntax	<code>RevertAllToInitialCondition</code>
VB Example	<code>oModule.RevertAllToInitialCondition</code>

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^3.
	<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	<code>oDesign.SARSetup(1.0, 1.0, 678, 1.0, 0)</code>

VB Syntax	SARSetup <TissueMass>, <MaterialDensity>, <TissueObjectListID>, <VoxelSize>, <AverageSARMethod>
VB Example	<code>oDesign.SARSetup 1.0, 1.0, 678, 1.0, 0</code>

SetActiveEditor

Sets the active editor.

UI Access	N/A.

Parameters	Name	Type	Description
	<EditorName>	String	Text of the design's notes.
Return Value	Editor object.		

Python Syntax	SetActiveEditor(<EditorName>)
Python Example	<code>oDesign.SetActiveEditor('3D Modeler')</code>

VB Syntax	SetActiveEditor <EditorName>
VB Example	<code>oDesign.SetActiveEditor "3D Modeler"</code>

SetAllowMaterialOverride

Sets the option to allow material override.

UI Access	N/A.		
Parameters	Name	Type	Description
	<allowMaterialOverride>	Integer	1 - allow material override. 0 - does not allow material override.
Return Value	None.		

Python Syntax	SetAllowMaterialOverride(<allowMaterialOverride>)
Python Example	<code>oDesign.SetAllowMaterialOverride(1)</code>

VB Syntax	SetAllowMaterialOverride <allowMaterialOverride>
VB Example	oDesign.SetAllowMaterialOverride 1

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".

UI Access	From the Project Manager , right-click the design and select Set Background Material .		
Parameters	Name	Type	Description
	<matName>	String	Material name.
Return Value	None.		

Python Syntax	SetBackgroundMaterial(<matName>)
Python Example	oDesign.SetBackgroundMaterial('vacuum')

VB Syntax	SetBackgroundMaterial <matName>
VB Example	oDesign.SetBackgroundMaterial "vacuum"

SetDesignMode

Switches between HFSS 3D Layout operating modes.

UI Access	Edit Layout...		
Parameters	Name	Type	Description
	<modeName>	String	Enter "IC" or "General" to change operating modes.
Return Value	None.		

Python Syntax	SetDesignMode(<modeName>)
Python Example	oDesign.SetDesignMode("IC")

VB Syntax	SetDesignMode <modeName>
VB Example	oDesign.SetDesignMode "IC"

SetDesignSettings (Mechanical)

Sets the design settings for Mechanical designs. The available parameters depend on the current solution type (Modal, Steady-State Thermal, Transient Thermal, or Structural).

UI Access	Mechanical > Design Settings			
Parameters	Name	Type	Solution Type*	Description
	Mechanical Parameters Array:			
	<NAME:MechanicalParams>	string	1, 2, 3, 4	"NAME:Design Settings Data"
	<Allow Material Override>	bool	2, 3	True False
	<Perform Minimal Validation>	bool	1, 2, 3, 4	True False
	<AmbientTemperature>	string	2, 3	Global ambient temperature for convection

				boundaries (with units)
	<EnvironmentTemperature>	string	4	Global stress-free temperature for structural solutions (with units)
	<ExportOnSimulationComplete>	bool	3	True False
	<ExportDirectory>	string	3	Path to export folder – If string is empty, files are saved to a subfolder: "<project_name>.aedtresults," under the current project's folder
	Model Parameters Array:			
	<NAME:ModelParams>	string	1, 2, 3, 4	Array "NAME:Model Validation Settings"
	<EntityCheckLevel>	string	1, 2, 3, 4	"None" "WarningOnly" "Basic" "Strict"
	<IgnoreUnclassifiedObjects>	bool	1, 2, 3, 4	True False
	<SkipIntersectionChecks>	bool	1, 2, 3, 4	True False
Return Value	None			

Note: *

In the above table, the Solution Type numbers are defined as follows:

1. Modal
2. Steady-State Thermal
3. Transient Thermal
4. Structural

Python Syn-	SetDesignSettings ([<NAME:MechanicalParams>, <Allow Material Override>, <Perform Minimal Validation>,

tax	[<AmbientTemperature>, <EnvironmentTemperature>, <ExportOnSimulationComplete>, <ExportDirectory>], [<NAME:ModelParams>, <EntityCheckLevel>, <IgnoreUnclassifiedObjects>, <SkipIntersectionChecks>]
Python Example [Transient Thermal]	<pre> oDesign.SetDesignSettings(["NAME:Design Settings Data", "Allow Material Override:=" , True, "Perform Minimal validation:=" , False, "AmbientTemperature:=" , "20cel", "ExportOnSimulationComplete:=" , True, "ExportDirectory:=" , "D:/Ansys/Mech-TT/TransThermal5.ae- dtexport/"], ["NAME:Model Validation Settings" , "EntityCheckLevel:=" , "Strict", "IgnoreUnclassifiedObjects:=" , False, "SkipIntersectionChecks:=" , False]) </pre>
Python Example [Structural]	<pre> oDesign.SetDesignSettings(["NAME:Design Settings Data", "Allow Material Override:=" , True, "Perform Minimal validation:=" , False, "EnvironmentTemperature:=" , "183cel"], ["NAME:Model Validation Settings" , "EntityCheckLevel:=" , "Strict", "IgnoreUnclassifiedObjects:=" , False, "SkipIntersectionChecks:=" , False]) </pre>

VB Syntax	SetDesignSettings Array(<NAME:MechanicalParams>,<Allow Material Override>, <Perform Minimal Validation>, [<AmbientTemperature>, <EnvironmentTemperature>, <ExportOnSimulationComplete>, <ExportDirectory>], Array(<NAME:ModelParams>, <EntityCheckLevel>, <IgnoreUnclassifiedObjects>, <SkipIntersectionChecks>)
VB Example [Transient Thermal]	<pre>oDesign.SetDesignSettings Array ("NAME:Design Settings Data", _ "Allow Material Override:=", true, "Perform Minimal validation:=", false, _ "AmbientTemperature:=", "20cel", "ExportOnSimulationComplete:=", _ "ExportDirectory:=", "D:/Ansys/Mech-TT/TransThermal5.aedtexport/"), _ Array("NAME:Model Validation Settings", "EntityCheckLevel:=", "Strict", _ "IgnoreUnclassifiedObjects:=", false, "SkipIntersectionChecks:=", false)</pre>
VB Example [Structural]	<pre>oDesign.SetDesignSettings Array ("NAME:Design Settings Data", _ "Allow Material Override:=", true, "Perform Minimal validation:=", false, _ "EnvironmentTemperature:=", "183cel", Array("NAME:Model Validation Settings", _ "EntityCheckLevel:=", "Strict", "IgnoreUnclassifiedObjects:=", false, _ "SkipIntersectionChecks:=", false)</pre>

SetFastTransformationForLayoutComponent

Toggles whether layout components in show outlines of objects across layers during view changes. This improves the visualization performance.

UI Access	Fast Transformation		
Parameters	Name	Type	Description
	<Layout ComponentName>	String	Name of the Layout Component.
	Boolean	String	True or False
Return Value	None.		

Python Syntax	<code>SetFastTransformationForLayoutComponent ("<LayoutComponentName", <boolean>)</code>
Python Example	<code>oDesign.SetFastTransformationForLayoutComponent ("LC1_1", False)</code>

SetLengthSettings

Sets the distributed and lumped lengths of the design, as well as the rise time.

UI Access	N/A		
Parameters	Name	Type	Description
	<DistributedUnits>	String	Length units used for post-processing.
	<LumpedLength>	String	Length of design in post-processing.
	<RiseTime>	String	Time used in post-processing.
Return Value	None.		

Python Syntax	<code>SetLengthSettings(<DistributedUnits>, <LumpedLength>, <RiseTime>)</code>
Python Example	<code>oDesign.SetLengthSettings('mm', '7meter', '1s')</code>

VB Syntax	<code>SetLengthSettings<DistributedUnits>, <LumpedLength>, <RiseTime>)</code>
VB Example	<code>oDesign.SetLengthSettings "mm", "7meter", "1s"</code>

SetObjectAttributesForLayoutComponent

Sets visualization attributes for layout components in for a named component for layers, nets, and objects.

UI Access	Object Attributes		
Parameters	Name	Type	Description
	<Layout ComponentName>	String	Name of the Layout Component.
Return Value	None.		

Python Syntax	SetObjectAttributesForLayoutComponent (["<LayoutComponentName">",[<ObjectAttributesinNetMode>][<ObjectAttributesinLayerMode>],[<ObjectAttributesinObjectMode>])
Python Example	<pre>oDesign.SetObjectAttributesForLayoutComponent (["Name:=" , "LC1_1", "ShowDielectric:=" , False, "DisplayMode:=" , 2, ["NAME:ObjectAttributesInLayerMode", "GND_1_L2:=" , [True,True,0], "GND_2_L4:=" , [True,True,0], "GND_3_L6:=" , [True,True,0], "GND_4_L9:=" , [True,True,0], "GND_5_L11:=" , [True,True,0], "Inner_Layer_3_L10:=" , [True,True,0],</pre>

```

"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],

```

	<pre>"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]])</pre>

SetPropValue [Design]

Sets the property value for the active property object.

UI Access	Edit properties on Project Tree objects.		
Parameters	Name	Type	Description
	<propPath>	String	A child object's property path. See: Object Property Script Function Summary .
	<Value>	String	New property value.
Return Value	Boolean: • True - property found and the new value is valid. • False - property not found.		

Python Syntax	SetPropValue(<propPath>, <Value>)
Python Example	<pre>oDesign.SetPropValue("offset", "12mm") oDesign.SetPropValue("Results/S Parameter Plot 1/Display Type", "Rectangular Plot")</pre>

VB Syntax	SetPropValue <propPath>, <Value>
VB Example	<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	One of the following, where tab titles are shown in parentheses: - PassedParameterTab ("Parameter Values") - DefinitionParameterTab (Parameter Defaults") - LocalVariableTab ("Variables" or "Local Variables") - ProjectVariableTab ("Project variables")

			• ConstantsTab ("Constants") • BaseElementTab ("Symbol" or "Footprint") • ComponentTab ("General") • Component("Component") • CustomTab ("Intrinsic Variables") • Quantities ("Quantities") • Signals ("Signals")
	<code><propServer></code>	String	An object identifier, generally returned from another script method, such as <code>CompInst@R;2;3</code>
	<code><propName></code>	String	Name of the property.
	<code><propValue></code>	String	The value for the property
Return Value	None.		

Python Syntax	<code>SetPropertyValue(<propTab>, <propServer>, <propName>, <propValue>)</code>
Python Example	<code>oEditor.SetPropertyValue("PassedParameterTab", "k", "R", "2200")</code>

VB Syntax	<code>SetPropertyValue <propTab>, <propServer>, <propName>, <propValue></code>
VB Example	<code>oEditor.SetPropertyValue "PassedParameterTab", "k", "R", "2200"</code>

SetShowLayoutForLayoutComponent

Layout visualization, rather than bounding box, for a Layout Component in design.

UI Access	Edit Layout...		
Parameters	Name	Type	Description
	<Layout ComponentName>	String	Name of the Layout Component.
	Boolean	String	True or False
Return Value	None.		

Python Syntax	SetShowLayoutForLayoutComponent ("<LayoutComponentName", <boolean>)
Python Example	<code>oDesign.SetShowLayoutForLayoutComponent ("LC1_1", True)</code>

SetShowAllLayoutComponents

This command will show/hide all layout components in design.

UI Access	Edit Layout... View > Visibility > Show All or Hide All		
Parameters	Name	Type	Description
	Boolean	String	True or False
Return Value	None		

Python Syntax	SetShowAllLayoutComponents (<boolean>)
Python Example	<pre>oDesign = oProject.GetActiveDesign() oDesign.SetShowAllComponentLayouts (True)</pre>

SetSolutionType

Sets the solution type for the design.

UI Access	HFSS > Solution Type.		
Parameters	Name	Type	Description
	<SolutionType>	String	Possible values are: "SBR+", "HFSS [Modal Terminal] [Network Composite]", "Transient [Network Composite]", "Eigenmode", or "Characteristic".
	EnableAutoOpen:=,	Boolean	Only . • True - turn on auto open mode. • False - turn off auto open mode.
	<ModelExteriorAsIE>	String	Only Applies for Driven Solution Type with Auto Open Mode as True. Possible values are: Radiation, FEBI, PML
Return Value	None.		

Python Syntax	SetSolutionType(<SolutionType>, <AutoOpenMode>, <ModelExteriorAsIE>)
Python Example	<pre>oDesign.SetSolutionType("HFSS Modal Network", ["NAME:Options", "EnableAutoOpen:=" , False])</pre>

```
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  
    ])
```

**VB
Syntax****SetSolutionType** <SolutionType>, <AutoOpenMode>, <ModelExteriorAsIE>

VB Example	<code>oDesign.SetSolutionType "Transient Network", Array("NAME:Options", "EnableAutoOpen:=", true, "BoundaryType:=", "Radiation")</code>
	<code>oDesign.SetSolutionType "SBR+", Array("NAME:Options", "EnableAutoOpen:=", false)</code>
	<code>oDesign.SetSolutionType "HFSS Terminal Composite", Array("NAME:Options", "EnableAutoOpen:=", false)</code>

SetSolveInsideThreshold

Sets the solve inside threshold to the specified double.

UI Access	N/A		
Parameters	Name	Type	Description
	<threshold>	Double	Siemens/m.
Return Value	None.		

Python Syntax	<code>SetSolveInsideThreshold(<threshold>)</code>
Python Example	<code>oDesign.SetSolveInsideThreshold(100000)</code>

VB Syntax	<code>SetSolveInsideThreshold <threshold></code>
VB Example	<code>oDesign.SetSolveInsideThreshold 100000</code>

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.

UI Access	HFSS > Fields > Edit Sources.		
Parameters	Name	Type	Description
	<SourceId>	Array	Name of modes/terminals to be set as source context.
Return Value	None.		

Python Syntax	SetSourceContexts(<SourceId>)		
Python Example	<pre>oModule.SetSourceContexts(["Box1_T1", "Box1_T2", "Box1_T3", "Current1", "IncPWave1"])</pre>		

VB Syntax	SetSourceContexts <SourceId>		
VB Example	<pre>oModule.SetSourceContexts _ Array("Box1_T1", "Box1_T2", "Box1_T3", "Current1", "IncPWave1")</pre>		

SetVariableValue

Sets the value of a variable. To set the value of a project variable, execute this command using `oProject`. To set the value of a local variable, use `oDesign`.

UI Access	N/A		
Parameters	Name	Type	Description
	<VarName>	String	Variable name.
	<VarValue>	Value	New value for the variable.
Return Value	None.		

Python Syntax	SetVariableValue (<VarName>, <VarValue>)
Python Example	<code>oProject.SetVariableValue('\$Var1', '3mm')</code>

VB Syntax	SetVariableValue <VarName>, <VarValue>
VB Example	<code>oProject.SetVariableValue "\$Var1", "3mm"</code>

Solve

Performs one or more simulation. The next script command will not be executed until the simulation(s) are complete.

UI Access	Select solution setup(s). Right-click and select Analyze .

Parameters	Name	Type	Description
	<SimulationNames>	Array	Array containing string simulation names.
Return Value	Integer: • 0 – Simulation(s) completed. • 1 – Simulation error. • -1 – Command execution error.		

Python Syntax	Solve <SimulationNames>
Python Example	<code>oDesign.Solve(['Setup1','Setup2','Setup3'])</code>

VB Syntax	Solve <SimulationNames>
VB Example	<code>oDesign.Solve Array("Setup1", "Setup2", "Setup3")</code>

StopSimLink

Stops linked simulation.

UI Access	N/A		
Parameters	Name	Type	Description
	<simId>	Integer	ID of specified simulation.
	<abort>	Boolean	Whether to stop the running simulation: • True - abort the running simulation. • False - do not abort running simulation.
Return Value	None.		

Python Syntax	StopSimLink(<simId>, <abort>)
Python Example	oDesign.StopSimLink(18, True)

VB Syntax	StopSimLink <simId>, <abort>
VB Example	oDesign.StopSimLink 18, true

Undo [Design]

Cancels the last design-level command.

UI Access	Edit > Undo
Parameters	None.
Return Value	None.

Python Syntax	Undo()
Python Example	oDesign.Undo()

VB Syntax	Undo
VB Example	oDesign.Undo

ValidateLink

Note:

This command is for internal Ansys use only.

Python Syntax	ValidateLink()
Python Example	oDesign.ValidateLink()

ValidateDesign

Returns whether a design is valid.

UI Access	Mechanical > Validation Check.
Parameters	None.
Return Value	Integer: • 1 – Validation passed. • 0 – Validation failed.

Python Syntax	ValidateDesign()
Python Example	oDesign.ValidateDesign()

VB Syntax	ValidateDesign
VB Example	oDesign.ValidateDesign

ValidateLink

Note:
This command is for internal Ansys use only.

Python Syntax	ValidateLink()
Python Example	oDesign.ValidateLink()

8 - 3D Modeler Editor Script Commands

3D Modeler commands should be executed by the "3D Modeler" editor:

```
Set oEditor = oDesign.SetActiveEditor("3D Modeler")
```

```
oEditor.<CommandName>
```

Conventions Used in this Chapter:

<AttributesArray>

<AttributesArray> takes the following structure:

```
Array("NAME:Attributes",  
      "Name:=", <string>,  
      "Flags:=", <string>,  
      "Color:=", <string>,  
      "Transparency:=", <value>,  
      "PartCoordinateSystem:=", <string>,  
      "UDMId:=", <string>,  
      "MaterialValue:=", <string>,  
      "SurfaceMaterialValue:=", <string>,  
      "Solveinside:=", <boolean>,  
      "ShellElement:=", <boolean>,  
      "ShellElementThickness:=", <string>,  
      "ReferenceTemperature:=", <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.
- **ReferenceTemperature** – String containing the temperature at which an object is in a stress and strain-free state (applicable to Mechanical–Structural solutions). The default value is "EnvTemp" (the environment temperature, which is a global design variable).
- **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](#)

[Other oEditor Commands](#)

Draw Menu Commands

[Create3DComponent](#)

[CreateBondwire](#)

[CreateBox](#)

[CreateCircle](#)

[CreateCone](#)

[CreateCutplane](#)

[CreateCylinder](#)

[CreateEllipse](#)

[CreateEquationCurve](#)

[CreateEquationSurface](#)

[CreateHelix](#)

[CreatePoint](#)

[CreatePolyline](#)

[CreateRectangle](#)

[CreateRegularPolygon](#)

[CreateRegularPolyhedron](#)

[CreateSphere](#)

[CreateSpiral](#)

[CreateTorus](#)

[CreateUserDefinedModel](#)

[CreateUserDefinedPart](#)

[Edit3DComponent](#)

[EditPolyline](#)

[EditNativeComponentDefinition \[Beta – Layout Components in Mechanical\]](#)

[Get3DComponentDefinitionNames](#)

[Get3DComponentInstanceNames](#)

[Get3DComponentMaterialNames](#)

[Get3DComponentMaterialProperties](#)

[Get3DComponentParameters](#)

[Insert3DComponent](#)

[InsertNativeComponentDefinition \[Beta – Layout Components in Mechanical\]](#)

[InsertPolylineSegment](#)

[SweepAlongPath](#)

[SweepAlongVector](#)

[SweepAroundAxis](#)

[SweepFacesAlongNormal](#)

[SweepFacesAlongNormalWithAttributes](#)

[UpdateComponentDefinition](#)

Create3DComponent

Creates a 3D component.

UI Access	Draw > 3D Component Library > Create 3D Component.		
Parameters	Name	Type	Description
	<CreateData>	Array	Structured array containing geometry data: (Array["NAME:CreateData", "ComponentName:=", "<string>", "Company:=", "<string>", "Company URL:=", "<string>", "Model Number:=", "<string>", "Help URL:=" , "<string>", "Version:=", "<string>",

			<pre> "Notes:=", "<string>", "IconType:=", "<string>", "Owner:=", "<string>", "Email:=", "<string>", "Date:=", "<string>", #Component creation date "HasLabel:=", <boolean>, "IsEncrypted:=", <boolean>, "AllowEdit:=", <boolean>, "SecurityMessage:=", "<string>", "Password:=", "<string>", "EditPassword:=", "<string>", "PasswordType:=", "<string>", "HideContents:=", <boolean>, "ReplaceNames:=", <boolean>, #for objects and materials "ComponentOutline:=", "<string>","#None" or "Bounding Box" "IncludedParts:=", [Array of partnames], "HiddenParts:=", [Array of hidden parts], "IncludedCS:=", [Array of included CS], "DefaultHandle:=", "<reference CS>" "IncludedParameters:=", [Array of included parameters], "IncludedDependentParameters:=", [Array of dependent </pre>

			<pre>parameters], "ParameterDescription:=", <array> "IsLicensed:=", <boolean> "LicensingDllName:=", "<string>" "VendorComponentIdentifier:=", "<string>" "PublicKeyFile:=", "<string>"]</pre>
	<DesignData>	Array	Structured array containing design data: <pre>Array["NAME:DesignData", "Boundaries:=", <array>, "Excitations:=", <array>, "MeshOperations:=", <array>]</pre>
	<FileName>	String	Full file path to 3D component.
	<ImageFile>	Array	Optional. Structured array containing file path of 3D component image: <pre>Array["NAME:ImageFile", "ImageFile:=", <string>])</pre>
Return Value	None.		

Python Syntax	Create3DComponent(<CreateData>, <DesignData>, <FileName>, <ImageFile>)
Python	oEditor.Create3DComponent

Example

```
(["NAME:CreateData",
  "ComponentName:=", "Connector",
  "Company:=", "",
  "Version:=", "1.0",
  "Notes:=", "",
  "Owner:=", "",
  "Email:=", "",
  "Date:=", "11:41:01 AM Aug 28, 2014",
  "HasLabel:=", false,
  "IncludedParts:=", ["Box1", "Cylinder1", "Cone1"],
  "IncludedCS:=", ["RelativeCS1"],
  "DefaultHandle:=", "Global",
  "IncludedParameters:=", ["htcone", "lr", "htcyl", "zs", "radcyl", "xs", "$rp",
"$con"],
  "ParameterDescription:=", [],
  "IsLicensed:=",      , False,
  "LicensingDllName:=",      , "",
  "VendorComponentIdentifier:=", "",
  "PublicKeyFile:=",      , ""
],
["NAME:DesignData",
  "Boundaries:=", ["PerfE1", "FiniteCond1"],
```


	<pre> "Excitations:=", ["1"], "MeshOperations:=", [], "C:/tmp/Connector.a3dcomp", ["NAME:ImageFile", "ImageFile:=", ""]) </pre>

VB Syntax	Create3DComponent <CreateData>, <DesignData>, <FileName>, <ImageFile>
VB Example	<pre> oEditor.Create3DComponent Array("NAME:CreateData", "ComponentName:=", "Connector", "Company:=", "", "Version:=", "1.0", "Notes:=", "", "Owner:=", "", "Email:=", "", "Date:=", "11:41:01 AM Aug 28, 2014", "HasLabel:=", false, "IncludedParts:=", Array("Box1", "Cylinder1", "Conel"), "IncludedCS:=", Array("RelativeCS1"), "DefaultHandle:=", "Global", </pre>

	<pre>"IncludedParameters:=", Array("htcone", "lr", "htcyl", "zs", "radcyl", "xs", "\$rp", "\$con"), "ParameterDescription:=", Array() "IsLicensed:=" , False, "LicensingDllName:=" , "", "VendorComponentIdentifier:=", "", "PublicKeyFile:=" , ""), Array("NAME:DesignData", "Boundaries:=", Array("PerfE1", "FiniteCond1"), "Excitations:=", Array("1"), "MeshOperations:=", Array()), "C:/tmp/Connector.a3dcomp", Array("NAME:ImageFile", "ImageFile:=", ""))</pre>

CreateBondwire

Creates a bondwire.

UI Access	Draw > Bondwire.		
Parameters	Name	Type	Description
	<Parameters>	Array	Structured array. Array("NAME:BondwireParameters", "WireType:=", <string("JEDEC_4Points", "JEDEC_5Points", or "LOW")>,

			<pre>"WireDiameter:=", <string>, "NumSides:=", <value>, "XPadPos:=", <value>, "YPadPos:=", <value>, "ZPadPos:=", <value>, "XDir:=", <value>, "YDir:=", <value>, "ZDir:=", <value>, "Distance:=", <value>, "h1:=", <value>, "h2:=", <value>, "alpha:=", <value>, "beta:=", <value>, "WhichAxis:=", <string("X","Y", or "Z")>, "ReverseDirection:=", <boolean></pre>
	<AttributesArray>	Array	Structured array. See: AttributesArray .
Return Value	None.		

Python Syntax	CreateBondwire(<Parameters>, <Attributes>)
Python Example	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",
  "ReverseDirection:="  , True
],
["NAME:Attributes",
  "Name:="              , "Bondwire1",
  "Flags:="             , "",

```

	<pre> "Color:=" , "(143 175 143)", "Transparency:=" , 0, "PartCoordinateSystem:=" , "Global", "UDMId:=" , "", "MaterialValue:=" , "\"vacuum\"", "SurfaceMaterialValue:=" , "\"\"", "SolveInside:=" , True, "ShellElement:=" , False, "ShellElementThickness:=" , "0mm", "IsMaterialEditable:=" , True, "UseMaterialAppearance:=" , False, "IsLightweight:=" , False] </pre>

VB Syntax	CreateBondwire <Parameters>, <Attributes>
VB Example	<pre> oEditor.CreateBondwire Array("NAME:BondwireParameters", "WireType:=", "LOW", "WireDia- meter:=", _ "0.025mm", "NumSides:=", "6", "XPadPos:=", "6mm", "YPadPos:=", "-2.5mm", "ZPadPos:=", _ "0mm", "XDir:=", "-10mm", "YDir:=", "4mm", "ZDir:=", "0mm", "Distance:=", _ "10.770329614269mm", "h1:=", "0.2mm", "h2:=", "0mm", "alpha:=", "80deg", "beta:=", _ </pre>

	<pre>"80deg", "WhichAxis:=", "Z", "ReverseDirection:=", false), Array("NAME:Attributes", "Name:=", _ "Bondwire2", "Flags:=", "", "Color:=", "(143 175 143)", "Transparency:=", 0, "PartCoordinateSystem:=", _ "Global", "UDMId:=", "", "MaterialValue:=", "" & Chr(34) & "copper" & Chr(34) & "", "Sur- faceMaterialValue:=", _ "" & Chr(34) & "" & Chr(34) & "", "SolveInside:=", false, "ShellElement:=", _ false, "ShellElementThickness:=", "0mm", "IsMaterialEditable:=", true, "UseMa- terialAppearance:=", _ false, "IsLightweight:=", false)</pre>

CreateBox

Creates a box.

UI Access	Draw > Box.		
Parameters	Name	Type	Description
	<Parameters>	Array	Structured array. Array("NAME:BoxParameters", "XPosition:=", <string>, "YPosition:=", <string>, "ZPosition:=", <string>, "XSize:=" , <string>, "YSize:=" , <string>, "ZSize:=" , <string>)
	<AttributesArray>	Array	Structured array. See: AttributesArray .

Return Value	None.

Python Syntax	CreateBox(<Parameters>, <Attributes>)
Python Example	<pre>oEditor.CreateBox(["NAME:BoxParameters", "XPosition:=" , "0.5mm", "YPosition:=" , "-6.5mm", "ZPosition:=" , "0mm", "XSize:=" , "2mm", "YSize:=" , "1.5mm", "ZSize:=" , "1.5mm"], ["NAME:Attributes", "Name:=" , "Box3", "Flags:=" , "", "Color:=" , "(143 175 143)", "Transparency:=" , 0, "PartCoordinateSystem:=" , "Global", "UDMId:=" , ""</pre>

	<pre> "MaterialValue:=" , "\"copper\"", "SurfaceMaterialValue:=" , "\"\"", "SolveInside:=" , False, "ShellElement:=" , False, "ShellElementThickness:=", "0mm", "IsMaterialEditable:=" , True, "UseMaterialAppearance:=", False, "IsLightweight:=" , False] </pre>

VB Syntax	CreateBox <Parameters>, <Attributes>
VB Example	<pre> oEditor.CreateBox Array("NAME:BoxParameters", "XPosition:=", "0mm", "YPosition:=", _ "4mm", "ZPosition:=", "0mm", "XSize:=", "-4mm", "YSize:=", "1mm", "ZSize:=", _ "2mm"), Array("NAME:Attributes", "Name:=", "Box4", "Flags:=", "", "Color:=", _ "(143 175 143)", "Transparency:=", 0, "PartCoordinateSystem:=", "Global", "UDMId:=", _ "", "MaterialValue:=", "" & Chr(34) & "copper" & Chr(34) & "", "Sur- faceMaterialValue:=", _ "" & Chr(34) & "" & Chr(34) & "", "SolveInside:=", false, "ShellElement:=", _ false, "ShellElementThickness:=", "0mm", "IsMaterialEditable:=", true, _ "UseMaterialAppearance:=", false, "IsLightweight:=", false) </pre>

CreateCircle

Creates a circle.

UI Access	Draw > Circle.		
Parameters	Name	Type	Description
	<Parameters>	Array	Structured array. Array("NAME:CircleParameters", "IsCovered:=", <boolean>, "XCenter:=", <value>, "YCenter:=", <value>, "ZCenter:=", <value>, "Radius:=", <value>, "WhichAxis:=", <string> "NumSegments:=", <string containing integer>)
	<AttributesArray>	Array	Structured array. See: AttributesArray .
Return Value	None.		

Python Syntax	CreateCircle(<Parameters>, <Attributes>)
Python Example	<pre>oEditor.CreateCircle(["NAME:CircleParameters", "IsCovered:=" , True,</pre>

```

"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\"",
  "SurfaceMaterialValue:=", "\"\"",
  "SolveInside:="       , False,
  "ShellElement:="      , False,
  "ShellElementThickness:=", "0mm",
  "IsMaterialEditable:=" , True,
  "UseMaterialAppearance:=", False,

```

	<code>"IsLightweight:=" , False</code> <code>l)</code>

VB Syntax	CreateCircle <Parameters>, <Attributes>
VB Example	<pre>oEditor.CreateCircle Array("NAME:CircleParameters", "IsCovered:=", true, "XCenter:=", _ "7mm", "YCenter:=", "-6mm", "ZCenter:=", "0mm", "Radius:=", "0.5mm", "WhichAxis:=", _ "Z", "NumSegments:=", "0"), Array("NAME:Attributes", "Name:=", "Circle2", "Flags:=", _ "", "Color:=", "(143 175 143)", "Transparency:=", 0, "PartCoordinateSystem:=", _ "Global", "UDMId:=", "", "MaterialValue:=", "" & Chr(34) & "copper" & Chr(34) & "", "Sur- faceMaterialValue:=", _ "" & Chr(34) & "" & Chr(34) & "", "SolveInside:=", false, "ShellElement:=", _ false, "ShellElementThickness:=", "0mm", "IsMaterialEditable:=", true, "UseMa- terialAppearance:=", _ false, "IsLightweight:=", false)</pre>

CreateCone

Creates a cone.

UI Access	Draw > Cone.		
Parameters	Name	Type	Description
	<Parameters>	Array	Structured array. Array("NAME:ConeParameters",

			"XCenter:=", <string>, "YCenter:=", <string>, "ZCenter:=", <string>, "WhichAxis:=", <string>, "Height:=", <string>, "BottomRadius:=", <string>, "TopRadius:=", <string>)
	<AttributesArray>	Array	Structured array. See: AttributesArray .
Return Value	None.		

Python Syntax	CreateCone(<Parameters>, <Attributes>)
Python Example	<pre>oEditor.CreateCone(["NAME:ConeParameters", "XCenter:=" , "3mm", "YCenter:=" , "-4.5mm", "ZCenter:=" , "0mm", "WhichAxis:=" , "Z", "Height:=" , "2.5mm", "BottomRadius:=" , "2.82842712474619mm", "TopRadius:=" , "2.23606797749979mm"],</pre>

```
[ "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 Exempl-	oEditor.CreateCone Array("NAME:ConeParameters", "XCenter:=", "3mm", "YCenter:=", _

e	<pre>"-3mm", "ZCenter:=", "0mm", "WhichAxis:=", "Z", "Height:=", "4mm", "BottomRadius:=", _ "1.11803398874989mm", "TopRadius:=", "2.06155281280883mm"), Array("NAME:Attributes", "Name:=", _ "Cone1", "Flags:=", "", "Color:=", "(143 175 143)", "Transparency:=", 0, "PartCoordin- ateSystem:=", _ "Global", "UDMId:=", "", "MaterialValue:=", "" & Chr(34) & "copper" & Chr(34) & "", "Sur- faceMaterialValue:=", _ "" & Chr(34) & "" & Chr(34) & "", "SolveInside:=", false, "ShellElement:=", _ false, "ShellElementThickness:=", "0mm", "IsMaterialEditable:=", true, "UseMa- terialAppearance:=", _ false, "IsLightweight:=", false)</pre>

CreateCutplane

Creates a cutplane.

UI Access	Draw > Plane.		
Parameters	Name	Type	Description
	<Parameters>	Array	Structured array. Array("NAME:PlaneParameters", "PlaneBaseX:=", <string>, "PlaneBaseY:=", <string>, "PlaneBaseZ:=", <string>, "PlaneNormalX:=", <string>, "PlaneNormalY:=", <string>),

			"PlaneNormalZ:=", <string>)
	<AttributesArray>	Array	See: AttributesArray . CreateCutplane only takes the Name and Color attributes.
Return Value	None.		

Python Syntax	CreateCutplane(<Parameters>, <Attributes>)
Python Example	<pre>oEditor.CreateCutplane(["NAME:PlaneParameters", "PlaneBaseX:=" , "-0.6mm", "PlaneBaseY:=" , "-0.8mm", "PlaneBaseZ:=" , "0mm", "PlaneNormalX:=" , "1.2mm", "PlaneNormalY:=" , "0.2mm", "PlaneNormalZ:=" , "0mm"], ["NAME:Attributes", "Name:=" , "Plane1", "Color:=" , "(143 175 143)"])</pre>

VB Syntax	CreateCutplane <Parameters>, <Attributes>
VB Example	<pre>oEditor.CreateCutplane Array("NAME:PlaneParameters", "PlaneBaseX=", "0.6mm", "PlaneBaseY=", _ "1.2mm", "PlaneBaseZ=", "0mm", "PlaneNormalX=", "0.4mm", "PlaneNormalY=", _ "0.6mm", "PlaneNormalZ=", "0mm"), Array("NAME:Attributes", "Name=", "Plane2", "Col- or=", _ " (143 175 143) ")</pre>

CreateCylinder

Creates a cylinder.

UI Access	Draw > Cylinder.		
Parameters	Name	Type	Description
	<Parameters>	Array	Structured array. Array("NAME:CylinderParameters", "XCenter=", <string>, "YCenter=", <string>, "ZCenter=", <string>, "Radius=", <string>, "Height=", <string>, "WhichAxis=", <string>, "NumSides=", <string containing integer>)
	<AttributesArray>	Array	Structured array. See: AttributesArray .

Return Value	None.

Python Syntax	CreateCylinder(<Parameters>, <Attributes>)
Python Example	<pre>oEditor.CreateCylinder(["NAME:CylinderParameters", "XCenter:=" , "6mm", "YCenter:=" , "-4.5mm", "ZCenter:=" , "0mm", "Radius:=" , "0.5mm", "Height:=" , "4.5mm", "WhichAxis:=" , "Z", "NumSides:=" , "0"], ["NAME:Attributes", "Name:=" , "Cylinder1", "Flags:=" , "", "Color:=" , "(143 175 143)", "Transparency:=" , 0, "PartCoordinateSystem:=" , "Global", "UDMId:=" , ""</pre>

	<pre>"MaterialValue:=" , "\"copper\"", "SurfaceMaterialValue:=", "\"\"", "SolveInside:=" , False, "ShellElement:=" , False, "ShellElementThickness:=", "0mm", "IsMaterialEditable:=" , True, "UseMaterialAppearance:=", False, "IsLightweight:=" , False])</pre>

VB Syntax	CreateCylinder <Parameters>, <Attributes>
VB Example	<pre>oEditor.CreateCylinder Array("NAME:CylinderParameters", "XCenter:=", "1.5mm", "YCenter:=", _ "6.5mm", "ZCenter:=", "0mm", "Radius:=", "1mm", "Height:=", "1mm", "WhichAxis:=", _ "Z", "NumSides:=", "0"), Array("NAME:Attributes", "Name:=", "Cylinder2", "Flags:=", _ "", "Color:=", "(143 175 143)", "Transparency:=", 0, "PartCoordinateSystem:=", _ "Global", "UDMId:=", "", "MaterialValue:=", "" & Chr(34) & "copper" & Chr(34) & "", "SurfaceMaterialValue:=", _ "" & Chr(34) & "" & Chr(34) & "", "SolveInside:=", false, "ShellElement:=", _ false, "ShellElementThickness:=", "0mm", "IsMaterialEditable:=", true, "UseMaterialAppearance:=", _ false, "IsLightweight:=", false)</pre>

CreateEllipse

Creates an ellipse.

UI Access	Draw > Ellipse.		
Parameters	Name	Type	Description
	<Parameters>	Array	Structured array. Array("NAME:EllipseParameters", "IsCovered:=", <string>, "XCenter:=", <string>, "YCenter:=", <string>, "ZCenter:=", <string>, "MajRadius:=", <string>, "Ratio:=", <string>, "WhichAxis:=", <string>, "NumSegments:=", <string>)
	<AttributesArray>	Array	Structured array. See: AttributesArray .
Return Value	None.		

Python Syntax	CreateEllipse(<Parameters>, <Attributes>)
Python Example	<pre>oEditor.CreateEllipse(["NAME:EllipseParameters",</pre>

```

    "IsCovered:="          , True,
    "XCenter:="            , "0.6mm",
    "YCenter:="            , "-0.6mm",
    "ZCenter:="            , "0mm",
    "MajRadius:="          , "0.2mm",
    "Ratio:="              , "7",
    "WhichAxis:="          , "Z",
    "NumSegments:="        , "0"
],
["NAME:Attributes",
    "Name:="                , "Ellip1",
    "Flags:="               , "",
    "Color:="               , "(143 175 143)",
    "Transparency:="        , 0,
    "PartCoordinateSystem:=", "Global",
    "UDMId:="               , "",
    "MaterialValue:="       , "\"copper\"",
    "SurfaceMaterialValue:=", "\"\"",
    "SolveInside:="         , False,
    "ShellElement:="        , False,
    "ShellElementThickness:=", "0mm",

```

	<pre> "IsMaterialEditable:=" , True, "UseMaterialAppearance:=", False, "IsLightweight:=" , False 1) </pre>

VB Syntax	CreateEllipse <Parameters>, <Attributes>
VB Example	<pre> oEditor.CreateEllipse Array("NAME:EllipseParameters", "IsCovered:=", true, "XCenter:=", _ "-0.4mm", "YCenter:=", "-3.2mm", "ZCenter:=", "0mm", "MajRadius:=", "0.4mm", "Ratio:=", _ "0.5", "WhichAxis:=", "Z", "NumSegments:=", "0"), Array("NAME:Attributes", "Name:=", _ "Ellipse2", "Flags:=", "", "Color:=", "(143 175 143)", "Transparency:=", 0, "PartCoordin- ateSystem:=", _ "Global", "UDMId:=", "", "MaterialValue:=", "" & Chr(34) & "copper" & Chr(34) & "", "Sur- faceMaterialValue:=", _ "" & Chr(34) & "" & Chr(34) & "", "SolveInside:=", false, "ShellElement:=", _ false, "ShellElementThickness:=", "0mm", "IsMaterialEditable:=", true, "UseMa- terialAppearance:=", _ false, "IsLightweight:=", false) </pre>

CreateEquationCurve

Creates an equation-based curve.

UI Access	Draw > Equation-Based Curve.		
Parameters	Name	Type	Description
	<Parameters>	Array	Structured array. <pre>Array("NAME:EquationBasedCurveParameters", "XtFunction:=", <string>, "YtFunction:=", <string>, "ZtFunction:=", <string>, "tStart:=", <string>, "tEnd:=", <string>, "NumOfPointsOnCurve:=", <string>, "Version:=", <integer>), <polylineArray(optional)>)</pre>
	<AttributesArray>	Array	Structured array. See: AttributesArray .
Return Value	None.		

Python Syntax	CreateEquationCurve(<Parameters>, <Attributes>)
Python Example	<pre>oEditor.CreateEquationCurve(["NAME:EquationBasedCurveParameters", "XtFunction:=" , "1", "YtFunction:=" , "3", "ZtFunction:=" , "32", "tStart:=" , "1",</pre>

```

    "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:="               , "",

```

	<pre>"MaterialValue:=" , "\"copper\"", "SurfaceMaterialValue:=", "\"\"", "SolveInside:=" , False, "ShellElement:=" , False, "ShellElementThickness:=", "0mm", "IsMaterialEditable:=" , True, "UseMaterialAppearance:=", False, "IsLightweight:=" , False])</pre>

VB Syntax	CreateEquationCurve <Parameters>, <Attributes>
VB Example	<pre>oEditor.CreateEquationCurve Array("NAME:EquationBasedCurveParameters", "XtFunction:=", _ "1", "YtFunction:=", "3", "ZtFunction:=", "32", "tStart:=", "1", "tEnd:=", "3", "NumOfPointsOnCurve:=", _ "0", "Version:=", 1, Array("NAME:PolylineXSection", "XSectionType:=", "None", "XSec- tionOrient:=", _ "Auto", "XSectionWidth:=", "0", "XSectionTopWidth:=", "0", "XSectionHeight:=", _ "0", "XSectionNumSegments:=", "0", "XSectionBendType:=", "Corner")), Array("NAME:At- tributes", "Name:=", _ "EquationCurve2", "Flags:=", "", "Color:=", "(143 175 143)", "Transparency:=", _ 0, "PartCoordinateSystem:=", "Global", "UDMId:=", "", "MaterialValue:=", _</pre>

	<pre>"" & Chr(34) & "copper" & Chr(34) & "", "SurfaceMaterialValue:=", "" & Chr(34) & "" & Chr(34) & "", "SolveInside:=", _ false, "ShellElement:=", false, "ShellElementThickness:=", "0mm", "IsMa- terialEditable:=", _ true, "UseMaterialAppearance:=", false, "IsLightweight:=", false)</pre>

CreateEquationSurface

Creates an equation-based surface.

UI Access	Draw > Equation-Based Surface.		
Parameters	Name	Type	Description
	<Parameters>	Array	Structured array. <pre>Array("NAME:EquationBasedSurfaceParameters", "XuvFunction:=" , <string equation containing Func- tion, Operators and/or quantities _u, _v, or PI>, "YuvFunction:=" , <string equation containing Func- tion, Operators and/or quantities _u, _v, or PI>, "ZuvFunction:=" , <string equation containing Func- tion, Operators and/or quantities _u, _v, or PI>, "uStart:=" , <string>, "uEnd:=" , <string>, "vStart:=" , <string>, "vEnd:=" , <string>, "Version:=" , <integer>)</pre>

	<table><tr><td><AttributesArray></td><td>Array</td><td>Structured array. See: AttributesArray.</td></tr></table>			<AttributesArray>	Array	Structured array. See: AttributesArray .
<AttributesArray>	Array	Structured array. See: AttributesArray .				
Return Value	None.					

Python Syntax	CreateEquationSurface(<Parameters>, <Attributes>)
Python Example	<pre>oEditor.CreateEquationSurface(["NAME:EquationBasedSurfaceParameters", "XuvFunction:=" , "_u", "YuvFunction:=" , "_v", "ZuvFunction:=" , "sin(_u)+cos(_v)", "uStart:=" , "1", "uEnd:=" , "10", "vStart:=" , "1", "vEnd:=" , "10", "Version:=" , 1], ["NAME:Attributes", "Name:=" , "EquationSurface1", "Flags:=" , "", "Color:=" , "(143 175 143)", "Transparency:=" , 0, "PartCoordinateSystem:=" , "Global",</pre>

	<pre> "UDMId:=" , "", "MaterialValue:=" , "\"copper\"", "SurfaceMaterialValue:=", "\"\"", "SolveInside:=" , False, "ShellElement:=" , False, "ShellElementThickness:=", "0mm", "IsMaterialEditable:=" , True, "UseMaterialAppearance:=", False, "IsLightweight:=" , False] </pre>

VB Syntax	CreateEquationSurface <Parameters>, <Attributes>
VB Example	<pre> oEditor.CreateEquationSurface Array("NAME:EquationBasedSurfaceParameters", "XuvFunction:=", _ "_u", "YuvFunction:=", "_v", "ZuvFunction:=", "sin(_u)+cos(_v)", "uStart:=", "1", "uEnd:=", _ "10", "vStart:=", "1", "vEnd:=", "10", "Version:=", 1), Array("NAME:Attributes", "Name:=", _ "EquationSurface2", "Flags:=", "", "Color:=", "(143 175 143)", "Transparency:=", _ 0, "PartCoordinateSystem:=", "Global", "UDMId:=", "", "MaterialValue:=", _ "" & Chr(34) & "copper" & Chr(34) & "", "SurfaceMaterialValue:=", "" & Chr(34) & "" & </pre>

	<pre>Chr(34) & "", "SolveInside:=", _ false, "ShellElement:=", false, "ShellElementThickness:=", "0mm", "IsMa- terialEditable:=", _ true, "UseMaterialAppearance:=", false, "IsLightweight:=", false)</pre>

CreateHelix

Creates a helix based on a sweep of specified objects.

UI Access	Draw > Helix.		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: SelectionsArray .
	<Parameters>	Array	Structured array. Array("NAME:HelixParameters", "XCenter:=" , <string>, "YCenter:=" , <string>, "ZCenter:=" , <string>, "XStartDir:=" , <string>, "YStartDir:=" , <string>, "ZStartDir:=" , <string>, "NumThread:=" , <string>, "RightHand:=" , <boolean>, "RadiusIncrement:=" , <string>, "Thread:=" , <string>)

Return Value	None.

Python Syntax	CreateHelix(<SelectionsArray>, <Parameters>)
Python Example	<pre>oEditor.CreateHelix(["NAME:Selections", "Selections:=" , "EquationSurface2", "NewPartsModelFlag:=" , "Model"], ["NAME:HelixParameters", "XCenter:=" , "10000mm", "YCenter:=" , "40000mm", "ZCenter:=" , "0mm", "XStartDir:=" , "0mm", "YStartDir:=" , "10000mm", "ZStartDir:=" , "0mm", "NumThread:=" , "1", "RightHand:=" , True, "RadiusIncrement:=" , "0mm", "Thread:=" , "1mm"])</pre>

VB Syntax	CreateHelix <SelectionsArray>, <Parameters>
VB Example	<pre>oEditor.CreateHelix Array("NAME:Selections", "Selections:=", "EquationSurface1", "NewPartsModelFlag:=", _ "Model"), Array("NAME:HelixParameters", "XCenter:=", "1000mm", "YCenter:=", "10000mm", "ZCenter:=", _ "2.39945573144418mm", "XStartDir:=", "-41000mm", "YStartDir:=", "-10000mm", "ZStartDir:=", _ "-2.39945573144418mm", "NumThread:=", "1", "RightHand:=", true, "RadiusIncrement:=", _ "0mm", "Thread:=", "1mm")</pre>

CreatePoint

Creates a point.

UI Access	Draw > Point.		
Parameters	Name	Type	Description
	<Parameters>	Array	Structured array. Array("NAME:PointParameters", "PointX:=", <value>, "PointY:=", <value>, "PointZ:=", <value>)
	<AttributesArray>	Array	Structured array. See: AttributesArray . CreatePoint takes only the Name and Color attributes.
Return Value	None.		

Python Syntax	CreatePoint(<Parameters>, <Attributes>)
Python Example	<pre> oEditor.CreatePoint(["NAME:PointParameters", "PointX:=" , "0.2mm", "PointY:=" , "-0.2mm", "PointZ:=" , "0mm"], ["NAME:Attributes", "Name:=" , "Point1", "Color:=" , "(143 175 143) "]) </pre>

VB Syntax	CreatePoint <Parameters>, <Attributes>
VB Example	<pre> oEditor.CreatePoint Array("NAME:PointParameters", </pre>

	<pre>"PointX:=" , "0.2mm", "PointY:=" , "-0.2mm", "PointZ:=" , "0mm") , Array("NAME:Attributes", "Name:=" , "Point1", "Color:=" , "(143 175 143)")</pre>

CreatePolyline

Creates a polyline.

UI Access	Draw > Line.		
Parameters	Name	Type	Description
	<Parameters>	Array	Structured array. 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>)
	<AttributesArray>	Array	Structured array. See: AttributesArray .
Return Value	None.		

Python Syntax	CreatePolyline(<Parameters>, <Attributes>)
Python Example	<pre>oEditor.CreatePolyline(["NAME:PolylineParameters", "IsPolylineCovered:=" , True, "IsPolylineClosed:=" , False, ["NAME:PolylinePoints", ["NAME:PLPoint", "X:=" , "20000mm", "Y:=" , "-20000mm",</pre>

```
"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:="             , "",

```

	<pre> "MaterialValue:=" , "\"copper\"", "SurfaceMaterialValue:=", "\"\"", "SolveInside:=" , False, "ShellElement:=" , False, "ShellElementThickness:=", "0mm", "IsMaterialEditable:=" , True, "UseMaterialAppearance:=", False, "IsLightweight:=" , False]) </pre>

VB Syntax	CreatePolyline <Parameters>, <Attributes>
VB Example	<pre> oEditor.CreatePolyline Array("NAME:PolylineParameters", "IsPolylineCovered:=", true, "IsPolylineClosed:=", _ false, Array("NAME:PolylinePoints", Array("NAME:PLPoint", "X:=", "40000mm", "Y:=", _ "50000mm", "Z:=", "0mm"), Array("NAME:PLPoint", "X:=", "-150000mm", "Y:=", "-50000mm", "Z:=", _ "0mm")), Array("NAME:PolylineSegments", Array("NAME:PLSegment", "SegmentType:=", "Line", "StartIndex:=", _ 0, "NoOfPoints:=", 2)), Array("NAME:PolylineXSection", "XSectionType:=", "None", "XSec- tionOrient:=", _ "Auto", "XSectionWidth:=", "0mm", "XSectionTopWidth:=", "0mm", "XSectionHeight:=", _ "0mm", "XSectionNumSegments:=", "0", "XSectionBendType:=", "Corner")), Array("NAME:At- tributes", "Name:=", _ </pre>

	<pre>"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)</pre>

CreateRectangle

Creates a rectangle.

UI Access	Draw > Rectangle.		
Parameters	Name	Type	Description
	<Parameters>	Array	Structured array. <pre>Array("NAME:RectangleParameters", "IsCovered:=" , <boolean>, "XStart:=" , <string>, "YStart:=" , <string>, "ZStart:=" , <string>, "Width:=" , <string>, "Height:=" , <string>, "WhichAxis:=" , <string "X", "Y", or "Z">)</pre>

	<AttributesArray>	Array	Structured array. See: AttributesArray .
Return Value	None.		

Python Syntax	CreateRectangle(<Parameters>, <Attributes>)
Python Example	<pre> oEditor.CreateRectangle(["NAME:RectangleParameters", "IsCovered:=" , True, "XStart:=" , "-80000mm", "YStart:=" , "-90000mm", "ZStart:=" , "0mm", "Width:=" , "20000mm", "Height:=" , "30000mm", "WhichAxis:=" , "Z"], ["NAME:Attributes", "Name:=" , "Rectangle1", "Flags:=" , "", "Color:=" , "(143 175 143)", "Transparency:=" , 0, "PartCoordinateSystem:=" , "Global", "UDMId:=" , "", </pre>

	<pre> "MaterialValue:=" , "\"copper\"", "SurfaceMaterialValue:=", "\"\"", "SolveInside:=" , False, "ShellElement:=" , False, "ShellElementThickness:=", "0mm", "IsMaterialEditable:=" , True, "UseMaterialAppearance:=", False, "IsLightweight:=" , False]) </pre>

VB Syntax	CreateRectangle <Parameters>, <Attributes>
VB Example	<pre> oEditor.CreateRectangle Array("NAME:RectangleParameters", "IsCovered:=", true, "XStart:=", _ "10000mm", "YStart:=", "-40000mm", "ZStart:=", "0mm", "Width:=", "40000mm", "Height:=", _ "10000mm", "WhichAxis:=", "Z"), Array("NAME:Attributes", "Name:=", "Rectangle2", "Flags:=", _ "", "Color:=", "(143 175 143)", "Transparency:=", 0, "PartCoordinateSystem:=", _ "Global", "UDMId:=", "", "MaterialValue:=", "" & Chr(34) & "copper" & Chr(34) & "", "SurfaceMaterialValue:=", _ "" & Chr(34) & "" & Chr(34) & "", "SolveInside:=", false, "ShellElement:=", _ </pre>

	<pre>false, "ShellElementThickness:=", "0mm", "IsMaterialEditable:=", true, "UseMa- terialAppearance:=", _ false, "IsLightweight:=", false)</pre>

CreateRegularPolygon

Creates a regular polygon.

UI Access	Draw > Regular Polygon.		
Parameters	Name	Type	Description
	<Parameters>	Array	Structured array. Array("NAME:RegularPolygonParameters", "IsCovered:=" , <boolean>, "XCenter:=" , <string>, "YCenter:=" , <string>, "ZCenter:=" , <string>, "XStart:=" , <string>, "YStart:=" , <string>, "ZStart:=" , <string>, "NumSides:=" , <string containing number greater than 2>, "WhichAxis:=" , <string "X", "Y", or "Z">)
	<AttributesArray>	Array	Structured array. See: AttributesArray .
Return Value	None.		

Python Syntax	CreateRegularPolygon(<Parameters>, <Attributes>)
Python Example	<pre>oEditor.CreateRegularPolygon(["NAME:RegularPolygonParameters", "IsCovered:=" , True, "XCenter:=" , "-70000mm", "YCenter:=" , "-100000mm", "ZCenter:=" , "0mm", "XStart:=" , "-50000mm", "YStart:=" , "-80000mm", "ZStart:=" , "0mm", "NumSides:=" , "12", "WhichAxis:=" , "Z"], ["NAME:Attributes", "Name:=" , "Polygon1", "Flags:=" , "", "Color:=" , "(143 175 143)", "Transparency:=" , 0, "PartCoordinateSystem:=" , "Global", "UDMId:=" , ""</pre>

	<pre> "MaterialValue:=" , "\"copper\"", "SurfaceMaterialValue:=", "\"\"", "SolveInside:=" , False, "ShellElement:=" , False, "ShellElementThickness:=", "0mm", "IsMaterialEditable:=" , True, "UseMaterialAppearance:=", False, "IsLightweight:=" , False]) </pre>

VB Syntax	CreateRegularPolygon <Parameters>, <Attributes>
VB Example	<pre> oEditor.CreateRegularPolygon Array("NAME:RegularPolygonParameters", "IsCovered:=", _ true, "XCenter:=", "-60000mm", "YCenter:=", "40000mm", "ZCenter:=", "0mm", "XStart:=", _ - "-50000mm", "YStart:=", "50000mm", "ZStart:=", "0mm", "NumSides:=", "8", "WhichAxis:=", _ - "Z"), Array("NAME:Attributes", "Name:=", "Polygon2", "Flags:=", "", "Color:=", _ "(143 175 143)", "Transparency:=", 0, "PartCoordinateSystem:=", "Global", "UDMId:=", _ "", "MaterialValue:=", "" & Chr(34) & "copper" & Chr(34) & "", "Sur- faceMaterialValue:=", _ "" & Chr(34) & "" & Chr(34) & "", "SolveInside:=", false, "ShellElement:=", _ false, "ShellElementThickness:=", "0mm", "IsMaterialEditable:=", true, "UseMa- </pre>

```
terialAppearance:=", _  
false, "IsLightweight:=", false)
```

CreateRegularPolyhedron

Creates a regular polyhedron.

UI Access	Draw > Regular Polyhedron.		
Parameters	Name	Type	Description
	<Parameters>	Array	Structured array. Array("NAME:PolyhedronParameters", "XCenter:=" , <string>, "YCenter:=" , <string>, "ZCenter:=" , <string>, "XStart:=" , <string>, "YStart:=" , <string>, "ZStart:=" , <string>, "Height:=" , <string>, "NumSides:=" , <string containing number greater than 2>, "WhichAxis:=" , <string "X", "Y", or "Z">)
	<AttributesArray>	Array	Structured array. See: AttributesArray .
Return Value	None.		

Python Syntax	CreateRegularPolyhedron(<Parameters>, <Attributes>)
Python Example	<pre>oEditor.CreateRegularPolyhedron(["NAME:PolyhedronParameters", "XCenter:=" , "40000mm", "YCenter:=" , "-80000mm", "ZCenter:=" , "0mm", "XStart:=" , "50000mm", "YStart:=" , "-70000mm", "ZStart:=" , "0mm", "Height:=" , "50000mm", "NumSides:=" , "8", "WhichAxis:=" , "Z"], ["NAME:Attributes", "Name:=" , "RegularPolyhedron1", "Flags:=" , "", "Color:=" , "(143 175 143)", "Transparency:=" , 0, "PartCoordinateSystem:=" , "Global", "UDMId:=" , ""</pre>

	<pre> "MaterialValue:=" , "\"copper\"", "SurfaceMaterialValue:=", "\"\"", "SolveInside:=" , False, "ShellElement:=" , False, "ShellElementThickness:=", "0mm", "IsMaterialEditable:=" , True, "UseMaterialAppearance:=", False, "IsLightweight:=" , False l) </pre>

VB Syntax	CreateRegularPolyhedron <Parameters>, <Attributes>
VB Example	<pre> oEditor.CreateRegularPolyhedron Array("NAME:PolyhedronParameters", "XCenter:=", _ "-10000mm", "YCenter:=", "-50000mm", "ZCenter:=", "0mm", "XStart:=", "0mm", "YStart:=", _ "-50000mm", "ZStart:=", "0mm", "Height:=", "90000mm", "NumSides:=", "8", "WhichAxis:=", _ "Z"), Array("NAME:Attributes", "Name:=", "RegularPolyhedron2", "Flags:=", "", "Col- or:=", _ "(143 175 143)", "Transparency:=", 0, "PartCoordinateSystem:=", "Global", "UDMId:=", _ "", "MaterialValue:=", "" & Chr(34) & "copper" & Chr(34) & "", "Sur- faceMaterialValue:=", _ </pre>

	<pre>"" & Chr(34) & "" & Chr(34) & "", "SolveInside:=", false, "ShellElement:=", _ false, "ShellElementThickness:=", "0mm", "IsMaterialEditable:=", true, "UseMa- terialAppearance:=", _ false, "IsLightweight:=", false)</pre>

CreateSphere

Creates a sphere.

UI Access	Draw > Sphere.		
Parameters	Name	Type	Description
	<Parameters>	Array	Structured array. Array("NAME:SphereParameters", "XCenter:=", <string>, "YCenter:=", <string>, "ZCenter:=", <string>, "Radius:=", <string>)
	<AttributesArray>	Array	Structured array. See: AttributesArray .
Return Value	None.		

Python Syntax	CreateSphere(<Parameters>, <Attributes>)
Python Example	<pre>oEditor.CreateSphere(["NAME:SphereParameters", "XCenter:=" , "-40000mm",</pre>

```

"YCenter:="          , "-130000mm",
"ZCenter:="          , "0mm",
"Radius:="           , "22360.6797749979mm"
],
["NAME:Attributes",
  "Name:="            , "Sphere1",
  "Flags:="           , "",
  "Color:="           , "(143 175 143)",
  "Transparency:="    , 0,
  "PartCoordinateSystem:=", "Global",
  "UDMId:="           , "",
  "MaterialValue:="    , "\"copper\"",
  "SurfaceMaterialValue:=", "\"\"",
  "SolveInside:="      , False,
  "ShellElement:="     , False,
  "ShellElementThickness:=", "0mm",
  "IsMaterialEditable:=" , True,
  "UseMaterialAppearance:=", False,
  "IsLightweight:="    , False
])

```

VB Syntax	CreateSphere <Parameters>, <Attributes>
VB Example	<pre>oEditor.CreateSphere Array("NAME:SphereParameters", "XCenter:=", "-120000mm", "YCenter:=", _ "90000mm", "ZCenter:=", "0mm", "Radius:=", "10000mm"), Array("NAME:Attributes", "Name:=", _ "Sphere2", "Flags:=", "", "Color:=", "(143 175 143)", "Transparency:=", 0, "PartCoordinateSystem:=", _ "Global", "UDMId:=", "", "MaterialValue:=", "" & Chr(34) & "copper" & Chr(34) & "", "SurfaceMaterialValue:=", _ "" & Chr(34) & "" & Chr(34) & "", "SolveInside:=", false, "ShellElement:=", _ false, "ShellElementThickness:=", "0mm", "IsMaterialEditable:=", true, "UseMaterialAppearance:=", _ false, "IsLightweight:=", false)</pre>

CreateSpiral

Creates a spiral by sweeping the specified object(s).

UI Access	Draw > Spiral.		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: SelectionsArray .
	<Parameters>	Array	Structured array.
			<pre>Array("NAME:SpiralParameters", "XCenter:=" , <string>, "YCenter:=" , <string>,</pre>

			<code>"ZCenter:=" , <string>, "YStartDir:=" , <string>, "ZStartDir:=" , <string>, "NumThread:=" , <string>, "RightHand:=" , <boolean>, "RadiusIncrement:=" , <string>)</code>
Return Value	None.		

Python Syntax	<code>CreateSpiral(<Parameters>, <Attributes>)</code>
Python Example	<pre>oEditor.CreateSpiral(["NAME:Selections", "Selections:=" , "Polygon2", "NewPartsModelFlag:=" , "Model"], ["NAME:SpiralParameters", "XCenter:=" , "-70000mm", "YCenter:=" , "50000mm", "ZCenter:=" , "0mm", "XStartDir:=" , "-60000mm", "YStartDir:=" , "-10000mm",</pre>

	<pre>"ZStartDir:=" , "0mm", "NumThread:=" , "1", "RightHand:=" , True, "RadiusIncrement:=" , "1mm"])</pre>

VB Syntax	CreateSpiral <Parameters>, <Attributes>
VB Example	<pre>oEditor.CreateSpiral Array("NAME:Selections", "Selections:=", "Rectangle2", "NewPartsModelFlag:=", _ "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	Name	Type	Description
	<Parameters>	Array	Structured array. Array("NAME:TorusParameters", "XCenter:=" , <string>,

			"YCenter:=" , <string>, "ZCenter:=" , <string>, "MajorRadius:=" , <string>, "MinorRadius:=" , <string>, "WhichAxis:=" , <string "X", "Y", or "Z">)
	<AttributesArray>	Array	Structured array. See: AttributesArray .
Return Value	None.		

Python Syntax	CreateTorus(<Parameters>, <Attributes>)
Python Example	<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",</pre>

	<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] </pre>

VB Syntax	CreateTorus <Parameters>, <Attributes>
VB Example	<pre> oEditor.CreateTorus Array("NAME:TorusParameters", "XCenter:=", "0.6mm", "YCenter:=", _ "-2mm", "ZCenter:=", "0mm", "MajorRadius:=", "0.365028153987288mm", "MinorRadius:=", _ "0.0821854415126694mm", "WhichAxis:=", "Z"), Array("NAME:Attributes", "Name:=", _ "Torus2", "Flags:=", "", "Color:=", "(143 175 143)", "Transparency:=", 0, "PartCoordin- ateSystem:=", _ </pre>

	<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)</pre>

CreateUserDefinedModel

Creates a user-defined model.

Note: This option can be used to create a UDM from a Discovery model (**Modeler** > [Discovery Link](#)).

UI Access	Draw > User-Defined Model > [Model]		
Parameters	Name	Type	Description
	<Parameters>	Array	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: <pre>Array("NAME:GeometryParameters", Array("NAME:UDMParam", "Name:=" , <string>, "Value:=" , <string>, "PropType2:=" , <integer>, "PropFlag2:=" , <integer>))</pre> 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 • 8 – Hidden 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. These values are further described in the <code>UserDefinedPrimitiveStructures.h</code> file included with the installation under "...\\ANSYS Inc\\v251\\AnsysEM\\UserDefinedPrimitives\\Examples\\Headers"
Return Value	None.	

Python Syntax	CreateUserDefinedModel(<Parameters>)
Python Example	

VB Syntax	CreateUserDefinedModel <Parameters>
VB Example	

Python Example of Creating a UDM from a [Discovery](#) Model

```
oEditor.CreateUserDefinedModel(
    [
        "NAME:UserDefinedModelParameters",
        [
            "NAME:Definition",
            [
                "NAME:UDMParam",
```

```

        "Name:=" , "GeometryFilePath",
        "Value:=" , "\"C:/Users/chenwan/Desktop/restored_files/tee.dsco\"",
        "DataType:=" , "String",
        "PropType2:=" , 0,
        "PropFlag2:=" , 1
    ],
    [
        "NAME:UDMParam",
        "Name:=" , "ProcessID",
        "Value:=" , "255988",
        "DataType:=" , "Int",
        "PropType2:=" , 2,
        "PropFlag2:=" , 8
    ]
],
[
    "NAME:Options",
    [
        "NAME:UDMParam",
        "Name:=" , "IsDiscoveryLink",
        "Value:=" , "1",
        "DataType:=" , "Int",
        "PropType2:=" , 5,
        "PropFlag2:=" , 8
    ],
    [
        "NAME:UDMParam",
        "Name:=" , "Solid Bodies",
        "Value:=" , "1",
        "DataType:=" , "Int",
        "PropType2:=" , 5,
        "PropFlag2:=" , 0
    ],
    [
        "NAME:UDMParam",
        "Name:=" , "Surface Bodies",
        "Value:=" , "1",
        "DataType:=" , "Int",

```



```
    "PropType2:=" , 5,
    "PropFlag2:=" , 0
  ],
  [
    "NAME:UDMParam",
    "Name:=" , "Parameters",
    "Value:=" , "\"eCADImportParameterType_Independent,eCADImportParameterType_None,eCADIm-
portParameterType_Independent,eCADImportParameterType_All\"",
    "DataType:=" , "String",
    "PropType2:=" , 1,
    "PropFlag2:=" , 0
  ],
  [
    "NAME:UDMParam",
    "Name:=" , "Parameter Key",
    "Value:=" , "\"\"",
    "DataType:=" , "String",
    "PropType2:=" , 0,
    "PropFlag2:=" , 0
  ],
  [
    "NAME:UDMParam",
    "Name:=" , "Named Selections",
    "Value:=" , "1",
    "DataType:=" , "Int",
    "PropType2:=" , 5,
    "PropFlag2:=" , 8
  ],
  [
    "NAME:UDMParam",
    "Name:=" , "Materials",
    "Value:=" , "\"None,None,Assignments,Assignments and Properties\"",
    "DataType:=" , "String",
    "PropType2:=" , 1,
    "PropFlag2:=" , 0
  ],
],
```

```
[
  "NAME:UDMParam",
  "Name:=" , "Import As Lightweight",
  "Value:=" , "0",
  "DataType:=" , "Int",
  "PropType2:=" , 5,
  "PropFlag2:=" , 0
],
[
  "NAME:UDMParam",
  "Name:=" , "Facet Level",
  "Value:=" , "\"3,1,2,3,4,5\"",
  "DataType:=" , "String",
  "PropType2:=" , 1,
  "PropFlag2:=" , 0
],
[
  "NAME:UDMParam",
  "Name:=" , "Cleaning",
  "Value:=" , "0",
  "DataType:=" , "Int",
  "PropType2:=" , 5,
  "PropFlag2:=" , 0
],
[
  "NAME:UDMParam",
  "Name:=" , "Use Parasolid For Transfer",
  "Value:=" , "1",
  "DataType:=" , "Int",
  "PropType2:=" , 5,
  "PropFlag2:=" , 8
],
[
  "NAME:UDMParam",
  "Name:=" , "Tolerant Stitching",
  "Value:=" , "1",
  "DataType:=" , "Int",
  "PropType2:=" , 5,
```

```
    "PropFlag2:=" , 8
  ],
  [
    "NAME:UDMParam",
    "Name:=" , "Stitch Tolerance",
    "Value:=" , "0.1",
    "PropType2:=" , 3,
    "PropFlag2:=" , 8
  ],
  [
    "NAME:UDMParam",
    "Name:=" , "Tighten Gaps",
    "Value:=" , "1",
    "DataType:=" , "Int",
    "PropType2:=" , 5,
    "PropFlag2:=" , 8
  ],
  [
    "NAME:UDMParam",
    "Name:=" , "Tighten Gaps Tolerance",
    "Value:=" , "1e-06",
    "PropType2:=" , 3,
    "PropFlag2:=" , 8
  ],
  [
    "NAME:UDMParam",
    "Name:=" , "Use Associativity",
    "Value:=" , "1",
    "DataType:=" , "Int",
    "PropType2:=" , 5,
    "PropFlag2:=" , 8
  ],
  [
    "NAME:UDMParam",
    "Name:=" , "Attributes",
    "Value:=" , "1",
```

```

    "DataType:=" , "Int",
    "PropType2:=" , 5,
    "PropFlag2:=" , 8
  ],
  [
    "NAME:UDMParam",
    "Name:=" , "Import Coordinate Systems",
    "Value:=" , "1",
    "DataType:=" , "Int",
    "PropType2:=" , 5,
    "PropFlag2:=" , 8
  ],
  [
    "NAME:UDMParam",
    "Name:=" , "Decompose Disjoint Faces",
    "Value:=" , "1",
    "DataType:=" , "Int",
    "PropType2:=" , 5,
    "PropFlag2:=" , 8
  ],
  [
    "NAME:UDMParam",
    "Name:=" , "Import Using Instances",
    "Value:=" , "1",
    "DataType:=" , "Int",
    "PropType2:=" , 5,
    "PropFlag2:=" , 8
  ],
  [
    "NAME:UDMParam",
    "Name:=" , "Line Bodies",
    "Value:=" , "1",
    "DataType:=" , "Int",
    "PropType2:=" , 5,
    "PropFlag2:=" , 8
  ],
  [
    "NAME:UDMParam",

```

```
"Name:=" , "Mixed Import Resolution",
"Value:=" , "\"eMixedImport_None\"",
"DataType:=" , "String",
"PropType2:=" , 0,
"PropFlag2:=" , 8
],
[
  "NAME:UDMParam",
  "Name:=" , "Enclosure and Symmetry Processing",
  "Value:=" , "1",
  "DataType:=" , "Int",
  "PropType2:=" , 5,
  "PropFlag2:=" , 8
],
[
  "NAME:UDMParam",
  "Name:=" , "Reader Mode Saves Updated File",
  "Value:=" , "0",
  "DataType:=" , "Int",
  "PropType2:=" , 5,
  "PropFlag2:=" , 8
],
[
  "NAME:UDMParam",
  "Name:=" , "Smart CAD Update",
  "Value:=" , "0",
  "DataType:=" , "Int",
  "PropType2:=" , 5,
  "PropFlag2:=" , 8
],
[
  "NAME:UDMParam",
  "Name:=" , "Import Work Points",
  "Value:=" , "1",
  "DataType:=" , "Int",
  "PropType2:=" , 5,
```

```
        "PropFlag2:=" , 8
    ],
    [
        "NAME:GeometryParams"
    ],
    "DllName:=" , "SACADIntegUDM",
    "Library:=" , "installLib",
    "Version:=" , "1.0",
    "ConnectionID:=" , ""
])
```

CreateUserDefinedPart

Creates a user-defined part.

UI Access	Draw > User-Defined Primitive > [Part]		
Parameters	Name	Type	Description
	<Parameters>	Array	Structured array. Array("NAME:UserDefinedPrimitiveParameters", "DllName:=" , <string>, "Version:=" , <string>, "NoOfParameters:=" , <integer>, "Library:=" , <string>, <paramVectorArray>)
	<paramVectorArray>	Array	Structured array containing arrays for each pair: Array("NAME:ParamVector", <pair>, <pair>, <pair>,...)

	<code><pair></code>	Array	Structured array: Array("NAME:Pair", "Name:=" , <string>, "Value:=" , <string>)
	<code><Attributes></code>	Array	Structured array. See: AttributesArray .
Return Value	None.		

Python Syntax	CreateUserDefinedPart(<Parameters>, <Attributes>)
Python Example	<pre>oEditor.CreateUserDefinedPart(["NAME:UserDefinedPrimitiveParameters", "DllName:=" , "RMxpert/LapCoil.dll", "Version:=" , "16.0", "NoOfParameters:=" , 22, "Library:=" , "syslib", ["NAME:ParamVector", ["NAME:Pair", "Name:=" , "DiaGap", "Value:=" , "100mm"], ["NAME:Pair",</pre>

```
"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",
```


	<pre> "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", </pre>

```
"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"]
```

	<pre>]]], ["NAME:Attributes", "Name:=" , "LapCoil1", "Flags:=" , "", "Color:=" , "(143 175 143)", "Transparency:=" , 0, "PartCoordinateSystem:=", "Global", "UDMId:=" , "", "MaterialValue:=" , "\"copper\"", "SurfaceMaterialValue:=", "\"\"", "SolveInside:=" , False, "ShellElement:=" , False, "ShellElementThickness:=", "0mm", "IsMaterialEditable:=" , True, "UseMaterialAppearance:=", False, "IsLightweight:=" , False])</pre>

VB Syntax	CreateUserDefinedPart < <i>Parameters</i> >, < <i>Attributes</i> >

**VB
Example**

```

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:=", _
"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 by saving it to a new name so that its properties are accessible. To change the properties on an existing component without changing its name, see [ChangeProperty](#).

UI Access	N/A		
Parameters	Name	Type	Description
	<compName>	String	Component name.
	<Parameters>	Array	Structured array. Array("NAME:EditComponentParametersData", "NewComponentName:=", <string>, "GeometryParameters:=", <string>, "MaterialParameters:=", <string>, "DesignParameters:=", <string>, <ComponentMeshing>, <Excitations>)
	<ComponentMeshing>	Array	Structured array. Array("NAME:Component Meshing", "MeshAssembly:=", <boolean>)
	<Excitations>	Array	Structured array containing array of suppressed excitations. Array("NAME:Excitations", "Suppressed:=", <array>)
Return Value	None.		

Python Syntax	Edit3DComponent (<compName>, <Parameters>)
Python Example	<pre>oEditor.Edit3DComponent("Connector1", ["NAME:EditComponentParametersData", "NewComponentName:=", "Connector2", "GeometryParameters:=", "", "MaterialParameters:=", "", "DesignParameters:=", "", ["NAME:Component Meshing", "MeshAssembly:=", False], ["NAME:Excitations", "Suppressed:=", []]])</pre>
VB Syntax	Edit3DComponent <compName>, <Parameters>
VB Example	<pre>oEditor.Edit3DComponent "Connector1", Array("NAME:EditComponentParametersData", "NewComponentName:=", "Connector2",</pre>

	<pre>"GeometryParameters:=", "", "MaterialParameters:=", "", "DesignParameters:=", "", Array("NAME:Component Meshing", "MeshAssembly:=", false), Array("NAME:Excitations", "Suppressed:=", Array()))</pre>

Edit3DComponentDefinition

Edits definitions of a specified 3D component.

UI Access	N/A		
Parameters	Name	Type	Description
	<Parameters>	Array	Structured array. Array("NAME:EditComponentParametersData", "OriginalComponentName:=", <string>, "NewComponentName:=", <string>, "GeometryParameters:=", <string>, "MaterialParameters:=", <string>, "DesignParameters:=", <string>, <ComponentMeshing>, <Excitations>)
	<ComponentMeshing>	Array	Structured array.

			Array("NAME:Component Meshing", "MeshAssembly:=", <boolean>)
	<Excitations>	Array	Structured array containing array of suppressed excitations. Array("NAME:Excitations", "Suppressed:=", <array>)
Return Value	None.		
Python Syntax	Edit3DComponent (<Parameters>)		
Python Example	<pre>oEditor.Edit3DComponent(["NAME:EditComponentParametersData", "OriginalComponentName:=", "Connector1", "NewComponentName:=", "Connector2", "GeometryParameters:=", "", "MaterialParameters:=", "", "DesignParameters:=", "", ["NAME:Component Meshing", "MeshAssembly:=", False], ["NAME:Excitations", "Suppressed:=", []]])</pre>		

VB Syntax	Edit3DComponent <Parameters>
VB Example	<pre> oEditor.Edit3DComponent Array("NAME:EditComponentParametersData", "OriginalComponentName:=", "Connector1", "NewComponentName:=", "Connector2", "GeometryParameters:=", "", "MaterialParameters:=", "", "DesignParameters:=", "", Array("NAME:Component Meshing", "MeshAssembly:=", false), Array("NAME:Excitations", "Suppressed:=", Array())) </pre>

EditNativeComponentDefinition [Beta – Layout Components in Mechanical]

Use this command to edit the definition of an existing Layout Component in a Mechanical design. Layout components are supported for steady-state thermal, transient thermal, and structural solution types.

UI Access	In the Project Manager, expand 3D Components , right-click on the component, and select Edit Definition..		
Parameters	Name	Type	Description
	<LinkedParameters>	Array	Structured array containing data of inserted layout component
Return Value	None		

Python Syntax	EditNativeComponentDefinition (<LinkedParameters>)
Python Example Thermal Solution (with a custom grid resolution)	<pre>oEditor.EditNativeComponentDefinition(["NAME:EditNativeComponentDefinitionData", "DefinitionName:=" , "LC1", ["NAME:GeometryDefinitionParameters", ["NAME:VariableOrders"]], ["NAME:DesignDefinitionParameters", ["NAME:VariableOrders"]], ["NAME:MaterialDefinitionParameters", ["NAME:VariableOrders"]], "NextUniqueID:=" , 0, "MoveBackwards:=" , False, "DatasetType:=" , "ComponentDatasetType",])</pre>

```

[
  "NAME:DatasetDefinitions"
],
[
  "NAME:NativeComponentDefinitionProvider",
  "Type:="                , "Layout Component",
  "Unit:="                , "mm",
  "Version:="             , 1.1,
  "EDBDefinition:="       , "RPS14_3",
  [
    "NAME:VariableMap"
  ],
  "ReferenceCS:="         , "",
  "CSToImport:="         , [],
  "BoardCutoutMaterial:=" , "air",
  "ViaHoleMaterial:="     , "FR4_epoxy",
  "ExtentsType:="        , "Polygon",
  "CustomResolution:="    , True, # Use custom grid resolution:
  "CustomResolutionCol:=" , 1350, # Number of columns (X-resolution)
  "CustomResolutionRow:=" , 375, # Number of rows (Y-resolution)
  "UseThermalLink:="     , False,
  [
    "NAME:TopBoundary",
    "BoundaryType:="     , "Convection",
    "FilmCoefficient:="   , "20w_per_m2kel",
    "ReferenceTemperature:=", "AmbientTemp"
    "UseRadiation:="     , True,
    "ViewFactor:="       , 1,
    "Emissivity:="       , 0.8,
    "RadiationTemperature:=", "AmbientTemp"
  ],
  [
    "NAME:BottomBoundary",
    "BoundaryType:="     , "Temperature",

```

Python Example Structural Solution (with resolution set by the slider and with imported temperatures)	<pre>oEditor.EditNativeComponentDefinition(["NAME:InsertNativeComponentData", "DefinitionName:=" , "LC1", ["NAME:GeometryDefinitionParameters", ["NAME:VariableOrders"]], ["NAME:DesignDefinitionParameters", ["NAME:VariableOrders"]], ["NAME:MaterialDefinitionParameters", ["NAME:VariableOrders"]], "NextUniqueID:=" , 0, "MoveBackwards:=" , False, "DatasetType:=" , "ComponentDatasetType", ["NAME:DatasetDefinitions"],],)</pre>

```

[
  "NAME:NativeComponentDefinitionProvider",
  "Type:="                                , "Layout Component",
  "Unit:="                                , "mm",
  "Version:="                             , 1.1,
  "EDBDefinition:="                       , "TraceMappingTest_EMDesign2",
  [
    "NAME:VariableMap"
  ],
  "ReferenceCS:="                         , "",
  "BoardCutoutMaterial:="                 , "air", # 400, or 800, respectively,
along the short dimension
  "ViaHoleMaterial:="                     , "FR4_epoxy",
  "ExtentsType:="                         , "Polygon",
  "CSToImport:="                          , [],
  "CustomResolution:="                    , False, # <--- Not custom; use resolution
slider:
  "SliderResolution:="                    , 3, # <--- Tick mark (0, 1, 2, 3, or 4)
represents 50, 100, 200
  [
    "NAME:TopBoundary",
    "BoundaryType:="                       , "Force",
    "ForceX:="                             , "2.5newton",
    "ForceY:="                             , "-4newton",
    "ForceZ:="                             , "0newton"
  ],
  [
    "NAME:BottomBoundary",
    "BoundaryType:="                       , "FrictionLessSupport"
  ],
  "EnableThermalCondtion:="               , True, #
  "BoundaryType:="                       , "ThermalCondtion", #
  "Uniform:="                             , False, #

```

	<pre>lativeTo:=" , "TargetProject" #], "ComponentName:=" , "LC1", "Company:=" , "", "Company URL:=" , "", "Model Number:=" , "", "Help URL:=" , "", "Version:=" , "1.1", "Notes:=" , "", "IconType:=" , "Layout Component"])</pre>

EditPolyline

Modifies a specified polyline. See: [CreatePolyline](#).

UI Access	N/A		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: SelectionsArray .
	<Parameters>	Array	Structured array. 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>)
Return Value	None.		

Python Syntax	EditPolyline(<SelectionsArray>, <Parameters>)
Python Example	<pre>oEditor.EditPolyline(["NAME:Selections", "Selections:=", "Polyline1"] ["NAME:PolylineParameters", "IsPolylineCovered:=" , True, "IsPolylineClosed:=" , False,</pre>


```
[ "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"
]]
)

```

VB
Syn-

EditPolyline <SelectionsArray>, <Parameters>

tax	
VB Exa- mple	<pre> oEditor.EditPolyline Array("NAME:Selections", "Selections:=", "Polyline1") Array ("NAME:PolylineParameters", "IsPolylineCovered:=", true, "IsPolylineClosed:=", _ false, Array("NAME:PolylinePoints", Array("NAME:PLPoint", "X:=", "40000mm", "Y:=", _ "50000mm", "Z:=", "0mm"), Array("NAME:PLPoint", "X:=", "-150000mm", "Y:=", "-50000mm", "Z:=", _ "0mm")), Array("NAME:PolylineSegments", Array("NAME:PLSegment", "SegmentType:=", "Line", "StartIndex:=", _ 0, "NoOfPoints:=", 2)), Array("NAME:PolylineXSection", "XSectionType:=", "None", "XSec- tionOrient:=", _ "Auto", "XSectionWidth:=", "0mm", "XSectionTopWidth:=", "0mm", "XSectionHeight:=", _ "0mm", "XSectionNumSegments:=", "0", "XSectionBendType:=", "Corner")), Array("NAME:At- tributes", "Name:=", _ "Polyline2", "Flags:=", "", "Color:=", "(143 175 143)", "Transparency:=", 0, "PartCoordin- ateSystem:=", _ "Global", "UDMId:=", "", "MaterialValue:=", "" & Chr(34) & "copper" & Chr(34) & "", "Sur- faceMaterialValue:=", _ "" & Chr(34) & "" & Chr(34) & "", "SolveInside:=", false, "ShellElement:=", _ false, "ShellElementThickness:=", "0mm", "IsMaterialEditable:=", true, "UseMa- terialAppearance:=", _ false, "IsLightweight:=", false) </pre>

Get3DComponentDefinitionNames

Gets names of 3D component definitions.

UI Access	N/A
Parameters	None.
Return Value	Array of strings containing component definition names.

Python Syntax	Get3DComponentDefinitionNames()
Python Example	<code>oEditor.Get3DComponentDefinitionNames()</code>

VB Syntax	Get3DComponentDefinitionNames()
VB Example	<pre>Dim defNames defNames = oEditor.Get3DComponentDefinitionNames()</pre>

Get3DComponentInstanceNames

Returns instance names of 3D component definitions.

UI Access	N/A		
Parameters	Name	Type	Description
	<DefinitionName>	String	Definition name.
Return Value	Array containing instance names.		

Python Syntax	Get3DComponentInstanceNames(<DefinitionName>)
Python Example	<code>oEditor.Get3DComponentInstanceNames("Connector")</code>

VB Syntax	Get3DComponentInstanceNames <DefinitionName>
VB Example	oEditor.Get3DComponentInstanceNames "Connector"

Get3DComponentMaterialNames

Returns material names for a specified *.a3dcomp format 3D component.

UI Access	N/A		
Parameters	Name	Type	Description
	<InstanceName>	String	Component instance name.
Return Value	Array containing material names.		

Python Syntax	Get3DComponentMaterialNames(<InstanceName>)
Python Example	oEditor.Get3DComponentMaterialNames ("Connector1.a3dcomp")

VB Syntax	Get3DComponentMaterialNames <InstanceName>
VB Example	oEditor.Get3DComponentMaterialNames "Connector1.a3dcomp"

Get3DComponentMaterialProperties

Returns material properties for a specified 3D component.

UI Access	N/A

Parameters	Name	Type	Description
	<MaterialName>	String	Material name.
Return Value	Array containing material properties.		

Python Syntax	Get3DComponentMaterialProperties(<MaterialName>)
Python Example	<code>oEditor.Get3DComponentMaterialProperties('Connector1:Material01')</code>

VB Syntax	Get3DComponentMaterialProperties <MaterialName>
VB Example	<code>oEditor.Get3DComponentMaterialProperties "Connector1:Material01"</code>

Get3DComponentParameters

Returns parameters for a specified 3D component.

UI Access	N/A		
Parameters	Name	Type	Description
	<compName>	String	3D component name.
Return Value	Array containing component parameters.		

Python Syntax	Get3DComponentParameters(<compName>)
Python Example	<code>oEditor.Get3DComponentParameters('Connector')</code>

VB Syntax	Get3DComponentParameters <compName>

VB Example	<code>oEditor.Get3DComponentParameters "Connector"</code>

Get3DComponentPartNames

Returns part names for a specified 3D component.

UI Access	N/A		
Parameters	Name	Type	Description
	<InstanceName>	String	3D component instance.
Return Value	Array containing part names.		

Python Syntax	<code>Get3DComponentParameters(<InstanceName>)</code>
Python Example	<code>oEditor.Get3DComponentPartNames('Connector')</code>

VB Syntax	<code>Get3DComponentParameters <InstanceName></code>
VB Example	<code>oEditor.Get3DComponentPartNames "Connector"</code>

Insert3DComponent

Inserts a 3D component.

UI Access	Draw > 3D Component Library > Browse > [Component].		
Parameters	Name	Type	Description
	<ComponentData>	Array	Structured array. <code>Array("NAME:InsertComponentData",</code>

			"Parameters:=", <string>, "TargetCS:=", <string>, "ComponentFile:=", <string filepath>
Return Value	None.		

Python Syntax	Insert3DComponent(<ComponentData>)
Python Example	<pre>oEditor.Insert3DComponent(["NAME:InsertComponentData", "Parameters:=", "", "TargetCS:=", "Global", "ComponentFile:=", "C:\tmp\Connector.a3dcomp"])</pre>

VB Syntax	Insert3DComponent <ComponentData>
VB Example	<pre>oEditor.Insert3DComponent Array("NAME:InsertComponentData", "Parameters:=", "", "TargetCS:=", "Global", "ComponentFile:=", "C:\tmp\Connector.a3dcomp")</pre>

InsertComponent

Inserts a component.

UI Access	N/A		
Parameters	Name	Type	Description
	< <i>ComponentData</i> >	Array	Structured array. Array("NAME:InsertComponentData", "Parameters:=", <string>, "TargetCS:=", <string>, "ComponentFile:=", <string filepath>)
Return Value	None.		

Python Syntax	InsertComponent(< <i>ComponentData</i> >)
Python Example	<pre>oEditor.InsertComponent(["NAME:InsertComponentData", "Parameters:=", "", "TargetCS:=", "Global", "ComponentFile:=", "C:\tmp\Connector.a3dcomp"])</pre>

VB Syntax	InsertComponent < <i>ComponentData</i> >
VB Example	<pre>oEditor.InsertComponent Array("NAME:InsertComponentData", "Parameters:=", "",</pre>

	<pre>"TargetCS:=", "Global", "ComponentFile:=", "C:\tmp\Connector.a3dcomp")</pre>

InsertNativeComponent [Beta – Layout Component in Mechanical]

Use this command to insert a Layout Component into a Mechanical design. Layout components are supported for steady-state thermal, transient thermal, and structural solution types.

Note:

When the layout component being inserted into the Mechanical design is not already present in the project, the *oEditor.InsertNativeComponent* command must be preceded by the [oComponentManager.Add](#) command to add the component to the project's *Components* Library. (See the **Thermal Solution** Python example below.

Similarly, when the source design contains geometry variables that you want to map to local mechanical design variables or AEDT project variables, one of the following commands must precede *oEditor.InsertNativeComponent*, depending on the preferred local variable type:

- *oDesign.ChangeProperty* (see the **Thermal Solution** Python example below)
- *oProject.ChangeProperty* (see the **Structural Solution** Python example below)

UI Access	In the Project Manager, right-click 3D Components and select Browse Layout Component .		
Parameters	Name	Type	Description
	<LinkedParameters>	Array	Structured array containing data of inserted layout component. See Python examples for details. Most parameters are self-explanatory.
	The following partial list includes parameters that need further explanation:		
	<MapInstanceParameters>	String	Local variable type: "DesignVariable" or "ProjectVariable"

			This parameter is applicable when source geometry variables are mapped to layout component instance parameters and you have chosen to create local design or project variables of the instance parameters.
	<CustomResolution>	Bool	Use slider position for trace mapping grid resolution when False Specify X & Y grid divisions numerically when True
	<UseThermalLink>	Bool	Import power dissipation from a source DCIR analysis into the layout component (Applicable to <i>Steady-State Thermal</i> and <i>Transient Thermal Mechanical</i> solutions) When the source HFSS 3D Layout includes a DCIR analysis , setting this option to True causes power dissipated as heat (determined by the source DCIR solution) to be imported into the target Mechanical layout component.
	<SliderResolution>	Integer	Slider position (tick mark number) used when <CustomResolution> = False (Controls resolution along the shorter board dimension: Values 0–4 represent resolutions of 50, 100, 200, 400, and 800, respectively)
	<CustomResolutionCol>	Integer	Number of columns (X resolution) when <CustomResolution> = True
	<CustomResolutionRow>	Integer	Number of rows (Y resolution) when <CustomResolution> = True
Return Value	None		

Python Syntax	InsertNativeComponent (<LinkedParameters>)

**Python
Example
Thermal
Solution**
(with
a
cus-
tom
grid
res-
olu-
tion)

```
# Add an HFSS 3D Layout component (from TMT_03.aedb) to the current project:
```

```
oComponentManager.Add(
[
    "NAME:TMT_03",
    "Info:=",
    [
        "Type:="                , 29,
        "NumTerminals:="        , 0,
        "DataSource:="          , "",
        "ModifiedOn:="          , 1711743004,
        "Manufacturer:="        , "",
        "Symbol:="              , "TMT_03",
        "ModelNames:="          , "",
        "Footprint:="           , "",
        "Description:="         , "",
        "InfoTopic:="           , "",
        "InfoHelpFile:="        , "",
        "IconFile:="            , "BlueDot.bmp",
        "Library:="             , "",
        "OriginalLocation:="     , "Project",
        "IEEE:="                , "",
        "Author:="              , "",
        "OriginalAuthor:="       , "",
        "CreationDate:="         , 1711743004,
        "ExampleFile:="         , "",
        "HiddenComponent:="     , 0,
```

```

        "CircuitEnv:="          , 0,
        "GroupID:="            , 0
    ],
    "CircuitEnv:="              , 0,
    "Refbase:="                 , "U",
    "NumParts:="                , 1,
    "ModSinceLib:="             , True,
    "CompExtID:="               , 9,
    "ModelEDBFilePath:="        , "D:\\Ansys\\2024R2\\TMT_03.aedb",
    "EDBCompPassword:="        , ""
])

```

Add a **design** variable (*L1_Thickness*) for the mapped source parameter used to control the thickness of the top PCB layer:

```

oDesign = oProject.SetActiveDesign("MechanicalDesign1")
oDesign.ChangeProperty(
[
    "NAME:AllTabs",
    [
        "NAME:LocalVariableTab",
        [
            "NAME:PropServers",
            "LocalVariables"
        ],
        "NAME:NewProps",

```

```

        [
            "NAME:L1_Thickness",
            "PropType:=", "VariableProp",
            "UserDef:=" , True,
            "Value:="    , "4mm"
        ]
    ]
]
])

# Add the layout component to the current mechanical design - thermal solution:

oEditor.InsertNativeComponent(
    [
        "NAME:InsertNativeComponentData",
        "TargetCS:="                , "Global",
        "SubmodelDefinitionName:="  , "LC1",
        [
            "NAME:ComponentPriorityLists"
        ],
        "NextUniqueID:="            , 0,
        "MoveBackwards:="           , False,
        "DatasetType:="             , "ComponentDatasetType",
        [
            "NAME:DatasetDefinitions"
        ],
        [
            "NAME:BasicComponentInfo",
            "ComponentName:="        , "LC1",
            "Company:="              , "",

```

```

"NAME:VariableOrders"
]
],
[
  "NAME:DesignDefinitionParameters",
  [
    "NAME:VariableOrders"
  ]
],
[
  "NAME:MaterialDefinitionParameters",
  [
    "NAME:VariableOrders"
  ]
],
"DefReferenceCSID:=" , 1,
"MapInstanceParameters:=" , "DesignVariable",
"UniqueDefinitionIdentifier:=", "13d6e53d-2f8c-4d0f-86ce-26b2583e17e4",
"OriginFilePath:=" , "",
"IsLocal:=" , False,
"ChecksumString:=" , "",
"ChecksumHistory:=" , [],
"VersionHistory:=" , [],
[
  "NAME:NativeComponentDefinitionProvider",
  "Type:=" , "Layout Component",
  "Unit:=" , "mm",
  "Version:=" , 1.1,
  "EDBDefinition:=" , "TMT_03",
  [
    "NAME:VariableMap"
  ]
]

```

Python Example Structural Solution (with resolution set by the slider and with imported temperatures)	<pre># Add a project variable (\$L1_Thickness) for the mapped source parameter used to control the thickness of the top PCB layer: # Note: When specifying the variable name in the UI (<i>Browse Layout Component: Variable Mapping</i>), do not include the dollar sign (\$) prefix. It will be added automatically to the beginning of the name you specify for project variables. oProject = oDesktop.SetActiveProject("InstanceVariationTest") oProject.ChangeProperty(["NAME:AllTabs", ["NAME:ProjectVariableTab", ["NAME:PropServers", "ProjectVariables"], ["NAME:NewProps", ["NAME:\$L1_Thickness", "PropType:=", "UserDef:=", "Value:="], ["VariableProp", True, "4mm"]]]])</pre>


```
# Add the layout component to the current mechanical design - structural solution:
```

```
oDesign = oProject.SetActiveDesign("MechanicalDesign2")
oEditor = oDesign.SetActiveEditor("3D Modeler")
oEditor.InsertNativeComponent(
    [
        "NAME:InsertNativeComponentData",
        "TargetCS:="                , "Global",
        "SubmodelDefinitionName:="  , "LC1",
        [
            "NAME:ComponentPriorityLists"
        ],
        "NextUniqueID:="            , 0,
        "MoveBackwards:="           , False,
        "DatasetType:="              , "ComponentDatasetType",
        [
            "NAME:DatasetDefinitions"
        ],
        [
            "NAME:BasicComponentInfo",
            "ComponentName:="        , "LC1",
            "Company:="               , "",
            "Company URL:="           , "",
            "Model Number:="          , "",
            "Help URL:="              , "",
            "Version:="               , "1.0",
            "Notes:="                 , "",

```

```

"",
  "IsLocal:=" , False,
  "ChecksumString:=" , "",
  "ChecksumHistory:=" , [],
  "VersionHistory:=" , [],
  [
    "NAME:NativeComponentDefinitionProvider",
    "Type:=" , "Layout Component",
    "Unit:=" , "mm",
    "Version:=" , 1.1,
    "EDBDefinition:=" , "TMT_03",
    [
      "NAME:VariableMap"
    ],
    "ReferenceCS:=" , "",
    "CSToImport:=" , [],
    "CustomResolution:=" , False, # Slider controls resolution
    "SliderResolution:=" , 2, # Slider tick mark number
    "BoardCutoutMaterial:=" , "air",
    "ViaHoleMaterial:=" , "FR4_epoxy",
    "ExtentsType:=" , "BoundingBox",
    [
      "NAME:TopBoundary",
      "BoundaryType:=" , "Force",
      "ForceX:=" , "2.5newton",
      "ForceY:=" , "-4newton",
      "ForceZ:=" , "0newton"
    ],
    [
      "NAME:BottomBoundary",
      "BoundaryType:=" , "FrictionLessSupport"
    ],
    "EnableThermalCondtion:=" , True, #
  ]

```


InsertPolylineSegment

Inserts a polyline segment before or after a specified existing segment.

UI Access	Draw > Line Segment > [Selection].		
Parameters	Name	Type	Description
	<Parameters>	Array	Structured array. 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: CreatePolyline .
Return Value	None.		

Python Syntax	InsertPolylineSegment(<Parameters>)
Python Example	<pre>oEditor.InsertPolylineSegment(["NAME:Insert Polyline Segment",</pre>

```

"Selections:="          , "Polyline1:CreatePolyline:1",
"Segment Indices:="     , [0],
"At Start:="            , True,
"SegmentType:="         , "Line",
["NAME:PolylinePoints",
  ["NAME:PLPoint",
    "X:="                , "1.1mm",
    "Y:="                , "0.8mm",
    "Z:="                , "0mm"
  ],
  ["NAME:PLPoint",
    "X:="                , "0.6mm",
    "Y:="                , "-0.8mm",
    "Z:="                , "0mm"
  ]
]
]
]
)

```

VB Syntax	InsertPolylineSegment < <i>Parameters</i> >
VB Example	oEditor.InsertPolylineSegment Array("NAME:Insert Polyline Segment", "Selections:=", _ "Polyline1:CreatePolyline:1", "Segment Indices:=", Array(1), "At Start:=", _

	<pre> false, "SegmentType:=", "Spline", Array("NAME:PolylinePoints", Array("NAME:PLPoint", "X:=", _ "-1.4mm", "Y:=", "0.4mm", "Z:=", "0mm"), Array("NAME:PLPoint", "X:=", "0.5mm", "Y:=", _ "1.1mm", "Z:=", "0mm"), Array("NAME:PLPoint", "X:=", "0.7mm", "Y:=", "-2.1mm", "Z:=", _ "0mm"), Array("NAME:PLPoint", "X:=", "0.4mm", "Y:=", "-1.1mm", "Z:=", "0mm"))) </pre>

SweepAlongPath

Sweeps the specified 1D or 2D parts along a path. The last 1D object specified is the path for the sweep.

UI Access	Draw > Sweep > Along Path.		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: SelectionsArray .
	<PathSweepParametersArray>	Array	<pre> Array("NAME:PathSweepParameters", "DraftAngle:=", <value>, "DraftType:=", <string>, "CheckFaceFaceIntersection:=", <bool>, "TwistAngle:=", <value>) </pre> Possible values for DraftType are "Extended", "Round", and "Natural".
Return Value	None.		

Python Syntax	SweepAlongPath(<SelectionsArray>, <PathSweepParametersArray>)
Python Example	<pre> oEditor.SweepAlongPath(["NAME:Selections", "Selections:=" , "Rectangle1,Polyline1", "NewPartsModelFlag:=" , "Model"], ["NAME:PathSweepParameters", "DraftAngle:=" , "0deg", "DraftType:=" , "Round", "CheckFaceFaceIntersection:=", False, "TwistAngle:=" , "0deg"]) </pre>

VB Syntax	SweepAlongPath <SelectionsArray>, <PathSweepParametersArray>
VB Example	<pre> oEditor.SweepAlongPath Array("NAME:Selections", "Selections:=", "Polygon1,Polyline1"),_ Array("NAME:PathSweepParameters", _ </pre>

	<pre>"DraftAngle:=", "0deg",_ "DraftType:=", "Round",_ "CheckFaceFaceIntersection:=", False,_ "TwistAngle:=", "30deg")</pre>

SweepAlongVector

Sweeps the specified 1D or 2D parts along a vector.

UI Access	Draw > Sweep > Along Vector.		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: SelectionsArray .
	<VecSweepParametersArray>	Array	<pre>Array("NAME:VectorSweepParameters", "DraftAngle:=", <value>, "DraftType:=", <string>, "CheckFaceFaceIntersection:=", <bool>, "SweepVectorX:=", <value> "SweepVectorY:=", <value> "SweepVectorZ:=", <value>)</pre> Possible values for DraftType are "Extended", "Round", and "Natural".
Return Value	None.		

Python Syntax	SweepAlongVector(<SelectionsArray>, <VecSweepParametersArray>)
Python Example	<pre> oEditor.SweepAlongVector(["NAME:Selections", "Selections:=" , "Rectangle1", "NewPartsModelFlag:=" , "Model"], ["NAME:VectorSweepParameters", "DraftAngle:=" , "0deg", "DraftType:=" , "Round", "CheckFaceFaceIntersection:=", False, "SweepVectorX:=" , "0mm" "SweepVectorY:=" , "0mm" "SweepVectorZ:=" , "12mm"]) </pre>

VB Syntax	SweepAlongVector <SelectionsArray>, <VecSweepParametersArray>
VB Example	<pre> oEditor.SweepAlongPath Array("NAME:Selections",_ </pre>

	<pre>"Selections:=", "Rectangle1",_ "NewPartsModelFlag:=" , "Model") Array("NAME:VectorSweepParameters", _ "DraftAngle:=", "0deg",_ "DraftType:=", "Round",_ "CheckFaceFaceIntersection:=", False,_ "SweepVectorX:=" , "0mm",_ "SweepVectorY:=" , "0mm",_ "SweepVectorZ:=" , "12mm")</pre>

SweepAroundAxis

Sweeps the specified 1D or 2D parts around an axis.

UI Access	Draw > Sweep > Around Axis.		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: SelectionsArray .
	<AxisSweepParametersArray>	Array	Array("NAME:AxisSweepParameters",
			"DraftAngle:=", <value>, "DraftType:=", <string>, "CheckFaceFaceIntersection:=", <bool>, "SweepAxis:=", <value> "SweepAngle:=", <value>

	"NumOfSegments:=", <value>) Possible values for DraftType are "Extended", "Round", and "Natural". Possible values for SweepAxis are "X", "Y", and "Z".
Return Value	None.

Python Syntax	SweepAroundAxis(<SelectionsArray>, <AxisSweepParametersArray>)
Python Example	<pre>oEditor.SweepAroundAxis(["NAME:Selections", "Selections:=" , "Rectangle1", "NewPartsModelFlag:=" , "Model"], ["NAME:AxisSweepParameters", "DraftAngle:=" , "0deg", "DraftType:=" , "Round", "CheckFaceFaceIntersection:=", False, "SweepAxis:=" , "X" "SweepAngle:=" , "360deg" "NumOfSegments:=" , "12"</pre>

	l)

VB Syntax	SweepAroundAxis <SelectionsArray>, <AxisSweepParametersArray>
VB Example	<pre>oEditor.SweepAroundAxis Array("NAME:Selections",_ "Selections:=", "Rectangle1",_ "NewPartsModelFlag:=" , "Model") Array("NAME:AxisSweepParameters", _ "DraftAngle:=", "0deg",_ "DraftType:=", "Round",_ "CheckFaceFaceIntersection:=", False,_ "SweepAxis:=" , "X",_ "SweepAngle:=" , "360deg",_ "NumOfSegments:=" , "12")</pre>

SweepFacesAlongNormal

Sweep the specified face(s) along normal.

UI Access	Modeler > Surface > Sweep Faces Along Normal		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: SelectionsArray .

	<code><parameters></code>	Array	Structured array. <pre>Array("NAME:Parameters", "NAME:SweepFaceAlongNormalToParameters", "FacesToDetach:=", <faceIDarray>, "LengthOfSweep:=", "<value><units>")</pre>
Return Value	None		

Python Syntax	<code>SweepFacesAlongNormal(<SelectionsArray> <parameters>)</code>
Python Example	<pre>oEditor.SweepFacesAlongNormal(["NAME:Selections", "Selections:=", "Rectangle1", "NewPartsModelFlag:=", "Model"], ["NAME:Parameters", "NAME:SweepFaceAlongNormalToParameters", "FacesToDetach:=", [183], "LengthOfSweep:=", "0.1mm"])</pre>

VB Syntax	<code>SweepFacesAlongNormal <SelectionsArray> <parameters></code>
VB Example	<code>oEditor.SweepFacesAlongNormal</code>

	<pre>Array("NAME:Selections", "Selections:=", "Rectangle1", "NewPartsModelFlag:=", "Model"), Array("NAME:Parameters", "NAME:SweepFaceAlongNormalToParameters", "FacesToDetach:=", Array(57), "LengthOfSweep:=", "0.5mm"))</pre>

SweepFacesAlongNormalWithAttributes

Sweep a face along normal, and specify attributes of the new object.

UI Access	Modeler > Surface > Sweep Faces Along Normal		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: SelectionsArray .
	<parameters>	Array	Array("NAME:Parameters", "NAME:SweepFaceAlongNormalToParameters", "FacesToDetach:=", <faceIDarray>, "LengthOfSweep:=", "<value><units>")
	<AttributesArray>	Array	Structured array. See: AttributesArray .
Return Value	None		

Python Syntax	SweepFacesAlongNormalWithAttributes(<SelectionsArray>, <parameters>, <AttributesArray>)
Python Example	<pre> oEditor.SweepFacesAlongNormalWithAttributes(["NAME:Selections", "Selections:=", "Rectangle1", "NewPartsModelFlag:=", "Model"], ["NAME:Parameters", "NAME:SweepFaceAlongNormalToParameters", "FacesToDetach:=", [183], "LengthOfSweep:=", "0.1mm"], ["NAME:Attributes", "Name:=", "Box3", "Flags:=", "", "Color:=", "(143 175 143)", "Transparency:=", 0, "PartCoordinateSystem:=", "Global", "UDMId:=", "", "MaterialValue:=", "\"copper\"", "SurfaceMaterialValue:=", "\"\"", "SolveInside:=", False, "ShellElement:=", False, "ShellElementThickness:=", "0mm", </pre>

	<pre>"IsMaterialEditable:=", True, "UseMaterialAppearance:=", False, "IsLightweight:=", False])</pre>

VB Syntax	SweepFacesAlongNormalWithAttributes <SelectionsArray>, <parameters>, <AttributesArray>
VB Example	<pre>oEditor.SweepFacesAlongNormalWithAttributes Array("NAME:Selections", "Selections:=", "Rectangle1", "NewPartsModelFlag:=", "Model"), Array("NAME:Parameters", "NAME:SweepFaceAlongNormalToParameters", "FacesToDetach:=", Array(57), "LengthOfSweep:=", "0.5mm"), Array("NAME:Attributes", "Name:=", "Box3", "Flags:=", "", "Color:=", "(143 175 143)", "Transparency:=", 0, "PartCoordinateSystem:=", "Global",</pre>

	<pre>"UDMId:=" , "", "MaterialValue:=" , "\"copper\"", "SurfaceMaterialValue:=", "\"\"", "SolveInside:=" , false, "ShellElement:=" , false, "ShellElementThickness:=", "0mm", "IsMaterialEditable:=" , true, "UseMaterialAppearance:=", false, "IsLightweight:=" , false))</pre>

UpdateComponentDefinition

Updates a 3D component's definition.

UI Access	Draw > 3D Component Library > Definitions.		
Parameters	Name	Type	Description
	<data>	Array	Structured array. Array("NAME:UpdateDefinitionData", "DefinitionNames:=", <string>, "Passwords:=", <array of strings>)
Return Value	None.		
Python Syntax	UpdateComponentDefinition(<data>)		

Python Example	<pre>oEditor.UpdateComponentDefinition(['NAME:UpdateDefinitionData', 'DefinitionNames:=', 'Connector, Magic_Tee', 'Passwords:=', ['', '']])</pre>

VB Syntax	UpdateComponentDefinition <data>
VB Example	<pre>oEditor.UpdateComponentDefinition Array("NAME:UpdateDefinitionData", "DefinitionNames:=", " Connector, Magic_Tee", "Passwords:=", Array("", ""))</pre>

Edit Menu Commands

[Copy](#)

[DeletePolylinePoint](#)

[DuplicateAlongLine](#)

[DuplicateAroundAxis](#)

[DuplicateMirror](#)

[Mirror](#)

[Move](#)

[OffsetFaces](#)[Paste](#)[Rotate](#)[Scale](#)

Copy

Copies specified part(s) to the clipboard.

UI Access	N/A		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: SelectionsArray .
Return Value	None.		

Python Syntax	Copy(<SelectionsArray>)		
Python Example	<pre>oEditor.Copy(["NAME:Selections", "Selections:=", "Box1"])</pre>		

VB Syntax	Copy <SelectionsArray>		
VB Example	<pre>oEditor.Copy Array("NAME:Selections", "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		
Parameters	Name	Type	Description
	<DeletePointArray>	Array	Structured array. Array("NAME:Delete Point", "Selections:=", <string "<PolylineName>:<PolylineAction>:<int>">, "Segment Index:=", <integer>, "At Start:=", <bool True for start point; False for end point>)
Return Value	None.		

Python Syntax	DeletePolylinePoint (<DeletePointArray>)
Python Example	<pre>oEditor.DeletePolylinePoint(["NAME:Delete Point", "Selections:=", "Polyline1:CreatePolyline:1", "Segment Index:=", 1, "At Start:=", True])</pre>

VB Syntax	DeletePolylinePoint <DeletePointArray>

VB Example	<pre>oEditor.DeletePolylinePoint Array("NAME>Delete Point", "Selections:=", "Polyline1>CreatePolyline:1", "Segment Index:=", 1, "At Start:=", True)</pre>

DuplicateAlongLine

Duplicates specified parts along a line.

UI Access	Edit > Duplicate > Along Line.		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. <pre>Array("NAME:Selections", "Selections:=" , <string>, "NewPartsModelFlag:=" , <string>)</pre>
	<ParametersArray>	Array	Structured array. <pre>Array("NAME:DuplicateToAlongLineParameters", "CreateNewObjects:=" , <boolean>, "XComponent:=" , <string>, "YComponent:=" , <string>, "ZComponent:=" , <string>, "NumClones:=" , <string containing number greater than 1>)</pre>
	<OptionsArray>	Array	Structured array. <pre>Array("NAME:Options", "DuplicateAssignments:=" , <boolean>)</pre>
	<CreateGroup>	Array	Optional. Structured array.

	Array("CreateGroupsForNewObjects:=", <boolean>)
Return Value	None.

Python Syntax	DuplicateAlongLine (<SelectionsArray>, <ParametersArray>, <OptionsArray>, <CreateGroup>)
Python Example	<pre>oEditor.DuplicateAlongLine(["NAME:Selections", "Selections:=" , "Box1", "NewPartsModelFlag:=" , "Model"], ["NAME:DuplicateToAlongLineParameters", "CreateNewObjects:=" , False, "XComponent:=" , "1mm", "YComponent:=" , "-0.7mm", "ZComponent:=" , "0mm", "NumClones:=" , "2"], ["NAME:Options", "DuplicateAssignments:=", False],)</pre>

	<pre>["CreateGroupsForNewObjects:=", False]</pre>

VB Syntax	DuplicateAlongLine <SelectionsArray>, <ParametersArray>, <OptionsArray>, <CreateGroup>
VB Example	<pre>oEditor.DuplicateAlongLine Array("NAME:Selections", "Selections:=", "Box1", "NewPartsModelFlag:=", "Model") Array("NAME:DuplicateToAlongLineParameters", "CreateNewObjects:=", false, "XComponent:=", "1mm", "YComponent:=", "-0.7mm", "ZComponent:=", "0mm", "NumClones:=", "2") Array("NAME:Options", "DuplicateAssignments:=", false) Array("CreateGroupsForNewObjects:=", false)</pre>

DuplicateAroundAxis

Duplicates specified parts around an axis.

UI Access	Edit > Duplicate > Around Axis.

Parameters	Name	Type	Description
	<code><SelectionsArray></code>	Array	Structured array. <pre>Array("NAME:Selections", "Selections:=" , <string>, "NewPartsModelFlag:=" , <string>)</pre>
	<code><ParametersArray></code>	Array	Structured array. <pre>Array("NAME:DuplicateAroundAxisParameters", "CreateNewObjects:=" , <boolean>, "WhichAxis:=" , <string>, "AngleStr:=" , <string>, "NumClones:=" , <string containing number greater than 1>)</pre>
	<code><OptionsArray></code>	Array	Structured array. <pre>Array("NAME:Options", "DuplicateAssignments:=" , <boolean>)</pre>
	<code><CreateGroup></code>	Array	Optional. Structured array. <pre>Array("CreateGroupsForNewObjects:=" , <boolean>)</pre>
Return Value	None.		

Python Syntax	<code>DuplicateAroundAxis (<SelectionsArray>, <ParametersArray>, <OptionsArray>, <CreateGroup>)</code>
Python Example	<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  
])
```

VB Syntax	DuplicateAroundAxis <SelectionsArray>, <ParametersArray>, <OptionsArray>, <CreateGroup>
VB Example	<pre>oEditor.DuplicateAroundAxis Array("NAME:Selections", "Selections:=", "Box1", "NewPartsModelFlag:=", "Model")</pre>

	<pre>Array("NAME:DuplicateAroundAxisParameters", "CreateNewObjects:=", true, "WhichAxis:=", "Z", "AngleStr:=", "90deg", "NumClones:=", "2") Array("NAME:Options", "DuplicateAssignments:=", false) Array("CreateGroupsForNewObjects:=", false)</pre>

DuplicateMirror

Duplicates specified parts according to a mirror plane.

UI Access	Edit > Duplicate > Mirror.		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. Array("NAME:Selections", "Selections:=" , <string>, "NewPartsModelFlag:=" , <string>)
	<ParametersArray>	Array	Structured array. Array("NAME:DuplicateToMirrorParameters", "DuplicateMirrorBaseX:=", <string>, "DuplicateMirrorBaseY:=", <string>,

			"DuplicateMirrorNormalX=", <string>, "DuplicateMirrorNormalY=", <string>, "DuplicateMirrorNormalZ=", <string>)
	<OptionsArray>	Array	Structured array. Array("NAME:Options", "DuplicateAssignments=", <boolean>)
	<CreateGroup>	Array	Optional. Structured array. Array("CreateGroupsForNewObjects=", <boolean>)
Return Value	None.		

Python Syntax	DuplicateMirror (<SelectionsArray>, <ParametersArray>, <OptionsArray>, <CreateGroup>)
Python Example	<pre> oEditor.DuplicateMirror(["NAME:Selections", "Selections:=", , "Box1", "NewPartsModelFlag:=", , "Model"], ["NAME:DuplicateToMirrorParameters", "DuplicateMirrorBaseX:=", "-0.4mm", "DuplicateMirrorBaseY:=", "-1.2mm", "DuplicateMirrorBaseZ:=", "0mm", "DuplicateMirrorNormalX:=", "0.124034734589208mm", </pre>

	<pre> "DuplicateMirrorNormalY:=", "0.992277876713668mm", "DuplicateMirrorNormalZ:=", "0mm"], ["NAME:Options", "DuplicateAssignments:=", False], ["CreateGroupsForNewObjects:=", False]) </pre>

VB Syntax	DuplicateMirror <SelectionsArray>, <ParametersArray>, <OptionsArray>, <CreateGroup>
VB Example	<pre> oEditor.DuplicateMirror Array("NAME:Selections", "Selections:=", "Box1", "NewPartsModelFlag:=", "Model") Array("NAME:DuplicateToMirrorParameters", "DuplicateMirrorBaseX:=", "-0.4mm", "DuplicateMirrorBaseY:=", "-1.2mm", "DuplicateMirrorBaseZ:=", "0mm", "DuplicateMirrorNormalX:=", "0.124034734589208mm", </pre>

	<pre>"DuplicateMirrorNormalY:=", "0.992277876713668mm", "DuplicateMirrorNormalZ:=", "0mm") Array("NAME:Options", "DuplicateAssignments:=", false) Array("CreateGroupsForNewObjects:=", false)</pre>

Mirror

Mirrors specified part(s).

UI Access	Edit > Arrange > Mirror.		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: SelectionsArray .
	<MirrorParameters>	Array	Structured array. <pre>Array("NAME:MirrorParameters", "MirrorBaseX:=" , <string>, "MirrorBaseY:=" , <string>, "MirrorBaseZ:=" , <string>, "MirrorNormalX:=" , <string>, "MirrorNormalY:=" , <string>, "MirrorNormalZ:=" , <string>)</pre>
Return Value	None.		

Python Syntax	Mirror(<SelectionsArray>, <MirrorParameters>)

Python Example	<pre>oEditor.Mirror(["NAME:Selections", "Selections:=", "Box1_1", "NewPartsModelFlag:=", "Model"], ["NAME:MirrorParameters", "MirrorBaseX:=", "-0.2mm", "MirrorBaseY:=", "-1.2mm", "MirrorBaseZ:=", "0mm", "MirrorNormalX:=", "-0.316227766016838mm", "MirrorNormalY:=", "0.948683298050514mm", "MirrorNormalZ:=", "0mm"])</pre>

VB Syntax	Mirror <SelectionsArray> <MirrorParameters>
VB Example	<pre>oEditor.Mirror Array("NAME:Selections", "Selections:=", "Box1_1", "NewPartsModelFlag:=", "Model") Array("NAME:MirrorParameters",</pre>

	<pre>"MirrorBaseX:=", "-0.2mm", "MirrorBaseY:=", "-1.2mm", "MirrorBaseZ:=", "0mm", "MirrorNormalX:=", "-0.316227766016838mm", "MirrorNormalY:=", "0.948683298050514mm", "MirrorNormalZ:=", "0mm")</pre>

Move

Moves specified part(s).

UI Access	Edit > Arrange > Move.		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: SelectionsArray .
	<TranslateParameters>	Array	Structured array. Array(["NAME:TranslateParameters", "TranslateVectorX:=" , <string>, "TranslateVectorY:=" , <string>, "TranslateVectorZ:=" , <string>)
Return Value	None.		

Python Syntax	Move(<SelectionsArray>, <TranslateParameters>)
Python Example	oEditor.Move(

	<pre>["NAME:Selections", "Selections:=" , "Box1_1", "NewPartsModelFlag:=" , "Model"], ["NAME:TranslateParameters", "TranslateVectorX:=" , "-0.5mm", "TranslateVectorY:=" , "0.1mm", "TranslateVectorZ:=" , "0mm"]</pre>

VB Syntax	Move <SelectionsArray> <TranslateParameters>
VB Example	<pre>oEditor.Move Array("NAME:Selections", "Selections:=", "Box1_1", "NewPartsModelFlag:=", "Model") Array("NAME:TranslateParameters", "TranslateVectorX:=", "-0.5mm", "TranslateVectorY:=", "0.1mm", "TranslateVectorZ:=", "0mm")</pre>

OffsetFaces

Offsets the faces of selected part(s).

UI Access	Edit > Arrange > Offset.		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: SelectionsArray .
	<OffsetParameters>	Array	Structured array. Array("NAME:OffsetParameters", "OffsetDistance:=" , <string>)
Return Value	None.		

Python Syntax	OffsetFaces (<SelectionsArray>, <OffsetParameters>)
Python Example	<pre>oEditor.OffsetFaces(["NAME:Selections", "Selections:=", "Box1_1", "NewPartsModelFlag:=", "Model"], ["NAME:OffsetParameters", "OffsetDistance:=", "16mm"])</pre>

VB Syntax	OffsetFaces <SelectionsArray> <OffsetParameters>

VB Example	<pre>oEditor.OffsetFaces Array("NAME:Selections", "Selections:=", "Box1_1", "NewPartsModelFlag:=", "Model") Array("NAME:OffsetParameters", "OffsetDistance:=", "16mm")</pre>

Paste (Model Editor)

Pastes previously copied object(s). See: [Copy](#).

UI Access	Edit > Paste.
Parameters	None.
Return Value	None.

Python Syntax	Paste()
Python Example	<pre>oEditor.Copy(["NAME:Selections", "Selections:=" , "Box1_2"]) oEditor.Paste()</pre>

VB Syntax	Paste
VB Example	<pre>oEditor.Copy Array("NAME:Selections", "Selections:=" , "Box1_2") oEditor.Paste</pre>

Rotate

Rotates specified object(s).

UI Access	Edit > Arrange > Rotate.		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: SelectionsArray .
	<RotateParameters>	Array	Structured array. <pre>Array("NAME:RotateParameters", "RotateAxis:=" , <string "X", "Y", or "Z">, "RotateAngle:=" , <string>)</pre>
Return Value	None.		

Python Syntax	Rotate(<SelectionsArray>, <RotateParameters>)
Python Example	<pre>oEditor.Rotate(["NAME:Selections", "Selections:=", "Box1_1", "NewPartsModelFlag:=", "Model"], ["NAME:RotateParameters",</pre>

	<pre>"RotateAxis:=", "Z", "RotateAngle:=", "90deg"])</pre>

VB Syntax	Rotate <SelectionsArray> <RotateParameters>
VB Example	<pre>oEditor.Rotate Array("NAME:Selections", "Selections:=", "Box1_1", "NewPartsModelFlag:=", "Model") Array("NAME:RotateParameters", "RotateAxis:=", "Z", "RotateAngle:=", "90deg")</pre>

Scale

Scales specified object(s).

UI Access	Edit > Scale.		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: SelectionsArray .
	<ScaleParameters>	Array	Structured array. Array("NAME:ScaleParameters", "ScaleX:=" , <string containing scale factor> ,

			<code>"ScaleY:=" , <string containing scale factor>, "ScaleZ:=" , <string containing scale factor>)</code>
Return Value	None.		

Python Syntax	Scale (<SelectionsArray>, <ScaleParameters>)
Python Example	<pre>oEditor.Scale(["NAME:Selections", "Selections:=", "Box1", "NewPartsModelFlag:=", "Model"], ["NAME:ScaleParameters", "ScaleX:=", "2", "ScaleY:=", "2", "ScaleZ:=", "2"])</pre>

VB Syntax	Scale <SelectionsArray> <ScaleParameters>
VB Example	<pre>oEditor.Scale Array("NAME:Selections", "Selections:=", "Box1", "NewPartsModelFlag:=", "Model")</pre>

	<pre>Array("NAME:ScaleParameters", "ScaleX:=", "2", "ScaleY:=", "2", "ScaleZ:=", "2")</pre>

Modeler Menu Commands

[AssignMaterial](#)

[Chamfer](#)

[Connect](#)

[CoverLines](#)

[CoverSurfaces](#)

[CreateEntityList](#)

[CreateFaceCS](#)

[CreateGroup](#)

[CreateObjectCS](#)

[CreateObjectFromEdges](#)

[CreateObjectFromFaces](#)

[CreateRelativeCS](#)

[DeleteEmptyGroups](#)

[DeleteLastOperation](#)

[DetachFaces](#)

[EditEntityList](#)

[EditFaceCS](#)

[EditObjectCS](#)

[EditRelativeCS](#)

[Export](#)

[ExportModelImageToFile](#)

[ExportModelMeshToFile](#)

[Fillet](#)

[FlattenGroup](#)

[Generate History](#)

[GetActiveCoordinateSystem](#)

[GetCoordinateSystems](#)

[HealObject](#)

[Import](#)

[ImportDXF](#)

[ImportGDSII \[Modeler\]](#)

[Intersect](#)

[MoveCStoEnd](#)

[MoveEntityToGroup](#)

[MoveFaces](#)

[ProjectSheet](#)

- [PurgeHistory](#)
- [ReplaceWith3DComponent](#)
- [Section](#)
- [SeparateBody](#)
- [SetModelUnits](#)
- [SetWCS](#)
- [ShowWindow](#)
- [Split](#)
- [Subtract](#)
- [SweepFacesAlongNormal](#)
- [ThickenSheet](#)
- [UncoverFaces](#)
- [Unite](#)
- [Ungroup](#)
- [WrapSheet](#)

AlignFaces

Aligns the adjacent selected faces of imported objects which have only one operation in their History Tree.

UI Access	Modeler > Model Preparation > Align Faces		
Parameters	Name	Type	Description
	<argFaceList>	Array	["NAME:<SpecifiedName>",

			"BaseFaces:=", <FaceNumberArray>, "SnapFaces:=", <FaceNumberArray>]
	<FaceNumberArray>	Array	Array of number represents specified faces.
Return Value	None.		

Python Syntax	AlignFaces(<argFaceList>)		
Python Example	<pre>oEditor.AlignFaces(["NAME:Entity List", "BaseFaces:=" , [9], "SnapFaces:=" , [18]])</pre>		

VB Syntax	AlignFaces <argFaceList>		
VB Example	<pre>oEditor.AlignFaces Array("NAME:Entity List", "BaseFaces:=" , [9], "SnapFaces:=" , [18])</pre>		

AssignMaterial

Assigns a material to specified object(s).

UI Access	Modeler > Assign Material.

Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: SelectionsArray .
	<AttributesArray>	Array	Structured array. See: AttributesArray . This script supports the following attributes: • MaterialValue • SolveInside • ShellElement • ShellElementThickness • IsMaterialEditable • UseMaterialAppearance • IsLightweight
Return Value	None.		
Python Syntax	AssignMaterial(<SelectionsArray>, <AttributesArray>)		

Python Example	<pre>oEditor.AssignMaterial(["NAME:Selections", "AllowRegionDependentPartSelectionForPMLCreation:=", True, "AllowRegionSelectionForPMLCreation:=", True, "Selections:=" , "Box1"], ["NAME:Attributes", "MaterialValue:=" , "diamond", "SolveInside:=" , False, "ShellElement:=" , False, "ShellElementThickness:=" , "nan", "IsMaterialEditable:=" , True, "UseMaterialAppearance:=" , False, "IsLightweight:=" , False])</pre>
VB Syntax	AssignMaterial <SelectionsArray> <AttributesArray>

VB Example	<pre>oEditor.AssignMaterial Array("NAME:Selections", "AllowRegionDependentPartSelectionForPMLCreation:=", true, "AllowRegionSelectionForPMLCreation:=", true, "Selections:=" , "Box1") Array("NAME:Attributes", "MaterialValue:=" , "diamond", "SolveInside:=" , false, "ShellElement:=" , false, "ShellElementThickness:=" , "nan", "IsMaterialEditable:=" , true, "UseMaterialAppearance:=" , false, "IsLightweight:=" , false)</pre>

Chamfer

Creates a chamfer.

UI Access	Modeler > Chamfer.		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: SelectionsArray .
	<Parameters>	Array	Structured array.

			<pre>Array("NAME:Parameters", Array("NAME:ChamferParameters", "Edges:=" , <array containing integer>, "Vertices:=" , <array>, "LeftDistance:=" , <string>, "RightDistance:=" , <string>, "ChamferType:=" , <string "Symmetric", "Left Distance-Angle", "Right Distance-Angle", or "Left Distance-Right Distance">))</pre>
Return Value	None.		

Python Syntax	Chamfer(<SelectionsArray>, <Parameters>)
Python Example	<pre>oEditor.Chamfer(["NAME:Selections", "Selections:=" , "Box2", "NewPartsModelFlag:=" , "Model"], ["NAME:Parameters", ["NAME:ChamferParameters", "Edges:=" , [42], "Vertices:=" , [],</pre>

	<pre> "LeftDistance:=" , "0.1mm", "RightDistance:=" , "0.1mm", "ChamferType:=" , "Symmetric"]] </pre>

VB Syntax	Chamfer <SelectionsArray> <Parameters>
VB Example	<pre> oEditor.Chamfer Array("NAME:Selections", "Selections:=" , "Box2", "NewPartsModelFlag:=" , "Model") Array("NAME:Parameters", Array("NAME:ChamferParameters", "Edges:=" , [42], "Vertices:=" , [], "LeftDistance:=" , "0.1mm", "RightDistance:=" , "0.1mm", "ChamferType:=" , "Symmetric")) </pre>

CleanUpModel

Cleans up history tree operations.

UI Access	Modeler > Cleanup Model History.
Parameters	None.
Return Value	None.

Python Syntax	<code>CleanUpModel()</code>
Python Example	<code>oEditor.CleanUpModel()</code>

VB Syntax	<code>CleanUpModel</code>
VB Example	<code>oEditor.CleanUpModel</code>

Connect

Connects two or more 1D polyline objects or 2D sheet objects.

UI Access	Modeler > Surface > Connect.		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: SelectionsArray .
Return Value	None.		

Python Syntax	<code>Connect(<SelectionsArray>)</code>
Python Example	<pre>oEditor.Connect (["NAME:Selections", "Selections:=", "Polyline2,Polyline1"])</pre>

VB Syntax	Connect <SelectionsArray>
VB Example	<pre>oEditor.Connect Array("NAME:Selections", "Selections:=", "Polyline2,Polyline1")</pre>

CoverLines

Covers two or more 1D objects to form a sheet.

UI Access	Modeler > Surface > Cover Lines.		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: SelectionsArray .
Return Value	None.		

Python Syntax	CoverLines(<SelectionsArray>)
Python Example	<pre>oEditor.CoverLines(["NAME:Selections", "Selections:=", "Polyline3,Polyline4", "NewPartsModelFlag:=", "Model"])</pre>

VB Syntax	CoverLines <SelectionsArray>

VB Example	<pre>oEditor.CoverLines Array("NAME:Selections", "Selections:=", "Polyline3,Polyline4", "NewPartsModelFlag:=", "Model")</pre>

CoverSurfaces

Covers two or more faces to form a solid object.

UI Access	Modeler > Surface > Cover Faces.		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: SelectionsArray .
Return Value	None.		

Python Syntax	CoverSurfaces (<SelectionsArray>)		
Python Example	<pre>oEditor.CoverSurfaces (["NAME:Selections", "Selections:=", "Obj1_Face1,Obj2_Face2", "NewPartsModelFlag:=", "Model"])</pre>		

VB Syntax	CoverSurfaces <SelectionsArray>		
VB Example	<pre>oEditor.CoverSurfaces Array("NAME:Selections",</pre>		

	<pre>"Selections:=", "Obj1_Face1,Obj2_Face2", "NewPartsModelFlag:=", "Model")</pre>

CreateEntityList

Creates a list of entities containing objects or faces (not both).

UI Access	Modeler > List > Create > [Object / Face] List.		
Parameters	Name	Type	Description
	<Parameters>	Array	Structured array. Array("NAME:GeometryEntityListParameters", "EntityType:=" , <string "Object" or "Face">, "EntityList:=" , <string of object names or IDs>) See GetObjectIDByName for returning object IDs.
	<AttributesArray>	Array	Structured array. See: AttributesArray . CreateEntityList takes only the "Name" parameter.
Return Value	None.		

Python Syntax	CreateEntityList(<Parameters>,<AttributesArray>)
Python Example	<pre>oEditor.CreateEntityList(["NAME:GeometryEntityListParameters", "EntityType:=", "Object",</pre>

	<pre> "EntityList:=", "Bondwire1,Bondwire2"], ["NAME:Attributes", "Name:=", "Objectlist1"]) </pre>

VB Syntax	CreateEntityList<Parameters> <AttributesArray>
VB Example	<pre> oEditor.CreateEntityList Array("NAME:GeometryEntityListParameters", "EntityType:=", "Object", "EntityList:=", "Bondwire1,Bondwire2") Array("NAME:Attributes", "Name:=", "Objectlist1") </pre>

CreateFaceCS

Creates a Face Coordinate System from a selected face.

UI Access	Modeler > Coordinate System > Create > FaceCS.		
Parameters	Name	Type	Description
	<Parameters>	Array	Structured array. <pre> Array("NAME:FaceCSParameters", <OriginArray>, "MoveToEnd:=", <boolean>, </pre>

			<pre>"FaceID:=", <integer>, <AxisPosnArray>, "WhichAxis:=", <string "X", "Y", or "Z">, "ZRotationAngle:=", <string>, "XOffset:=", <string>, "YOffset:=", <string>, "AutoAxis:=", <boolean></pre>
	<i><OriginArray></i>	Array	Structured array. <pre>Array("NAME:Origin", "IsAttachedToEntity:=", <boolean>, "EntityID:=", <integer>, "FacetedBodyTriangleIndex:=", <integer>, "TriangleVertexIndex:=", <integer>, "PositionType:=", <string "FaceCenter", "EdgeCenter", "OnVertex", "OnEdge", or "OnFace">, "UParam:=", <float between 0 and 1 representing the relative position of the point on the edge or face>, "VParam:=", <float between 0 and 1 representing the relative position of the point on the edge or face>, "XPosition:=", <string>,</pre>

			"YPosition:=" , <string>, "ZPosition:=" , <string>) IsAttachedToEntity specifies whether the point is anchored to a vertex, edge, or face. If True, provide UParam and VParam. If False, provide XPosition, YPosition, and ZPosition to provide fixed position. Pass "0" for unused parameters.
	<AxisPosnArray>	Array	Structured array. Array("NAME:AxisPosn", "IsAttachedToEntity:=" , <boolean>, "EntityID:=" , <integer>, "FacetedBodyTriangleIndex:=" , <integer>, "TriangleVertexIndex:=" , <integer>, "PositionType:=" , <string "FaceCenter", "EdgeCenter", "OnVertex", "OnEdge", or "OnFace">, "UParam:=" , <float>, "VParam:=" , <float>, "XPosition:=" , <string>, "YPosition:=" , <string>, "ZPosition:=" , <string>)
	<AttributesArray>	Array	Structured array. See: AttributesArray .
Return Value	None.		

Python Syntax	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,
```

	<pre>"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"])</pre>

VB Syntax	CreateFaceCS <Parameters> <AttributesArray>
VB	oEditor.CreateFaceCS Array("NAME:FaceCSParameters", Array("NAME:Origin", "IsAt-

Example	<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	Type	Description
	<Parameters>	Array	Structured array. Array("NAME:GroupParameter", "ParentGroupID:=" , <string>, "Parts:=" , <string>,

			"SubmodelInstances:=" , <string>, "Groups:=" , <string>)
Return Value	None.		

Python Syntax	CreateGroup(<Parameters>)
Python Example	<pre>oEditor.CreateGroup(["NAME:GroupParameter", "ParentGroupID:=" , "Model", "Parts:=" , "Box1,Box2,Box3", "SubmodelInstances:=" , "", "Groups:=" , ""])</pre>

VB Syntax	CreateGroup <Parameters>
VB Example	<pre>oEditor.CreateGroup Array("NAME:GroupParameter", "ParentGroupID:=", "Model", _ "Parts:=", "Box1,Box2,Box3", "SubmodelInstances:=", "", "Groups:=", "")</pre>

CreateObjectCS

Creates an object coordinate system from a selected object.

UI Access	Modeler > Coordinate System > Create > Object > [Offset / Rotated / Both].

Parameters	Name	Type	Description
	<Parameters>	Array	Structured array. <pre>Array("NAME:ObjectCSParameters", <OriginArray> "MoveToEnd:=" , <boolean>, "ReverseXAxis:=" , <boolean>, "ReverseYAxis:=" , <boolean>, <xAxisArray / xAxisPosArray> <yAxisArray / yAxisPosArray>)</pre> Note: xAxisArray and xAxisPosArray differ. Use xAxisArray for absolute position and xAxisPosArray for relative position. Do the same for yAxisArray and yAxisPosArray.
	<OriginArray>	Array	Structured array. <pre>Array("NAME:Origin", "IsAttachedToEntity:=" , <boolean>, "EntityID:=" , <integer>, "FacetedBodyTriangleIndex:=", <integer>, "TriangleVertexIndex:=" , <integer>, "PositionType:=" , <string "OnVertex", "EdgeCenter", "FaceCenter", "OnEdge", or "AbsolutePosition">, "UParam:=" , <float between 0 and 1 representing the relative position of the point on the edge or</pre>

		<pre>face>, "VParam:=" , <float between 0 and 1 representing the relative position of the point on the edge or face>, "XPosition:=" , <string>, "YPosition:=" , <string>, "ZPosition:=" , <string>) IsAttachedToEntity specifies whether the point is anchored. If True, provide UParam and VParam. If False, provide XPosition, YPosition, and ZPosition to provide fixed position. Pass "0" for unused parameters.</pre>
<xAxisArray>	Array	<pre>Structured array for absolute position: Array("NAME:xAxis", "DirectionType:=" , "AbsoluteDirection", "EdgeID:=" , <integer>, "FaceID:=" , <integer>, "xDirection:=" , <string>, "yDirection:=" , <string>, "zDirection:=" , <string>, "UParam:=" , <float>, "VParam:=" , <float>)</pre>
<xAxisPosArray>	Array	<pre>Structured array for relative position: Array("NAME:xAxisPos", "IsAttachedToEntity:=" , <boolean>,</pre>

			<pre>"EntityID:=" , <integer>, "FacetedBodyTriangleIndex:=", <integer>, "TriangleVertexIndex:=" , <integer>, "PositionType:=" , <string "OnVertex", "EdgeCenter", "FaceCenter", or "OnEdge">, "UParam:=" , <float>, "VParam:=" , <float>, "XPosition:=" , <string>, "YPosition:=" , <string>, "ZPosition:=" , <string>)</pre>
	<yAxisArray>	Array	Structured array for absolute position: <pre>Array("NAME:yAxis", "DirectionType:=" , "AbsoluteDirection", "EdgeID:=" , <integer>, "FaceID:=" , <integer>, "xDirection:=" , <string>, "yDirection:=" , <string>, "zDirection:=" , <string>, "UParam:=" , <float>, "VParam:=" , <float>)</pre>
	<yAxisPosArray>	Array	Structured array for relative position:

			<pre>Array("NAME:yAxisPos", "IsAttachedToEntity:=" , <boolean>, "EntityID:=" , <integer>, "FacetedBodyTriangleIndex:=", <integer>, "TriangleVertexIndex:=" , <integer>, "PositionType:=" , <string "OnVertex", "EdgeCenter", "FaceCenter", or "OnEdge">, "UParam:=" , <float>, "VParam:=" , <float>, "XPosition:=" , <string>, "YPosition:=" , <string>, "ZPosition:=" , <string>)</pre>
	<AttributesArray>	Array	Structured array. See: AttributesArray .
Return Value	None.		

Python Syntax	CreateObjectCS(<Parameters>,<AttributesArray>)
Python Example	<pre>oEditor.CreateObjectCS(["NAME:ObjectCSParameters", ["NAME:Origin", "IsAttachedToEntity:=" , True, "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

```

	<pre>], ["NAME:yAxis", "DirectionType:=" , "AbsoluteDirection", "EdgeID:=" , -1, "FaceID:=" , -1, "xDirection:=" , "0", "yDirection:=" , "1", "zDirection:=" , "0", "UParam:=" , 0, "VParam:=" , 0]], ["NAME:Attributes", "Name:=" , "ObjectCS1", "PartName:=" , "Box2"])</pre>

VB Syntax	CreateObjectCS <Parameters> <AttributesArray>
VB Example	<pre>oEditor.CreateObjectCS Array("NAME:ObjectCSParameters", Array("NAME:Origin", "IsAt- tachedToEntity:=", _ false, "EntityID:=", -1, "FacetedBodyTriangleIndex:=", -1, "TriangleVertexIndex:=", _</pre>

	<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") </pre>

CreateObjectFromEdges

Creates an object from the specified object edge.

UI Access	Modeler > Edge > Create Object From Edge		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: SelectionsArray .
	<ParametersArray>	Array	Structured array.
			Array("NAME:Parameters",

			<pre>Array("NAME:BodyFromEdgeToParameters", "Edges:=" , <array containing integer edges>))</pre>
	<i><CreateGroupsForNewObjects></i>	Array	Structured array. <pre>Array("CreateGroupsForNewObjects:=", <boolean True to create groups for new objects; else False>)</pre>
Return Value	None.		

Python Syntax	CreateObjectFromEdges(<SelectionsArray>, <ParametersArray>, <CreateGroupsForNewObjects>)
Python Example	<pre>oEditor.CreateObjectFromEdges(["NAME:Selections", "Selections:=", "Box2", "NewPartsModelFlag:=", "Model"], ["NAME:Parameters", ["NAME:BodyFromEdgeToParameters", "Edges:=", [41]]], ["CreateGroupsForNewObjects:=", False])</pre>

	1)

VB Syntax	CreateObjectFromEdges <SelectionsArray> <ParametersArray> <CreateGroupsForNewObjects>
VB Example	oEditor.CreateObjectFromEdges Array("NAME:Selections", "Selections:=", "Box1", "NewPartsModelFlag:=", "Model"), Array("NAME:Parameters", Array("NAME:BodyFromEdgeToParameters", "Edges:=", Array(13))), Array("CreateGroupsForNewObjects:=", false)

CreateObjectFromFace

Creates 2D objects from specified face(s).

UI Access	N/A		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: SelectionsArray .
	<Parameters>	Array	Structured array. Array("NAME:Parameters", <BodyFromFaceToParameters>)
	<CreateGroupsForNewObjects>	Array	Optional. Structured array. Array("CreateGroupsForNewObjects:=", <boolean>)
Return Value	None.		

Python Syntax	CreateObjectFromFace (<SelectionsArray>,<Parameters>,<CreateGroupsForNewObjects>)
Python Example	<pre> oEditor.CreateObjectFromFace (["NAME:Selections", "Selections:=" , "Box3", "NewPartsModelFlag:=" , "Model"], ["NAME:Parameters", ["NAME:BodyFromFaceToParameters", "FacesToDetach:=" , [68]]], ["CreateGroupsForNewObjects:=", False]) </pre>

VB Syntax	CreateObjectFromFace <SelectionsArray> <Parameters> <CreateGroupsForNewObjects>
VB Example	<pre> oEditor.CreateObjectFromFace Array("NAME:Selections", "Selections:=", "Box3", "NewPartsModelFlag:=" , "Model") Array("NAME:Parameters", Array("NAME:BodyFromFaceToParameters", "FacesToDetach:=", Array(68))) Array("CreateGroupsForNewObjects:=", false) </pre>

CreateObjectFromFaces

Creates 2D objects from specified face(s).

UI Access	Modeler > Surface > Create Object from Face		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: SelectionsArray .
	<Parameters>	Array	Structured array. Array("NAME:Parameters", <BodyFromFaceToParameters>)
	<CreateGroupsForNewObjects>	Array	Structured array. Array("CreateGroupsForNewObjects:=", <boolean>)
Return Value	None.		

Python Syntax	CreateObjectFromFaces (<SelectionsArray>,<Parameters>, <CreateGroupsForNewObjects>)
Python Example	<pre>oEditor.CreateObjectFromFaces(["NAME:Selections", "Selections:=" , "Box3", "NewPartsModelFlag:=" , "Model"], ["NAME:Parameters", ["NAME:BodyFromFaceToParameters",</pre>

	<pre>"FacesToDetach:=" , [68]]], ["CreateGroupsForNewObjects:=", False])</pre>

VB Syntax	CreateObjectFromFaces <SelectionsArray> <Parameters> <CreateGroupsForNewObjects>
VB Example	<pre>oEditor.CreateObjectFromFaces Array("NAME:Selections", "Selections:=", "Box3", "NewPartsModelFlag:=" , "Model") Array("NAME:Parameters", Array("NAME:BodyFromFaceToParameters", "FacesToDetach:=", Array(68))) Array("CreateGroupsForNewObjects:=", False)</pre>

CreateRelativeCS

Creates a Relative Coordinate System.

UI Access	Modeler > Coordinate System > Create > Relative CS > [Offset / Rotated / Both].		
Parameters	Name	Type	Description
	<Parameters>	Array	Structured array. Array("NAME:RelativeCSParameters", "Mode:=" , "Axis/Position", "OriginX:=" , <string>, "OriginY:=" , <string>,

			"OriginZ:=" , <string>, "XAxisXvec:=" , <string>, "XAxisYvec:=" , <string>, "XAxisZvec:=" , <string>, "YAxisXvec:=" , <string>, "YAxisYvec:=" , <string>, "YAxisZvec:=" , <string>)
	<AttributesArray>	Array	Structured array. See: AttributesArray . CreateRelativeCS supports only the "Name" parameter.
Return Value	None.		

Python Syntax	CreateRelativeCS(<Parameters>,<AttributesArray>)
Python Example	<pre>oEditor.CreateRelativeCS(["NAME:RelativeCSParameters", "Mode:=" , "Axis/Position", "OriginX:=" , "0.62mm", "OriginY:=" , "-0.7mm", "OriginZ:=" , "0mm", "XAxisXvec:=" , "1mm", "XAxisYvec:=" , "0mm",</pre>

	<pre>"XAxisZvec:=" , "0mm", "YAxisXvec:=" , "0mm", "YAxisYvec:=" , "1mm", "YAxisZvec:=" , "0mm"], ["NAME:Attributes", "Name:=" , "RelativeCS1"])</pre>

VB Syntax	CreateRelativeCS <Parameters> <AttributesArray>
VB Example	<pre>oEditor.CreateRelativeCS Array("NAME:RelativeCSParameters", "Mode:=", "Axis/Position", "OriginX:=", _ "0mm", "OriginY:=", "0mm", "OriginZ:=", "0mm", "XAxisXvec:=", "0.66mm", "XAxisYvec:=", _ "0.28mm", "XAxisZvec:=", "0mm", "YAxisXvec:=", "0.06mm", "YAxisYvec:=", "0.14mm", "YAxisZvec:=", _ "0mm"), Array("NAME:Attributes", "Name:=", "RelativeCS1")</pre>

DeleteEmptyGroups

Deletes group(s) from the history tree.

UI Access	Modeler > Group > Delete Empty.		
Parameters	Name	Type	Description

	<table><tr><td><Parameters></td><td>Array</td><td>Structured array. Array("Groups:=" , <array of string group IDs>)</td></tr></table>	<Parameters>	Array	Structured array. Array("Groups:=" , <array of string group IDs>)
<Parameters>	Array	Structured array. Array("Groups:=" , <array of string group IDs>)		
Return Value	None.			

Python Syntax	DeleteEmptyGroups(<Parameters>)
Python Example	<pre>oEditor.DeleteEmptyGroups(["Groups:=", ["Group1", "Group2", "Group3"]])</pre>

VB Syntax	DeleteEmptyGroups <Parameters>
VB Example	<pre>oEditor.DeleteEmptyGroups Array("Groups:=", Array("Group1", "Group2", "Group3"))</pre>

DeleteLastOperation

Deletes the last operation performed on the specified object(s).

UI Access	Modeler > Delete Last Operation.		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: SelectionsArray .
Return Value	None.		

Python Syntax	DeleteLastOperation(<SelectionsArray>)

Python Example	<pre>oEditor.DeleteLastOperation(["NAME:Selections", "Selections:=", "Box3", "NewPartsModelFlag:=", "Model"])</pre>

VB Syntax	DeleteLastOperation <SelectionsArray>
VB Example	<pre>oEditor.DeleteLastOperation Array("NAME:Selections", "Selections:=", "Box3", "NewPartsModelFlag:=", "Model")</pre>

DeleteOperation

Deletes specified operation performed on a selected object.

UI Access	Select an operation in the project tree, then press Delete on the keyboard.		
Parameters	Name	Type	Description
	<ParamsArray>	Array	Structured array. Array("NAME:Parameters", Array("NAME:PartOperations", Array("NAME:<PartName>", "OperationIndices:=", <array of operation indices>)), Array("NAME:UDMOperations"))

Return Value	None.

Python Syntax	DeleteOperation(<ParamsArray>)
Python Example	<pre>oEditor.DeleteOperation(["NAME:Parameters", ["NAME:PartOperations", ["NAME:Coil_0", "OperationIndices:=", [1]]], ["NAME:UDMOperations"])</pre>

VB Syntax	DeleteOperation <ParamsArray>
VB Example	<pre>oEditor.DeleteOperation _ Array("NAME:Parameters", _</pre>

	<pre> Array("NAME:PartOperations", _ Array("NAME:Coil_0", _ "OperationIndices:=", Array(1))), _ Array("NAME:UDMOperations")) </pre>

DetachEdges

Detaches the specified edge(s) from an object.

UI Access	Modeler > Edge > Detach Edges.		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: SelectionsArray .
	<Parameters>	Array	Structured array. <pre> Array("NAME:Parameters", <DetachEdgesArray>) </pre>
	<DetachEdgesArray>	Array	Structured array. <pre> Array("NAME:DetachEdgesToParameters", "EdgesToDetach:=" , <array containing integer edge IDs>) </pre>
Return Value	None.		

Python Syntax	DetachEdges(<SelectionsArray>, <Parameters>)
Python Example	<pre> oEditor.DetachEdges(["NAME:Selections", "Selections:=", "Rectangle1", </pre>

	<pre>"NewPartsModelFlag:=", "Model"], ["NAME:Parameters", ["NAME:DetachEdgesToParameters", "EdgesToDetach:=", [18,17]]]</pre>

VB Syntax	DetachEdges <SelectionsArray> <Parameters>
VB Example	<pre>oEditor.DetachEdges Array("NAME:Selections", "Selections:=", "Rectangle1", "NewPartsModelFlag:=", "Model") Array("NAME:Parameters", Array("NAME:DetachEdgesToParameters", "EdgesToDetach:=", [18,17]))</pre>

DetachFaces

Detaches the specified face(s) from an object.

UI Access	Modeler > Surface > Detach Faces.		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: SelectionsArray .
	<Parameters>	Array	Structured array. Array("NAME:Parameters",

			<DetachFacesArray>
	<DetachFacesArray>	Array	Structured array. Array("NAME:DetachFacesToParameters", "FacesToDetach:=" , <array containing integer face IDs>)
Return Value	None.		

Python Syntax	DetachFaces(<SelectionsArray>, <Parameters>)
Python Example	<pre>oEditor.DetachFaces(["NAME:Selections", "Selections:=", "Box3", "NewPartsModelFlag:=", "Model"], ["NAME:Parameters", ["NAME:DetachFacesToParameters", "FacesToDetach:=", [68,67]]]) </pre>

VB Syntax	DetachFaces <SelectionsArray> <Parameters>
VB	oEditor.DetachFaces Array("NAME:Selections", "Selections:=", "Box3",

Example	<pre>"NewPartsModelFlag:=", "Model") Array("NAME:Parameters", Array("NAME:DetachFacesToParameters", "FacesToDetach:=", [68,67]))</pre>

EditEntityList

Modifies an entity list.

UI Access	Modeler > List > Reassign.		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: SelectionsArray .
	<Parameters>	Array	Structured array. Array("NAME:GeometryEntityListParameters", "EntityType:=" , <string "Object" or "Face">, "EntityList:=" , <string list>)
Return Value	None.		

Python Syntax	EditEntityList(<SelectionsArray>, <Parameters>)
Python Example	<pre>oEditor.EditEntityList(["NAME:Selections", "Selections:=" , "Objectlist1"</pre>

	<pre>], ["NAME:GeometryEntityListParameters", "EntityType:=" , "Object", "EntityList:=" , "Box1, Box2, Box3"]) </pre>

VB Syntax	EditEntityList <SelectionsArray> <Parameters>
VB Example	<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](#).

UI Access	Modeler > Coordinate System > Edit.		
Parameters	Name	Type	Description
	<Parameters>	Array	Structured array. <pre> Array("NAME:FaceCSParameters", <OriginArray>, "MoveToEnd:=", <boolean>, </pre>

			<pre>"FaceID:=", <integer>, <AxisPosnArray>, "WhichAxis:=", <string "X", "Y", or "Z">, "ZRotationAngle:=", <string>, "XOffset:=", <string>, "YOffset:=", <string>, "AutoAxis:=", <boolean></pre>
	<i><OriginArray></i>	Array	<pre>Structured array. Array("NAME:Origin", "IsAttachedToEntity:=", <boolean>, "EntityID:=", <integer>, "FacetedBodyTriangleIndex:=", <integer>, "TriangleVertexIndex:=", <integer>, "PositionType:=", <string "FaceCenter", "EdgeCenter", "OnVertex", "OnEdge", or "OnFace">, "UParam:=", <float between 0 and 1 representing the relative position of the point on the edge or face>, "VParam:=", <float between 0 and 1 representing the relative position of the point on the edge or face>, "XPosition:=", <string>,</pre>

			<pre>"YPosition:=" , <string>, "ZPosition:=" , <string>)</pre> IsAttachedToEntity specifies whether the point is anchored to a vertex, edge, or face. If True, provide UParam and VParam. If False, provide XPosition, YPosition, and ZPosition to provide fixed position. Pass "0" for unused parameters.
	<AxisPosnArray>	Array	Structured array. <pre>Array("NAME:AxisPosn", "IsAttachedToEntity:=" , <boolean>, "EntityID:=" , <integer>, "FacetedBodyTriangleIndex:=", <integer>, "TriangleVertexIndex:=" , <integer>, "PositionType:=" , <string "FaceCenter", "EdgeCenter", "OnVertex", "OnEdge", or "OnFace">, "UParam:=" , <float>, "VParam:=" , <float>, "XPosition:=" , <string>, "YPosition:=" , <string>, "ZPosition:=" , <string>)</pre>
	<AttributesArray>	Array	Structured array. See: AttributesArray . Use to select the coordinate system to edit.
Return Value	None.		
Python Syntax	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"
])

```

VB Syntax	EditFaceCS <Parameters> <AttributesArray>
VB Example	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",

```

	<pre>"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")</pre>

EditObjectCS

Edits an existing object coordinate system. See: [CreateObjectCS](#).

UI Access	Modeler > Coordinate System > Edit.		
Parameters	Name	Type	Description
	<Parameters>	Array	Structured array. Array("NAME:ObjectCSParameters", <OriginArray> "MoveToEnd:=" , <boolean>,

		<pre> "ReverseXAxis:=" , <boolean>, "ReverseYAxis:=" , <boolean>, <xAxisArray / xAxisPosArray> <yAxisArray / yAxisPosArray> </pre> Note: xAxisArray and xAxisPosArray differ. Use xAxisArray for absolute position and xAxisPosArray for relative position. Do the same for yAxisArray and yAxisPosArray.
	<OriginArray>	Array Structured array. <pre> Array("NAME:Origin", "IsAttachedToEntity:=" , <boolean>, "EntityID:=" , <integer>, "FacetedBodyTriangleIndex:=", <integer>, "TriangleVertexIndex:=" , <integer>, "PositionType:=" , <string "OnVertex", "EdgeCenter", "FaceCenter", "OnEdge", or "AbsolutePosition">, "UParam:=" , <float between 0 and 1 representing the relative position of the point on the edge or face>, "VParam:=" , <float between 0 and 1 representing the relative position of the point on the edge or face>, "XPosition:=" , <string>, </pre>

		<pre>"YPosition:=" , <string>, "ZPosition:=" , <string>)</pre> IsAttachedToEntity specifies whether the point is anchored. If True, provide UParam and VParam. If False, provide XPosition, YPosition, and ZPosition to provide fixed position. Pass "0" for unused parameters.
<xAxisArray>	Array	Structured array for absolute position: <pre>Array("NAME:xAxis", "DirectionType:=" , "AbsoluteDirection", "EdgeID:=" , <integer>, "FaceID:=" , <integer>, "xDirection:=" , <string>, "yDirection:=" , <string>, "zDirection:=" , <string>, "UParam:=" , <float>, "VParam:=" , <float>)</pre>
<xAxisPosArray>	Array	Structured array for relative position: <pre>Array("NAME:xAxisPos", "IsAttachedToEntity:=" , <boolean>, "EntityID:=" , <integer>, "FacetedBodyTriangleIndex:=" , <integer>, "TriangleVertexIndex:=" , <integer>, "PositionType:=" , <string "OnVertex", "EdgeCenter", "FaceCenter", or "OnEdge">,</pre>

		<pre>"UParam:=" , <float>, "VParam:=" , <float>, "XPosition:=" , <string>, "YPosition:=" , <string>, "ZPosition:=" , <string>)</pre>
<code><yAxisArray></code>	Array	Structured array for absolute position: <pre>Array("NAME:yAxis", "DirectionType:=" , "AbsoluteDirection", "EdgeID:=" , <integer>, "FaceID:=" , <integer>, "xDirection:=" , <string>, "yDirection:=" , <string>, "zDirection:=" , <string>, "UParam:=" , <float>, "VParam:=" , <float>)</pre>
<code><yAxisPosArray></code>	Array	Structured array for relative position: <pre>Array("NAME:yAxisPos", "IsAttachedToEntity:=" , <boolean>, "EntityID:=" , <integer>, "FacetedBodyTriangleIndex:=" , <integer>, "TriangleVertexIndex:=" , <integer>,</pre>

			"PositionType:=" , <string "OnVertex", "EdgeCenter", "FaceCenter", or "OnEdge">, "UParam:=" , <float>, "VParam:=" , <float>, "XPosition:=" , <string>, "YPosition:=" , <string>, "ZPosition:=" , <string>)
	<AttributesArray>	Array	Structured array. See: AttributesArray . Use to select the coordinate system to edit.
Return Value	None.		

Python Syntax	EditObjectCS(<Parameters>,<AttributesArray>)
Python Example	<pre>oEditor.EditObjectCS(["NAME:ObjectCSParameters", ["NAME:Origin", "IsAttachedToEntity:=" , True, "EntityID:=" , 59, "FacetedBodyTriangleIndex:=", -1, "TriangleVertexIndex:=" , -1, "PositionType:=" , "OnVertex", "UParam:=" , 0, "VParam:=" , 0,</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,

```

	<pre>"FaceID:=" , -1, "xDirection:=" , "0", "yDirection:=" , "1", "zDirection:=" , "0", "UParam:=" , 0, "VParam:=" , 0]], ["NAME:Attributes", "Name:=" , "ObjectCS1", "PartName:=" , "Box2"])</pre>

VB Syntax	EditObjectCS <Parameters> <AttributesArray>
VB Example	<pre>oEditor.EditObjectCS Array("NAME:ObjectCSParameters", Array("NAME:Origin", "IsAt- tachedToEntity:=", _ false, "EntityID:=", -1, "FacetedBodyTriangleIndex:=", -1, "TriangleVertexIndex:=", _ -1, "PositionType:=", "AbsolutePosition", "UParam:=", 0, "VParam:=", 0, "XPosition:=", _ "0mm", "YPosition:=", "0mm", "ZPosition:=", "0mm"), "MoveToEnd:=", false, "ReverseXAx- is:=", _ false, "ReverseYAxis:=", false, Array("NAME:xAxisPos", "IsAttachedToEntity:=", true,</pre>

	<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")</pre>

EditRelativeCS

Edits an existing Relative Coordinate System. See: [CreateRelativeCS](#).

UI Access	Modeler > Coordinate System > Edit.		
Parameters	Name	Type	Description
	<Parameters>	Array	Structured array. Array("NAME:RelativeCSParameters", "Mode:=" , "Axis/Position", "OriginX:=" , <string>, "OriginY:=" , <string>, "OriginZ:=" , <string>,

			"XAxisXvec:=" , <string>, "XAxisYvec:=" , <string>, "XAxisZvec:=" , <string>, "YAxisXvec:=" , <string>, "YAxisYvec:=" , <string>, "YAxisZvec:=" , <string>)
	<AttributesArray>	Array	Structured array. See: AttributesArray . Use to select the coordinate system to edit.
Return Value	None.		

Python Syntax	EditRelativeCS(<Parameters>,<AttributesArray>)
Python Example	<pre>oEditor.EditRelativeCS(["NAME:RelativeCSParameters", "Mode:=" , "Axis/Position", "OriginX:=" , "0.62mm", "OriginY:=" , "-0.7mm", "OriginZ:=" , "0mm", "XAxisXvec:=" , "1mm", "XAxisYvec:=" , "0mm", "XAxisZvec:=" , "0mm", "YAxisXvec:=" , "0mm", "YAxisYvec:=" , "1mm",</pre>

	<pre>"YAxisZvec:=" , "0mm"], ["NAME:Attributes", "Name:=" , "RelativeCS1"])</pre>

VB Syntax	EditRelativeCS <Parameters> <AttributesArray>
VB Example	<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", "YAxisZvec:=", _ "0mm"), Array("NAME:Attributes", "Name:=", "RelativeCS1")</pre>

Export

Exports the model to a file.

Note: This script does not export image file types or GDSII files. See: [ExportModelImageToFile](#) and ExportGDSII.

UI Access	Modeler > Export

	Name	Type	Description
	<Parameters>	Array	Structured array. <pre>Array("NAME:ExportParameters", "AllowRegionDependentPartSelectionForPMLCreation:=", <boolean>, "AllowRegionSelectionForPMLCreation:=", <boolean>, "Selections:=" , <string list>, "File Name:=" , <string filepath>, "Major Version:=" , <integer (-1 if not applic- able)>, "Minor Version:=" , <integer (-1 if not applic- able)>)</pre>
Parameters			
Return Value	None.		

Python Syntax	Export(<Parameters>)
Python Example	<pre>oEditor.Export(["NAME:ExportParameters", "AllowRegionDependentPartSelectionForPMLCreation:=", True, "AllowRegionSelectionForPMLCreation:=", True, "Selections:=" , "Box1,Box2,Box3", "File Name:=" , "C:/Users/jdoe/Desktop/export.sab", "Major Version:=" , 25,</pre>

	<pre>"Minor Version:=" , 0 l)</pre>

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 Ensign use *.avz. For export to Ensign in -ng mode, the corresponding version of Ensign must be installed. On Linux, it might need manual set environment variable AWP_ROOT212 to its installation path, e.g. AWP_ROOT212-2=/installations/ansys_inc/v212/ for AnsysEDT v21.2 and Ensign 21.2.

ExportModelImageToFile supports export overlay of polar plot 3D with existing transformation (scaling, rotation and translation) in -ng (non-graphical) mode.

UI Access	Modeler > Export.

Parameters	Name	Type	Description
	<path>	String	Full file path including extension.
	<width>	Integer	Width in pixels (use 0 for default).
	<height>	Integer	Height in pixels (use 0 for default).
	<Parameters>	Array	Structured array.
			<pre>Array("NAME:SaveImageParams", "ShowAxis:=" , <string containing boolean>, "ShowGrid:=" , <string containing boolean>, "ShowRuler:=" , <string containing boolean>, "ShowRegion:=" , <string>, "Selections:=" , <string>, "FieldPlotSelections:=" , <string>' # Comma delimited string. #Use to set which field plot to show. "FitToSelections:=" , "" , "FitToFieldPlotSelection:=" , ""</pre> Note: "FitToSelections" specify geometry objects for the "Fit" operation. Note: "FitToFieldPlotSelections" specifies field plots for the "Fit" operation.

			"AutoFit:=" , "True", Note: If FitToSlections or FitToFieldPlotSelections are used , then then AutoFit is true, it makes sure color key does not overlap field plot. It is true by default. If neither is used, then "AutoFit" will "Fit" to full model. "Orientation:=" , <string> "ShowOrientationGadget:=" , <False>
Return Value	None.		

Python Syntax	ExportModelImageToFile(<path> <width> <height> <Parameters>)
Python Example	<pre>oEditor.ExportModelImageToFile ("C:/Users/jdoe/Desktop/export.bmp", 1920, 1080, ["NAME:SaveImageParams", "ShowAxis:=" , "True", "ShowGrid:=" , "True", "ShowRuler:=" , "True",</pre>

	<pre>"ShowRegion:=" , "Default", "Selections:=" , "", "FieldPlotSelections:=" , "", "FitToSelections:=" , "", "FitToFieldPlotSelections:=" , "", "AutoFit:=" , "True", "Orientation:=" , "" 1)</pre>

VB Syntax	ExportModelImageToFile <path> <width> <height> <Parameters>
VB Example	<pre>oEditor.ExportModelImageToFile "C:/Users/jdoe/Desktop/export.bmp", 1383, 512, Array("NAME:SaveImageParams", "ShowAxis:=", "True", "ShowGrid:=", "True", "ShowRuler:=", "True", "ShowRegion:=", "Default", "Selections:=", "", "FitToFieldPlotSelections:=", "",</pre>

	<code>"AutoFit:="</code> <code>"Orientation:=", "</code>

ExportModelMeshToFile

Exports geometry model to a 3D model file (e.g. *.obj, *.wrl, etc.).

UI Access	N/A		
Parameters	Name	Type	Description
	<filePath>	String	Full file path, including extension *.obj, *.wrl, etc
	<selections>	Array	Selected parts.
Return Value	None.		

Python Syntax	<code>ExportModelMeshToFile <filePath>, <selections>)</code>
Python Example	<code>oEditor.ExportModelMeshToFile("E:/MyDir/scriptRun/2Selected-ng.obj", ["BotCover-", "AveragingVolumeAtPeakRMSEfieldLocation"])</code>

VB Syntax	<code>ExportModelMeshToFile <filePath>, <selections></code>
VB Example	<code>oEditor.ExportModelMeshToFile("E:/MyDir/scriptRun/2Selected-ng.obj", ["BotCover-", "AveragingVolumeAtPeakRMSEfieldLocation"])</code>

Fillet

Performs a fillet on specified edge(s).

UI Access	Modeler > Fillet.		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: SelectionsArray .
	<Parameters>	Array	Structured array. Array("NAME:Parameters", <FilletParametersArray>)
	<FilletParametersArray>	Array	Structured array. Array("NAME:FilletParameters", "Edges:=" , <array containing integer edge IDs>, "Vertices:=" , <empty array>, "Radius:=" , <string>, "Setback:=" , <string>)
Return Value	None.		

Python Syntax	Fillet(<SelectionsArray>, <Parameters>)
Python Example	<pre>oEditor.Fillet(["NAME:Selections", "Selections:=" , "Box1", "NewPartsModelFlag:=" , "Model"], ["NAME:Parameters",</pre>

	<pre> ["NAME:FilletParameters", "Edges:=" , [13], "Vertices:=" , [], "Radius:=" , "1mm", "Setback:=" , "0mm"]] </pre>

VB Syntax	Fillet <SelectionsArray> <Parameters>
VB Example	<pre> oEditor.Fillet Array("NAME:Selections", "Selections:=" , "Box1", "NewPartsModelFlag:=" , "Model") Array("NAME:Parameters", Array("NAME:FilletParameters", "Edges:=" , Array(13), "Vertices:=" , Array(), "Radius:=" , "1mm", "Setback:=" , "0mm")) </pre>

FlattenGroup

Flattens a specified history tree group.

UI Access	Modeler > Group > Flatten.		
Parameters	Name	Type	Description
	<GroupID>	Array	Structured array. Array("Groups:=", Array(<string list of group IDs>))
Return Value	None.		

Python Syntax	FlattenGroup (<GroupID>)
Python Example	<code>oEditor.FlattenGroup(["Groups:=", ["Group1"]])</code>

VB Syntax	FlattenGroup <GroupID>
VB Example	<code>oEditor.FlattenGroup Array("Groups:=", Array("Group1"))</code>

GenerateHistory

Generates the history for specified 1D object(s).

UI Access	Modeler > Generate History.		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: SelectionsArray .

Return Value	None.

Python Syntax	GenerateHistory (<SelectionsArray>)
Python Example	<pre>oEditor.GenerateHistory(["NAME:Selections", "Selections:=" , "Polyline1", "NewPartsModelFlag:=" , "Model", "UseCurrentCS:=" , True]) </pre>

VB Syntax	GenerateHistory <SelectionsArray>
VB Example	<pre>oEditor.GenerateHistory Array("NAME:Selections", "Selections:=" , "Polyline1", "NewPartsModelFlag:=" , "Model", "UseCurrentCS:=" , True) </pre>

GetActiveCoordinateSystem

Returns the active coordinate system.

UI Access	None.
Parameters	None.
Return Value	String name of active coordinate system.

Python Syntax	<code>GetActiveCoordinateSystem()</code>
Python Example	<code>oEditor.GetActiveCoordinateSystem()</code>

VB Syntax	<code>GetActiveCoordinateSystem</code>
VB Example	<pre>dim csName csName = oEditor.GetActiveCoordinateSystem</pre>

GetActiveCoordinateSystemTransform

Returns transformation information for the active coordinate system (with respect to the global coordinate system), which consists of the affine matrix for rotation of the coordinate system and a 3D vector with the translation of the coordinate system's origin with respect to the global coordinate system's origin.

UI Access	None
Parameters	None
Return Value	Array of strings containing transformation information

Python Syntax	<code>GetActiveCoordinateSystemTransform()</code>
Python Example	<code>oEditor.GetActiveCoordinateSystemTransform()</code>

VB Syntax	GetActiveCoordinateSystemTransform
VB Example	<code>oEditor.GetActiveCoordinateSystemTransform</code>

GetCoordinateSystems

Returns the names of coordinate systems in the design.

UI Access	None.
Parameters	None.
Return Value	Array containing string names of coordinate systems.

Python Syntax	GetCoordinateSystems()
Python Example	<code>oEditor.GetCoordinateSystems()</code>

VB Syntax	GetCoordinateSystems()
VB Example	<pre>dim csNames csNames = oEditor.GetCoordinateSystems</pre>

HealObject

Heals an imported object.

UI Access	Modeler > Model Preparation > Heal.		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: SelectionsArray .

<Parameters>	Array	Structured array. <pre> Array("NAME:ObjectHealingParameters", "Version:=" , <integer>, "AutoHeal:=" , <boolean>, "TolerantStitch:=" , <boolean>, "SimplifyGeom:=" , <boolean>, "TightenGaps:=" , <boolean>, "HealToSolid:=" , <boolean>, "StopAfterFirstStitchError:=" , <boolean>, "MaxStitchTol:=" , <float>, "ExplodeAndStitch:=" , <boolean>, "GeomSimplificationTol:=" , <integer>, "MaximumGeneratedRadiusForSimplification:=" , <integer>, "SimplifyType:=" , <integer>, "TightenGapsWidth:=" , <float>, "RemoveSliverFaces:=" , <boolean>, "RemoveSmallEdges:=" , <boolean>, "RemoveSmallFaces:=" , <boolean>, "SliverFaceTol:=" , <integer>, "SmallEdgeTol:=" , <integer>, "SmallFaceAreaTol:=" , <integer>,</pre>

			"BoundingBoxScaleFactor:=", <integer>, "RemoveHoles:=" , <boolean>, "RemoveChamfers:=" , <boolean>, "RemoveBlends:=" , <boolean>, "HoleRadiusTol:=" , <integer>, "ChamferWidthTol:=" , <integer>, "BlendRadiusTol:=" , <integer>, "AllowableSurfaceAreaChange:=", <integer>, "AllowableVolumeChange:=", <integer>)
Return Value	None.		

Python Syntax	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,
```

	<pre> "HoleRadiusTol:=" , 0, "ChamferWidthTol:=" , 0, "BlendRadiusTol:=" , 0, "AllowableSurfaceAreaChange:=", 5, "AllowableVolumeChange:=", 5] </pre>

VB Syntax	HealObject <SelectionsArray> <Parameters>
VB Example	<pre> oEditor.HealObject Array("NAME:Selections", "Selections:=", "Box2", "NewPartsModelFlag:=", _ "Model"), Array("NAME:ObjectHealingParameters", "Version:=", 1, "AutoHeal:=", false, "TolerantStitch:=", _ true, "SimplifyGeom:=", true, "TightenGaps:=", true, "HealToSolid:=", false, "StopAfter- FirstStitchError:=", _ 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:=", _ </pre>

	<pre>1250, "RemoveHoles:=", false, "RemoveChamfers:=", false, "RemoveBlends:=", _ false, "HoleRadiusTol:=", 0, "ChamferWidthTol:=", 0, "BlendRadiusTol:=", 0, "Allow- ableSurfaceAreaChange:=", _ 5, "AllowableVolumeChange:=", 5)</pre>

Import

Imports a 3D model file.

Note:

This script does not import DXF or GDSII models. See: [ImportDXF](#) and [ImportGDSII](#).

UI Access	Modeler > Import.		
Parameters	Name	Type	Description
	<Parameters>	Array	Structured array. <pre>Array("NAME:NativeBodyParameters", "HealOption:=" , <integer>, "Options:=" , <string>, "FileType:=" , <string "UnRecognized">, "MaxStitchTol:=" , <integer>, "ImportFreeSurfaces:=" , <boolean>, "GroupByAssembly:=" , <boolean>, "CreateGroup:=" , <boolean>, "STLFileUnit:=" , <string>,</pre>

			<pre>"MergeFacesAngle:=" , <float>, "HealSTL:=" , <boolean>, "ReduceSTL:=" , <boolean>, "ReduceMaxError:=" , <integer>, "ReducePercentage:=" , <integer>, "PointCoincidenceTol:=" , <float>, "CreateLightweightPart:=", <boolean>, "ImportMaterialNames:=" , <boolean>, "SeparateDisjointLumps:=", <boolean>, "SourceFile:=" , <string>)</pre>
Return Value	None.		

Python Syntax	Import(<Parameters>)
Python Example	<pre>oEditor.Import(["NAME:NativeBodyParameters", "HealOption:=" , 0, "Options:=" , "-1", "FileType:=" , "UnRecognized", "MaxStitchTol:=" , -1, "ImportFreeSurfaces:=" , False,</pre>

	<pre> "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, "SourceFile:=" , "C:\\Users\\jdoe\\Desktop\\export.model" l) </pre>

VB Syntax	Import <Parameters>
VB Example	<pre> oEditor.Import Array("NAME:NativeBodyParameters", "HealOption:=" , 0, "Options:=" , "-1", "FileType:=" , "UnRecognized", "MaxStitchTol:=" , -1, </pre>

	<pre>"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, "SourceFile:=" , "C:\\Users\\jdoe\\Desktop\\export.model")</pre>

ImportDXF [Modeler]

Imports a DXF model file.

UI Access	Modeler > Import.		
Parameters	Name	Type	Description
	<Parameters>	Array	Structured array. Array("NAME:options",

		<pre> "FileName:=" , <string>, "Scale:=" , <float>, "AutoDetectClosed:=" , <boolean>, "SelfStitch:=" , <boolean>, "DefeatureGeometry:=" , <boolean>, "DefeatureDistance:=" , <integer>, "RoundCoordinates:=" , <boolean>, "RoundNumDigits:=" , <integer>, "WritePolyWithWidthAsFilledPoly:=" , <boolean>, "ImportMethod:=" , <integer>, "2DSheetBodies:=" , <boolean>, <layersArray>) </pre>
<layersArray>	Array	Structured array. <pre> Array("NAME:LayerInfo", <layer>, <layer>, <layer>, ...) </pre>
<layer>	Array	Structured array. <pre> Array("NAME:<layerName>", "source:=" , <string>, "display_source:=" , <string>, "import:=" , <boolean>, "dest:=" , <string>, "dest_selected:=" , <boolean>, </pre>

	<table><tr><td></td><td></td><td>"layer_type:=" , <string>)</td></tr></table>					"layer_type:=" , <string>)
		"layer_type:=" , <string>)				
Return Value	None.					

Python Syntax	ImportDXF(<Parameters>)
Python Example	<pre>oEditor.ImportDXF(["NAME:options", "FileName:=", "C:/Users/jdoe/Desktop/export.dxf", "Scale:=" , 0.001, "AutoDetectClosed:=" , True, "SelfStitch:=" , True, "DefeatureGeometry:=" , False, "DefeatureDistance:=" , 0, "RoundCoordinates:=" , False, "RoundNumDigits:=" , 4, "WritePolyWithWidthAsFilledPoly:=", False, "ImportMethod:=" , 1, "2DSheetBodies:=" , False, ["NAME:LayerInfo", ["NAME:0", "source:=" , "0",</pre>

```

"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"

```

	<pre>]]]) </pre>

VB Syntax	ImportDXF <Parameters>
VB Example	<pre> oEditor.ImportDXF Array("NAME:options", "FileName:=", _ "C:/Users/jdoe/Desktop/export.dxf", "Scale:=", 0.001, "AutoDetectClosed:=", _ true, "SelfStitch:=", true, "DefeatureGeometry:=", false, "DefeatureDistance:=", _ 0, "RoundCoordinates:=", false, "RoundNumDigits:=", 4, "WritePolyWithWidthAsFilledPoly:=", _ false, "ImportMethod:=", 1, "2DSheetBodies:=", false, Array("NAME:LayerInfo", Array ("NAME:0", "source:=", _ "0", "display_source:=", "0", "import:=", true, "dest:=", "0", "dest_selected:=", _ false, "layer_type:=", "signal"), Array("NAME:LAYER_1", "source:=", "LAYER_1", "dis- play_source:=", _ "LAYER_1", "import:=", true, "dest:=", "LAYER_1", "dest_selected:=", false, "layer_ type:=", _ "signal"), Array("NAME:LAYER_2", "source:=", "LAYER_2", "display_source:=", "LAYER_2", "import:=", _ true, "dest:=", "LAYER_2", "dest_selected:=", false, "layer_type:=", "signal"))) </pre>

ImportFromClipboard

Imports a model from clipboard.

UI Access	Modeler > Import From Clipboard.
Parameters	None.
Return Value	None.

Python Syntax	ImportFromClipboard ()
Python Example	<code>oEditor.ImportFromClipboard()</code>

VB Syntax	ImportFromClipboard
VB Example	<code>oEditor.ImportFromClipboard</code>

ImportGDSII [Modeler]

Imports a GDSII model file.

UI Access	Modeler > Import.		
Parameters	Name	Type	Description
	<Parameters>	Array	Structured array. <code>Array("NAME:options", "FileName:=" , <string>, "FlattenHierarchy:=" , <boolean>, "ImportMethod:=" , <integer>,"</code>

			<pre><layerMapArray>, <layerMapArray>, <layerMapArray>, ... "OrderMap:=" , ["entry:=" , <entry>, <entry>, <entry>, ...]])</pre>
	<layerMapArray>	Array	Structured array. Array("NAME:LayerMapInfo", "LayerNum:=" , <integer>, "DestLayer:=" , <string>, "layer_type:=" , <string>)
	<entry>	Array	Structured array. Array("order:=" , <integer LayerNum>, "layer:=" , <string DestLayer>)
Return Value	None.		

Python Syntax	ImportGDSII(<Parameters>)
Python Example	<pre>oEditor.ImportGDSII(["NAME:options", "FileName:=", "C:/Users/coil2.gds", "FlattenHierarchy:=" , True,</pre>

```
"ImportMethod:="          , 1,
["NAME:LayerMap",
  ["NAME:LayerMapInfo",
    "LayerNum:="          , 12,
    "DestLayer:="         , "Signal12",
    "layer_type:="        , "signal"
  ],
  ["NAME:LayerMapInfo",
    "LayerNum:="          , 13,
    "DestLayer:="         , "Signal13",
    "layer_type:="        , "signal"
  ],
  ["NAME:LayerMapInfo",
    "LayerNum:="          , 14,
    "DestLayer:="         , "Signal14",
    "layer_type:="        , "signal"
  ]
],
"OrderMap:=",
[
  "entry:=",
```



```
        ["order:=", 0, "layer:=", "Signal12"],  
        "entry:=",  
        ["order:=", 1, "layer:=", "Signal13"],  
        "entry:=",  
        ["order:=", 2, "layer:=", "Signal14"]  
    ]  
])
```

VB Syntax	ImportGDSII <Parameters>
VB Example	<pre>oEditor.ImportGDSII Array("NAME:options", "FileName:=", "C:/export.gds", "FlattenHierarchy:=", True, "ImportMethod:=", 1, Array("NAME:LayerMap", Array("NAME:LayerMapInfo", "LayerNum:=", 1, "DestLayer:=", "Signal1", "layer_type:=", "signal"]))) "OrderMap:=", Array("entry:=", Array(</pre>

	<pre>"order:=", 0, "layer:=", "Signal1")))</pre>

Imprint

Imprints the geometry of one object upon another.

UI Access	Modeler > Boolean > Imprint...		
Parameters	Name	Type	Description
	<Selections>	Array	Structured array. <pre>Array("NAME:Selections", "Blank Parts:=", <string, object name>, "Tool Parts:=", <string, object name>)</pre>
	<Parameters>	Array	Structured array. <pre>Array("NAME:ImprintParameters", "KeepOriginals:=", <boolean>)</pre>
Return Value	None.		

Python Syntax	Imprint (<Selections>, <Parameters>)
Python Example	<pre>oEditor.Imprint(["NAME:Selections", "Blank Parts:=", "Cylinder1", "Tool Parts:=", "Box1"],</pre>

	<pre>["NAME:ImprintParameters", "KeepOriginals:=", False])</pre>

VB Syntax	Imprint <Selections>, <Parameters>
VB Example	<pre>oEditor.Imprint Array("NAME:Selections", "Blank Parts:=", "Cylinder1", "Tool Parts:=", "Box1"), Array("NAME:ImprintParameters", "KeepOriginals:=", false)</pre>

ImprintProjection

Projects the form of a sheet object onto the face or faces of another object (either solid or sheet).

UI Access	Modeler > Boolean > Imprint Projection.		
Parameters	Name	Type	Description
	<Selections>	Array	Structured array. Array("NAME:Selections", "Selections:=", <string, selected objects>)
	<Parameters>	Array	Structured array. Array("NAME:ImprintProjectionParameters",

			<code>"KeepOriginals:=", <boolean>, "NormalProjection:=", <boolean>, "Distance:=", , <float>, "DirectionX:=", , <float>, "DirectionY:=", , <float>, "DirectionZ:=", , <float>)</code>
Return Value	None.		

Python Syntax	<code>ImprintProjection (<Selections>, <Parameters>)</code>
Python Example	<pre>oEditor.ImprintProjection(["NAME:Selections", "Selections:=", "Rectangle1,Cylinder1"], ["NAME:ImprintProjectionParameters", "KeepOriginals:=", False, "NormalProjection:=", True, "Distance:=", "1.36014705087354mm", "DirectionX:=", "0.882257546512569", "DirectionY:=", "0.294085848837523", "DirectionZ:=", "0.367607311046904"])</pre>

VB Syntax	ImprintProjection <Selections>, <Parameters>
VB Example	<pre>oEditor.ImprintProjection Array("NAME:Selections", "Selections:=", "Rectangle1,Cylinder1"), Array("NAME:ImprintProjectionParameters", "KeepOriginals:=", false, "NormalProjection:=", true, "Distance:=", "1.36014705087354mm", "DirectionX:=", "0.882257546512569", "DirectionY:=", "0.294085848837523", "DirectionZ:=", "0.367607311046904")</pre>

Intersect

Intersects specified objects.

UI Access	Modeler > Boolean > Intersect.		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: SelectionsArray .
	<Parameters>	Array	Structured array. Array("NAME:IntersectParameters", "KeepOriginals:=" , <boolean True to keep original objects; False to delete>)

Return Value	None.

Python Syntax	<code>Intersect(<SelectionsArray>, <Parameters>)</code>
Python Example	<pre>oEditor.Intersect(["NAME:Selections", "Selections:=", "Rectangle1,Rectangle2"], ["NAME:IntersectParameters", "KeepOriginals:=", False]) </pre>

VB Syntax	<code>Intersect <SelectionsArray> <Parameters></code>
VB Example	<pre>oEditor.Intersect Array("NAME:Selections", "Selections:=", "Rectangle1,Rectangle2") Array("NAME:IntersectParameters", "KeepOriginals:=", False) </pre>

MoveCStoEnd

Moves a specified Object Coordinate System to the end of the History tree.

UI Access	Modeler > Coordinate System > Move CS to End.

Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: SelectionsArray .
Return Value	None.		

Python Syntax	MoveCStoEnd (<SelectionsArray>)
Python Example	<pre>oEditor.MoveCStoEnd(["NAME:Selections", "Selections:=" , "ObjectCS1"])</pre>

VB Syntax	MoveCStoEnd <SelectionsArray>
VB Example	<pre>oEditor.MoveCStoEnd Array("NAME:Selections", "Selections:=", "ObjectCS1")</pre>

MoveEntityToGroup

Moves a specified entity or entities to a specified group.

UI Access	Drag item into the group in the history tree.		
Parameters	Name	Type	Description
	<Objects>	Array	Structured array. Selects the entity/entities to move. Array("Objects:=" , <array containing string object IDs>)

	<code><MoveEntityToGroup></code>	Array	Structured array. <code>Array("ParentGroup:=" , <string group name>)</code>
Return Value	None.		

Python Syntax	<code>MoveEntityToGroup (<Objects>, <MoveEntityToGroup>)</code>		
Python Example	<pre>oEditor.MoveEntityToGroup(["Objects:=", ["Box3"]], ["ParentGroup:=", "Box_Group"])</pre>		

VB Syntax	<code>MoveEntityToGroup <Objects> <MoveEntityToGroup></code>		
VB Example	<pre>oEditor.MoveEntityToGroup Array("Objects:=", Array("Box3")) Array("ParentGroup:=", "Box_Group")</pre>		

MoveFaces

Moves the specified faces along normal or along a vector.

UI Access	Modeler > Surface > Move Faces > Along [Normal/Vector].		
Parameters	Name	Type	Description
	<code><SelectionsArray></code>	Array	Structured array. See: SelectionsArray .
	<code><MoveFacesParametersArray></code>	Array	Structured array. <code>Array("NAME:Parameters",</code>

			<FacesOfOneObjToMove>, <FacesOfOneObjToMove>, ...)
	<FacesOfOneObjToMove>	Array	Structured array. <pre>Array ("Name:MoveFacesParameters", "MoveAlongNormalFlag:=", <boolean>, "OffsetDistance:=", <string>, "MoveVectorX:=", <string>, "MoveVectorY:=", <string>, "MoveVectorZ:=", <string>, "FacesToMove:=", <array>)</pre> MoveAlongNormalFlag specifies whether to move along the face normal or along a vector. If false, provide the MoveVectorX, MoveVectorY, and MoveVectorZ parameters. FacesToMove is an array of integers (the IDs of the faces to move).
Return Value	None		

Python Syntax	MoveFaces (<SelectionsArray>, <MoveFacesParametersArray>)
Python Example	<pre>oEditor.MoveFaces (["NAME:Selections", "Selections:=" , "Rectangle1",</pre>

	<pre> "NewPartsModelFlag:=" , "Model"], ["NAME:Parameters", ["NAME:MoveFacesParameters", "MoveAlongNormalFlag:=" , True, "OffsetDistance:=" , "1mm", "MoveVectorX:=" , "0mm", "MoveVectorY:=" , "0mm", "MoveVectorZ:=" , "0mm", "FacesToMove:=" , [183]]]])</pre>

VB Syntax	MoveFaces <SelectionsArray>, <MoveFacesParametersArray>
VB Example	<pre>oEditor.MoveFaces _ Array("NAME:Selections", "Selections:=", _ "Box2,Box1"), _ Array("NAME:Parameters", _ Array("NAME:MoveFacesParameters", _ "MoveAlongNormalFlag:=", true, _ "OffsetDistance:=", "1mm", _ "FacesToMove:=", Array(218)), Array("NAME:MoveFacesParameters", _ "MoveAlongNormalFlag:=", false, _</pre>

	<pre>"OffsetDistance:=", "1mm", _ "MoveVectorX:=", "1mm", _ "MoveVectorY:=", "0mm", _ "MoveVectorZ:=", "0mm", _ "FacesToMove:=", Array(185))</pre>

ProjectSheet

Project a sheet object, typically for modeling thin conformal deposits. Typically followed by **Thicken Sheet**.

UI Access	Click Modeler > Surface > Project Sheet .		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: SelectionsArray .
	<Parameters>	Array	Array containing string. Array("NAME:ProjectSheetParameters")
Return Value	None.		

Python Syntax	ProjectSheet (<SelectionsArray>, <Parameters>)
Python Example	<pre>oEditor.ProjectSheet(['NAME:Selections', 'Selections:=', 'Box1,Box2,Polyline1'], ['NAME:ProjectSheetParameters'])</pre>

	)

VB Syntax	ProjectSheet <SelectionsArray> <Parameters>
VB Example	<pre>oEditor.ProjectSheet Array("NAME:Selections", "Selections:=", "Box1,Box2,Polyline1"), Array("NAME:ProjectSheetParameters")</pre>

PurgeHistory

Purges the history of a specified object.

UI Access	Modeler > Purge History.		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: SelectionsArray .
Return Value	None.		

Python Syntax	PurgeHistory (<SelectionsArray>)
Python Example	<pre>oEditor.PurgeHistory(["NAME:Selections", "Selections:=", "Box2", "NewPartsModelFlag:=", "Model"]</pre>

	l)

VB Syntax	PurgeHistory <SelectionsArray>
VB Example	<pre>oEditor.PurgeHistory Array("NAME:Selections", "Selections:=", "Box2", "NewPartsModelFlag:=", "Model")</pre>

ReplaceWith3DComponent

Replaces the selection with a 3D component.

UI Access	None.		
Parameters	Name	Type	Description
	<CreateData>	Array	Structured array.
	<DesignData>	Array	Structured array.
	<ImageFile>	Array	Structured array.
Return Value	None.		

Python Syntax	ReplaceWith3DComponent(<CreateData>, <DesignData>, <ImageFile>)
Python Example	<pre>oEditor.ReplaceWith3DComponent(["NAME>CreateData",</pre>

```

"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"],
"HiddenParts:=", [],
"IncludedCS:=", [],
"DefaultHandle:=", "Global",
"IncludedParameters:=", ["bend_angle"],
"ParameterDescription:=", ["bend_angle:=", ""]
],
["NAME:DesignData",
"Excitations:=", ["1","2"]

```

	<pre>], ["NAME:ImageFile", "ImageFile:=", ""]) </pre>

VB Syntax	ReplaceWith3DComponent <CreateData>, <DesignData>, <ImageFile>
VB Example	<pre> oEditor.ReplaceWith3DComponent Array("NAME:CreateData", "ComponentName:=", _ "ConnectorOnly_wBCs", "Company:=", "", "Company URL:=", "", "Model Number:=", "", "Help URL:=", _ "", "Version:=", "1.0", "Notes:=", "", "IconType:=", "", "Owner:=", _ "MyName", "Email:=", "My@email.com", "Date:=", "3:07:37 PM Jun 25, 2019", _ "HasLabel:=", false, "IsEncrypted:=", false, "AllowEdit:=", false, _ "SecurityMessage:=", "", "Password:=", "", "EditPassword:=", "", _ "PasswordType:=", "UnknownPassword", "HideContents:=", true, "ReplaceNames:=", true, _ "ComponentOutline:=", "None", "IncludedParts:=", _ Array("pin", "solder1", "ConnectorDielectric", _ "ConnectorOuter", "pcap", "solderr"), "HiddenParts:=", Array(), _ "IncludedCS:=", Array(), "DefaultHandle:=", _ </pre>

	<pre>"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:=", "")</pre>

Section

Creates a 2D cross-section of the selection in the specified plane.

UI Access	Modeler > Surface > Section.		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: SelectionsArray .
	<Parameters>	Array	Structured array. Array("NAME:SectionToParameters", "CreateNewObjects:=" , <boolean>, "SectionPlane:=" , <string "XY", "YZ", or "ZX">, "SectionCrossObject:=" , <boolean>)
Return Value	None.		

Python Syntax	Section (<SelectionsArray>, <Parameters>)
Python Example	<pre> oEditor.Section(["NAME:Selections", "Selections:=" , "Cone1", "NewPartsModelFlag:=" , "Model"], ["NAME:SectionToParameters", "CreateNewObjects:=" , True, "SectionPlane:=" , "XY", "SectionCrossObject:=" , False]) </pre>

VB Syntax	Section <SelectionsArray> <Parameters>
VB Example	<pre> oEditor.Section Array("NAME:Selections", "Selections:=" , "Cone1", "NewPartsModelFlag:=" , "Model") Array("NAME:SectionToParameters", "CreateNewObjects:=" , True, </pre>

	<pre>"SectionPlane:=" , "XY", "SectionCrossObject:=" , False)</pre>

SeparateBody

Separates the bodies of specified multi-lump objects.

UI Access	Modeler > Boolean > Separate Bodies.		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: SelectionsArray .
Return Value	None.		

Python Syntax	SeparateBody (<SelectionsArray>)
Python Example	<pre>oEditor.SeparateBody(["NAME:Selections", "Selections:=", "Rectangle1,Circle1", "NewPartsModelFlag:=", "Model"])</pre>

VB Syntax	SeparateBody <SelectionsArray>
VB Example	<pre>oEditor.SeparateBody Array("NAME:Selections",</pre>

	<pre>"Selections:=", "Rectangle1,Circle1", "NewPartsModelFlag:=", "Model")</pre>

SetModelUnits

Sets the model units.

UI Access	Modeler > Units.		
Parameters	Name	Type	Description
	<Parameters>	Array	Structured array. Array("NAME:Units Parameter", "Units:=", <string>, "Rescale:=", <boolean True to rescale model; else False>) To see valid unit strings, select Modeler > Units .
Return Value	None.		

Python Syntax	SetModelUnits (<Parameters>)
Python Example	<pre>oEditor.SetModelUnits(["NAME:Units Parameter", "Units:=", "km", "Rescale:=", False</pre>

	l)

VB Syntax	SetModelUnits <Parameters>
VB Example	<pre>oEditor.SetModelUnits Array("NAME:Units Parameter", "Units:=", "km", "Rescale:=", False)</pre>

SetWCS

Sets the working coordinate system.

UI Access	Modeler > Coordinate System > Set Working CS.		
Parameters	Name	Type	Description
	<Parameters>	Array	Structured array. <pre>Array("NAME:SetWCS Parameter", "Working Coordinate System:=", <string CS name>, "RegionDepCSOk:=" , <boolean True if region-dependent; else False)</pre>
Return Value	None.		

Python Syntax	SetWCS (<Parameters>)
Python Example	oEditor.SetWCS (

	<pre>["NAME:SetWCS Parameter", "Working Coordinate System:=", "Global", "RegionDepCSOk:=", False]</pre>

VB Syntax	SetWCS <Parameters>
VB Example	<pre>oEditor.SetWCS Array("NAME:SetWCS Parameter", "Working Coordinate System:=", "Global", "RegionDepCSOk:=", False)</pre>

ShowWindow

Opens the active 3D Modeler window.

UI Access	None.
Parameters	None.
Return Value	None.

Python Syntax	ShowWindow
Python Example	<code>oEditor.ShowWindow()</code>

VB Syntax	ShowWindow
VB Example	oEditor.ShowWindow

Simplify

Converts a complex MCAD object into simpler primitives which are easy to mesh and solve.

UI Access	Modeler > Model Preparation > Simplify.		
Parameters	Name	Type	Description
	<Selections>	Array	Structured array. See: SelectionsArray .
	<Parameters>	Array	Structured array. <pre>Array("NAME:SimplifyParameters", "Type:=", <string fit type>, "ExtrusionAxis:=", <string>, "Cleanup:=", <boolean>, "Splitting:=", <boolean>, "SeparateBodies:=", <boolean>, "CloneBody:=", <boolean>, "Generate Primitive History:=", <boolean>, "NumberPointsCurve:=", <integer>, "LengthThresholdCurve:=", <integer>)</pre>
	<CreateGroup>	Array	Structured array. <pre>Array("CreateGroupsForNewObjects:=", <boolean>)</pre>

Return Value	None.

Python Syntax	Simplify (<Selections>, <Parameters>, <CreateGroup>)
Python Example	<pre>oEditor.Simplify(["NAME:Selections", "Selections:=", "Cylinder1,Box2", "NewPartsModelFlag:=", "Model"], ["NAME:SimplifyParameters", "Type:=", "Polygon Fit", "ExtrusionAxis:=", "Auto", "Cleanup:=", True, "Splitting:=", True, "SeparateBodies:=", False, "CloneBody:=", False, "Generate Primitive History:=", True, "NumberPointsCurve:=", 3, "LengthThresholdCurve:=", 20], ["CreateGroupsForNewObjects:=", True])</pre>

VB Syntax	Simplify <Selections>, <Parameters>, <CreateGroup>
VB Example	<pre> oEditor.Simplify Array("NAME:Selections", "Selections:=", "Cylinder1,Box2", "NewPartsModelFlag:=", "Model"), Array("NAME:SimplifyParameters", "Type:=", "Polygon Fit", "ExtrusionAxis:=", "Auto", "Cleanup:=", true, "Splitting:=", true, "SeparateBodies:=", false, "CloneBody:=", false, "Generate Primitive History:=", true, "NumberPointsCurve:=", 3, "LengthThresholdCurve:=", 20), Array("CreateGroupsForNewObjects:=", true) </pre>

Split

Splits the specified object(s) along a plane.

UI Access	Modeler > Boolean > Split.

Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: SelectionsArray .
	<Parameters>	Array	Structured array. <pre>Array("NAME:SplitToParameters", "SplitPlane:=" , <string "XY", "YZ", "ZX", or "Dummy">, "WhichSide:=" , <string "PositiveOnly", "Neg- ativeOnly" or "Both">, "ToolType:=" , <string "PlaneTool" or "FaceTool">, "ToolEntityID:=" , <-1 if using SplitPlane; else FaceID>, "SplitCrossingObjectsOnly:=", <boolean>, "DeleteInvalidObjects:=", <boolean>)</pre> You can split an object either along an existing plane , or by creating a plane using an edge. For existing plane: • SplitPlane is "XY", "YZ", or "ZX" • ToolType is "PlaneTool" • ToolEntityID is -1 For an edge: • SplitPlane is "Dummy" • ToolType is "EdgeTool" • ToolEntityID is the face ID

Return Value	None.

Python Syntax	Split(<SelectionsArray>, <Parameters>)
Python Example	<pre> oEditor.Split(["NAME:Selections", "Selections:=" , "Box1", "NewPartsModelFlag:=" , "Model"], ["NAME:SplitToParameters", "SplitPlane:=" , "XY", "WhichSide:=" , "PositiveOnly", "ToolType:=" , "PlaneTool", "ToolEntityID:=" , -1, "SplitCrossingObjectsOnly:=", False, "DeleteInvalidObjects:=", True]) </pre>

VB Syntax	Split <SelectionsArray> <Parameters>
VB Example	<pre> oEditor.Split Array("NAME:Selections", "Selections:=" , "Box1", </pre>

	<pre>"NewPartsModelFlag:=" , "Model") Array("NAME:SplitToParameters", "SplitPlane:=" , "XY", "WhichSide:=" , "PositiveOnly", "ToolType:=" , "PlaneTool", "ToolEntityID:=" , -1, "SplitCrossingObjectsOnly:=", False, "DeleteInvalidObjects:=", True)</pre>

Stitch

Stitches selected sheets.

UI Access	Modeler > Model Preparation > Stitch Sheets.		
Parameters	Name	Type	Description
	<Selections>	Array	Structured array. See: SelectionsArray .
	<Parameters>	Array	Structured array. Array("NAME:StitchParameters", "MaxTol:=", <integer maximum tolerance>)
Return Value	None.		

Python Syntax	Stitch (<Selections>, <Parameters>)

Python Example	<pre>oEditor.Stitch(["NAME:Selections", "Selections:=", "Rectangle3,Rectangle4", "NewPartsModelFlag:=", "Model"], ["NAME:StitchParameters", "MaxTol:=", -1])</pre>

VB Syntax	Stitch <Selections>, <Parameters>
VB Example	<pre>oEditor.Stitch Array("NAME:Selections", "Selections:=", "Rectangle3,Rectangle4", "NewPartsModelFlag:=", "Model"), Array("NAME:StitchParameters", "MaxTol:=", -1)</pre>

Subtract

Subtracts the specified object(s).

UI Access	Modeler > Boolean > Subtract.

Parameters	Name	Type	Description
	<Selections>	Array	Structured array. 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>)
Return Value	None.		

Python Syntax	Subtract(<Selections>, <Parameters>)
Python Example	<pre>oEditor.Subtract(["NAME:Selections", "Blank Parts:=", "Rectangle1", "Tool Parts:=", "Rectangle2"], ["NAME:SubtractParameters",</pre>

	<pre>"KeepOriginals:=", False]</pre>

VB Syntax	Subtract <Selections> <Parameters>
VB Example	<pre>oEditor.Subtract Array("NAME:Selections", "Blank Parts:=", "Rectangle1", "Tool Parts:=", "Rectangle2") Array("NAME:SubtractParameters", "KeepOriginals:=", false)</pre>

SweepFacesAlongNormal

Sweep the specified face(s) along normal.

UI Access	Modeler > Surface > Sweep Faces Along Normal		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: SelectionsArray .
	<parameters>	Array	Structured array. Array("NAME:Parameters", "NAME:SweepFaceAlongNormalToParameters", "FacesToDetach:=", <faceIDarray>, "LengthOfSweep:=", "<value><units>")
Return Value	None		

Python Syntax	SweepFacesAlongNormal(<SelectionsArray> <parameters>)
Python Example	<pre>oEditor.SweepFacesAlongNormal(["NAME:Selections", "Selections:=", "Rectangle1", "NewPartsModelFlag:=", "Model"], ["NAME:Parameters", "NAME:SweepFaceAlongNormalToParameters", "FacesToDetach:=", [183], "LengthOfSweep:=", "0.1mm"])</pre>

VB Syntax	SweepFacesAlongNormal <SelectionsArray> <parameters>
VB Example	<pre>oEditor.SweepFacesAlongNormal Array("NAME:Selections", "Selections:=", "Rectangle1", "NewPartsModelFlag:=", "Model"), Array("NAME:Parameters", "NAME:SweepFaceAlongNormalToParameters",</pre>

	<pre>"FacesToDetach:=", Array(57), "LengthOfSweep:=", "0.5mm"))</pre>

ThickenSheet

Thickens a sheet object to convert it to a 3D object.

UI Access	Modeler > Surface > Thicken Sheet		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: SelectionsArray .
	<parameters>	Array	Structured array. Array("NAME:SheetThickenParameters", "Thickness:=" , <string>, "BothSides:=" , <boolean>)
Return Value	None.		

Python Syntax	ThickenSheet (<SelectionsArray> <parameters>)
Python Example	<pre>oEditor.ThickenSheet(["NAME:Selections", "Selections:=" , "Rectangle3", "NewPartsModelFlag:=" , "Model"],</pre>

	<pre>["NAME:SheetThickenParameters", "Thickness:=" , "0.01mm", "BothSides:=" , False]</pre>

VB Syntax	ThickenSheet <SelectionsArray> <parameters>
VB Example	<pre>oEditor.ThickenSheet Array("NAME:Selections", "Selections:=" , "Rectangle3", "NewPartsModelFlag:=" , "Model") Array("NAME:SheetThickenParameters", "Thickness:=" , "0.01mm", "BothSides:=" , false)</pre>

UncoverFaces

Uncovers the specified face(s).

UI Access	Modeler > Surface > Uncover Faces		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: SelectionsArray .
	<parameters>	Array	Structured array.

			<pre>Array("NAME:Parameters", Array("NAME:UncoverFacesParameters", "FacesToUncover:=" , <array of face IDs>))</pre>
Return Value	None.		

Python Syntax	<code>UncoverFaces(<SelectionsArray> <parameters>)</code>
Python Example	<pre>oEditor.UncoverFaces(["NAME:Selections", "Selections:=" , "Box1", "NewPartsModelFlag:=" , "Model"], ["NAME:Parameters", ["NAME:UncoverFacesParameters", "FacesToUncover:=" , [12,16,18]]])</pre>

VB Syntax	<code>UncoverFaces <SelectionsArray> <parameters></code>
VB Example	<code>oEditor.UncoverFaces</code>

	<pre>Array("NAME:Selections", "Selections:=" , "Box1", "NewPartsModelFlag:=" , "Model") Array("NAME:Parameters", Array("NAME:UncoverFacesParameters", "FacesToUncover:=" , Array(12,16,18)))</pre>

Unite

Unites the specified objects.

UI Access	Modeler > Boolean > Unite		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: SelectionsArray .
	<parameters>	Array	Structured array. Array("NAME:UniteParameters", "KeepOriginals:=" , <boolean>, "TurnOnNBodyBoolean:=", <boolean>)
			Note: TurnOnNBodyBoolean is set to True by default. Enabled by Parasolid, this unites <i>n</i> bodies in a single step, rather than uniting them one by one.

Return Value	None.

Python Syntax	<code>Unite(<SelectionsArray>, <parameters>)</code>
Python Example	<pre>oEditor.Unite(["NAME:Selections", "Selections:=", "Rectangle1,Rectangle2"], ["NAME:UniteParameters", "KeepOriginals:=", False, "TurnOnNBodyBoolean:=", True]) </pre>

VB Syntax	<code>Unite <SelectionsArray> <parameters></code>
VB Example	<pre>oEditor.Unite Array("NAME:Selections", "Selections:=", "Rectangle1,Rectangle2") Array("NAME:UniteParameters", "KeepOriginals:=", false, "TurnOnNBodyBoolean:=", true) </pre>

Ungroup

Ungroups a specified history tree group.

UI Access	Modeler > Group > Ungroup		
Parameters	Name	Type	Description
	<Groups>	Array	Structured array.
	Array("Groups:=", <array of group names to ungroup>)		
Return Value	None		

Python Syntax	Ungroup(<Groups>)
Python Example	oEditor.Ungroup(["Groups:=", ["Group1"]])

VB Syntax	Ungroup <Groups>
VB Example	oEditor.Ungroup Array("Groups:=", Array("Group1"))

WrapSheet

Wraps a sheet object to another object.

UI Access	None.		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: SelectionsArray .

	<i><Parameters></i>	Array	Structured array. Array("NAME:WrapSheetParameters", "Imprinted:=", <boolean>)
Return Value	None.		

Python Syntax	WrapSheet (<SelectionsArray>, <Parameters>)
Python Example	<pre>oEditor.WrapSheet(["NAME:Selections", "Selections:=", "Rectangle1,Box1"], ["NAME:WrapSheetParameters", "Imprinted:=", True])</pre>

VB Syntax	WrapSheet <SelectionsArray> <Parameters>
VB Example	<pre>oEditor.WrapSheet Array("NAME:Selections", "Selections:=", "Rectangle1,Box1"), Array("NAME:WrapSheetParameters",</pre>

	"Imprinted:=", true)

Other oEditor Commands

[AddDefinitionFromBlock](#)

[AddDefinitionFromLibFile](#)

[BreakUDMConnection](#)

[ChangeProperty](#)

[Delete](#)

[FitAll](#)

[GetBodyNamesByPosition](#)

[GetChildNames \[Modeler\]](#)

[GetChildObject \[Modeler\]](#)

[GetChildTypes \[Modeler\]](#)

[GetEdgeByPosition](#)

[GetEdgeIDsFromObject](#)

[GetEdgeIDsFromFace](#)

[GetEntityListIDByName](#)

[GetExtendedDefinitionObject](#)

[GetFaceArea](#)

[GetFaceByPosition](#)

[GetFaceCenter](#)

[GetFaceIDs](#)

GetGeometryModelerMode

[GetMatchedObjectName](#)

[GetModelBoundingBox](#)

[GetModelUnits](#)

[GetNumObjects](#)

[GetObjectIDByName](#)

[GetObjectName](#)

[GetObjectNameByFaceID](#)

[GetObjectsByMaterial](#)

[GetObjectsInGroup](#)

[GetObjectVolume](#)

[GetPropertyValue](#)

[GetPropEvaluatedValue](#)

[GetPropNames](#)

[GetPropSIValue](#)

[GetPropValue](#)

[GetSelections](#)

[GetUserPosition](#)

[GetVertexIDsFromEdge](#)

[GetVertexIDsFromFace](#)

[GetVertexIDsFromObject](#)

[GetVertexPosition](#)

[PageSetup](#)

[RenamePart](#)

[SetPropValue \[Modeler\]](#)

AddDefinitionFromBlock

Adds a material definition from block text (same definition format as would be contained in the material library file) by library type (using definition folder name). This scripting command directly supports the .AMAT (or .ASURF) definition formats.

UI Access	N/A		
Parameters	Name	Type	Description
	<defBlock>	String	Text of the new material definition in block form.
	<defFolderName>	String	Library type (by definition folder name)
	<newTimeStamp>	String	New timestamp (time_t as integer number of seconds since 1/1/1970 12:00am, as string), default is current time
	<replaceExisting>	Boolean	True to replace existing, False to choose a new unique name if an existing definition is found
Return Value	A property scripting object for the definition.		

P- yt- h- o-	AddDefinitionFromBlock(<defBlock>, <defFolderName>, <newTimeStamp>, <replaceExisting>)

n S- y- nt- ax	
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") 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.

UI Access	N/A		
Parameters	Name	Type	Description
	<FilePath>	String	Path of the library file (i.e. AMAT file or ASURF file)
	<defName>	String	Which definition to use, required because a lib file can have multiple definitions.
	<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
Return Value	Property scripting object for the definition.		

Python Syntax	AddDefinitionFromLibFile(<FilePath>, <defName>, <defFolderName>, <newTimeStamp>,<replaceExisting>)
Python Example	<pre>oProject = oDesktop.NewProject() oProject.InsertDesign("HFSS", "HFSSDesign1", "DrivenModal", "") oDesign = oProject.SetActiveDesign("HFSSDesign1")</pre>

```
oEditor = oDesign.SetActiveEditor("3D Modeler")
oEditor.CreateBox(

    [

        "NAME:BoxParameters",

        "XPosition:="          , "-0.4mm",

            "YPosition:="      , "-1mm",
            "ZPosition:="      , "0mm",
            "XSize:="          , "1.4mm",
            "YSize:="          , "1.6mm",
            "ZSize:="          , "0.6mm"

    ],

    [

        "NAME:Attributes",

            "Name:="            , "Box1",

        "Flags:="              , "",
```

```
"Color:="                , "(143 175 143)",

"Transparency:="         , 0,

"PartCoordinateSystem:=", "Global",

"UDMId:="                , "",

"MaterialValue:="        , "\"vacuum\"",

"SurfaceMaterialValue:=", "\"\"",

"SolveInside:="          , True,

    "ShellElement:="      , False,

"ShellElementThickness:=", "0mm",

"IsMaterialEditable:="    , True,

"UseMaterialAppearance:=", False,

"IsLightweight:="         , False

])

oDefinitionManager = oProject.GetDefinitionManager()
```



```
scriptDir = os.path.dirname(os.path.realpath(__file__))
amatFilePath = scriptDir + '/material0.txt'
materialName = "material0"
materialNameInQuotes = "\"" + materialName + "\""
added = oDefinitionManager.AddDefinitionFromLibFile(amatFilePath, materialName, "Materials", "")
addedName = ''
    if isinstance(added, basestring):
addedName = added
    elif isinstance(added, list):
addedName = added[0]
else:
    addedName = added.GetName().replace("Materials:", "")
AddInfoMessage(os.path.basename(__file__) + " result: " + addedName)
oEditor.ChangeProperty(
[
    "NAME:AllTabs",
    [
        "NAME:Geometry3DAttributeTab",
        [
```

```
        "NAME:PropServers",
        "Box1 "
    ],
    [
        "NAME:ChangedProps",
        [
            "NAME:Material",
            "Value:=", materialNameInQuotes
        ]
    ]
]
])
```

AssignSurfaceMaterial

Assigns a material to specified surfaces.

UI Access	N/A		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: SelectionsArray .
	<AttributesArray>	Array	Structured array. See: AttributesArray . This script supports the following attributes: - MaterialValue - SolveInside

			• ShellElement • ShellElementThickness • IsMaterialEditable • UseMaterialAppearance • IsLightweight
Return Value	None.		

Python Syntax	AssignSurfaceMaterial(<SelectionsArray>, <AttributesArray>)

Python Example	<pre>oEditor.AssignSurfaceMaterial(["NAME:Selections", "AllowRegionDependentPartSelectionForPMLCreation:=", True, "AllowRegionSelectionForPMLCreation:=", True, "Selections:=" , "Rectangle1"], ["NAME:Attributes", "MaterialValue:=" , "diamond", "SolveInside:=" , False, "ShellElement:=" , False, "ShellElementThickness:=" , "nan", "IsMaterialEditable:=" , True, "UseMaterialAppearance:=" , False, "IsLightweight:=" , False])</pre>
VB Syntax	AssignSurfaceMaterial <SelectionsArray> <AttributesArray>

VB Example	<pre>oEditor.AssignSurfaceMaterial Array("NAME:Selections", "AllowRegionDependentPartSelectionForPMLCreation:=", true, "AllowRegionSelectionForPMLCreation:=", true, "Selections:=" , "Rectangle1") Array("NAME:Attributes", "MaterialValue:=" , "diamond", "SolveInside:=" , false, "ShellElement:=" , false, "ShellElementThickness:=" , "nan", "IsMaterialEditable:=" , true, "UseMaterialAppearance:=" , false, "IsLightweight:=" , false)</pre>

BreakUDMConnection

Breaks a current UDM connection to Discovery.

UI Access	Break Connection.		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: SelectionsArray .
Return Value	None.		

Python Syntax	BreakUDMConnection (<SelectionsArray>)
Python Example	<pre>oEditor.BreakUDMConnection (["NAME:Selections", "Selections:=" , "Discovery1"]) </pre>

VB Syntax	BreakUDMConnection <SelectionsArray>
VB Example	<pre>oEditor.BreakUDMConnection Array("NAME:Selections", "Selections:=", "") </pre>

ChangeProperty

Changes the properties of an object in the history tree.

UI Access	Right-click an object in the History Tree and select Properties .		
Parameters	Name	Type	Description
	<propertyArgs>	Array	Structured array. The properties vary depending on the object. Due to the number of potential configurations, it is recommended that you generate this script using the UI's Automation tab.
Return Value	None.		

Python Syntax	ChangeProperty(<propertyArgs>)
Python Example	Example: Changing the Position of a Box and the Reference Temperature (for Structural Solutions only) <pre>oEditor.ChangeProperty(["NAME:AllTabs", ["NAME:Geometry3DCmdTab", ["NAME:PropServers" , "Box1:CreateBox:1"], ["NAME:ChangedProps", ["NAME:Position", "X:=" , "0.35in", "Y:=" , "0.55in", "Z:=" , "0in"], ["NAME:Reference Temperature", # Applicable only to Mechanical- "Value:=" , "27cel" # Structural solutions]]])</pre> Example: Changing a Box's Material and Wireframe Display <pre>oEditor.ChangeProperty(</pre>

```
[
  "NAME:AllTabs",
  [
    "NAME:Geometry3DAttributeTab",
    [
      "NAME:PropServers" ,
      "Box1"
    ],
    [
      "NAME:ChangedProps",
      [
        "NAME:Material",
        "Value:="          , "\"vacuum\""
      ],
      [
        "NAME:Display Wireframe",
        "Value:="          , True
      ]
    ]
  ]
]
))
```

Example: Change Default Handle (reference coordinate system)

```
oEditor.ChangeProperty(
  [
    "NAME:AllTabs",
    [
      "NAME:General",
      [
        "NAME:PropServers",
        "HFSSDesign1_1"
      ],
      [
```



```

        "NAME:ChangedProps",
        [
            "NAME:Handle",
            "Value:="                , " NotchOutY"
        ]
    ]
]
])

oEditor.ChangeProperty(
[
    "NAME:AllTabs",
    [
        "NAME:Component Data",
        [
            "NAME:PropServers",
            "BoundarySetup:LC1_1_Port1"
        ],
        [
            "NAME:ChangedProps",
            [
                "NAME:Voltage",
                "Value:="            , "10mV"
            ]
        ]
    ]
]
])
...

```

Example: Change Material Import for a Discovery Model

```

oEditor = oDesign.SetActiveEditor("3D Modeler")
oEditor.ChangeProperty(
[

```

	<pre> "NAME:AllTabs", ["NAME:Options", ["NAME:PropServers", "Discovery1"], ["NAME:ChangedProps", ["NAME:Materials", "Value:=" , "Assignments and Properties"]]]]) </pre>

VB Syntax	ChangeProperty <propertyArgs>
VB Example	Example: Changing the Position of a Box and the Reference Temperature (for Structural Solutions only) <pre> oEditor.ChangeProperty Array("NAME:AllTabs", Array("NAME:Geometry3DCmdTab", Array("NAME:PropServers" , "Box1:CreateBox:1"), Array("NAME:ChangedProps", Array("NAME:Position", "X:=" , "0.35in", "Y:=" , "0.55in", "Z:=" , "0in"), </pre>

```

        Array("NAME:Reference Temperature", '
        "Value:=", "27cel"                '
    )
    )
)

```

Applicable only to Mechanical-
Structural solutions

Example: Changing a Box's Material and Wireframe Display

```

oEditor.ChangeProperty(
Array("NAME:AllTabs",
    Array("NAME:Geometry3DAttributeTab",
        Array("NAME:PropServers",
            "Box1"
        ),
        Array("NAME:ChangedProps",
            Array("NAME:Material",
                "Value:=", "\"vacuum\""
            ),
            Array("NAME:Display Wireframe",
                "Value:=", True
            )
        )
    )
)

```

Example: Change Default Handle (reference coordinate system)

```

oEditor.ChangeProperty
Array("NAME:AllTabs",
    Array("NAME:General",
        Array("NAME:PropServers",

```

	<pre> "HFSSDesign1_1"), Array("NAME:ChangedProps", Array("NAME:Handle", "Value:=" , " NotchOutY")))</pre>

CloseAllWindows[Editor]

Closes all windows belong to current 3D Modeler editor.

UI Access	N/A
Parameters	None.
Return Value	None.

Python Syntax	CloseAllWindows()
Python Example	oEditor.CloseAllWindows()

VB Syntax	CloseAllWindows
VB Example	oEditor.CloseAllWindows

Defeature

Removes irrelevant features from a primitive.

UI Access	N/A		
Parameters	Name	Type	Description
	<Selections>	Array	Structured array. See: SelectionsArray .
	<Options>	Array	Structured array. Array("NAME:options", "tolerance:=", <double containing tolerance value>, "fix:=", <boolean, if true, then self-intersections are fixed.>)
Return Value	None.		

Python Syntax	Defeature(<Selections>, <Options>)
Python Example	<pre>oEditor.Defeature(["NAME:Selections", "Selections:=", "Box1"], ["NAME:Options", "Tolerance:=", 1E-006, "Fix:=", True])</pre>

VB Syntax	Defeature <Selections>, <Options>
VB Example	<pre>oEditor.Defeature</pre>

```
Array("NAME:Selections", "Selections:=", "Box1"),
Array("NAME:Options", "tolerance:=", 1E-006, "fix:=", true)
```

Delete

Deletes the specified object(s).

UI Access	Edit > Delete.		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: SelectionsArray .
Return Value	None.		

Python Syntax	Delete(<SelectionsArray>)
Python Example	<pre>oEditor.Delete(["NAME:Selections", "Selections:=", "Rectangle1,Rectangle2"]) </pre>

VB Syntax	Delete <SelectionsArray>
VB Example	<pre>oEditor.Delete Array("NAME:Selections", "Selections:=", "Rectangle1,Rectangle2") </pre>

FitAll

Fits the design to the modeling area.

UI Access	View > Fit All > All Views.
Parameters	None.
Return Value	None.

Python Syntax	FitAll()
Python Example	<code>oEditor.FitAll()</code>

VB Syntax	FitAll()
VB Example	<code>Set oEditor = oDesign.SetActiveEditor("3D Modeler") oEditor.FitAll()</code>

GenerateAllUserDefinedModels

Generates all user defined models.

UI Access	N/A
Parameters	None.
Return Value	Array of models generated.

Python Syntax	GenerateAllUserDefinedModels ()

Python Example	<code>oEditor.GenerateAllUserDefinedModels()</code>

VB Syntax	<code>GenerateAllUserDefinedModels</code>
VB Example	<code>oEditor.GenerateAllUserDefinedModels</code>

GenerateUserDefinedModel

Generates specified user defined model(s).

UI Access	N/A		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: SelectionsArray .
Return Value	Array of models generated.		

Python Syntax	<code>GenerateUserDefinedModel (<SelectionsArray>)</code>
Python Example	<code>oEditor.GenerateUserDefinedModel(["NAME:Selections", "Selections:=", "model1, model2"])</code>

VB Syntax	<code>GenerateAllUserDefinedModels</code>
VB Example	<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	Type	Description
	<positionParameters>	Array	Structured array containing position coordinates for active coordinate system. Array ("NAME:Parameters", "XPosition=", <string>, "YPosition=", <string>, "ZPosition=", <string>)
Return Value	Array containing string object names.		

Python Syntax	GetBodyNamesByPosition (<positionParameters>)
Python Example	<pre>oEditor.GetBodyNamesByPosition(["NAME:Parameters", "XPosition=", "0mm", "YPosition=", "15mm", "ZPosition=", "0mm"]</pre>

	<pre>]) </pre>

VB Syntax	GetBodyNamesByPosition <positionParameters>
VB Example	<pre> oEditor.GetBodyNamesByPosition Array("NAME:Parameters", _ "XPosition:=", "0mm", _ "YPosition:=", "15mm", _ "ZPosition:=", "0mm") </pre>

GetChildNames [Modeler]

Returns the names of children for a specified input. For 3D Components and UDMs, these commands do not return parts, coordinate systems, plans, as top-level modeler children.

Note:

This command is not supported by the EMIT and Circuit design types.

UI Access	N/A		
Parameters	Name	Type	Description
	<category>	String	<i>Optional.</i> Passing no input returns the list of possible strings: "AllParts" – Returns the names of all parts. "CoordinateSystems" – Returns the names of all coordinate systems. "Groups" – Returns the names of all groups.

			• "Lists" – Returns the names of all lists. • "ModelParts" – Returns names of model parts. • "NonModelParts" – Returns the names of non-model parts. • "Planes" – Returns the names of all planes. • "Points" – Returns the names of all points. • "SubmodelDefinitions" – Returns the names of sub-model definitions.
Return Value	Array containing object names in the selected category.		

Python Syntax	<code>GetChildNames(<category>)</code>
Python Example	Standalone Example: <pre>oEditor.GetChildNames("ModelParts") oEditor.GetChildNames("Points")</pre> Example used in Conjunction with GetChildObject and GetPropNames: <pre>oDesign = oProject.GetActiveDesign() oModel = oDesign.GetChildObject("3D Modeler") oModel.GetChildTypes()</pre> • This returns an array containing strings: "ModelParts", "AllParts", "NonModelParts", "CoordinateSystems", "Points", "Planes", "SubmodelDefinitions", "Groups", and "Lists". <pre>oModel.GetChildNames("ModelParts")</pre>

	- This returns an array containing string model parts. For example: "WG_Interior", "WG", "MT_Interior", "MT", "Box2", "Cylinder1". <pre>oWGi = oModel.GetChildObject("WG_Interior")</pre> - This sets the object WG_Interior to variable oWGi. <pre>oWGi.GetPropNames()</pre> - This returns property names from child object assigned to oWGi. In this case: "Name", "Material", "Material/SIValue", "Material/EvaluatedValue", "Solve Inside", "Orientation", "Orientation/Choices", "Model", "Display Wireframe", "Material Appearance", "Color", "Color/Red", "Color/Green", "Color/Blue", and "Transparent".

VB Syntax	GetChildNames <category>
VB Example	Standalone Example: <pre>oEditor.GetChildNames "ModelParts" oEditor.GetChildNames "Points"</pre> Example used in Conjunction with GetChildObject and GetPropNames: <pre>Dim oAnsoftApp Dim oDesktop Dim oProject Dim oDesign Dim oEditor Set oDesign = oProject.GetActiveDesign() Set oModel = oDesign.GetChildObject "3D Modeler"</pre>

	<pre>oModel.GetChildTypes()</pre> - This returns an array containing strings: "ModelParts", "AllParts", "NonModelParts", "CoordinateSystems", "Points", "Planes", "SubmodelDefinitions", "Groups", and "Lists". <pre>oModel.GetChildNames "ModelParts"</pre> - This returns an array containing string model parts. For example: "WG_Interior", "WG", "MT_Interior", "MT", "Box2", "Cylinder1". <pre>Set oWGi = oModel.GetChildObject "WG_Interior"</pre> - This sets the object WG_Interior to variable oWGi. <pre>oWGi.GetPropNames()</pre> - This returns property names from child object assigned to oWGi. In this case: "Name", "Material", "Material/SIValue", "Material/EvaluatedValue", "Solve Inside", "Orientation", "Orientation/Choices", "Model", "Display Wireframe", "Material Appearance", "Color", "Color/Red", "Color/Green", "Color/Blue", and "Transparent".

GetChildObject [Modeler]

Returns a 3D modeler child object, which can be assigned to a variable. Will return normally if there are no active objects. For 3D Components and UDMs, these commands do not return parts, coordinate systems, plans, as top-level modeler children.

Note: This command is not supported by the EMIT and Circuit design types.

UI Access	N/A		
Parameters	Name	Type	Description

	<code><object></code>	String	In this case, "3D Modeler".
Return Value	Returns the 3D Modeler object. In the examples below, it is assigned to the variable oEditor.		

Python Syntax	GetChildObject(<object>)
Python Example	Standalone Example: <pre>oDesign = oProject.GetActiveDesign() oEditor = oDesign.GetChildObject("3D Modeler")</pre> Example used in Conjunction with GetChildNames and GetPropNames: <pre>oDesign = oProject.GetActiveDesign() oModel = oDesign.GetChildObject("3D Modeler") oModel.GetChildTypes()</pre> - This returns an array containing strings: "ModelParts", "AllParts", "NonModelParts", "CoordinateSystems", "Points", "Planes", "SubmodelDefinitions", "Groups", and "Lists". <pre>oModel.GetChildNames("ModelParts")</pre> - This returns an array containing string model parts. For example: "WG_Interior", "WG", "MT_Interior", "MT", "Box2", "Cylinder1". <pre>oWGi = oModel.GetChildObject("WG_Interior")</pre> - This sets the object WG_Interior to variable oWGi. <pre>oWGi.GetPropNames()</pre> - This returns property names from child object assigned to oWGi. In this case: "Name", "Material", "Material/SIValue", "Material/EvaluatedValue", "Solve Inside", "Orientation", "Orientation/Choices", "Model", "Dis-

	play Wireframe", "Material Appearance", "Color", "Color/Red", "Color/Green", "Color/Blue", and "Transparent".

VB Syntax	GetChildObject<object>
VB Example	Standalone Example: <pre>Dim oAnsoftApp Dim oDesktop Dim oProject Dim oDesign Dim oEditor Set oDesign = oProject.GetActiveDesign() Set oEditor = oDesign.GetChildObject "3D Modeler"</pre> Example used in Conjunction with GetChildNames and GetPropNames: <pre>Dim oAnsoftApp Dim oDesktop Dim oProject Dim oDesign Dim oEditor Set oDesign = oProject.GetActiveDesign() Set oModel = oDesign.GetChildObject "3D Modeler"</pre>

	<pre>oModel.GetChildTypes()</pre> • This returns an array containing strings: "ModelParts", "AllParts", "NonModelParts", "CoordinateSystems", "Points", "Planes", "SubmodelDefinitions", "Groups", and "Lists". <pre>oModel.GetChildNames "ModelParts"</pre> • This returns an array containing string model parts. For example: "WG_Interior", "WG", "MT_Interior", "MT", "Box2", "Cylinder1". <pre>Set oWGi = oModel.GetChildObject "WG_Interior"</pre> • This sets the object WG_Interior to variable oWGi. <pre>oWGi.GetPropNames()</pre> • This returns property names from child object assigned to oWGi. In this case: "Name", "Material", "Material/SIValue", "Material/EvaluatedValue", "Solve Inside", "Orientation", "Orientation/Choices", "Model", "Display Wireframe", "Material Appearance", "Color", "Color/Red", "Color/Green", "Color/Blue", and "Transparent".

GetChildTypes [Modeler]

Gets child types of queried designs or editors obtained by [GetActiveProject\(\)](#) and [GetActiveDesign\(\)](#) commands. For 3D Components and UDMs, these commands do not return parts, coordinate systems, plans, as top-level modeler children.

Note:

This command is not supported by the EMIT and Circuit design types.

UI Access	N/A
Parameters	None.
Return Value	Array containing the names of child types.

Python Syntax	GetChildTypes()
Python Example	Standalone Example: <pre>oDesign = oProject.GetActiveDesign() oEditor = oDesign.SetActiveEditor("3D Modeler") oEditor.GetChildTypes()</pre> • This returns an array containing strings: "ModelParts", "AllParts", "NonModelParts", "CoordinateSystems", "Points", "Planes", "SubmodelDefinitions", "Groups", and "Lists". Example Used in Conjunction with GetChildNames: <pre>oProject = oDesktop.GetActiveProject() oDesign = oProject.GetActiveDesign() oDesign.GetChildNames()</pre> • This returns an array containing names of child objects for the design. For example: "Boundaries", "Nets", "Analysis", "Optimetrics", "Radiation", "Results", and "3D Modeler". <pre>oDesign.GetChildTypes()</pre> • This returns an array containing the child types. For example: "Module", "Editor", and "Variable".

VB Syntax	GetChildTypes()
VB Example	Standalone Example:

```

Dim oDesign
Dim oProject
Dim oEditor
Set oDesign = oProject.GetActiveDesign()
Set oEditor = oDesign.SetActiveEditor "3D Modeler"
oEditor.GetChildTypes()

```

- This returns an array containing strings: "ModelParts", "AllParts", "NonModelParts", "CoordinateSystems", "Points", "Planes", "SubmodelDefinitions", "Groups", and "Lists".

Example Used in Conjunction with [GetChildNames](#):

```

Dim oDesign
Dim oProject
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	Type	Description
	<positionParameters>	Array	Structured array. <pre>Array("NAME:EdgeParameters", "BodyName:=", <string object name>, "Xposition:=", <string>, "YPosition:=", <string>, "ZPosition:=", <string>)</pre> Note: For 2D XY Designs, ZPosition should be set to "0". For 2D RZ Designs, YPosition should be set to "0".
Return Value	Array containing string edge IDs.		

Python Syntax	GetEdgeByPosition (<positionParameters>)
Python Example	<pre>oEditor.GetEdgeByPosition(["NAME:EdgeParameters", "BodyName:=", "Box1", "Xposition:=", "10mm", "YPosition:=", "0mm",</pre>

	<pre> "ZPosition:=", "10mm"]) </pre>

VB Syntax	GetEdgeByPosition <positionParameters>
VB Example	<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.

UI Access	N/A		
Parameters	Name	Type	Description
	<PartName>	String	Name of specified part.
	<EdgeName>	String	Name of specified edge.
Return Value	Integer edge ID		

Python Syntax	GetEdgeIDFromNameForFirstOperation(<PartName>, <EdgeName>)
Python Example	<pre> oEditor.GetEdgeIDFromNameForFirstOperation("Coil_1", "Edge_4510") </pre>

VB Syntax	GetEdgeIDFromNameForFirstOperation <PartName>, <EdgeName>
VB Example	<code>oEditor.GetEdgeIDFromNameForFirstOperation "Coil_1", "Edge_4510"</code>

GetEdgeIDsFromFace

Returns the edge IDs for a specified Face ID.

UI Access	N/A		
Parameters	Name	Type	Description
	<faceID>	Integer	ID of the specified face.
Return Value	Array containing string edge IDs.		

Python Syntax	GetEdgeIDsFromFace (<faceID>)
Python Example	<code>oEditor.GetEdgeIDsFromFace (20)</code>

VB Syntax	GetEdgeIDsFromFace <faceID>
VB Example	<code>oEditor.GetEdgeIDsFromFace 20</code>

GetEdgeIDsFromObject

Returns the edge IDs for a specified object.

UI Access	N/A		
Parameters	Name	Type	Description
	<object>	String	Object name.
Return Value	Array containing string edge IDs.		

Python Syntax	GetEdgeIDsFromObject (<object>)		
Python Example	oEditor.GetEdgeIDsFromObject ("Box1")		

VB Syntax	GetEdgeIDsFromObject <object>		
VB Example	oEditor.GetEdgeIDsFromObject "Box1"		

GetEdgeLength

Returns the length of edges for a specified Edge ID.

UI Access	N/A		
Parameters	Name	Type	Description
	<edgeID>	Integer	ID of the specified edge.
Return Value	Integer containing the length of found edges.		

Python Syntax	GetEdgeLength (<EdgeID>)		
Python Example	oEditor.GetEdgeLength (20)		

VB Syntax	GetEdgeLength <edgeID>
VB Example	oEditor.GetEdgeLength 20

GetEntityListIDByName

Returns a list of IDs in a given entity list (either object or face). See: [CreateEntityList](#).

UI Access	N/A		
Parameters	Name	Type	Description
	<listName>	String	List name.
Return Value	Array containing string object IDs.		

Python Syntax	GetEntityListIDByName (<listName>)
Python Example	oEditor.GetEntityListIDByName ("MyList")

VB Syntax	GetEntityListIDByName <listName>
VB Example	oEditor.GetEntityListIDByName "MyList"

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.

UI Access	N/A		
Parameters	Name	Type	Description
	<defName>	String	Definition to retrieve.
	<defFolderName>	String	Library type (by definition folder name).
Return Value	An extended material wrapper script object (see material wrapper script object functions above).		

P- y- t- h- o- n S- y- n- t- a- x	GetExtendedDefinitionObject(<defName>, <defFolderName>)
P- y- t- h- o- n E- x- a- m-	<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)</pre>

pl-e	<code>oMat = oDefMgr.GetExtendedDefinitionObject('vacuum2', 'Materials')</code>
	<code># use extended definition object function(s) to manipulate the definition</code>
	<code>oMat.SetModTime("1630611487")</code>

GetFaceArea

Returns the area of a specified face.

UI Access	N/A		
Parameters	Name	Type	Description
	<faceID>	Integer	ID of specified face.
Return Value	Long integer of face area.		

Python Syntax	<code>GetFaceArea (<faceID>)</code>
Python Example	<code>oEditor.GetFaceArea(19)</code>

VB Syntax	<code>GetFaceArea <faceID></code>
VB Example	<code>oEditor.GetFaceArea 19</code>

GetFaceCenter

Returns the center position of a specified face.

UI Access	N/A		
Parameters	Name	Type	Description
	<planarFaceID>	Long	Face ID
Return Value	Array containing string X, Y, Z coordinates.		

Python Syntax	GetFaceCenter (<planarFaceID>)		
Python Example	oEditor.GetFaceCenter(12)		

VB Syntax	GetFaceCenter <planarFaceID>		
VB Example	oEditor.GetFaceCenter 12		

GetFaceByPosition

Returns the face ID located at a specified position.

Note:

The coordinates must point to exactly one face, not a vertex or edge where two or more faces join.

UI Access	N/A		
Parameters	Name	Type	Description
	<FaceParameters>	Array	Structured Array. Array("NAME:FaceParameters",

			"BodyName:=", <string>, "XPosition:=", <string>, "YPosition:=", <string>, "ZPosition:=", <string>)
Return Value	Integer Face ID.		

Python Syntax	GetFaceByPosition(<FaceParameters>)
Python Example	<pre>oEditor.GetFaceByPosition(["NAME:FaceParameters", "BodyName:=", "Box1", "XPosition:=", "0.2mm", "YPosition:=", "-0.2mm", "ZPosition:=", "0.4mm"])</pre>

VB Syntax	GetFaceByPosition <FaceParameters>
VB Example	<pre>dim FaceID FaceID = oEditor.GetFaceByPosition Array("NAME:FaceParameters", "BodyName:=", "Box1",</pre>

	<pre>"XPosition:=", "0.2mm", "YPosition:=", "-0.2mm", "ZPosition:=", "0.4mm")</pre>

GetFaceIDFromNameForFirstOperation

Gets face ID from first operation of a part.

UI Access	N/A		
Parameters	Name	Type	Description
	<PartName>	String	Name of specified part.
	<FaceName>	String	Name of specified face.
Return Value	Integer face ID		

Python Syntax	GetFaceIDFromNameForFirstOperation(<PartName>, <FaceName>)
Python Example	oEditor.GetFaceIDFromNameForFirstOperation("Rotor", "Face_14343")

VB Syntax	GetFaceIDFromNameForFirstOperation <PartName>, <FaceName>
VB Example	oEditor.GetFaceIDFromNameForFirstOperation "Rotor", "Face_14343"

GetFaceIDs

Returns the face IDs associated with a specified object.

UI Access	N/A

Parameters	Name	Type	Description
	<objectName>	String	Object name.
Return Value	Array containing string face IDs.		

Python Syntax	GetFaceIDs (<objectName>)
Python Example	oEditor.GetFaceIDs ('Box1')

VB Syntax	GetFaceIDs <objectName>
VB Example	oEditor.GetFaceIDs "Box1"

GetFaceIDsOfSheet

Returns the face IDs associated with a specified sheet object.

UI Access	N/A		
Parameters	Name	Type	Description
	<sheetName>	String	Name of specified sheet object.
Return Value	Array containing string face IDs.		

Python Syntax	GetFaceIDsOfSheet(<sheetName>)
Python Example	oEditor.GetFaceIDsOfSheet ('Sheet1')

VB Syntax	GetFaceIDsOfSheet < <i>sheetName</i> >
VB Example	<code>oEditor.GetFaceIDsOfSheet "Sheet1"</code>

GetMatchedObjectName

Returns all object names containing the input text string.

UI Access	N/A		
Parameters	Name	Type	Description
	< <i>wildcardText</i> >	String	Text string present in object name(s).
Return Value	Array containing string object names.		

Python Syntax	GetMatchedObjectName (< <i>wildcardText</i> >)
Python Example	<code>oEditor.GetMatchedObjectName('Box*')</code>
	<code>oEditor.GetMatchedObjectName('?ox?')</code>

VB Syntax	GetMatchedObjectName < <i>wildcardText</i> >
VB Example	<code>oEditor.GetMatchedObjectName "Box*"</code>

GetModelBoundingBox

Returns the bounding box of the current model using the global coordinate system as reference. The outputs are in the defined modeler units. Only modeled objects are considered for this computation (non-modeled objects are ignored).

UI Access	N/A
Parameters	None.
Return Value	Array containing string Xmin, Ymin, Zmin, Xmax, Ymax, and Zmax values of the bounding box in the global coordinate system

Python Syntax	GetModelBoundingBox()
Python Example	<code>oEditor.GetModelBoundingBox()</code>

VB Syntax	GeModelBoundingBox
VB Example	<code>oEditor.GeModelBoundingBox</code>

GetModelUnits

Returns the model's unit of measure.

UI Access	N/A
Parameters	None.
Return Value	String containing unit of measure.

Python Syntax	GetModelUnits()

Python Example	<code>oEditor.GetModelUnits()</code>

VB Syntax	<code>GetModelUnits</code>
VB Example	<code>oEditor.GetModelUnits</code>

GetNumObjects

Returns the number of objects in a design.

UI Access	N/A
Parameters	None.
Return Value	Integer number of objects.

Python Syntax	<code>GetNumObjects()</code>
Python Example	<code>oEditor.GetNumObjects()</code>

VB Syntax	<code>GetNumObjects</code>
VB Example	<code>oEditor.GetNumObjects(</code>

GetObjectIDByName

Given an object's name, returns its ID. IDs are used with [CreateEntityList](#).

UI Access	N/A		
Parameters	Name	Type	Description
	<objectName>	String	Object name.
Return Value	Integer object ID.		

Python Syntax	GetObjectIDByName(<objectName>)		
Python Example	oEditor.GetObjectIDByName('Box1')		

VB Syntax	GetObjectIDByName <objectName>		
VB Example	oEditor.GetObjectIDByName "Box1"		

GetObjectName

Returns an object's name from its specified base index (creation order).

UI Access	N/A		
Parameters	Name	Type	Description
	<index>	Integer	Base index (where '0' is the first item created)
Return Value	String containing object name.		

Python Syntax	GetObjectName(<index>)		

Python Example	<code>oEditor.GetObjectName(3)</code>

VB Syntax	<code>GetObjectName <index></code>
VB Example	<code>oEditor.GetObjectName 3</code>

GetObjectNameByEdgeID

Returns an object name given an edge ID.

UI Access	N/A		
Parameters	Name	Type	Description
	<EdgeID>	Integer	The edge ID.
Return Value	String containing object name.		

Python Syntax	<code>GetObjectNameByEdgeID(<EdgeID>)</code>
Python Example	<code>oEditor.GetObjectNameByEdgeID(88)</code>

VB Syntax	<code>GetObjectNameByEdgeID <EdgeID></code>
VB Example	<code>oEditor.GetObjectNameByEdgeID 10</code>

GetObjectNameByFaceID

Returns an object name given a face ID.

UI Access	N/A		
Parameters	Name	Type	Description
	<FaceID>	Integer	The face ID.
Return Value	String containing object name.		

Python Syntax	GetObjectByNameByFaceID (<FaceID>)
Python Example	oEditor.GetObjectByNameByFaceID(88)

VB Syntax	GetObjectByNameByFaceID <FaceID>
VB Example	oEditor.GetObjectByNameByFaceID 10

GetObjectByNameID

Returns an object name given an ID.

UI Access	N/A		
Parameters	Name	Type	Description
	<ObjID>	Integer	The object ID.
Return Value	String containing object name.		

Python Syntax	GetObjectByNameID(<ObjID>)

Python Example	<code>oEditor.GetObjectNameByID(88)</code>

VB Syntax	<code>GetObjectNameByID <ObjID></code>
VB Example	<code>oEditor.GetObjectNameByID 10</code>

GetObjectNameByVertexID

Returns an object name given a vertex ID.

UI Access	N/A		
Parameters	Name	Type	Description
	<VertexID>	Integer	The vertex ID.
Return Value	String containing object name.		

Python Syntax	<code>GetObjectNameByVertexID(<VertexID>)</code>
Python Example	<code>oEditor.GetObjectNameByVertexID(88)</code>

VB Syntax	<code>GetObjectNameByVertexID <VertexID></code>
VB Example	<code>oEditor.GetObjectNameByVertexID 10</code>

GetObjectsByMaterial

Returns a list of objects of a specified material.

UI Access	N/A		
Parameters	Name	Type	Description
	<material>	String	Material name.
Return Value	Array containing string object names.		

Python Syntax	GetObjectsByMaterial (<material>)		
Python Example	oEditor.GetObjectsByMaterial ('copper')		

VB Syntax	GetObjectsByMaterial <material>		
VB Example	oEditor.GetObjectsByMaterial "copper"		

GetObjectShapeType

Returns the shape type of a specified object.

UI Access	N/A		
Parameters	Name	Type	Description
	<objName>	String	Name of specified object.
	<csName>	String	Name of coordinate system of the object.
Return Value	String containing shape type.		

Python Syntax	<code>GetObjectShapeType(<objName>, <csName>)</code>
Python Example	<code>oEditor.GetObjectShapeType("box1", "Global")</code>

VB Syntax	<code>GetObjectShapeType <objName>, <csName></code>
VB Example	<code>oEditor.GetObjectShapeType "box1", "Global"</code>

GetObjectsInGroup

Returns a list of objects in a specified group.

UI Access	N/A		
Parameters	Name	Type	Description
	<group>	String	Group name. One of: • "<materialName>" • "<assignmentName>" • "Non Model" • "Solids" • "Unclassified" • "Sheets" • "Lines"
Return Value	Array containing string object names.		

Python Syntax	GetObjectsInGroup (<group>)
Python Example	oEditor.GetObjectsInGroup ('Sheets')

VB Syntax	GetObjectsInGroup <group>
VB Example	oEditor.GetObjectsInGroup "Sheets"

GetObjectVolume

Returns an object's volume from its name).

UI Access	N/A		
Parameters	Name	Type	Description
	<name>	String	Object name
Return Value	.Real		

Python Syntax	GetObjectVolume(<name>)
Python Example	oEditor.GetObjectVolume ("Box1")

VB Syntax	GetObjectVolume <name>
VB Example	oEditor.GetObjectVolume "Box1"

GetObjPath [Editor]

Obtains the path to the 3D modeler.

UI Access	N/A
Parameters	None.
Return Value	String containing the path.

Python Syntax	GetObjPath()
Python Example	<code>oEditor.GetObjPath()</code>

VB Syntax	GetObjPath
VB Example	<code>oEditor.GetObjPath</code>

GetPartsForUserDefinedModel

Obtains parts from a specified user defined model.

UI Access	N/A		
Parameters	Name	Type	Description
	<udmName>	String	Name of user defined model.
Return Value	Array of strings containing associated parts.		

Python Syntax	<code>GetPartsForUserDefinedModel (<udmName>)</code>
Python Example	<code>oEditor.GetPartsForUserDefinedModel ("OnDieSpiralInductor1")</code>

VB Syntax	<code>GetPartsForUserDefinedModel <udmName></code>
VB Example	<code>oEditor.GetPartsForUserDefinedModel "OnDieSpiralInductor1"</code>

GetPoints [3D Modeler Editor]

Returns all the points defined in current 3D modeler.

UI Access	N/A
Parameters	None.
Return Value	Array of strings containing names of points.

Python Syntax	<code>GetPoints ()</code>
Python Example	<code>oEditor.GetPoints ()</code>

VB Syntax	<code>GetPoints</code>
VB Example	<code>oEditor.GetPoints</code>

GetPropEvaluatedValue

Returns the Evaluated-Value for Value-Property and Variable. Returns the Property-value as text string for other property types

Note:

This command is not supported by the EMIT and Circuit design types.

UI Access	N/A		
Parameters	Name	Type	Description
	<PropName>	String	Name of the property.
Return Value	String value of the evaluated value.		

Python Syntax	GetPropEvaluatedValue (<PropName>)		
Python Example	<pre>oVar = oDesign.GetChildObject(" Variables/var") oVar.GetPropEvaluatedValue()</pre>		

GetPropertyValue

Returns the value of a single property belonging to a specific <PropServer> and <PropTab>. This function is available with the Project, Design or Editor objects, including definition editors.

Tip: Use the script recording feature and edit a property, and then view the resulting script to see the format for that property.

UI Access	N/A		
Parameters	Name	Type	Description
	<PropTab>	String	One of the following, where tab titles are shown in parentheses: - PassedParameterTab ("Parameter Values") - DefinitionParameterTab (Parameter Defaults") - LocalVariableTab ("Variables" or "Local Variables") - ProjectVariableTab ("Project variables") - ConstantsTab ("Constants") - BaseElementTab ("Symbol" or "Footprint") - ComponentTab ("General") - Component("Component") - CustomTab ("Intrinsic Variables") - Quantities ("Quantities") - Signals ("Signals")
	<PropServer>	String	An object identifier, generally returned from another script method, such as CompInst@R;2;3
	<PropName>	String	Name of the property.
Return Value	String value of the property.		

Python Syntax	GetPropertyValue (<PropTab>, <PropServer>, <PropName>)
Python Example	<pre> selectionArray = oEditor.GetSelections() for k in selectionArray: val = oEditor.GetPropertyValue("PassedParameterTab", k, "R") ... </pre>

VB Syntax	GetPropertyValue (<PropTab>, <PropServer>, <PropName>)
VB Example	<pre> selectionArray = oEditor.GetSelections for k in selectionArray: val = oEditor.GetPropertyValue("PassedParameterTab", k, "R") ... </pre>

GetPropNames [Modeler]

Returns the property names for the active model object, or specified property values.

Note:

This command is not supported by the EMIT and Circuit design types.

UI Access	N/A		
Parameters	Name	Type	Description
	<IncludeReadOnly>	Boolean	Optional.

			• True - includes all properties. • False - returns only property names that can be changed. True by default.
Return Value	Returns property names of the current 3D Model object.		

Python Syntax	<code>GetPropNames(<IncludeReadOnly>)</code>
Python Example	<code>oEditor.GetPropNames(True)</code>

VB Syntax	<code>GetPropNames(<IncludeReadOnly>)</code>
VB Example	<code>oEditor.GetPropName True</code>

GetPropSIValue

Returns the SI-Value for Value-Property and Variable. Return NAN for other property type if its value is not able to convert to be a double-floating point value.

Note:

This command is not supported by the EMIT and Circuit design types.

UI Access	N/A		
Parameters	Name	Type	Description

	<table><tr><td><PropName></td><td>String</td><td>Name of the property.</td></tr></table>	<PropName>	String	Name of the property.
<PropName>	String	Name of the property.		
Return Value	Property value as a double floating value, or NAN if the property value cannot be converted to double floating point.			

Python Syntax	GetPropSIValue (<PropName>)
Python Example	<pre>oCreateBox = oDesign.GetChildObject("3D Modeler/Box1/CreateBox:1") oCreateBox.GetPropValue("xSize") return "length / 2" oCreateBox.GetPropEvaluatedValue("xSize") return '0.4mm' oCreateBox.GetPropSIValue("xSize") return 0.0004</pre>

GetPropValue [Modeler]

Returns the property value for the active model object, or specified property values.

Note:

This command is not supported by the EMIT and Circuit design types.

UI Access	N/A		
Parameters	Name	Type	Description
	<propPath>		Optional. Child object's property path. See: Object Property Function Summary .

Return Value	The property value.

Python Syntax	<code>GetPropValue(<propPath>)</code>
Python Example	<pre>Rh1 = oDesign.GetChildObject('Box1') Rh1.GetPropValue('Orientation') Returns the specified Property Value: 'Global'</pre>

VB Syntax	<code>GetPropValue(<propPath>)</code>
VB Example	<pre>Rh1 = oDesign.GetChildObject("Box1") GetPropValue("Orientation")</pre>

GetRelativeCoordinateSystems

Returns the relative coordinate systems in the current 3D modeler.

UI Access	N/A
Parameters	None.
Return Value	Array containing name of relative coordinate systems.

Python Syntax	GetRelativeCoordinateSystems()
Python Example	<code>oEditor.GetRelativeCoordinateSystems()</code>

VB Syntax	GetRelativeCoordinateSystems
VB Example	<code>oEditor.GetRelativeCoordinateSystems</code>

GetSelections [Model Editor]

Returns an array of currently selected objects.

UI Access	N/A
Parameters	None.
Return Value	Array containing object IDs

Python Syntax	GetSelections()
Python Example	<code>oEditor.GetSelections()</code>

VB Syntax	GetSelections
VB Example	<code>oEditor.GetSelections</code>

GetSubGroupsInGroup

Returns subgroup names in a specified group.

UI Access	N/A		
Parameters	Name	Type	Description
	<group>	String	Group name. One of: • "<materialName>" • "<assignmentName>" • "Non Model" • "Solids" • "Unclassified" • "Sheets" • "Lines"
Return Value	Array containing subgroup names.		

Python Syntax	GetSubGroupsInGroup(<group>)
Python Example	<code>oEditor.GetSubGroupsInGroup('Sheets')</code>

VB Syntax	GetSubGroupsInGroup<group>
VB Example	<code>oEditor.GetSubGroupsInGroup "Sheets"</code>

GetUserPosition

Returns a user's current coordinates in the 3D Modeler window.

UI Access	N/A		
Parameters	Name	Type	Description
	<prompt>	String	"Enter a point." Then click a point in the 3D Modeler window.
Return Value	Array containing X, Y, Z coordinates.		

Python Syntax	GetUserPosition(<prompt>)		
Python Example	oEditor.GetUserPosition("Enter a point.")		

VB Syntax	GetUserPosition <prompt>		
VB Example	oEditor.GetUserPosition "Enter a point."		

GetVertexIDFromNameForFirstOperation

Returns vertex ID associated with a specified vertex belongs to the first operation of a part.

UI Access	N/A		
Parameters	Name	Type	Description
	<PartName>	String	Name of specified part.
	<VertexName>	String	Name of specified vertex.
Return Value	Integer representing vertex ID.		

Python Syntax	GetVertexIDFromNameForFirstOperation(<PartName>, <VertexName>)		

Python Example	<code>oEditor.GetVertexIDFromNameForFirstOperation("Part1", "Vertex1")</code>

VB Syntax	<code>GetVertexIDFromNameForFirstOperation <PartName>, <VertexName></code>
VB Example	<code>oEditor.GetVertexIDFromNameForFirstOperation "Part1", "Vertex1"</code>

GetVertexIDsFromEdge

Returns vertex IDs associated with a specified edge.

UI Access	N/A		
Parameters	Name	Type	Description
	<edgeID>	Integer	Edge ID.
Return Value	Array containing string vertex IDs.		

Python Syntax	<code>GetVertexIDsFromEdge(<edgeID>)</code>
Python Example	<code>oEditor.GetVertexIDsFromEdge(10)</code>

VB Syntax	<code>GetVertexIDsFromEdge <edgeID></code>
VB Example	<code>oEditor.GetVertexIDsFromEdge 10</code>

GetVertexIDsFromFace

Returns vertex IDs associated with a specified face.

UI Access	N/A		
Parameters	Name	Type	Description
	<faceID>	Integer	Face ID.
Return Value	Array containing string vertex IDs.		

Python Syntax	GetVertexIDsFromFace(<faceID>)		
Python Example	oEditor.GetVertexIDsFromFace(10)		

VB Syntax	GetVertexIDsFromFace <faceID>		
VB Example	oEditor.GetVertexIDsFromFace 10		

GetVertexIDsFromObject

Returns vertex IDs associated with a specified object.

UI Access	N/A		
Parameters	Name	Type	Description
	<objectName>	String	Object name.
Return Value	Array containing string vertex IDs.		

Python Syntax	GetVertexIDsFromObject(<objectName>)
Python Example	oEditor.GetVertexIDsFromObject('Box1')

VB Syntax	GetVertexIDsFromObject <objectName>
VB Example	oEditor.GetVertexIDsFromObject "Box1"

GetVertexPosition

Returns an array of coordinates for a specified vertex.

UI Access	N/A		
Parameters	Name	Type	Description
	<vertexID>	Integer	Vertex ID.
Return Value	Array containing X, Y, Z coordinates.		

Python Syntax	GetVertexPosition(<vertexID>)
Python Example	oEditor.GetVertexPosition(1)

VB Syntax	GetVertexPosition <vertexID>
VB Example	oEditor.GetVertexPosition 1

GetWireBodyNames

Returns the wire body names in current 3D modeler.

UI Access	N/A
Parameters	None.
Return Value	Array containing string of names.

Python Syntax	GetWireBodyNames()
Python Example	<code>oEditor.GetWireBodyNames()</code>

VB Syntax	GetWireBodyNames
VB Example	<code>oEditor.GetWireBodyNames</code>

PageSetup

Specifies page settings for printing.

UI Access	File > Page Setup.		
Parameters	Name	Type	Description
	<Parameters>	Array	Structured array. ["NAME:PageSetupData", "margins:=", <SetupArray>

			]
	<SetupArray>	Array	Structured Array ["left:=", <string>, "right:=", <string>, "top:=", <string>, "bottom:=", <string>)]
Return Value	None.		

Python Syntax	PageSetup(<Parameters>)
Python Example	<pre>oEditor.PageSetup(["NAME:PageSetupData", "margins:=", ["left:=", "10mm", "right:=", "10mm", "top:=", "10mm", "bottom:=", "10mm"]</pre>

	l)

VB Syntax	PageSetup <Parameters>
VB Example	<pre>oEditor.PageSetup Array("NAME:PageSetupData", "margins:=", Array("left:=", "10mm", "right:=", "10mm", "top:=", "10mm", "bottom:=", "10mm"))</pre>

RemoveBadEdges

Removes bad edges from specified list.

UI Access	N/A		
Parameters	Name	Type	Description
	<EdgeList>	Array	Array containing edge IDs.
Return Value	None.		

Python Syntax	RemoveBadEdges (<EdgeList>)
Python Example	oEditor.RemoveBadEdges([18,30])

VB Syntax	RemoveBadEdges <EdgeList>
VB Example	<code>oEditor.RemoveBadEdges Array(18, 30)</code>

RemoveBadFaces

Removes bad faces from specified list.

UI Access	N/A		
Parameters	Name	Type	Description
	<FaceList>	Array	Array containing face IDs.
Return Value	None.		

Python Syntax	RemoveBadFaces (<FaceList>)
Python Example	<code>oEditor.RemoveBadFaces ([328, 333])</code>

VB Syntax	RemoveBadFaces <FaceList>
VB Example	<code>oEditor.RemoveBadFaces Array(328, 333)</code>

RemoveBadVertices

Removes bad vertices from specified list.

UI Access	N/A		
Parameters	Name	Type	Description
	<VertexList>	Array	Array containing vertex IDs.
Return Value	None.		

Python Syntax	RemoveBadVertices (<VertexList>)		
Python Example	oEditor.RemoveBadVertices ([324, 325])		

VB Syntax	RemoveBadVertices <VertexList>		
VB Example	oEditor.RemoveBadVertices Array(324, 325)		

RenamePart

Renames an object.

UI Access	Enter new name in Name field.		
Parameters	Name	Type	Description
	<renameParametersArray>	Array	Structured array. Array("NAME:Rename Data", "Old Name:=", <string>, "New Name:=", <string>)

Return Value	None.

Python Syntax	<code>oEditor.RenamePart(<renameParametersArray>)</code>
Python Example	<pre>oEditor.RenamePart (['NAME:Rename Data', 'Old Name:=' , 'partname', 'New Name:=' , 'newpartname',])</pre>

VB Syntax	<code>oEditor.RenamePart <RenameParametersArray></code>
VB Example	<pre>oEditor.RenamePart Array("NAME:Rename Data", "Old Name:=", "partname", "New Name:=", "newpartname",)</pre>

SetPropValue [Modeler]

Sets the property value for the active model property.

Note:

This command is not supported by the EMIT and Circuit design types.

UI Access	Edit Properties on History Tree objects		
Parameters	Name	Type	Description
	<propPath>	String	Child object's property path. See: Object Property Function Summary .
	<value>	Varies	New property value.
Return Value	Boolean: • True – property found. • False – property not found.		

Python Syntax	SetPropValue(<propPath>, <value>)
Python Example	<code>oEditor.SetPropValue("Color/r", 111)</code>

VB Syntax	SetPropValue <propPath>, <value>
VB Example	<code>oEditor.SetPropValue "Color/r", 111)</code>

SetTopDownViewDirectionForActiveView

Sets active view to top-down view direction.

UI Access	N/A

Parameters	Name	Type	Description
	<RelativeCS>	String	Name of relative coordinate system
Return Value	None.		

Python Syntax	SetTopDownViewDirectionForActiveView (<RelativeCS>)
Python Example	<code>oEditor.SetTopDownViewDirectionForActiveView ("Global")</code>

VB Syntax	SetTopDownViewDirectionForActiveView <RelativeCS>
VB Example	<code>oEditor.SetTopDownViewDirectionForActiveView "Global"</code>

SetTopDownViewDirectionForAllViews

Sets all views to top-down view direction.

UI Access	N/A		
Parameters	Name	Type	Description
	<RelativeCS>	String	Name of relative coordinate system
Return Value	None.		

Python Syntax	SetTopDownViewDirectionForAllViews (<RelativeCS>)
Python Example	<code>oEditor.SetTopDownViewDirectionForAllViews ("Global")</code>

VB Syntax	SetTopDownViewDirectionForAllViews <RelativeCS>
VB Example	oEditor.SetTopDownViewDirectionForAllViews "Global"

UpdatePriorityList

Updates specified priority lists.

UI Access	N/A		
Parameters	Name	Type	Description
	<Lists>	Array	Array of priority list names.
Return Value	None.		

Python Syntax	UpdatePriorityList (<Lists>)
Python Example	oEditor.UpdatePriorityList(["PriorityList1", "PriorityList2"])

VB Syntax	UpdatePriorityList <Lists>
VB Example	oEditor.UpdatePriorityList Array("PriorityList1", "PriorityList2")

UpgradeVersion

Upgrades legacy geometry to current version.

UI Access	Right-click on an operation icon in the history tree, select Upgrade Version .

Parameters	Name	Type	Description
	<Selections>	Array	Structured array. Array("NAME:Parameters", Array("NAME:PartOperations", Array("NAME:<string>",), "OperationIndices:=", <array of integers>), Array("NAME:UDMOperations"))
Return Value	None.		

Python Syntax	UpgradeVersion (<Selections>)
Python Example	<pre>oEditor.UpgradeVersion(["NAME:Parameters", ["NAME:PartOperations", ["NAME:source", "OperationIndices:=", [0]]], ["NAME:UDMOperations"]])</pre>

VB Syntax	UpgradeVersion <Lists>
VB Example	<pre> oEditor.UpgradeVersion Array("NAME:Parameters", Array("NAME:PartOperations", Array("NAME:source", "OperationIndices:=", Array(0))), Array("NAME:UDMOperations")) </pre>

Validate3DComponent

Validates a 3D component.

UI Access	N/A		
Parameters	Name	Type	Description
	<componentPath>	String	Path to a 3D component.
	<password>	String	Optional. Password to access the component.
Return Value	Boolean: • 1 - component is valid. • 0 - component is not valid.		

Python Syntax	Validate3DComponent (<componentPath>, <password>)
Python Example	oEditor.Validate3DComponent (

	"C:/temp/component.3dcomp", "")

VB Syntax	Validate3DComponent <componentPath>, <password>
VB Example	oEditor.Validate3DComponent "C:/temp/component.3dcomp", ""

WriteHistoryTreeLayoutForTest

Writes history tree layout to a file.

UI Access	N/A		
Parameters	Name	Type	Description
	<filePath>	String	Path to the file.
	<OrgByMaterial>	Integer	• 1 - organize objects by material. • 0 - do not organize by material.
	<OrgByAssignment>	Integer	• 1 - organize sheets by assignment. • 0 - do not organize by assignment.
	<OrgByCompDefinition>	Integer	• 1 - organize components by definition. • 0 - do not organize by definition.
	<DonotOrgUnderGroup>	Integer	• 1 - do not organize within group. • 0 - organize within group.
	<ShowGroup>	Integer	Optional. • 1 - show group.

	• 0 - do not show.
Return Value	None.
Python Syntax	<code>WriteHistoryTreeLayoutForTest (<filePath>, <OrgByMaterial>, <OrgByAssignment>, <OrgByCompDefinition>, <DonotOrgUnderGroup>, <ShowGroup>)</code>
Python Example	<pre>oEditor.WriteHistoryTreeLayoutForTest ('C:\temp', 1, 0, 0, 0, 1)</pre>
VB Syntax	<code>WriteHistoryTreeLayoutForTest <filePath>, <OrgByMaterial>, <OrgByAssignment>, <OrgByCompDefinition>, <DonotOrgUnderGroup>, <ShowGroup></code>
VB Example	<code>oEditor.WriteHistoryTreeLayoutForTest "C:\temp", 1, 0, 0, 0, 1</code>

This page intentionally
left blank.

9 - Output Variable Script Commands

Output variable commands should be executed by the "OutputVariable" module.

First, obtain the output variable module from oDesign and use it for output variable commands:

```
Set oModule = oDesign.GetModule("OutputVariable")

oModule.<CommandName><args>
```

The commands are:

[DeleteOutputVariable](#)

[DoesOutputVariableExist](#)

[EditOutputVariable](#)

[GetOutputVariableValue](#)

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	Mechanical > Results > Output Variables.		
Parameters	Name	Type	Description
	<OutputVarName>	String	Name of the new output variable.
	<Expression>	Value	Value to assign to the variable.

	<i><SolutionName></i>	String	Name of the solution, as seen in the output variable UI.
	<i><reportType></i>	String	The name of the report type as seen in the output variable UI.
	<i><ContextArray></i>	Array	Structured array containing context for which the output variable expression is being evaluated. <code>Array("Context:=", <string>)</code>
Return Value	None.		

Python Syntax	<code>CreateOutputVariable (<OutputVarName>, <Expression>, <SolutionName>, <reportType solutionType>, <contextArray domainArray>)</code>
Python Example	

VB Syntax	<code>CreateOutputVariable <OutputVarName>, <Expression>, <SolutionName>, <reportType solutionType>, <contextArray domainArray></code>
VB Example	

DeleteOutputVariable

Deletes an existing output variable. The variable can only be deleted if it is not in use by any traces.

UI Access	Mechanical > Results > Output Variables. In the Output Variables window, click Delete .

Parameters	Name	Type	Description
	<OutputVarName>	String	Name of the output variable.
Return Value	None.		

Python Syntax	DeleteOutputVariable (<OutputVarName>)
Python Example	<pre>oModule = oDesign.GetModule("OutputVariable") oModule.DeleteOutputVariable ("testNew")</pre>

VB Syntax	DeleteOutputVariable <OutputVarName>
VB Example	<pre>Set oModule = oDesign.GetModule("OutputVariable") oModule.DeleteOutputVariable "testNew"</pre>

DoesOutputVariableExist

Verifies whether or not a named output variable exists.

UI Access	N/A		
Parameters	Name	Type	Description
	<outputVariableName>	String	The output variable name.
Return Value	Boolean True if the variable exists; False if it does not.		

Python Syntax	DoesOutputVariableExist(<outputVariableName>)
Python Example	<pre>oProject = oDesktop.GetActiveProject() oDesign = oProject.GetActiveDesign() oModule = oDesign.GetModule("OutputVariable") oModule.DoesOutputVariableExist("MyTestVar")</pre>

VB Syntax	DoesOutputVariableExist(<outputVariableName>)
VB Example	<pre>Set oModule = oDesign.GetModule "OutputVariable" oModule.DoesOutputVariableExist "MyTestVar"</pre>

EditOutputVariable

Changes the name or expression of an existing output variable.

UI Access	N/A		
Parameters	Name	Type	Description
	<OrigVarName>	String	Name of the original output variable.
	<NewExpression>	String	New value to assign to the variable.
	<NewVarName>	String	New name of the variable if any, else pass empty string.
	<SolutionName>	String	Name of the solution as seen in the output variable UI. For example, "Setup1 : Last Adaptive".
	<ReportType>	String	The name of the report type as seen in the output vari-

	<ContextArray>	Array	able UI. Structured array containing context for which the output variable expression is being evaluated. Array("Context:=", <string>)
Return Value	None		

Python Syntax	EditOutputVariable (<OrigVarName>, <NewExpression>, <NewVarName>, <SolutionName>, <ReportType>, <ContextArray>)
Python Example	<pre>oModule = oDesign.GetModule("OutputVariable") oModule.EditOutputVariable ("test", "normalize(R1_0.V)", "testNew", "TR", "Standard", [])</pre>

VB Syntax	EditOutputVariable <OrigVarName>, <NewExpression>, <NewVarName>, <SolutionName>, <ReportType>, <ContextArray>
VB Example	<pre>Set oModule = oDesign.GetModule("OutputVariable") oModule.EditOutputVariable "test", "dB(S(WavePort1,WavePort1)) ", "testNew", "Setup1 : LastAdaptive", "Modal Solution Data", Array("Domain:=", "Sweep")</pre>

ExportOutputVariables

Exports output variables to a file.

UI Access	Click on Export in the Output Variables dialog.		
Parameters	Name	Type	Description

	<table><tr><td><FileName></td><td>String</td><td>Name of the file include path.</td></tr></table>	<FileName>	String	Name of the file include path.
<FileName>	String	Name of the file include path.		
Return Value	Boolean: • True - output variable successfully exported. • False - error when export output variables.			

Python Syntax	ExportOutputVariables(<FileName>)
Python Example	oModule.ExportOutputVariables("C:/output_var.aoutvar")

VB Syntax	ExportOutputVariables <FileName>
VB Example	oModule.ExportOutputVariables "C:/output_var.aoutvar"

GetOutputVariables

Returns the list of output variables.

UI Access	N/A
Parameters	None.
Return Value	Array containing all output variables.

Python Syntax	GetOutputVariables()
Python Example	<code>oDesign.GetOutputVariables()</code>

VB Syntax	GetOutputVariables
VB Example	<code>oDesign.GetOutputVariables</code>

GetOutputVariableValue

Returns the double value of an output variable. Only expressions that return a double value are supported. The expression is evaluated only for a single point.

UI Access	N/A		
Parameters	Name	Type	Description
	<code><OutputVarName></code>	String	Name of the output variable.
	<code><IntrinsicVariation></code>	String	A set of intrinsic variable value pairs to use when evaluating the output expression.
	<code><SolutionName></code>	String	Name of the solution as listed in the output variable UI. For example, "Setup1 : Last Adaptive".
	<code><ReportType></code>	String	The name of the report type as seen in the output variable UI.
	<code><ContextArray></code>	Array	Structured array containing context for which the output variable expression is being evaluated. Can be empty. <code>Array("Context:=", <string>)</code>
Return Value	Double value of the output variable.		

Python Syntax	<code>GetOutputVariableValue(<OutputVarName>, <IntrinsicVariation>, <SolutionName>, <ReportTypeName>,</code>

	<code><ContextArray>)</code>
Python Example	<pre>Val = oDesign.GetOutputVariableValue("test","Freq = '20Ghz' Theta='20deg' Phi='30deg'", "TR", "Standard", [])</pre>

VB Syntax	<code>GetOutputVariableValue <OutputVarName>, <IntrinsicVariation>, <SolutionName>, <ReportTypeName>, <ContextArray></code>
VB Example	<pre>Val = oDesign.GetOutputVariableValue "test","Freq = '20Ghz' Theta='20deg' Phi='30deg'", "TR", "Standard", Array()</pre>

ImportOutputVariables

Imports output variables from a file.

UI Access	Click on Import in the Output Variables dialog.		
Parameters	Name	Type	Description
	<code><FileName></code>	String	Name of the file include path.
Return Value	Boolean: • True - output variable successfully imported. • False - error when import output variables.		

Python Syntax	<code>ImportOutputVariables(<FileName>)</code>

Python Example	<code>oModule.ImportOutputVariables("C:/output_var.aoutvar")</code>

VB Syntax	<code>ImportOutputVariables <FileName></code>
VB Example	<code>oModule.ImportOutputVariables "C:/output_var.aoutvar"</code>

SimValueContext

SimValueContext holds context information for a trace, and describes how data for a trace should be extracted from the simulation. SimValueContext contains a list of 14 required initial values:

```
SimValueContext (  
Domain ID, Calculation Type, Number of Cycles, Rise Time,  
Step, Impulse, Context ID, Window Width,  
Window Type, TDR Kaiser Parameter, Hold Time, DeviceName,  
TDR Step Time, DR Maximum Time )
```

For example, the following indicates a trace in the Time Domain, Standard Calculation with the number of cycles being 2:

```
"SimValueContext:=", Array(1, 0, 2, 0, false, false, -1, 1, 0, 1, 1, "", 0, 0)
```

Additional, context-specific values may follow the required values, as described in subsection 15 below.

1. Domain ID

No Domain	0
Time Domain	1
Spectrum Domain	2

Sweep Domain	3
Device Domain	4
SinglePt Domain	5
LoadPull Domain	6
Transient Domain	7
Budget Domain	8
NetworkFunction Domain	9
Oscillator Domain	55802
Noise Domain	55803
Transfer Function Domain	55807
Time Frequency Domain	55808
Transient Time Domain	55809
Periodic AC Domain	55818
UI Domain	55819
Eye Measurement Domain	55823
Initial Response Domain	55824
Peak Distortion Domain	55825

2. Calculation Type

Standard Calculation	0
Device2_DCIV	1
Device3_DCIV_Output	2

Device3_DCIV_Input	3
Device3_DCIV_Transfer	4
Device3_DCIV_Reverse	5
Device2_ACLoad	6
Device3_ACLoad_Output	7
Device3_ACLoad_Input	8
Device3_ACLoad_Transfer	9
Device3_ACLoad_Reverse	10
Constellation	11
EyeDiagram	12
FreeX (Statistic Report)	13

3. **Number of Cycles** — Used in Time Domain in HarmonicBalance analysis.

4. **Rise Time** — Not used by Designer/Nexxim.

5. **Step** — Not used by Designer/Nexxim.

6. **Impulse** — Not used by Designer/Nexxim.

7. **Context ID** — Not used by Designer/Nexxim.

8. **Window Width** — Not used by Designer/Nexxim.

9. **Window Type** — Not used by Designer/Nexxim.

10. **TDR Kaiser Parameter** — Not used by Designer/Nexxim.

11. **Hold Time** — Not used by Designer/Nexxim.

12. **DeviceName** — Not used by Designer/Nexxim.

13. **TDR Step Time** — Not used by Designer/Nexxim.

14. **TDR Maximum Time** — Not used by Designer/Nexxim.

15. **Context-specific values** — Used in Time Domain in HarmonicBalance analysis.

Context-specific values are entered in the format "key, true/false, keyvalue", where:

- **"key"** is the name of the key being set.
- **"true/false"** indicates whether the key is a string value.
- **"keyvalue"** is the value of the key.
- The order of the context keys is not significant.
- Context keys have software defaults that will be used if not provided in the script.

VB Example:

```
"SimValueContext:=", Array(1, 0, 2, 0, false, false, -1, 1, 0, 1, 1, "", 0, 0,  
"DE", false, "0",  
"DP", false, "20000000",  
"DT", false, "0.001",  
"WE", false, "10ns",  
"WM", false, "10ns",  
"WN", false, "0ns",  
"WS", false, "0ns"))
```

a. Plotting Range for Time domain in Transient and QuickEye analysis:

Description	Key Name	Is a string?	Key Value
Start Time	WS	False	0ns
Stop Time	WE	False	10ns
Minimum Time	WM	False	0ns
Maximum Time	WN	False	10ns
Is Thinning Enabled?	DE	False	0
Dy/dx Tolerance	DT	False	0.001
Number of points	DP	False	20000000

b. Transient report context for Spectral domain in Transient analysis:

Description	Key Name	Is a string?	Key Value
Start Time	TS	False	0ns
Stop Time	TE	False	10ns
Max Harmonics	MH	False	100
Max Frequency	MF	False	*
Window type	WT	False	0
Width Percentage	WW	False	100
Kaiser Parameter	KP	False	0
Adjust Coherent Gain	CG	False	0

* Script can specify either MH or MF. If neither is specified, Max Harmonics is set to 100. If both are specified, MF is used.

Window Type	ID
Rectangular	0
Bartlett	1
Blackman	2
Hamming	3
Hanning	4
Kaiser	5
Welch	6
Weber	7
Lanzcos	8

c. Eyeprobe index context for UI domain, Time domain, Eye Measurement domain in VeriEye and QuickEye analysis:

Description	Key Name	Is a string?	Key Value
Eyeprobe compinst ID	PCID	False	0

d. Eyesource index context for Initial Response domain and Peak Distortion domain in VeriEye 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

This page intentionally
left blank.

10 - Reporter Editor Script Commands

Reporter commands should be executed by the oDesign object.

For example:

All Report and Trace properties can be edited using the **ChangeProperty** commands. This includes Title properties, General properties, and Background properties such as border color, fonts, X and Y axis scaling, and number display.

Note:

When you execute **Tools > Record Script**, operations performed in the Reporter are automatically recorded.

The list of commands is as follows:

[AddCartesianLimitLine](#)

[AddCartesianXMarker](#)

[AddCartesianYMarker](#)

[AddCartesianYMarkerToStack](#)

[AddDeltaMarker](#)

[AddMarker](#)

[AddNote](#)

[AddTraces](#)

[ClearAllMarkers](#)

[ClearAllTraceCharacteristics](#)

[CopyReportDefinitions](#)

[CopyReportData](#)

[CopyTraceDefinitions](#)

[CopyTracesData](#)

[CreateReport \[Mechanical\]](#)

[CreateReportFromFile](#)

[CreateReportFromTemplate](#)

[CreateReportOfAllQuantities](#)

[DeleteAllReports](#)

[DeleteReports](#)

[DeleteTraces](#)

[ExportImageToFile](#)

[ExportToFile \[Reporter\]](#)

[GetAllCategories](#)

[GetAllQuantities](#)

[GetAllReportNames](#)

[GetAvailableDisplayTypes](#)

[GetAvailableReportTypes](#)

[GetAvailableSolutions](#)

[GetChildNames](#)

[GetChildObject](#)

[GetChildTypes](#)

[GetPropertyValue](#)

[GetSolutionContexts](#)

[GetSolutionDataPerVariation](#)

[GroupPlotCurvesByGroupingStrategy](#)

[ImportIntoReport](#)

[MovePlotCurvesToGroup](#)

[MovePlotCurvesToNewGroup](#)

[PasteReports](#)

[PasteTraces](#)

[RenameReport](#)

[RenameTrace](#)

[ResetPlotSettings](#)

[SavePlotSettingsAsDefault](#)

[SetPropValue](#)

[UpdateAllFieldsPlots](#)

[UpdateAllReports](#)

[UpdateReports](#)

[UpdateTraces](#)

[UpdateTracesContextandSweeps](#)

AddAllEyeMeasurements

Displays all the eye measurements in tabular format

UI Access	Right-click the report and select Trace Characteristics > Add All Eye Measurements		
Parameters	Name	Type	Description
	<ReportName>	String	Name of the report.
Return Value	None.		

Python Syntax	AddAllEyeMeasurements(<ReportName>)		
Python Example	oModule.AddAllEyeMeasurements("DQS Eye")		

VB Syntax	AddAllEyeMeasurements <ReportName>		
VB Example	oModule.AddAllEyeMeasurements "DQS Eye"		

AddCartesianLimitLine

Adds a limit line to a report on the X axis.

UI Access	Report2D > Add Limit Line> Specify Points...		
Parameters	Name	Type	Description
	<ReportName>	String	Name of the report.
	<Def>	Array	Structured array: Array("NAME:CartesianLimitLine", Array("NAME:XValues", <integer X values>),

			<pre> "XUnits:=", <string unit of measure for X>, Array("NAME:YValues", <integer Y values>), "YUnits:=", "<string unit of measure for Y>", "YAxis:=", <string name of associated Y axis>) </pre>
Return Value	None.		

Python Syntax	AddCartesianLimitLine (<ReportName>, <Def>)
Python Example	<pre> oModule.AddCartesianLimitLine("Project Outputs", ["NAME:CartesianLimitLine", ["NAME:XValues",0, 2, 5, 7, 10, 15], "XUnits:=", "s", ["NAME:YValues",0.05, 0.3, 0.65, 0.825, 0.95, 1], "YUnits:=", "mV", "YAxis:=", "Y1"]) </pre>

VB Syntax	AddCartesianLimitLine <ReportName>, <Def>
VB Example	<pre> oModule.AddCartesianLimitLine "Project Outputs", Array("NAME:CartesianLimitLine", </pre>

	<pre>Array("NAME:XValues",0, 2, 5, 7, 10, 15), "XUnits:=", "s", Array("NAME:YValues",0.05, 0.3, 0.65, 0.825, 0.95, 1), "YUnits:=", "mV", "YAxis:=", "Y1")</pre>

AddCartesianLimitLineFromCurve

Adds a limit line to a report from selected curve on the plot.

UI Access	Report2D > Add Limit Line > From Selected Curve...		
Parameters	Name	Type	Description
	<ReportName>	String	Name of the report.
	<LimitLineParams>	String	Structured array. 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>, "CreateMode:=", "<AboveCurve BelowCurve Above and Below Curve>",

			"YShiftPercent:=", <value percentage to shift from Y>)
Return Value	None.		

Python Syntax	AddCartesianLimitLineFromCurve(<ReportName>, <LimitLineParams>)
Python Example	<pre>oModule.AddCartesianLimitLineFromCurve("Variables Plot 2", ["NAME:CartesianLimitLineFromCurve", "TraceName:=" , "Phase", "CurveName:=" , "", "Start:=" , "0deg", "Stop:=" , "375deg", "YAxis:=" , 1, "YOffset:=" , 0, "CreateMode:=" , "AboveCurve", "YShiftPercent:=" , 10])</pre>

VB Syntax	AddCartesianLimitLineFromCurve<ReportName>, <LimitLineParams>
VB Example	<pre>oModule.AddCartesianLimitLineFromCurve _ "Variables Plot 2", _</pre>

	<pre>Array("NAME:CartesianLimitLineFromCurve", _ "TraceName:=", "Phase", "CurveName:=", "", _ "Start:=", "0deg", "Stop:=", "375deg", _ "YAxis:=", 1, "YOffset:=", 0, _ "CreateMode:=", "AboveCurve", _ "YShiftPercent:=", 10)</pre>

AddCartesianLimitLineFromEquation

Adds a limit line to a report from a specified equation.

UI Access	Report2D > Add Limit Line > Specify Equation...		
Parameters	Name	Type	Description
	<ReportName>	String	Name of the report.
	<LimitLineParams>	Array	Structured array.
			<pre>Array("NAME:CartesianLimitLineFromEquation", "YAxis:=", <integer Y-Axis number>, "Start:=", <string start frequency with unit>, "Stop:=", <string end frequency with unit>, "Step:=", <string frequency resolution with unit>, "Equation:=", <string specified equation>) "XValuesUnit:=", "GHz"</pre>

Return Value	None.

Python Syntax	AddCartesianLimitLineFromEquation(<ReportName>, <LimitLineParams>)
Python Example	<pre>oModule.AddCartesianLimitLineFromEquation("S Parameter Plot 1", ["NAME:CartesianLimitLineFromEquation", "YAxis:=" , 1, "Start:=" , "9GHz", "Stop:=" , "11GHz", "Step:=" , "0.2GHz", "Equation:=" , "x+1" "XValuesUnit:=" , "GHz"])</pre>

VB Syntax	AddCartesianLimitLineFromEquation <ReportName>, <LimitLineParams>
VB Example	<pre>oModule.AddCartesianLimitLineFromEquation _ "S Parameter Plot 1", _ Array("NAME:CartesianLimitLineFromEquation", _ "YAxis:=", 1, _ "Start:=", "9GHz", "Stop:=", "11GHz", "Step:=", "0.2GHz", _</pre>

	"Equation:=", "x+1", "XValuesUnit:=", "GHz")

AddCartesianXMarker

Adds a marker to a report on the X axis.

UI Access	Report2D > Marker > Add X Marker.		
Parameters	Name	Type	Description
	<ReportName>	String	Name of report.
	<MarkerName>	String	Marker name, including any trailing number.
	<XValue>	Double	X coordinate.
Return Value	None		

Python Syntax	AddCartesianXMarker (<ReportName>, <MarkerName>, <XValue>)
Python Example	oModule.AddCartesianXMarker ("XY Plot 1", "MX1", 0)

VB Syntax	AddCartesianXMarker <ReportName>, <MarkerName>, <XValue>
VB Example	oModule.AddCartesianXMarker "XY Plot1", "MX1", 0

AddCartesianYMarker

Adds a marker to a report on the Y axis.

UI Access	Report2D > Marker > Add Y Marker.		
Parameters	Name	Type	Description
	<ReportName>	String	Name of report.
	<MarkerName>	String	Marker name, including any trailing number.
	<AxisName>	String	Name of axis.
	<YValue>	Double	Y coordinate.
	<CurveName>	String	Name of curve.
Return Value	None		

Python Syntax	AddCartesianYMarker (<ReportName>, <MarkerName>, <AxisName>, <YValue>, <CurveName>)		
Python Example	<pre>oModule.AddCartesianYMarker("XY Plot 1", "MY1", "Y1", 0, "dB() : Setup1 : Sweep1")</pre>		

VB Syntax	AddCartesianYMarker <ReportName>, <MarkerName>, <AxisName>, <YValue>, <CurveName>		
VB Example	<pre>oModule.AddCartesianYMarker "XY Plot 1", "MY1", "Y1", 0, _ "dB() : Setup1 : Sweep1"</pre>		

AddCartesianYMarkerToStack

Adds a marker to a stacked report on the Y axis.

UI Access	N/A		
Parameters	Name	Type	Description
	<ReportName>	String	Name of report.

	<MarkerName>	String	Marker name, including any trailing number.
	<AxisName>	String	Name of axis.
	<YValue>	Double	Y coordinate.
	<CurveName>	String	Name of curve.
	<StackOption>	Array	"Current" to create marker on the current stack. "All" to create marker on all stack.
Return Value	None		

Python Syntax	AddCartesianYMarkerToStack (<ReportName>, <MarkerName>, <AxisName>, <YValue>, <CurveName>, <StackOption>)		
Python Example	<pre>oModule.AddCartesianYMarker("XY Plot 1", "MY1", "Y1", 0, "dB() : Setup1 : Sweep1", ["All"])</pre>		

VB Syntax	AddCartesianYMarkerToStack <ReportName>, <MarkerName>, <AxisName>, <YValue>, <CurveName>, <StackOption>		
VB Example	<pre>oModule.AddCartesianYMarkerToStack "XY Plot 1", "MY1", "Y1", 0, _ "dB() : Setup1 : Sweep1", Array("All")</pre>		

AddDeltaMarker

Add markers to calculate differences between two trace points on a plot.

UI Access	Report2D > Marker > Add Delta Marker.

Parameters	Name	Type	Description
	<ReportName>	String	Name of report.
	<MarkerName1>	String	Marker name, including any trailing number, for the first marker.
	<CurveName1>	String	Full trace name for the first marker.
	<PrimarySweepValue1>	String	
	<MarkerName2>	String	Marker name, including any trailing number, for the second marker.
	<CurveName2>	String	Full trace name for the second marker.
	<PrimarySweepValue2>	String	
Return Value	None		

Python Syntax	AddDeltaMarker(<ReportName>, <MarkerName1>, <CurveName1>, <PrimarySweepValue1>, <MarkerName2>, <CurveName2>, <PrimarySweepValue2>)
Python Example	

VB Syntax	AddDeltaMarker <ReportName>, <MarkerName1>, <CurveName1>, <PrimarySweepValue1>, <MarkerName2>, <CurveName2>, <PrimarySweepValue2>
VB Example	

AddMarker

Adds a marker to a trace on a report.

UI Access	Report2D > Marker > Add Marker.		
Parameters	Name	Type	Description
	<ReportName>	String	Name of report.

	<MarkerName>	String	Marker name, including any trailing number.
	<CurveName>	String	Full trace name.
	<PrimarySweepValue>	String	Primary sweep value, including unit.
Return Value	None		

Python Syntax	AddMarker(<ReportName>, <MarkerName>, <CurveName>, <PrimarySweepValue>)		
Python Example	<pre>oModule.AddMarker("XY Plot 1", "m5", "GS1.VAL : TR4 : Cartesian", "3.615999999999997s")</pre>		

VB Syntax	AddMarker <ReportName>, <MarkerName>, <CurveName>, <PrimarySweepValue>		
VB Example	<pre>Set oModule = oDesign.GetModule("ReportSetup") oModule.AddMarker "XY Plot1", "m1", "mag(S(Port1 Port1)) : Setup1: LastAdaptive : Cartesian", "0.3in"</pre>		

AddNote

Adds a note at a specified location to a given report.

UI Access	Right-click on the plot and select Add Note .

Parameters	Name	Type	Description
	<ReportName>	String	Name of report
	<NoteDataArray>	Array	Structured array. Array("NAME:<StringDataName>", <NoteArray>)
	<NoteArray>	Array	Structured array: Array("NAME:<NoteDataSourceName>", "SourceName:=", <string source name>, "HaveDefaultPos:=", <boolean True for position 0,0; False to specify below>, "DefaultXPos:=", <int X position for note; 0 for default>, "DefaultYPos:=", <int Y position for note; 0 for default>, "String:=", <string note text>)
Return Value	None		

Python Syntax	AddNote (<ReportName>, <NoteDataArray>)
Python Example	<pre>oModule.AddNote("XY Plot1", ["NAME:NoteDataSource", "SourceName:=", "Note1", "HaveDefaultPos:=", False, "DefaultXPos:=", 1996, "DefaultYPos:=", 3177,</pre>

	<pre>"String:=", "This is a note."])</pre>

VB Syntax	AddNote <ReportName> <NoteDataArray>
VB Example	<pre>oModule.AddNote "XY Plot1", _ Array("NAME:NoteDataSource", _ "SourceName:=", "Note1", _ "HaveDefaultPos:=", false, _ "DefaultXPos:=", 1996, _ "DefaultYPos:=", 3177, _ "String:=", "This is a note.")</pre>

AddTraceCharacteristics

Adds a trace characteristics field to the legend on a report.

UI Access	Report2D > Trace Characteristics > All. This opens the Add Trace Characteristics dialog box.		
Parameters	Name	Type	Description
	<ReportName>	String	Name of report.
	<FunctionName>	String	The function name. See the Functions column of the Add Trace Characteristics dialog box.
	<FunctionArgs>	Array	Array containing string values for any function arguments. Pass empty array if

			no arguments exist. To see which argument(s) a function takes, see the bottom of the Add Trace Characteristics dialog box. For function with one argument: <pre>Array(<value>)</pre> For function with multiple arguments: <pre>Array(<value>, <value>, ...)</pre>
	<RangeArgs>	Array	Required. Array containing either string "Full" for a full sweep range, or "Specified" and strings containing the start and end values for the frequency range. For example: <pre>Array("Specified", "19.5GHz", "24.4GHz")</pre>
Return Value	None.		

Python Syntax	AddTraceCharacteristics (<ReportName>, <FunctionName>, <FunctionArgs>, <RangeArgs>)
Python Example	<pre>oModule.AddTraceCharacteristics("XY Plot 2", "delaytime", ["0"], ["Full"]) oModule.AddTraceCharacteristics("Differential S-parameters", "prms", ["0", "0"], ["Full"]) oModule.AddTraceCharacteristics("Rept2DRectFreq", "distortion", ["0"], ["Specified", "19.5GHz", "20.4GHz"])</pre>

VB Syntax	AddTraceCharacteristics <ReportName>, <FunctionName>, <FunctionArgs>, <RangeArgs>

VB Example	<pre>oModule.AddTraceCharacteristics "XY Plot 2", "delaytime", Array("0"), Array("Full") oModule.AddTraceCharacteristics "Differential S-parameters", "prms", Array("0", "0"), Array("Full") oModule.AddTraceCharacteristics "Rept2DRectFreq", "distortion", Array("0"), Array("Specified", "19.5GHz", "20.4GHz")</pre>

AddTraces

Creates a new trace and adds it to the specified report.

UI Access	Modify Report > Add Trace.		
Parameters	Name	Type	Description
	<ReportName>	String	Name of Report
	<SolutionName>	String	Name of the solution as listed in the Modify Report dialog box.
	<ContextArray>	Array	Context for which the expression is being evaluated. This can be an empty string if there is no context. <pre>Array("Domain:=", <DomainType>) <DomainType> ex. "Sweep" or "Time" Array("Context:=", <GeometryType>) <GeometryType> ex. "Infinite Spheren", "Spheren", "Polylinen"</pre>
	<FamiliesArray>	Array	Contains sweep definitions for the report. <pre>Array("<VariableName>:= ", <ValueArray>) <ValueArray> Array("All") or Array("Value1", "Value2",</pre>

			... "Valuen") examples of <VariableName> "Freq", "Theta", "Distance"
	<ReportDataArray>	Array	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))")
Return Value	None		

Python Syntax	Add Traces(<ReportName>, <SolutionName>, <ContextArray>, <FamiliesArray>, <ReportDataArray>)
Python Example	<pre>oModule.AddTraces("XY Plot1", "Setup1 : Sweep1", ["Domain:=", "Time", "HoldTime:=", 1, "RiseTime:=", 0, "StepTime:=", 6.24999373748E-012, "Step:=", False, "WindowWidth:=", 1, "WindowType:=", 0, "KaiserParameter:=", 1, "MaximumTime:=", 6.2437437437444E-009], ["Time:=", ["All"], "OverridingValues:=", ["0s", "6.24999373748188e-012s", ...]], ["X Component:=", "Time", "Y Component:=", ["TDRZ(WavePort1)"]], [])</pre>

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)")), _ Array() </pre>

ApplyReportTemplate

Applies settings to a report from a template file.

UI Access	Right-click on a report, select Report Templates > Apply Settings .		
Parameters	Name	Type	Description
	<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")

Return Value	None.

Python Syntax	<code>ApplyReportTemplate(<ReportName>, <TemplateFile>, <PropertyType>)</code>
Python Example	<code>oModule.ApplyReportTemplate("Variables Plot 1", "C:/MyTemplate.rpt", "Graphical")</code>

VB Syntax	<code>ApplyReportTemplate <ReportName>, <TemplateFile>, <PropertyType></code>
VB Example	<code>oModule.ApplyReportTemplate "Variables Plot 1", "C:/MyTemplate.rpt", "Graphical"</code>

ClearAllMarkers

Clears all markers from a report.

UI Access	Report2D > Markers > Clear All.		
Parameters	Name	Type	Description
	<ReportName>	String	Name of Report
Return Value	None		

Python Syntax	<code>ClearAllMarkers(<ReportName>)</code>
Python Example	<code>oModule.ClearAllMarkers("XY Plot 1")</code>

VB Syntax	<code>ClearAllMarkers <ReportName></code>
VB Example	<code>oModule.ClearAllMarkers "XY Plot 1"</code>

ClearAllTraceCharacteristics

Clears all trace characteristics from the legend in a report.

UI Access	Report2D > Trace Characteristics > Clear All.		
Parameters	Name	Type	Description
	<PlotName>	String	Name of the plot
Return Value	None		

Python Syntax	<code>ClearAllTraceCharacteristics(<PlotName>)</code>
Python Example	<code>oModule.ClearAllTraceCharacteristics("XY Plot 1")</code>

VB Syntax	<code>ClearAllTraceCharacteristics <PlotName></code>
VB Example	<code>oModule.ClearAllTraceCharacteristics "XY Plot 1"</code>

CloneReportsFromDatasetSolution

Clones a report for a solved solution from a dataset solution.

UI Access	Right-click the Project tree on the report and choose Clone from Dataset Solution > [Dataset_SolutionName]		
Parameters	Name	Type	Description
	<ReportsToClone>	Array	Array of report names to clone.
	<SoluNameToUse>	String	Name of the dataset solution.
Return Value	None.		

Python Syntax	CloneReportsFromDatasetSolution(<ReportsToClone>, <SoluNameToUse>)		
Python Example	oModule.CloneReportsFromDatasetSolution(["Rectangular Plot1"], "DatasetSolution_rsm_5812")		

VB Syntax	CloneReportsFromDatasetSolution <ReportsToClone>, <SoluNameToUse>		
VB Example	oModule.CloneReportsFromDatasetSolution _ Array("Rectangular Plot1"), "DatasetSolution_rsm_5812"		

CopyPlotSettings

Copies settings of a specified plot.

UI Access	Right-click a report, select Copy		
Parameters	Name	Type	Description
	<ReportName>	String	Name of specified report.
Return Value	None.		

Python Syntax	CopyPlotSettings(<ReportName>)
Python Example	<code>oModule.CopyPlotSettings("Plot 1")</code>

VB Syntax	CopyPlotSettings <ReportName>
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 Copy Definition		
Parameters	Name	Type	Description
	<ReportsArray>	Array	Names of reports from which to copy the definitions
Return Value	None		

Python Syntax	CopyReportDefinitions(<ReportsArray>)
Python Example	<code>oModule.CopyReportDefinitions(["Transmission", "Reflection"])</code>

VB Syntax	CopyReportDefinitions <ReportsArray>
VB Example	<pre>oModule.CopyReportDefinitionsv _ Array("Transmission", "Reflection")</pre>

CopyReportsData

Copy all data corresponding to the specified reports.

UI Access	Select a report in the Project tree, right-click and select Copy Data		
Parameters	Name	Type	Description
	<ReportsArray>	Array	Names of reports from which to copy data
Return Value	None		

Python Syntax	CopyReportsData (<ReportsArray>)
Python Example	<pre>oModule.CopyReportsData (["Transmission", "Reflection"])</pre>

VB Syntax	CopyReportsData <ReportsArray>
VB Example	<pre>oModule.CopyReportsData _ Array("Transmission", "Reflection")</pre>

CopyTraceDefinitions

Copy trace definitions for a paste operation.

UI Access	Select a trace in the Project tree, right-click and select Copy Definition		
Parameters	Name	Type	Description
	<ReportName>	String	Name of Report
	<TracesArray>	Array	Trace definitions to copy
Return Value	None		

Python Syntax	CopyTraceDefinitions(<ReportName>, <TracesArray>)		
Python Example	<pre>oModule.CopyTraceDefinitions ("Transmission", ["mag (S (Port1, Port2)) "])</pre>		

VB Syntax	CopyTraceDefinitions <ReportName>, <TracesArray>		
VB Example	<pre>oModule.CopyTraceDefinitions "Transmission", _ Array ("mag (S (Port1, Port2)) ")</pre>		

CopyTracesData

Copies trace data for a paste operation.

UI Access	Select a trace in the Project tree, right-click and select Copy Data .		
Parameters	Name	Type	Description
	<ReportName>	String	Name of Report.
	<TraceArray>	Array	Trace definitions from which to copy corresponding data.

Return Value	None.

Python Syntax	<code>CopyTracesData(<ReportName>, <TracesArray>)</code>
Python Example	<code>oModule.CopyTracesData ("Transmission", ["mag(S (Port1, Port2)) "])</code>

VB Syntax	<code>CopyTracesData <ReportName>, <TracesArray></code>
VB Example	<code>oModule.CopyTracesData _ "C11", Array("mag(S (Port1, Port2)) ")</code>

CreateReport [Mechanical]

Creates a new report with one or more traces and adds it to the **Results** branch in the Project Manager.

UI Access	- From the Results ribbon tab: Click [Fields Report Modal Report Monitor Report] > <i>{Report_Type}</i> - In the Project Manager: Right-click on Results > [Create Fields Report Create Modal Report Create Monitor Report] > <i>{Report_Type}</i> - From the Menu Bar: Click Mechanical > Results > [Fields Report Modal Report Monitor Report] > <i>{Report_Type}</i>		
Parameters	Name	Type	Description
	<ReportName>	String	Name of report.
	<ReportType>	String	Type of report ("Modal" or "Fields").
	<DisplayType>	String	Specifies the type of plot to create. Applicable choices for Mechanical designs include:

		"Rectangular Plot", "Rectangular Stacked Plot", and "Data Table"
<SolutionName>	String	Name of the solution on which the report is to be based
<ContextArray>	Array	Context where the expression is to be evaluated. The array consists of the geometry (<polyline_name> along which to report the quantity or the <point_name> at which to do so). Polylines can consist of any number of straight or curved segments. When specifying a polyline, the array also contains the number of points to calculate along its path. For example: <pre>Array("Context:=", "Polyline2", "PointCount:=", 101)</pre> This array can be empty if no context is required.
<FamiliesArray>	Array	Specifies Sweep and Families parameters. When the Sweep is the "Distance" along a polyline, specify either ["All"] for the full polyline path, or a range (["<min_distance>,<max_distance>"] for a portion of the polyline path. For example: <pre>["Distance:=", ["All"]] ["Distance:=", ["0.25mm", "33.5mm"]]</pre> For Modal reports, the Sweep is the vibration "Mode". In this case, specify ["All"] to report the quantities for all modes or ["<start_mode>,<end_mode>"] for a contiguous subset of the available modes. For example: <pre>["Mode:=", ["All"]] ["Mode:=", ["2", "5"]]</pre> For Modal fields reports, in which the Sweep is "Distance" along a polyline, the desired modes are specified in the <i>Families</i> tab of the UI's <i>Report</i> dialog box. In the script, you can list the desired modes, and they don't have to be contiguous, as shown in the first of the following two examples:

		<pre>["Distance:=", ["All"], "Mode:=", ["1", "2", "4", "5", "7"]] ["Distance:=", ["0.25mm", "33.5mm"], "Mode:=", ["All"]]</pre> Selection of families is similar for all solution types when there are multiple sets of results (such as with parametric solutions). In such cases, a different variable would be substituted for "Mode."
<ReportDataArray>	Array	This array contains the X variable and Y variable (report quantities) definitions. For example: <pre>["X Component:=", <VariableName>, "Y Component:=", <ReportQuantityArray>]]</pre>
<ReportQuantityArray>	Array	Report quantities are dependent on the solution type and include the following: • All Solution Types: ◦ "Volume(<object_name>)" – Volume of the named solid object • Modal Solutions – Modal Report: ◦ "ModeFreq" – Resonant frequency of the vibration modes ◦ "ParticipationFactor^[1]" – Modal participation factors ◦ "EffectiveMass^[1]" – Mass participating in the mode ◦ "EffectiveMassRatio^[1]" – EffectiveMass / total mass ◦ "CummEMassFraction^[1]" – Cumulative effective mass factor^[2] • Modal Solutions – Fields Report: ◦ "Mag_Displacement" – Displacement magnitude^[3] • Thermal Solutions – Fields Report: ◦ "Temperature" ◦ "Mag_HeatFlux" – Heat flux magnitude ◦ "Surface_Loss_Density" – Imported EM losses (surface-based) ◦ "Volume_Loss_Density" – Imported EM losses (volume-based)

			◦ "Linked_Heat_Transfer_Coefficient" – Imported HPCs from an Icepak or Mechanical source solution (for Convection boundary film coefficients) ◦ "Thermal_Conductivity^[4]" – Only applicable to designs with trace mapping • Thermal Solutions – Monitor Report (Transient only): ◦ "<monitor_point_name, Temperature" – Temperature vs. Time at the selected monitor points • Structural Solutions: ◦ "Mag_Displacement" – Displacement Magnitude ◦ "Equivalent_Stress" – von Mises equivalent stress ◦ "Equivalent_Strain" – von Mises equivalent strain ◦ "Temperature" – Assigned or imported temperatures The following four quantities are only applicable to designs with trace mapping: ◦ Youngs_Modulus^[4]" ◦ "Poisson_ratio^[4]" ◦ "Thermal_Expansion^[4]" – Coefficient of thermal expansion ◦ Shear_Modulus^[4]"
Return Value	None.		
Notes	[1] – Six different quantities are available for the indicated modal results—three translational ("Dir") and three rotational ("Rot"). After each <i>ReportQuantity</i> name, _DirX, _DirY, _DirZ, _RotX, _RotY, or _RotZ is appended to indicate the translational or rotational direction. For example, "ParticipationFactor_RotZ" is the modal participation factor for rotation about the Z axis. [2] – For a complete definition of the <i>Cumulative Effective Mass Fraction</i> result, see the Results (Modal) help page.		

	[3] – See the note concerning displacement magnitudes for modal solutions on the <i>Results (Modal)</i> help page. [4] – The indicated quantities are only applicable to Thermal or Structural solutions that include layout components (for trace mapping). Three different quantities are available for each listed result to provide separate material properties in each of the three axis directions (based on the mapped metal fraction data). After each <i>ReportQuantity</i> name, _X, _Y, or _Z is appended to indicate the direction of the result. For example, "Thermal_Conductivity_Y" is the mapped thermal conductivity in the Y direction.

Python Syntax	CreateReport(<ReportName>, <ReportType>, <DisplayType>, <SolutionName>, [<ContextArray>], [<FamiliesArray>], [<ReportDataArray>])
Python Example	Five examples are given below covering each solution type and report type: Example 1 – Modal (Modal Report): <pre>Set oModule = oDesign.GetModule("ReportSetup") oModule.CreateReport("Frequency Table 1", "Modal", "Data Table", "Setup1 : Solution", [], ["Mode:=", ["All"]], ["X Component:=", "Mode", "Y Component:=", ["ModeFreq", "ParticipationFactor_DirX", "ParticipationFactor_DirY", "ParticipationFactor_DirZ", "ParticipationFactor_RotX", "ParticipationFactor_RotY", "ParticipationFactor_RotZ"]])</pre>

Example 2 – Modal (Fields Report):

```
Set oModule = oDesign.GetModule("ReportSetup")
oModule.CreateReport("Calculator Expressions Plot 1",
    "Fields", "Rectangular Stacked Plot",
    "Setup1 : Solution",
    [
        "Context:=", "Polyline1",
        "PointCount:=", 101
    ],
    [
        "Distance:=", ["0.25mm", "3.35mm"],
        "Mode:=", ["1", "2", "4", "5", "7"]
    ],
    [
        "X Component:=", "Distance",
        "Y Component:=", ["Mag_Displacement"]
    ])

```

Example 3 – Steady-State Thermal (with Trace Mapping):

```
Set oModule = oDesign.GetModule("ReportSetup")
oModule.CreateReport("Calculator Expressions Plot 2",
    "Fields", "Rectangular Plot", "Setup1 : Solution",
    [
        "Context:=", "Polyline1",
        "PointCount:=", 101
    ],
    [
        "Distance:=", ["All"]
    ],
    [
        "X Component:=", "Distance",
    ])

```

```

        "Y Component:=" ,
    [
        "Temperature" ,
        "Mag_HeatFlux",
        "Volume_Loss_Density",
        "Thermal_Conductivity_X",
        "Thermal_Conductivity_Y",
        "Thermal_Conductivity_Z"
    ]
    ])

```

Example 4 – Transient Thermal (Monitor Report)

A monitor report example is given because fields reports for transient thermal solutions are the same as those for steady-state thermal solutions (see Example 3).

```

Set oModule = oDesign.GetModule("ReportSetup")
oModule.CreateReport "Monitor Plot 1", "Monitor",
    "Rectangular Plot", "Setup1 : Solution", [],
    [
        "Time:=" , ["All"]
    ]
    [
        "X Component:=", "Time",
        "Y Component:=",
        [
            "Global.Temperature",
            "HeatSinkCentroid.Temperature",
            "RodCentroid.Temperature"
        ]
    ]
    ])

```

Example 5 – Structural (with Trace Mapping)

```

Set oModule = oDesign.GetModule("ReportSetup")

```


	<pre> oModule.CreateReport("Calculator Expressions Plot 3", "Fields", "Rectangular Plot", "Setup1 : Solution", ["Context:=", "Polyline1", "PointCount:=", 101], ["Distance:=" , ["All"]], ["X Component:=", "Distance", "Y Component:=", ["Equivalent_Stress", "Equivalent_Strain"]]) </pre>

VB Syntax	CreateReport(<ReportName>, <ReportType>, <DisplayType>, <SolutionName>, Array(<ContextArray>), Array(<FamiliesArray>), Array(<ReportDataArray>))
VB Example	The same five examples presented in the preceding table are included below in the equivalent Visual Basic Script format: Example 1v: Modal (Modal Report): <pre> Set oModule = oDesign.GetModule("ReportSetup") oModule.CreateReport "Frequency Table 1", "Modal", _ "Data Table", "Setup1 : Solution", Array(), _ Array("Mode:=", Array("All")), _ Array("X Component:=", "Mode", "Y Component:=", _ Array("ModeFreq", "ParticipationFactor_DirX", _ "ParticipationFactor_DirY", _ </pre>

```
"ParticipationFactor_DirZ", _
"ParticipationFactor_RotX", _
"ParticipationFactor_RotY", _
"ParticipationFactor_RotZ"))
```

Example 2v – Modal (Fields Report):

```
Set oModule = oDesign.GetModule("ReportSetup")
oModule.CreateReport "Calculator Expressions Plot 1", _
"Fields", "Rectangular Stacked Plot", _
"Setup1 : Solution", Array("Context:=", _
"Polyline1", "PointCount:=", 101), _
Array("Distance:=", Array("0.25mm", "3.35mm"), _
"Mode:=", Array("1", "2", "4", "5", "7")) _
Array("X Component:=", "Distance", "Y Component:=", _
Array("Mag_Displacement"))
```

Example 3v – Steady-State Thermal (with Trace Mapping): ,

```
Set oModule = oDesign.GetModule("ReportSetup")
oModule.CreateReport "Calculator Expressions Plot 2", _
"Fields", "Rectangular Plot", "Setup1 : Solution", _
Array("Context:=", "Polyline1", "PointCount:=", _
101), Array("Distance:=", Array("All")), _
Array("X Component:=", "Distance", _
"Y Component:=", Array("Temperature", _
"Mag_HeatFlux", "Volume_Loss_Density", _
"Thermal_Conductivity_X", _
"Thermal_Conductivity_Y", _
"Thermal_Conductivity_Z"))
```

Example 4v – Transient Thermal (Monitor Report)

A monitor report example is given because fields reports for transient thermal solutions are the same as those for steady-state thermal solutions (see Example 3v).

	<pre>Set oModule = oDesign.GetModule("ReportSetup") oModule.CreateReport "Monitor Plot 1", "Monitor", _ "Rectangular Plot", "Setup1 : Solution", Array(), _ Array("Time:=", Array("All")), Array(_ "X Component:=", "Time", "Y Component:=", _ Array("Global.Temperature", _ "HeatSinkCentroid.Temperature", _ "RodCentroid.Temperature"))</pre> Example 5v – Structural (with Trace Mapping) <pre>Set oModule = oDesign.GetModule("ReportSetup") oModule.CreateReport "Calculator Expressions Plot 3", _ "Fields", "Rectangular Plot", "Setup1 : Solution", _ Array("Context:=", "Polyline1", "PointCount:=", _ 101), Array("Distance:=", Array("All")), _ Array("X Component:=", "Distance", "Y Component:=", _ Array("Equivalent_Stress", "Equivalent_Strain"))</pre>

CreateReportFromFile

Creates a new report from an .rdat file.

UI Access	Right-click on Results > Create Report From File...		
Parameters	Name	Type	Description
	<FilePathName>	String	Path to .rdat file.
Return Value	None.		

Python Syntax	CreateReportFromFile(<FilePathName>)
Python Example	oModule.CreateReportFromFile("C:/Users/MyDir/Documents/Return_Loss.rdat")

VB Syntax	CreateReportFromFile <FilePathName>
VB Example	oModule.CreateReportFromFile "C:/Users/MyDir/Documents/Return_Loss.rdat"

CreateReportFromTemplate

Creates a report from a saved template.

UI Access	[product] > Results > Report Templates > PersonalLib > [TemplateName]		
Parameters	Name	Type	Description
	<TemplatePath>	String	Path to report template
Return Value	None		

Python Syntax	CreateReportFromTemplate(<TemplatePath>)
Python Example	oModule.CreateReportFromTemplate("C:/MyHFSSProjects/PersonalLib/ReportTemplates/TestTemplate.rpt")

VB Syntax	CreateReportFromTemplate <TemplatePath>

VB Example	<pre>oModule.CreateReportFromTemplate "C:/MyHFSSProjects/PersonalLib/" _ "ReportTemplates/TestTemplate.rpt"</pre>

CreateReportOfAllQuantities

Creates a report including all quantities in a category. Cannot create a report with expressions.

UI Access	NA		
Parameters	Name	Type	Description
	<ReportType>	String	Report type name as input parameter
	<DisplayType>	String	Display type name as input parameter
	<SolutionName>	String	Solution name as input parameter
	<SimValueCtxt>	String	A context name, or array of string that encoded the context(I).
	<CategoryName>	String	Category name as input parameter
	<PointSet>	Array	Array of strings(II)
	<CommonComponentsOfTraces>	Array	Array of strings(III)
	<ExtTraceInfo>	Array	Array of strings(IV)
Return Value	None.		

Python Syntax	<pre>CreateReportOfAllQuantities(<ReportNameArg>, <ReportType>, <DisplayType>, <SolutionName>, <SimValueCtxt>, <CategoryName>, <PointSet>, <CommonComponentsOfTraces>, <ExtTraceInfo>)</pre>
Python Example	<pre>oModule.CreateReportOfAllQuantities("Smith Chart all", "Modal Solution Data", "Smith Chart", "Setup1 : LastAdaptive",</pre>

	<pre>[],"S Parameter",["Freq=", ["All"], "offset=", ["All"],"a=", ["Nominal"],"b=", ["Nominal"]],[],[])</pre>

VB Syntax	<code>CreateReportOfAllQuantities <ReportNameArg>, <ReportType>, <DisplayType>, <SolutionName>, <SimValueCtxt>, <CategoryName>, <PointSet>, <CommonComponentsOfTraces>, <ExtTraceInfo></code>
VB Example	<pre>oModule.CreateReportOfAllQuantities "Smith Chart all", _ "Modal Solution Data", "Smith Chart", "Setup1 : LastAdaptive", _ Array(),"S Parameter", _ Array("Freq=", Array("All"), "offset=", Array("All"), _ "a=", Array("Nominal"),"b=", Array("Nominal")), _ Array(), Array()</pre>

DeleteMarker

Use:Deletes the specified marker.

UI Access	[product] > Fields > Fields > Marker > Delete Marker .		
Parameters	Name	Type	Description
	<MarkerName>	String	Name of the marker.
Return Value	None.		

Python Syntax	DeleteMarker(<MarkerName>)
Python Example	<code>oModule.DeleteMarker("m1")</code>

VB Syntax	DeleteMarker <MarkerName>
VB Example	<code>oModule.DeleteMarker "m1"</code>

DeleteAllReports

Deletes all existing reports.

UI Access	Right-click the report to delete in the project tree, and then click Delete All Reports on the shortcut menu.
Parameters	None.
Return Value	None.

Python Syntax	DeleteAllReports()
Python Example	<code>oModule.DeleteAllReports()</code>

VB Syntax	DeleteAllReports
VB Example	<code>oModule.DeleteAllReports</code>

DeleteReports

Deletes an existing report or reports.

UI Access	Right-click the report to delete in the project tree, and then click Delete on the shortcut menu		
Parameters	Name	Type	Description
	<ReportNameArray>	Array	Array of report names to be deleted
Return Value	None.		

Python Syntax	DeleteReports(<ReportNameArray>)		
Python Example	oModule.DeleteReports (["Rept2DRectFreq"])		

VB Syntax	DeleteReports <ReportNameArray>		
VB Example	oModule.DeleteReports Array ("Rept2DRectFreq")		

DeleteTraceCharacteristics

Deletes a trace characteristics field from a report

UI Access	N/A		
Parameters	Name	Type	Description
	<ReportName>	String	Name of the report to delete from.
	<TraceCharsNames>	Array	Array of trace characteristics to delete.
Return Value	None.		

Python Syntax	DeleteTraceCharacteristics(<ReportName>, <TraceCharsNames>)
Python Example	<pre>oModule.DeleteTraceCharacteristics("Variables Plot 1", ["lowercutoff", "gain-crossover"])</pre>

VB Syntax	DeleteTraceCharacteristics <ReportName>, <TraceCharsNames>
VB Example	<pre>oModule.DeleteTraceCharacteristics "Variables Plot 1", _ Array("lowercutoff", "gaincrossover")</pre>

DeleteTraces

Deletes an existing traces or traces.

UI Access	Right-click the Trace to delete in the project tree, and then click Delete on the shortcut menu		
Parameters	Name	Type	Description
	<TraceSelection>	Array	Structured array define selections. Array("<ReportName>:=", <TracesArray>, <TracesArray>, ...)
	<ReportName>	String	Name of Report
	<TracesArray>	Array	Contains the traces to delete within a report Array(<Trace>, <Trace>, ...)
	<Trace>	String	A specific trace that the user wishes to delete

Return Value	None.

Python Syntax	DeleteTraces(<TraceSelection>)
Python Example	<pre>oModule.DeleteTraces (["XY Plot 1:=", ["dB(S(LumpPort1,LumpPort1))"], "XY Plot 2:=", ["Mag_E"]])</pre>

VB Syntax	DeleteTraces <TraceSelection>
VB Example	<pre>oModule.DeleteTraces Array("XY Plot 1:=", _ Array("dB(S(LumpPort1,LumpPort1))"),) "XY Plot 2:=", Array("Mag_E"))</pre>

DoesSupportTraceCharacteristics

Determines whether trace characteristics is supported in a specified display type.

UI Access	N/A		
Parameters	Name	Type	Description
	<DisplayType>	String	Specified display type to check.
Return Value	Integer • 1 - trace characteristics is supported. • 0 - trace characteristics not supported.		

Python Syntax	DoesSupportTraceCharacteristics(<DisplayType>)
Python Example	<code>oModule.DoesSupportTraceCharacteristics("Rectangular Plot")</code>

VB Syntax	DoesSupportTraceCharacteristics <DisplayType>
VB Example	<code>oModule.DoesSupportTraceCharacteristics "Rectangular Plot"</code>

DumpAllReportsData

Dumps all reports data to an Ansoft report data file.

UI Access	N/A		
Parameters	Name	Type	Description
	<FileName>	String	File name with path.
Return Value	None.		

Python Syntax	DumpAllReportsData(<FileName>)
Python Example	<code>oModule.DumpAllReportsData("C:/ReportsData.rdat")</code>

VB Syntax	DumpAllReportsData <FileName>

VB Example	<code>oModule.DumpAllReportsData "C:/ReportsData.rdat"</code>

EditCartesianXMarker

Edits an XMarker value.

UI Access	N/A		
Parameters	Name	Type	Description
	<i><ReportName></i>	String	Name of report.
	<i><MarkerName></i>	String	Marker name, including any trailing number.
	<i><XValue></i>	Double	X coordinate.
Return Value	None.		

Python Syntax	<code>EditCartesianXMarker (<ReportName>, <MarkerName>, <XValue>)</code>
Python Example	<code>oModule.EditCartesianXMarker ("XY Plot 1", "MX1", 0)</code>

VB Syntax	<code>EditCartesianXMarker <ReportName>, <MarkerName>, <XValue></code>
VB Example	<code>oModule.EditCartesianXMarker "XY Plot1", "MX1", 0</code>

EditCartesianYMarker

Edits a YMarker value.

UI Access	N/A

Parameters	Name	Type	Description
	<ReportName>	String	Name of report.
	<MarkerName>	String	Marker name, including any trailing number.
	<AxisName>	String	Name of axis.
	<YValue>	Double	Y coordinate.
	<CurveName>	String	Name of curve.
Return Value	None		

Python Syntax	EditCartesianYMarker (<ReportName>, <MarkerName>, <AxisName>, <YValue>, <CurveName>)
Python Example	<pre>oModule.EditCartesianYMarker("XY Plot 1", "MY1", "Y1", 0, "dB() : Setup1 : Sweep1")</pre>

VB Syntax	EditCartesianYMarker <ReportName>, <MarkerName>, <AxisName>, <YValue>, <CurveName>
VB Example	<pre>oModule.EditCartesianYMarker "XY Plot 1", "MY1", "Y1", 0, _ "dB() : Setup1 : Sweep1"</pre>

EditMarker

Edits a marker on a report.

UI Access	N/A		
Parameters	Name	Type	Description

	<i><ReportName></i>	String	Name of report.
	<i><MarkerName></i>	String	Marker name, including any trailing number.
	<i><CurveName></i>	String	Full trace name.
	<i><PrimarySweepValue></i>	String	Primary sweep value, including unit.
Return Value	None.		

Python Syntax	EditMarker(<i><ReportName></i> , <i><MarkerName></i> , <i><CurveName></i> , <i><PrimarySweepValue></i>)		
Python Example	<pre>oModule.EditMarker("XY Plot 1", "m5", "GS1.VAL : TR4 : Cartesian", "3.61599999999997s")</pre>		

VB Syntax	EditMarker <i><ReportName></i> , <i><MarkerName></i> , <i><CurveName></i> , <i><PrimarySweepValue></i>		
VB Example	<pre>Set oModule = oDesign.GetModule("ReportSetup") oModule.EditMarker "XY Plot1", "m1", "mag(S(Port1 Port1)) : Setup1: LastAdaptive : Cartesian", "0.3in"</pre>		

ExportEyeMaskViolation

Exports eye mask violations to a file.

UI Access	N/A		
Parameters	Name	Type	Description

	<ReportName>	String	Name of specified report.
	<ExportFileName>	String	File name to export to.
Return Value	N/A		

Python Syntax	ExportEyeMaskViolation(<ReportName>, <ExportFileName>)		
Python Example	oModule.ExportEyeMaskViolation("Variables Plot 1", "C:/eyemask1.csv")		

VB Syntax	ExportEyeMaskViolation <ReportName>, <ExportFileName>		
VB Example	oModule.ExportEyeMaskViolation "Variables Plot 1", "C:/eyemask1.csv"		

ExportImageToFile [Reporter]

Exports a report image in a specified format. This command is fully supports -ng (non-graphical) mode.

UI Access	N/A		
Parameters	Name	Type	Description
	<ReportName>	String	Name of report to be exported.
	<FileName>	String	Full path of the exported image file name; with extension of jpg, gif, tiff, tif, bmp, or wrl.
	<Width>	Integer	Image width in pixels; if width or height is less or equal to zero, use the report window width, or 500 pixels if report window is closed.
	<Height>	Integer	Image height in pixels; if width or height is less or equal to zero, use the report window height, or 500 pixels report window is closed.

Return Value	None.

Python Syntax	ExportImageToFile(<ReportName>, <FileName>, <Width>, <Height>)
Python Example	<pre>##### Script Recorded for Saving 2D oDesktop.OpenProject("E:/Ansoft/Tee2.aedt") oProject = oDesktop.SetActiveProject("Tee2") oDesign = oProject.SetActiveDesign("TeeModel") oModule = oDesign.GetModule("ReportSetup") oModule.ExportImageToFile("S Parameter Plot 1", "E:/Ansoft/S Parameter Plot 1.png", 1920, 1080) oModule.ExportImageToFile("S Parameter Plot 12", "E:/Ansoft/S Parameter Plot 12.png", 1920, 1080) oModule.ExportImageToFile("S Parameter Plot 1", "E:/Ansoft/S Parameter Plot 1.png", 1920, 1080, ["NAME:SaveImageParams", "ShowLegend:=" , "False", "ShowMarkerTable:=" , "False"]) oModule.ExportImageToFile("S Parameter Plot 1", "E:/Ansoft/S Parameter Plot 1.png", 1920, 1080, [</pre>


```
"NAME:SaveImageParams",
"ShowLegend:=" , "True",
"ShowMarkerTable:=" , "True"
])
oModule.ExportImageToFile("S Parameter Plot 12", "E:/Ansoft/S Parameter Plot 12.png",
1920, 1080,
[
"NAME:SaveImageParams",
"ShowLegend:=" , "False",
"ShowMarkerTable:=" , "Default"
])
oModule.ExportImageToFile("S Parameter Plot 12", "E:/Ansoft/S Parameter Plot 12.png",
1920, 1080,
[
"NAME:SaveImageParams",
"ShowLegend:=" , "True",
"ShowMarkerTable:=" , "Default"
])
oDesktop.CloseProject("Tee2")

### Script Recorded for saving as 3D:
```

```
oDesktop.OpenProject("E:/Ansoft/Tee2.aedt")
oProject = oDesktop.SetActiveProject("Tee2")
oDesign = oProject.SetActiveDesign("TeeModel")
oModule = oDesign.GetModule("ReportSetup")
oModule.ExportImageToFile("S Parameter Plot 6", "E:/Ansoft/S Parameter Plot 11.png",
1335, 466,
[
    "NAME:SaveImageParams",
    "AutoFit:=" , "False",
    "Orientation:=" , "",
    "ShowOrientationGadget:=", "Default"
])
oModule.ExportImageToFile("S Parameter Plot 6", "E:/Ansoft/S Parameter Plot 11.png",
1335, 466,
[
    "NAME:SaveImageParams",
    "AutoFit:=" , "False",
    "Orientation:=" , "",
    "ShowOrientationGadget:=", "Default"
    "ShowLegend:=" , "True"
])
```

```
oModule.ExportImageToFile("S Parameter Plot 6", "E:/Ansoft/S Parameter Plot 10.png",  
1335, 466,  
  
[  
    "NAME:SaveImageParams",  
    "AutoFit:=" , "False",  
    "Orientation:=" , "",  
    "ShowOrientationGadget:=", "Default",  
    "ShowLegend:=" , "False"  
])
```

VB Syntax	ExportImageToFile <ReportName>, <FileName>, <Width>, <Height>
VB Example	<pre>oModule.ExportImageToFile _ "Rectangular Contour Plot 1", _ "D:/work/2015/UV-Export/pp1.gif", 0, 0</pre>

ExportModelImageToFile

Exports the model as an image file (*.avz, *.bmp, *.gif, *.jpeg, *.tiff, *.wrl). In Release 23.1, this command is fully supports -ng (non-graphical) mode. To export to Ensignt use *.avz. For export to Ensignt in -ng mode, the corresponding version of Ensignt 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 Ensignt 21.2.

ExportModelImageToFile supports export overlay of polar plot 3D with existing transformation (scaling, rotation and translation) in -ng (non-graphical) mode.

UI Access	Modeler > Export.		
Parameters	Name	Type	Description
	<path>	String	Full file path including extension.
	<width>	Integer	Width in pixels (use 0 for default).
	<height>	Integer	Height in pixels (use 0 for default).
	<Parameters>	Array	Structured array.
			Array ("NAME:SaveImageParams", "ShowAxis:=" , <string containing boolean>, "ShowGrid:=" , <string containing boolean>, "ShowRuler:=" , <string containing boolean>, "ShowRegion:=" , <string>, "Selections:=" , <string>, "FieldPlotSelections:=", <string>' # Comma delimited string. #Use to set which field plot to show. "FitToSelections:=" , "" , Note: "FitToSelections" specify geometry objects for the "Fit" operation. "FitToFieldPlotSelection:=" , ""

			Note: "FitToFieldPlotSelections" specifies field plots for the "Fit" operation. "AutoFit:=" , "True", Note: If FitToSlections or FitToFieldPlotSelections are used , then then AutoFit is true, it makes sure color key does not overlap field plot. It is true by default. If neither is used, then "AutoFit" will "Fit" to full model. "Orientation:=" , <string> "ShowOrientationGadget:=" , <False>
Return Value	None.		

Python Syntax	ExportModelImageToFile(<path> <width> <height> <Parameters>)
Python Example	<pre>oEditor.ExportModelImageToFile ("C:/Users/jdoe/Desktop/export.bmp", 1920, 1080, ["NAME:SaveImageParams",</pre>

	<pre> "ShowAxis:=" , "True", "ShowGrid:=" , "True", "ShowRuler:=" , "True", "ShowRegion:=" , "Default", "Selections:=" , "", "FieldPlotSelections:=" , "", "FitToSelections:=" , "", "FitToFieldPlotSelections:=" , "", "AutoFit:=" , "True", "Orientation:=" , "" l) </pre>

VB Syntax	ExportModelImageToFile <path> <width> <height> <Parameters>
VB Example	<pre> oEditor.ExportModelImageToFile "C:/Users/jdoe/Desktop/export.bmp", 1383, 512, Array("NAME:SaveImageParams", "ShowAxis:=" , "True", "ShowGrid:=" , "True", "ShowRuler:=" , "True", </pre>

	<pre>"ShowRegion:=", "Default", "Selections:=", "", "FitToFieldPlotSelections:=", "", "AutoFit:=", "True", "Orientation:=", ""</pre>

ExportModelMeshToFile

Exports geometry model to a 3D model file (e.g. *.obj, *.wrl, etc.).

UI Access	N/A		
Parameters	Name	Type	Description
	<filePath>	String	Full file path, including extension *.obj, *.wrl, etc
	<selections>	Array	Selected parts.
Return Value	None.		

Python Syntax	ExportModelMeshToFile <filePath>, <selections>)
Python Example	<pre>oEditor.ExportModelMeshToFile("E:/MyDir/scriptRun/2Selected-ng.obj", ["BotCover- ", "AveragingVolumeAtPeakRMSEfieldLocation"])</pre>

VB Syntax	ExportModelMeshToFile <filePath>, <selections>
VB Example	<pre>oEditor.ExportModelMeshToFile("E:/MyDir/scriptRun/2Selected-ng.obj", ["BotCover- ", "AveragingVolumeAtPeakRMSEfieldLocation"])</pre>

ExportPlot3DToFile [Reporter]

Use: Exports 3D polar, spherical and rectangular plot to a case file. It works in both graphical and NG mode..

Command: None.

Syntax: ExportPlot3DToFile(<plotName>, <path>)

Return Value: A 3D plot file.

Parameters: <plotName>

Type: <string>

Plot name.

<Path>

Type: <string>

Path to file.

Python Syntax	ExportPlot3DToFile(<name> <Path>)
Python Example	<pre>oModule = oDesign.GetModule("ReportSetup") oModule.UpdateReports(["rE Plot 1"]) oModule.UpdateReports(["Directivity Plot 1"]) oModule.UpdateReports(["Gain Plot 1"]) oModule.ExportPlot3DToFile("rE Plot 1", "D:/projects/test2-output/rEPlot1.case") oModule.ExportPlot3DToFile("Directivity Plot 1", "D:/projects/test2-out- put/DirectivityPlot1.case") oModule.ExportPlot3DToFile("Gain Plot 1", "D:/projects/test2-out-</pre>

	put/GainPlot1.case")

VB Syntax	ExportPlot3DToFile <plotName> <Path>
VB Example	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"
```

Python Syntax	ExportReport(<ReportName>, <FileName>)
Python Example	oDesign.ExportReport ("Plot1", "c:\report1.dat")

ExportReportDataToFile

Exports report data to a file.

UI Access	N/A		
Parameters	Name	Type	Description
	<ReportName>	String	Name of specified report.
	<ExportFileName>	String	Name of export file. File extension "rdat" is expected.
Return Value	None.		

Python Syntax	ExportReportDataToFile(<ReportName>, <ExportFileName>)
Python Example	<code>oModule.ExportReportDataToFile("Plot 1", "C:/Plot1data.rdat")</code>

VB Syntax	ExportReportDataToFile <ReportName>, <ExportFileName>
VB Example	<code>oModule.ExportReportDataToFile "Plot 1", "C:/Plot1data.rdat"</code>

ExportTableToFile

Exports a marker table from a report to a file.

UI Access	Right-click on a plot, select Marker > Export Marker Table... or Export Delta Marker Table...		
Parameters	Name	Type	Description
	<ReportName>	String	Name of specified report.

	<ExportFileName>	String	Name of export file.
	<TableType>	String	Type of marker table to export. "Marker" or "DeltaMarker".
Return Value	None.		

Python Syntax	ExportTableToFile(<ReportName>, <ExportFileName>, <TableType>)		
Python Example	oModule.ExportTableToFile("Plot 1", "C:/Marker.csv", "Marker")		

VB Syntax	ExportTableToFile <ReportName>, <ExportFileName>, <TableType>		
VB Example	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.

UI Access	Right-click on report name in the Project tree and select Export .		
Parameters	Name	Type	Description
	<ReportName>	String	Name of the report
	<FileName>	String	Path and File Name
			Supported formats
			.txt Post processor format file
			.csv Comma-delimited data file
			.tab Tab-separated file

			.dat	Ansyes plot data file
			.rdat	Ansyes report data file
Return Value	None			

Python Syntax	ExportToFile (<ReportName>, <FileName>)
Python Example	<pre>oModule.ExportToFile("Plot1", "c:\report1.dat")</pre>

VB Syntax	ExportToFile <ReportName>, <FileName>
VB Example	<pre>oModule.ExportToFile "Plot1", "c:\report1.dat"</pre>

ExportUniformPointsToFile

Exports uniform points from a data table or plot report that includes the Export Uniform Points to File option enabled to text format, comma delimited, tab delimited, or .dat type output files.

File name:

erEPhi, Phi=0 plane For Phase Center Calc.csv

Save

Save as type:

Comma delimited data files(*.csv)

Cancel

Options

☒ Use Separate Columns for Curves

☒ Export Uniform Points

☒ Export with Full Sweep Range

Start:

-180

deg

Stop:

180

deg

Step:

36

deg

UI Access	Right-click on report name in the project tree and select Export Data .		
Parameters	Name	Type	Description
	<ReportName>	String	Name of report to be exported.
	<FileName>	String	Full path of the exported image file name; with extension of .txt - Post processor format file .csv - Comma-delimited data file .tab - Tab-separated file .dat - Ansys plot data file
	<SweepRange>	String	Start, stop, and step range with units, for sweep.
	<UnitSpec>	String	For example, "kV, Mhz, yd"
	<UseTraceNumberFormat>	Boolean	"True", "False"
Return Value	None.		

Python Syntax	<code>ExportUniformPointsToFile(<ReportName>, <FileName>, <SweepRange>)</code>
Python Example	<code>oModule.ExportUniformPointsToFile("S Parameter Table 2", "D:/MyFiles/cftt.csv", "4GHz", "5GHz", "1GHz", False, " kV, MHz, yd ", False)</code>

VB Syntax	<code>ExportUniformPointsToFile <ReportName>, <FileName>, <SweepRange></code>
VB Example	<code>oModule.ExportUniformPointsToFile "rETheta and rEPhi [dB]", "C:/MyFiles/rETheta and rEPhi [dB].csv", "-180deg", "180deg", "36deg"</code>

GetAllCategories

Get all available category names (not including variable and output-variables) in a solution for a report type and display type, returned as an array of text strings.

UI Access	NA		
Parameters	Name	Type	Description
	<code><ReportType></code>	String	Report type name.
	<code><DisplayType></code>	String	Name of display type.
	<code><SolutionName></code>	String	Name of solution.
	<code><SimValueCtxt></code>	Array	A context name, or array of strings that encode the contexts.
Return Value	Array of text strings		

Python Syntax	<code>GetAllCategories(<ReportType>, <DisplayType>, <SolutionName>, <SimValueCtxt>)</code>
Python Example	<pre>oModule.GetAllCategories("Far Fields", "Rectangular Plot", "Setup1 : LastAdaptive", "Infinite Sphere1")</pre>

VB Syntax	<code>GetAllCategories <ReportType>, <DisplayType>, <SolutionName>, <SimValueCtxt></code>
VB Example	<pre>oModule.GetAllCategories _ "Far Fields", "Rectangular Plot", _ "Setup1 : LastAdaptive", "Infinite Sphere1"</pre>

GetAllQuantities

Gets all available quantity names in category, returned as an array of text strings.

UI Access	NA		
Parameters	Name	Type	Description
	<ReportType>	String	Report type name.
	<DisplayType>	String	Display type name.
	<SolutionName>	String	Name of solution.
	<SimValueCtxt>	Array	A context name, or array of string that encoded the contexts(I).
	<CategoryName>	String	A category name as input parameter. a category name returned in GetAllCategories() or "Variables", or "Output Variables"
Return Value	Array of text strings		

Python Syntax	<code>GetAllQuantities(<ReportType>,<DisplayType>, <SolutionName>, <SimValueCtxt>, <CategoryName>)</code>
Python Example	<pre>oModule.GetAllQuantities("Far Fields", "Rectangular Plot", "Setup1 : LastAdaptive", "Infinite Sphere1", "Gain")</pre>

VB Syntax	<code>GetAllQuantities <ReportType>,<DisplayType>, <SolutionName>, <SimValueCtxt>, <CategoryName></code>
VB Example	<pre>oModule.GetAllQuantities _ "Far Fields", "Rectangular Plot", _ "Setup1 : LastAdaptive",) "Infinite Sphere1", "Gain"</pre>

GetAllReportNames

Gets the names of existing reports in a design

UI Access	N/A
Parameters	None.
Return Value	Array of report names

Python Syntax	GetAllReportNames()
Python Example	<code>oModule.GetAllReportNames()</code>

VB Syntax	GetAllReportNames
VB Example	<code>oModule.GetAllReportNames</code>

GetAvailableDisplayTypes

Retrieves all supported display types in report type as an array of text strings.

UI Access	N/A		
Parameters	Name	Type	Description
	<ReportType>	String	Report type name.
Return Value	Array of text strings		

Python Syntax	GetAvailableDisplayTypes(<ReportType>)
Python Example	<code>oModule.GetAvailableDisplayTypes("Far Fields")</code>

VB Syntax	GetAvailableDisplayTypes <ReportType>
VB Example	<code>oModule.GetAvailableDisplayTypes "Far Fields"</code>

GetAvailableReportTypes

Retrieves all available report types in the current Design as an array of text string.

UI Access	N/A
Parameters	None.
Return Value	Array of text strings

Python Syntax	GetAvailableReportTypes()
Python Example	<code>oModule.GetAvailableReportTypes()</code>

VB Syntax	GetAvailableReportTypes
VB Example	<code>oModule.GetAvailableReportTypes</code>

GetAvailableSolutions

Gets all available solutions in report type as an array of text strings.

UI Access	N/A		
Parameters	Name	Type	Description
	<ReportType>	String	Report type name.
Return Value	Array of text strings		

Python Syntax	GetAvailableSolutions(<ReportType>)
Python Example	<code>oModule.GetAvailableSolutions("Far Fields")</code>

VB Syntax	GetAvailableSolutions <ReportType>
VB Example	<code>oModule.GetAvailableSolutions "Far Fields"</code>

GetChildNames [Report Setup]

Gets a list of report names.

UI Access	N/A
Parameters	None.
Return Value	Array of strings containing all report names.

Python Syntax	GetChildNames()
Python Example	<code>oModule.GetChildNames()</code>

VB Syntax	GetChildNames
VB Example	<code>oModule.GetChildNames</code>

GetChildObject [Report Setup]

Gets a report object or report child object; the module's first level of child object is report. Report has trace, axis, header, Legend, and more children. Trace has curve as child etc. Those child objects can be accessed by calling all levels of parent object's GetChildObject (path) function.

UI Access	NA		
Parameters	Name	Type	Description
	<ObjectPath>	String	Report Name or an object path beginning with a report name.
Return Value	A ReportSetup(Results) Module Child Object, [ReportSetup(Results) Module Child Objects]		

Python Syntax	GetChildObject(<ObjectPath>)
Python Example	<pre>oRpt = oRptModule.GetChildObject("S Parameter Plot 1") oTrace = oRptModule.GetChildObject("S Parameter Plot 1/dB(S(Port1,Port1))") oAxisX = oRptModule.GetChildObject("S Parameter Plot 1/AxisX")</pre>

VB Syntax	GetChildObject <ObjectPath>
VB Example	<pre>set oRpt = oRptModule.GetChildObject "S Parameter Plot 1" set oTrace = oRptModule.GetChildObject "S Parameter Plot 1/dB(S(Port1,Port1))" set oAxisX = oRptModule.GetChildObject "S Parameter Plot 1/AxisX"</pre>

GetChildTypes [ReportSetup]

Gets child types of queried Report module object.

UI Access	N/A
Parameters	None.
Return Value	Array of strings containing child object types.

Python Syntax	GetChildTypes ()
Python Example	<code>oModule.GetChildTypes ()</code>

VB Syntax	GetChildTypes
VB Example	<code>oModule.GetChildTypes</code>

GetCurvePropServerName

Gets the PropServer (the owner of the properties, or the list containing them) name of a curve.

UI Access	N/A		
Parameters	Name	Type	Description
	<ReportName>	String	Name of specified report.
	<TraceName>	String	Name of specified trace.
Return Value	Array of string containing the PropServer name.		

Python Syntax	<code>GetCurvePropServerName(<ReportName>, <TraceName>)</code>
Python Example	<code>oModule.GetCurvePropServerName("Plot 1", "Phase")</code>

VB Syntax	<code>GetCurvePropServerName <ReportName>, <TraceName></code>
VB Example	<code>oModule.GetCurvePropServerName "Plot 1", "Phase"</code>

GetDisplayType

Gets the display type of a specified report.

UI Access	NA		
Parameters	Name	Type	Description
	<ReportName>	String	Report name
Return Value	String containing display type.		

Python Syntax	<code>GetDisplayType(<ReportName>)</code>
Python Example	<code>oModule.GetDisplayType("Design Plot 1")</code>

VB Syntax	<code>GetDisplayType <ReportName></code>
VB Example	<code>oModule.GetDisplayType "Design Plot 1"</code>

GetDynLinkIntrinsicVariables

Gets variable names from a trace included in dynamic link outputs.

UI Access	N/A		
Parameters	Name	Type	Description
	<TraceName>	String	Trace name with its report name.
Return Value	Array of variable names		

Python Syntax	GetDynLinkIntrinsicVariables(<TraceName>)		
Python Example	oModule.GetDynLinkIntrinsicVariables("Plot 1:Trace")		

VB Syntax	GetDynLinkIntrinsicVariables <TraceName>		
VB Example	oModule.GetDynLinkIntrinsicVariables "Plot 1:Trace")		

GetDynLinkQtyValueState

Gets the state of the quantity values from a source dynamic linked trace.

UI Access	N/A		
Parameters	Name	Type	Description
	<TraceName>	String	Name of specified source trace.
	<QtyName>	String	Name of specified quantity value.
Return Value	Array of strings containing states.		

Python Syntax	GetDynLinkQtyValueState(<TraceName>, <QtyName>)
Python Example	oModule.GetDynLinkQtyValueState("Plot 1:Trace1", "")

VB Syntax	GetDynLinkQtyValueState <TraceName>, <QtyName>
VB Example	oModule.GetDynLinkQtyValueState "Plot 1:Trace1", ""

GetDynLinkTraces

Gets the names of the dynamic linked traces.

UI Access	N/A		
Parameters	Name	Type	Description
	<SoluName>	String	Name of the source solution. If empty, refer to current solution.
Return Value	Array of strings containing trace names.		

Python Syntax	GetDynLinkTraces(<SoluName>)
Python Example	oModule.GetDynLinkTraces("")

VB Syntax	GetDynLinkTraces <SoluName>

VB Example	<code>oModule.GetDynLinkTraces ""</code>

GetDynLinkVariableValues

Gets the values of a variable from a dynamic linked trace.

UI Access	N/A		
Parameters	Name	Type	Description
	<code><TraceName></code>	String	Name of specified dynamic linked trace, with its report name.
	<code><VarName></code>	String	Name of specified variable.
Return Value	Array of strings containing variable values.		

Python Syntax	<code>GetDynLinkVariableValues(<TraceName>, <VarName>)</code>
Python Example	<code>oModule.GetDynLinkVariableValues("Plot 1:Trace", "Var1")</code>

VB Syntax	<code>GetDynLinkVariableValues <TraceName>, <VarName></code>
VB Example	<code>oModule.GetDynLinkVariableValues "Plot 1:Trace", "Var1"</code>

GetName

Returns the design name of the active design, in that order separated by a semicolon.

UI Access	N/A
Parameters	None.

Return Value	String indicating the name of the active design.

Python Syntax	GetName()
Python Example	design_name = oDesign.GetName()

VB Syntax	GetName
VB Example	design_name = oDesign.GetName

GetObjPath [Design]

Obtains the path to the design.

UI Access	N/A
Parameters	None.
Return Value	String containing the path to the design.

Python Syntax	GetObjPath()
Python Example	oDesign.GetObjPath()

VB Syntax	GetObjPath
VB Example	oDesign.GetObjPath

GetPropertyValue

Returns the value of a single property belonging to a specific *<PropServer>* and *<PropTab>*. This function is available with the Project, Design or Editor objects, including definition editors.

Tip: Use the script recording feature and edit a property, and then view the resulting script to see the format for that property.

UI Access	N/A		
Parameters	Name	Type	Description
	<i><PropTab></i>	String	One of the following, where tab titles are shown in parentheses: - PassedParameterTab ("Parameter Values") - DefinitionParameterTab (Parameter Defaults") - LocalVariableTab ("Variables" or "Local Variables") - ProjectVariableTab ("Project variables") - ConstantsTab ("Constants") - BaseElementTab ("Symbol" or "Footprint") - ComponentTab ("General") - Component("Component") - CustomTab ("Intrinsic Variables") - Quantities ("Quantities") - Signals ("Signals")
	<i><PropServer></i>	String	An object identifier, generally returned from another script method, such as <code>CompInst@R;2;3</code>

	<table><tr><td><PropName></td><td>String</td><td>Name of the property.</td></tr></table>	<PropName>	String	Name of the property.
<PropName>	String	Name of the property.		
Return Value	String value of the property.			

Python Syntax	<code>GetPropertyValue (<PropTab>, <PropServer>, <PropName>)</code>
Python Example	<pre>selectionArray = oEditor.GetSelections() for k in selectionArray: val = oEditor.GetPropertyValue("PassedParameterTab", k, "R") ...</pre>

VB Syntax	<code>GetPropertyValue (<PropTab>, <PropServer>, <PropName>)</code>
VB Example	<pre>selectionArray = oEditor.GetSelections for k in selectionArray: val = oEditor.GetPropertyValue("PassedParameterTab", k, "R") ...</pre>

GetPropNames [Reporter]

Report setup module does not have its own property, this function always returns empty array.

UI Access	N/A
Parameters	None.
Return Value	Empty array.

Python Syntax	GetPropNames()
Python Example	<code>oModule.GetPropNames()</code>

VB Syntax	GetPropNames
VB Example	<code>oModule.GetPropNames</code>

GetPropValue [Report Setup]

Gets the property value for a Report, or reports' child object.

UI Access	N/A		
Parameters	Name	Type	Description
	<PropPath>	String	A child object's property path. See property path discussion here .
Return Value	String containing the property value.		

Python Syntax	GetPropValue(<PropPath>)
Python Example	<code>oModule.GetPropValue("S Parameter Plot 1/Display Type")</code>

VB Syntax	GetPropValue <PropPath>
VB Example	oModule.GetPropValue "S Parameter Plot 1/Display Type"

GetQtyExpressionsForSourceTrace

Gets the quantity expressions from a specified source trace.

UI Access	N/A		
Parameters	Name	Type	Description
	<SourceTraceName>	String	Name of specified source trace.
Return Value	Array of strings containing quantity expressions.		

Python Syntax	GetQtyExpressionsForSourceTrace(<SourceTraceName>)
Python Example	oModule.GetQtyExpressionsForSourceTrace("Plot 1:Trace1")

VB Syntax	GetQtyExpressionsForSourceTrace <SourceTraceName>
VB Example	oModule.GetQtyExpressionsForSourceTrace "Plot 1:Trace1"

GetReportTraceNames

Gets the names of existing trace names in a plot.

UI Access	N/A		
Parameters	Name	Type	Description
	<PlotName>	String	Name of specified plot.
Return Value	Array of strings containing trace names.		

Python Syntax	GetReportTraceNames(<PlotName>)		
Python Example	<code>oModule.GetReportTraceNames("SParameter Plot 1")</code>		

VB Syntax	GetReportTraceNames <PlotName>		
VB Example	<code>oModule.GetReportTraceNames "SParameter Plot 1"</code>		

GetReportSummaryForRegressionTesting

Gets report summary from a dumped report file.

UI Access	N/A		
Parameters	Name	Type	Description
	<ReportName>	String	Name of a specified dumped report.
Return Value	String containing report summary.		

Python Syntax	GetReportSummaryForRegressionTesting(<ReportName>)		

Python Example	<code>oModule.GetReportSummaryForRegressionTesting("C:/report.rdat")</code>

VB Syntax	<code>GetReportSummaryForRegressionTesting <ReportName></code>
VB Example	<code>oModule.GetReportSummaryForRegressionTesting "C:/report.rdat"</code>

GetSolutionContexts

Gets all available solution context names in a solution as an array of text strings.

UI Access	N/A		
Parameters	Name	Type	Description
	<ReportType>	String	Report type name.
	<DisplayType>	String	Display type name.
	<SolutionName>	String	Name of solution.
Return Value	Array of text strings		

Python Syntax	<code>GetSolutionContexts(<ReportType>, <DisplayType>, <SolutionName>)</code>
Python Example	<code>oModule.GetSolutionContexts("Far Fields", "Rectangular Plot", "Setup1:LastAdaptive")</code>

VB Syntax	<code>GetSolutionContexts <ReportType>, <DisplayType>, <SolutionName></code>

VB Example	<pre>oModule.GetSolutionContexts _ "Far Fields", "Rectangular Plot", _ "Setup1:LastAdaptive"</pre>

GetSolutionDataPerVariation

Obtains solution data for a given report type and solution. You must have already run a simulation.

UI Access	N/A		
Parameters	Name	Type	Description
	<reportTypeArg>	String	Report type name as input parameter.
	<solutionNameArg>	String	Solution name as input parameter.
	<simValueCtxtArg>	Structured Array	Same as ContextArray values created in the relevant CreateReport script.
	<familiesArg>	Array of Strings	Same as FamiliesArray values created in the relevant CreateReport script.
	<expressionArg>	String or Array of Strings	Text string or array of text strings; valid expression, may validate it as the data-table Y-component.
Return Value	ARRAY of ISolutionDataResultComInterface objects, containing: • GetSweepNames() • GetSweepUnits() • GetSweepValues() • IsPerQuantityPrimarySweep() • GetDataExpressions() – should be the same as <expressionArg> • GetPerQuantityPrimarySweepValues() • IsDataComplex() • GetDataUnits()		

	• GetRealDataValues() • GetImagDataValues() • ReleaseData() • GetDesignVariableNames() • GetDesignVariableUnits() • GetDesignVariableValue() • GetDesignVariationKey()

Note:

- This command is *not* recordable from the UI, but its parameters are similar to CreateReport, so you may record a CreateReport script to get the parameter values.
- For the returned ISolutionDataResultComInterface object, some of its functions have an optional boolean parameter: SValue. SValue defaults to True. When the pass in value is True, return data values will be in Standard International values; when False, return data values will be in the current units.

Example: Freq Sweep with [1GHz, 2GHz,3GHz], GetSweepUnits("Freq") return "GHz"; GetSweepValues("Freq", True) return [1000000,2000000,3000000]; GetSweepValues("Freq", False) return [1,2,3].

Python Syntax	GetSolutionDataPerVariation(reportTypeArg, solutionNameArg, simValueCtxtArg, familiesArg, expressionArg)
Python Example	<pre>oModule = oDesign.GetModule("ReportSetup") arr = oModule.GetSolutionDataPerVariation('Modal Solution Data', 'Setup1 : Sweep', ['Domain:=' , 'Sweep'], ['Freq:=' , ['All']], ['offset:=' , ['All']], ['S (Port1,Port1) ' , 'dB(S(Port1,Port3)) '])</pre>

VB Syntax	<code>GetSolutionDataPerVariation(reportTypeArg, solutionNameArg, simValueCtxtArg, familiesArg, expressionArg)</code>
VB Example	<pre>set solutionData = oModule.GetSolutionDataPerVariation("Modal Solution Data", "Setup1 : Sweep", Array("Domain:=", "Sweep"), Array("Freq:=", Array("All"), "offset:=", Array("All")), Array("S(Port1,Port1)", "dB(S(Port1,Port3))"))</pre>

GetDataExpressions

Returns data expressions.

Important:

This is a member function of ISolutionDataResultComInterface object, which is the element of the returned array from function [GetSolutionDataPerVariation](#).

UI Access	N/A
Parameters	N/A
Return Value	Array of text strings

Python Syntax	<code>GetDataExpressions()</code>

Python Example	<code>expressions = oModule.GetDataExpressions()</code>

VB Syntax	<code>GetDataExpressions()</code>
VB Example	<code>dim expressions expressions = oModule.GetDataExpressions()</code>

GetDataUnits

Returns text string containing units.

Important:

This is a member function of ISolutionDataResultComInterface object, which is the element of the returned array from function [GetSolutionDataPerVariation](#).

UI Access	N/A		
Parameters	Name	Type	Description
	<expressionString>	String	Can be returned from GetDataExpressions()
Return Value	Text string of units; empty if no units		

Python Syntax	<code>GetDataUnits(<expressionString>)</code>
Python Example	<code>oModule.GetDataUnits(expressions)</code>

VB Syntax	GetDataUnits(<expressionString>)
VB Example	<code>oModule.GetDataUnits(expressions)</code>

GetDesignVariableNames

Returns array of strings containing design variable names.

Important:

This is a member function of ISolutionDataResultComInterface object, which is the element of the returned array from function [GetSolutionDataPerVariation](#).

UI Access	NA
Parameters	NA
Return Value	Array of strings

Python Syntax	GetDesignVariableNames()
Python Example	<code>names = oModule.GetDesignVariableNames()</code>

VB Syntax	GetDesignVariableNames()

VB Example	<pre>dim names names = oModule.GetDesignVariableNames()</pre>

GetDesignVariableUnits

Returns array of strings containing design variable units.

Important:

This is a member function of ISolutionDataResultComInterface object, which is the element of the returned array from function [GetSolutionDataPerVariation](#).

UI Access	N/A		
Parameters	Name	Type	Description
	<varName>	String	Can be returned from GetDesignVariableNames()
Return Value	Text string of units; empty if no units.		

Python Syntax	GetDesignVariableUnits(<varName>)
Python Example	<pre>units = oModule.GetDesignVariableUnits('Variable Name')</pre>

VB Syntax	GetDesignVariableUnits(<varName>)
VB Example	<pre>dim units</pre>

```
units = oModule.GetDesignVariableUnits("Variable Name")
```

GetDesignVariableValue

Returns a design variable's value.

Important:

This is a member function of ISolutionDataResultComInterface object, which is the element of the returned array from function [GetSolutionDataPerVariation](#).

UI Access	N/A		
Parameters	Name	Type	Description
	<varName>	String	Can be returned from GetDesignVariableNames()
	<siValue>	Boolean	True to return SI-value; False to return by GetDesignVariableUnits()
Return Value	Double value		

Python Syntax	GetDesignVariableValue(<varName>, <siValue>)
Python Example	<pre>value = oModule.GetDesignVariableValue('varName',1)</pre>

VB Syntax	GetDesignVariableValue(<varName>, <siValue>)
VB Example	<pre>dim value value = oModule.GetDesignVariableValue("varName", True)</pre>

GetDesignVariationKey

Returns a design's Variation Key.

Important:

This is a member function of ISolutionDataResultComInterface object, which is the element of the returned array from function [GetSolutionDataPerVariation](#).

UI Access	N/A
Parameters	N/A
Return Value	String containing variation key.

Python Syntax	GetDesignVariationKey()
Python Example	<pre>oModule.GetDesignVariationKey()</pre>

VB Syntax	GetDesignVariationKey()
VB Example	<pre>dim key set key = oModule.GetDesignVariationKey()</pre>

GetImagDataValues

Returns array of imaginary data values.

Important:

This is a member function of ISolutionDataResultComInterface object, which is the element of the returned array from function [GetSolutionDataPerVariation](#).

UI Access	N/A		
Parameters	Name	Type	Description
	<expressionString>	String	Can be returned from GetDataExpressions()
	<siValue>	Boolean	True to return SI-value; False to return with units returned in GetSweepUnits() .
Return Value	Array of doubles		

Python Syntax	GetImagDataValues(<expressionString>,<siValue>)
Python Example	<pre>imaginaryvalues = oModule.GetImagDataValues('expression',1)</pre>

VB Syntax	GetImagDataValues(<expressionString>,<siValue>)
VB Example	<pre>dim imaginaryvalues imaginaryvalues = oModule.GetImagDataValues("expression",True)</pre>

GetPerQuantityPrimarySweepValues

Returns per quantity primary sweep values.

Important:

This is a member function of ISolutionDataResultComInterface object, which is the element of the returned array from function [GetSolutionDataPerVariation](#).

UI Access	N/A		
Parameters	Name	Type	Description
	<expressionString>	String	Can be returned from GetDataExpressions() .
	<siValue>	Boolean	True to return SI-value; False to return by GetSweepUnits() .
Return Value	Array of doubles if IsPerQuantityPrimarySweep() returned True; error if returned False		

Python Syntax	GetPerQuantitySweepValues(<expressionString>, <siValue>)
Python Example	<pre>sweepvalues = oModule.GetPerQuantitySweepValues('0.111,0.201,0.345,0.231', 1)</pre>

VB Syntax	GetPerQuantitySweepValues(<expressionString>, <siValue>)
VB Example	<pre>dim sweepvalues sweepvalues = oModule.GetPerQuantitySweepValues("0.111,0.201,0.345,0.231", True)</pre>

GetRealDataValues

Returns array of real data values.

Important:

This is a member function of ISolutionDataResultComInterface object, which is the element of the returned array from function [GetSolutionDataPerVariation](#).

UI Access	N/A		
Parameters	Name	Type	Description
	<expressionString>	String	Can be returned from GetDataExpressions()
	<siValue>	Boolean	True to return SI-value; False to return with units returned in GetSweepUnits() .
Return Value	Array of doubles		

Python Syntax	GetRealDataValues(<expressionString>,<siValue>)
Python Example	<pre>realvalues = oModule.GetRealDataValues('expression',1)</pre>

VB Syntax	GetRealDataValues(<expressionString>,<siValue>)
VB Example	<pre>dim realvalues realvalues = oModule.GetRealDataValues("expression",True)</pre>

GetSweepNames

Returns array of text strings containing primary sweep name(s).

Important:

This is a member function of ISolutionDataResultComInterface object, which is the element of the returned array from function [GetSolutionDataPerVariation](#).

UI Access	N/A
Parameters	N/A
Return Value	Array of text strings

Python Syntax	GetSweepNames()
Python Example	<pre>sweepnames = arr[0].GetSweepNames()</pre>

VB Syntax	GetSweepNames()
VB Example	<pre>dim sweepnames sweepnames = arr[0].GetSweepNames()</pre>

GetSweepUnits

Returns text string containing units.

Important:

This is a member function of ISolutionDataResultComInterface object, which is the element of the returned array from function [GetSolutionDataPerVariation](#).

UI Access	N/A		
Parameters	Name	Type	Description
	<sweepName>	String	Primary sweep name
Return Value	Text string containing units		

Python Syntax	GetSweepUnits(<sweepName>)
Python Example	<pre>sweepunits = oModule.GetSweepUnits('Sweep 1')</pre>

VB Syntax	GetSweepUnits(<sweepName>)
VB Example	<pre>dim sweepunits sweepunits = oModule.GetSweepUnits("Sweep 1")</pre>

GetSweepValues

Returns sweep values.

Important:

This is a member function of ISolutionDataResultComInterface object, which is the element of the returned array from function [GetSolutionDataPerVariation](#).

UI Access	N/A		
Parameters	Name	Type	Description
	<sweepName>	String	Primary sweep name
	<siValue>	Boolean	True to return SI-value; False to return by GetSweepUnits() .
Return Value	Array of doubles		

Python Syntax	GetSweepValues(<sweepName>, <siValue>)
Python Example	<pre>sweepvalues = oModule.GetSweepValues('Sweep 1', True)</pre>

VB Syntax	GetSweepValues(<sweepName>, <siValue>)
VB Example	<pre>dim sweepvalues sweepvalues = oModule.GetSweepValues("Sweep 1", True)</pre>

IsDataComplex

Returns whether an expression is complex.

Important:

This is a member function of ISolutionDataResultComInterface object, which is the element of the returned array from function [GetSolutionDataPerVariation](#).

UI Access	NA		
Parameters	Name	Type	Description
	<expressionString>	String	Can be returned from GetDataExpressions() .
Return Value	Boolean (True if expression is Complex data; False if not)		

Python Syntax	IsDataComplex(<expressionString>)
Python Example	<code>oModule.IsDataComplex('.001,.234,.455,.434')</code>

VB Syntax	IsDataComplex(<expressionString>)
VB Example	<code>oModule.IsDataComplex('.001,.234,.455,.434')</code>

IsPerQuantityPrimarySweep

Returns whether data expressions have different primary sweep values.

Important:

This is a member function of ISolutionDataResultComInterface object, which is the element of the returned array from function [GetSolutionDataPerVariation](#).

UI Access	N/A
Parameters	N/A
Return Value	Boolean (True if data expressions have different primary sweep values)

Python Syntax	IsPerQuantityPrimarySweep()
Python Example	<pre>var = oModule.IsPerQuantityPrimarySweep()</pre>

VB Syntax	IsPerQuantityPrimarySweep()
VB Example	<pre>dim var var = oModule.IsPerQuantityPrimarySweep()</pre>

Release Data

Releases all cached data. After this function is called, all subsequent function calls to the object will fail.

Important:

This is a member function of ISolutionDataResultComInterface object, which is the element of the returned array from function [GetSolutionDataPerVariation](#).

UI Access	N/A
Parameters	N/A
Return Value	N/A

Python Syntax	ReleaseData()
Python Example	<code>oModule.ReleaseData()</code>

VB Syntax	ReleaseData()
VB Example	<code>oModule.ReleaseData()</code>

GroupPlotCurvesByGroupingStrategy

Groups curves in a Stacked Plot automatically based on a curve grouping strategy.

UI Access	N/A		
Parameters	Name	Type	Description
	<ReportName>	String	Name of report.
	<GroupStrategy>	String	Strategy for grouping, "Single", "By Trace" or "By Units".

Return Value	None.

Python Syntax	<code>GroupPlotCurvesByGroupingStrategy(<ReportName>, <GroupStrategy>)</code>
Python Example	<code>oModule.GroupPlotCurvesByGroupingStrategy("Transient Plot 1", "By Trace")</code>

VB Syntax	<code>GroupPlotCurvesByGroupingStrategy <ReportName>, <GroupStrategy></code>
VB Example	<code>oModule.GroupPlotCurvesByGroupingStrategy "Transient Plot 1", "By Trace"</code>

ImportIntoReport

Imports .tab, .csv, and .dat format files into a report.

UI Access	Right-click on report name in the Project tree and select Import....		
Parameters	Name	Type	Description
	<ReportName>	String	Name of the Report
	<FileName>	String	Path and File Name
		.csv	Comma-delimited data file
		.tab	Tab-separated file
		.dat	Ansys plot data file
Return Value	None		

Python Syntax	ImportIntoReport (<ReportName>, <FileName>)
Python Example	oDesign.ImportIntoReport ("Plot1", "c:\report1.dat")

VB Syntax	ImportIntoReport <ReportName>, <FileName>
VB Example	oDesign.ImportIntoReport "Plot1", "c:\report1.dat"

ImportReportDataIntoReport

Imports report data from a file into a specified report.

UI Access	N/A		
Parameters	Name	Type	Description
	<ReportName>	String	Name of specified report.
	<FileName>	String	Name of specified data file. File extension "rdat" is expected.
Return Value	None.		

Python Syntax	ImportReportDataIntoReport(<ReportName>, <FileName>)
Python Example	oModule.ImportReportDataIntoReport("Plot 1", "C:/Plot1data.rdat")

VB Syntax	ImportReportDataIntoReport <ReportName>, <FileName>
VB Example	oModule.ImportReportDataIntoReport "Plot 1", "C:/Plot1data.rdat"

MovePlotCurvesToGroup

In a Stacked Plot move curve(s) from its stack(s) to an existing stack. Here term 'group' is synonymous to 'stack' in the context of cartesian stacked plot.

UI Access	N/A		
Parameters	Name	Type	Description
	<ReportName>	String	Name of report.
	<CurveArray>	Array	Array of curve names to move.
	<StackName>	String	Name of stack to move to.
Return Value	None.		

Python Syntax	MovePlotCurvesToGroup(<ReportName>, <CurveArray>, <StackName>)
Python Example	<pre>oModule.MovePlotCurvesToGroup("XY Stacked Plot 1", ["R2.V : TR", "R2.I : TR"], "Stack 2")</pre>

VB Syntax	MovePlotCurvesToGroup <ReportName>, <CurveArray>, <StackName>
VB Example	<pre>oModule.MovePlotCurvesToGroup _ "XY Stacked Plot 1", _ Array("R2.V : TR", "R2.I : TR"), _ "Stack 2"</pre>

MovePlotCurvesToNewGroup

Move curve(s) from its stack(s) to a new stack. Here term 'group' is synonymous to 'stack' in the context of Cartesian stacked plot.

UI Access	N/A		
Parameters	Name	Type	Description
	<ReportName>	String	Name of report.
	<CurveArray>	Array	Array of curve names to move.
Return Value	None.		

Python Syntax	MovePlotCurvesToNewGroup (<ReportName>, <CurveArray>)		
Python Example	<pre>oModule.MovePlotCurvesToNewGroup ("XY Stacked Plot 1", ["R2.V : TR", "R2.I : TR"])</pre>		

VB Syntax	MovePlotCurvesToNewGroup <ReportName>, <CurveArray>		
VB Example	<pre>oModule.MovePlotCurvesToNewGroup _ "XY Stacked Plot 1", _ Array("R2.V : TR", "R2.I : TR")</pre>		

OpenWindowForAllReports

Opens windows for all reports belong to current design.

UI Access	N/A
Parameters	None.
Return Value	None.

Python Syntax	OpenWindowForAllReports()
Python Example	<code>oModule.OpenWindowForAllReports()</code>

VB Syntax	OpenWindowForAllReports
VB Example	<code>oModule.OpenWindowForAllReports</code>

OpenWindowForReports

Opens windows for specified reports.

UI Access	N/A		
Parameters	Name	Type	Description
	<i><ReportNames></i>	Array	Array of strings containing report names.
Return Value	None.		

Python Syntax	OpenWindowForReports(<i><ReportNames></i>)

Python Example	<code>oModule.OpenWindowForReports(["Report1", "Report2"])</code>

VB Syntax	<code>OpenWindowForReports <ReportNames></code>
VB Example	<code>oModule.OpenWindowForReports Array("Report1", "Report2")</code>

PastePlotSettings

Paste plot settings to a specified report.

UI Access	Right-click a report, select Paste		
Parameters	Name	Type	Description
	<ReportName>	String	Name of specified report.
	<PropTypeToApply>	String	Property type to paste. "Graphical", "Data", or "All".
Return Value	None.		

Python Syntax	<code>PastePlotSettings(<ReportName>, <PropTypeToApply>)</code>
Python Example	<code>oModule.PastePlotSettings("Plot 1", "Graphical")</code>

VB Syntax	<code>PastePlotSettings <ReportName>, <PropTypeToApply></code>
VB Example	<code>oModule.PastePlotSettings "Plot 1", "Graphical"</code>

PasteReports

Paste copied reports to results in the current project.

UI Access	Edit > Paste
Parameters	None.
Return Value	None.

Python Syntax	PasteReports ()
Python Example	<code>oModule.PasteReports ()</code>

VB Syntax	PasteReports
VB Example	<code>oModule.PasteReports</code>

PasteReportsWithLegacyNames

Pastes copied reports to results in the current project with legacy name definitions.

UI Access	N/A
Parameters	None.
Return Value	None.

Python Syntax	PasteReportsWithLegacyNames ()
Python Example	<code>oModule.PasteReportsWithLegacyNames ()</code>

VB Syntax	PasteReportsWithLegacyNames
VB Example	<code>oModule.PasteReportsWithLegacyNames</code>

PasteTraces

Pastes copied traces to a named plot.

UI Access	Paste		
Parameters	Name	Type	Description
	<ReportName>	String	Name of plot
Return Value	None		

Python Syntax	PasteTraces (<ReportName>)
Python Example	<code>oModule.PasteTraces ("XY Plot1")</code>

VB Syntax	PasteTraces <ReportName>

VB Example	<code>oModule.PasteTraces "XY Plot1"</code>

PasteTracesWithLegacyNames

Pastes copied traces to a named plot using legacy name definitions.

UI Access	N/A		
Parameters	Name	Type	Description
	<i><ReportName></i>	String	Name of plot
Return Value	None		

Python Syntax	<code>PasteTracesWithLegacyNames (<ReportName>)</code>
Python Example	<code>oModule.PasteTracesWithLegacyNames ("XY Plot1")</code>

VB Syntax	<code>PasteTracesWithLegacyNames <ReportName></code>
VB Example	<code>oModule.PasteTracesWithLegacyNames "XY Plot1"</code>

RenameReport

Renames an existing report.

UI Access	Select a report on the Project tree, right-click and select Rename

Parameters	Name	Type	Description
	<OldReportName>	String	Old Report Name
	<NewReportName>	String	New Report Name
Return Value	None.		

Python Syntax	RenameReport (<OldReportName>, <NewReportName>)
Python Example	<code>oModule.RenameReport("XY Plot1", "Reflection")</code>

VB Syntax	RenameReport <OldReportName>, <NewReportName>
VB Example	<code>oModule.RenameReport "XY Plot1", "Reflection"</code>

RenameTrace

To rename a trace in a plot

UI Access	N/A		
Parameters	Name	Type	Description
	<ReportName>	String	Name of report.
	<TraceName>	String	Name of Trace
	<NewName>	String	New trace name.
Return Value	None.		

Python Syntax	<code>RenameTrace(<ReportName>, <TraceName>, <NewName>)</code>
Python Example	<pre>oModule.RenameTrace ("XY Plot1", "dB(S(WavePort1,WavePort1))1", "Port1dbS")</pre>

VB Syntax	<code>RenameTrace <ReportName>, <TraceName>, <NewName></code>
VB Example	<pre>oModule.RenameTrace "XY Plot1", _ "dB(S(WavePort1,WavePort1))1", _ "Port1dbS"</pre>

ResetPlotSettings

Resets plot settings to defaults.

UI Access	Right-click on a plot, select Edit > Reset Plot Settings		
Parameters	Name	Type	Description
	<PlotName>	String	Name of specified plot.
Return Value	None.		

Python Syntax	<code>ResetPlotSettings(<PlotName>)</code>
Python Example	<pre>oModule.ResetPlotSettings("Differential S-parameters")</pre>

VB Syntax	<code>ResetPlotSettings <PlotName></code>
VB Example	<code>oModule.ResetPlotSettings("Differential S-parameters")</code>

SavePlotSettingsAsDefault

Saves report plot settings as default.

UI Access	Report Templates > Save Settings as Default		
Parameters	Name	Type	Description
	<PlotName>	String	Name of plot to use for plot defaults.
Return Value	None.		

Python Syntax	<code>SavePlotSettingsAsDefault("<PlotName>")</code>
Python Example	<code>oModule.SavePlotSettingsAsDefault ("XY Plot1")</code>

VB Syntax	<code>SavePlotSettingsAsDefault "<PlotName>"</code>
VB Example	<code>oModule.SavePlotSettingsAsDefault "XY Plot 1"</code>

SetLinkOutputTraces

Specifies dynamic link output traces from the current design.

UI Access	Right-click the Results , select Link Output...		
Parameters	Name	Type	Description

	<code><TraceArray></code>	Array	Array of traces to set. <code>Array("<ReportName>:=", <array of trace names>, "<ReportName>:=", <array of trace names>,...)</code>
Return Value	None.		

Python Syntax	<code>SetLinkOutputTraces(<TraceArray>)</code>		
Python Example	<pre>oModule.SetLinkOutputTraces(["Plot 1:=", ["Trace1"], "Plot 2:=", ["Trace1"]])</pre>		

VB Syntax	<code>SetLinkOutputTraces <TraceArray></code>		
VB Example	<pre>oModule.SetLinkOutputTraces _ Array("Plot 1:=", Array("Trace1"), _ "Plot 2:=", Array("Trace1"))</pre>		

SetPropValue [Report Setup]

Sets the property value for report module child object.

UI Access	Select Edit Properties on Report objects.		
Parameters	Name	Type	Description
	<PropPath>	String	A child object's property path. See property path discussion here .
	<NewValue>	String, Number, or Boolean	New value data type is depending on the property type,
Return Value	True if the property is found and the new value is valid. Otherwise return False.		

Python Syntax	SetPropValue(<PropPath>, <NewValue>)		
Python Example	oRptModule.SetPropValue("S Parameter Plot 1/Display Type", "DataTable")		
	oRptModule.SetPropValue("S Parameter Plot 1/db(S(port1,port2)/Primary sweep", "Freq")		

VB Syntax	SetPropValue <PropPath>, <NewValue>		
VB Example	oRptModule.SetPropValue "S Parameter Plot 1/Display Type", "DataTable"		
	oRptModule.SetPropValue "S Parameter Plot 1/db(S(port1,port2)/Primary sweep", "Freq"		

UnGroupPlotCurvesInGroup

From a Stacked Plot, ungroups curves in a stack.

UI Access	N/A		
Parameters	Name	Type	Description

	<ReportName>	String	Name of report.
	<GroupName>	String	Stack group name.
Return Value	None.		

Python Syntax	UnGroupPlotCurvesInGroup(<ReportName>, <GroupName>)		
Python Example	oModule.UngroupPlotCurvesInGroup("S Parameter Plot 3", "Stack 1")		

VB Syntax	UnGroupPlotCurvesInGroup <ReportName>, <GroupName>		
VB Example	oModule.UngroupPlotCurvesInGroup "S Parameter Plot 3", "Stack 1"		

UpdateAllReports

Updates all reports in the **Results** branch in the project tree.

UI Access	Right-click on Results in the project tree, select Update All Reports
Parameters	None
Return Value	None

Python Syntax	UpdateAllReports()
Python Example	oModule.UpdateAllReports()

VB Syntax	UpdateAllReports
VB Example	<code>oModule.UpdateAllReports</code>

UpdateReports

Updates specified reports.

UI Access	N/A		
Parameters	Name	Type	Description
	<ReportNames>	Array	Array of strings containing report names.
Return Value	None.		

Python Syntax	UpdateReports(<ReportNames>)
Python Example	<code>oModule.UpdateReports (["XY Plot 1", "XY Plot 4"])</code>

VB Syntax	UpdateReports <ReportNames>
VB Example	<code>oModule.UpdateReports Array("XY Plot 1", "XY Plot 4")</code>

UpdateTraces

Update the traces in a report for which traces are not automatically updated by the Report Traces dialog box, Update Report, Real Time selection.

UI Access	In Report dialog, click Apply Traces button		
Parameters	Name	Type	Description
	<ReportName>	String	Name of Report.
	<TraceNames>	Array	Array of strings containing trace names.
	<SolutionName>	String	Name of the solution.
	<ContextArray>	Array	Context for which the expression is being evaluated. This can be an empty string if there is no context. <pre>Array("Domain:=", <DomainType>) <DomainType> ex. "Sweep" or "Time" Array("Context:=", <GeometryType>) <GeometryType> ex. "Infinite Spheren", "Spheren", "Polylinen"</pre>
	<FamiliesArray>	Array	Contains sweep definitions for the report. <pre>Array("<VariableName>:= ", <ValueArray>) <ValueArray> Array("All") or Array("Value1", "Value2", ... "Valuen") examples of <VariableName> "Freq", "Theta", "Distance"</pre>
	<ReportDataArray>	Array	This array contains the report quantity and X, Y, and (Z) axis definitions. <pre>Array("X Component:=", <VariableName>, "Y Component:=", <VariableName> <ReportQuantityArray>) <ReportQuantityArray> ex. Array("dB(S(Port1, Port1))")</pre>
	<ExtTraceInfo>	Array	Optional. Array defines extended trace information.

Return Value	None.

Python Syntax	UpdateTraces(<ReportName>, <SolutionName>, <ContextArray>, <FamiliesArray>, <ReportDataArray>)
Python Example	<pre>oModule.UpdateTraces("XY Plot 1", ["NEG1.VAL"], "TR4", ["NAME:Context", "SimValueContext:=", [2,0,2,0,False,False, -1,1,0,1,1,"",0,0,"CG",False,"0","KP",False,"0","MH",False, "100","TE",False,"100s","TH",False,"40", "TS",False,"0ns","UF",False, "0","WT",False,"0","WW",False,"100"]], ["Spectrum:=", ["All"]], ["X Component:=", "Spectrum", "Y Component:=", ["mag (NEG1.VAL) "]], [])</pre>

VB Syntax	UpdateTraces <ReportName>, <SolutionName>, <ContextArray>, <FamiliesArray>, <ReportDataArray>
VB Example	<pre>oModule.UpdateTraces "XY Plot1", Array("dB(S(WavePort1,WavePort1))"), _ "Setup1 : Sweep1", Array("Domain:=", "Time", "HoldTime:=", 1, _ "RiseTime:=", 0, "StepTime:=", 0, "Step:=", false, _ "WindowWidth:=", 1, "WindowType:=", 0, "KaiserParameter:=",1, _ "MaximumTime:=", 0), Array("Time:=", Array("All")), _ Array("X Component:=", "Time",_ "Y Component:=", Array("dB(S(WavePort1,WavePort1))")), _ Array()</pre>

UpdateTracesContextAndSweeps

Edits sweeps and context of multiple traces without affecting their component expressions.

UI Access	Modify Report with multiple traces selected.		
Parameters	Name	Type	Description
	<ReportName>	String	Name of Report.
	<TraceNames>	Array	Array of strings containing trace names.
	<SolutionName>	String	Name of the solution as listed in the Modify Report dialog box. For example: "Setup1 : Last Adaptive"
	<ContextArray>	Array	Context for which the expression is being evaluated. This can be an empty string if there is no context. ex. "Sweep" or "Time"

	<code><PointSet></code>	Array	Point set for the selected traces, for example, X and Y values for the plot.
Return Value	None		

Python Syntax	<code>UpdateTracesContextAndSweeps(<ReportName>, <TraceNames>, <SolutionName>, <ContextArray>, <PointSet>)</code>
Python Example	<pre>oModule.UpdateTracesContextAndSweeps_ ("Active S Parameter Quick Report", ["dB(ActiveS(Port1:1))", "dB(ActiveS(Port2:1))"], "Setup1 : Sweep1", [], ["Freq:=", ["9GHz", "9.05GHz", "9.1GHz", "9.15GHz", "9.2GHz", "9.25GHz", "9.3GHz", "9.35GHz", "9.4GHz", "9.45GHz", "9.5GHz", "9.55GHz", "9.6GHz", "9.65GHz", "9.7GHz", "9.9GHz", "9.95GHz", "10GHz"], "offset:=", ["All"]])</pre>

VB Syntax	<code>UpdateTracesContextAndSweeps <ReportName>, <TraceNames>, <SolutionName>, <ContextArray>, <PointSet></code>

VB Example	<pre>oModule.UpdateTracesContextAndSweeps _ "Active S Parameter Quick Report",_ Array("dB(ActiveS(Port1:1))", "dB(ActiveS(Port2:1))"),_ "Setup1 : Sweep1", Array(), _ Array("Freq=", _ Array("9GHz", "9.05GHz", "9.1GHz", _ "9.15GHz", "9.2GHz", _ "9.25GHz", "9.3GHz", "9.35GHz", _ "9.4GHz", "9.45GHz", "9.5GHz", _ "9.55GHz", _ "9.6GHz", "9.65GHz", "9.7GHz", _ "9.75GHz", "9.8GHz", "9.85GHz", _ "9.9GHz", "9.95GHz", "10GHz"),_ "offset=", Array("All"))</pre>

11 - 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](#)

General Commands Recognized by the Boundary/Excitations Module

[AddAssignmentToBoundary](#)

[DeleteAllBoundaries](#)

[DeleteAllExcitations](#)

[DeleteBoundaries](#)

[GetBoundaryAssignment](#)

[GetBoundaries](#)

[GetBoundariesOfType](#)

[GetExcitations](#)

[GetExcitationsOfType](#)

[GetHybridRegions](#)

[GetHybridRegionsOfType](#)

[GetNumBoundaries](#)

[GetNumBoundariesOfType](#)

[GetNumExcitations](#)

[GetNumExcitationsOfType](#)

[ReassignBoundary](#)

[RemoveAssignmentFromBoundary](#)

[RenameBoundary](#)

[ReprioritizeBoundaries](#)

AddAssignmentToBoundary

Adds a new geometry assignment to a boundary.

UI Access	N/A		
Parameters	Name	Type	Description
	<Assignment>	Array	Structured array. <pre>Array("Name:<BoundName>", "Objects:=", <AssignmentObjects>, "Faces:=", <AssignmentFaces>)</pre>
Return Value	None.		

Python Syntax	AddAssignmentToBoundary(<Assignment>)
Python Example	<pre>oModule.AddAssignmentToBoundary((["NAME:PerfE1", "Faces:=", [12]]))</pre>

VB Syntax	AddAssignmentToBoundary <Assignment>
VB Example	<pre>oModule.AddAssignmentTOBoundary Array("NAME:PerfE1", _ "Objects:=", Array("Box2", "Box3"), _ "Faces:=", Array(12, 11))</pre>

AutoidentifyLatticePair

Automatically identifies coupled lattice pair within specified object.

UI Access	Boundaries > Assign > Coupled... > Auto Identify Lattice Pair...		
Parameters	Name	Type	Description
	<PlaneName>	String	Name of relative plane.
	<ObjectName>	String	Name of specified object.
Return Value	None.		

Python Syntax	AutoidentifyLatticePair (<PlaneName>, <ObjectName>)
Python Example	<code>oModule.AutoIdentifyLatticePair("Global:XY", "Cylinder1")</code>

VB Syntax	AutoidentifyLatticePair <PlaneName>, <ObjectName>
VB Example	<code>oModule.AutoIdentifyLatticePair "Global:XY", "Cylinder1"</code>

AutoidentifyNets

Use: Automatically identifies nets.

Command: **Q3D Extractor>Nets>Auto Identify Nets**

Syntax: AutoidentifyNets

Return Value: None

Command: oModule.AutoIdentifyNets

AutoIdentifyPorts

Automatically identifies ports in a terminal design.

UI Access	N/A		
Parameters	Name	Type	Description
	<FaceIDArray>	Array	Array of face IDs. Array ("NAME:Faces", <FaceID>, <FaceID>, ...)
	<IsWavePort>	Boolean	• True - waveport • False - lumped port.
	<ReferenceConductorsArray>	Array	Structured array. Array ("NAME:ReferenceConductors", <ConductorName>, <ConductorName>, ...)
	<BaseName-forCreatedPorts>	String	Optional. Base name to use for created ports
	<UseConductorNames>	Boolean	Optional. • True - use conductor names. • False - use port object name as base name
Return Value	None.		

Python Syntax	AutoIdentifyPorts (<FaceIDArray>, <IsWavePort>, <ReferenceConductorsArray>)

Python Example	<pre>oModule.AutoIdentifyPorts(["NAME:Faces", 52], True, ["NAME:ReferenceConductors", "Conductor1"], True)</pre>

VB Syntax	<code>AutoIdentifyPorts <FaceIDArray>, <IsWavePort>, <ReferenceConductorsArray></code>
VB Example	<pre>oModule.AutoIdentifyPorts Array("NAME:Faces", 52), true, _ Array("NAME:ReferenceConductors", "Conductor1"), true</pre>

AutoIdentifyTerminals

Automatically identifies the terminals within the given ports.

UI Access	N/A		
Parameters	Name	Type	Description
	<ConductorsArray>	Array	Structured array. Array("NAME:ReferenceConductors", <ConductorName>, <ConductorName>, ...)
	<PortNames>	String	Names of given ports.
	<UseConductorNames>	Boolean	• True - use conductor names.

	<table><tr><td></td><td></td><td> • False - use port object name as base name. </td></tr></table>			• False - use port object name as base name.
		• False - use port object name as base name.		
Return Value	None.			

Python Syntax	<code>AutoIdentifyTerminals (<ConductorsArray>, <PortNames>, <UseConductorNames>)</code>
Python Example	<pre>oModule.AutoIdentifyTerminals(["NAME:ReferenceConductors", "Conductor1"], "WavePort1", True)</pre>

VB Syntax	<code>AutoIdentifyTerminals <ConductorsArray>, <PortNames>, <UseConductorNames></code>
VB Example	<pre>oModule.AutoIdentifyTerminals _ Array("NAME:ReferenceConductors", "Conductor1"), _ "WavePort1", true</pre>

ChangeImpedanceMult

Modifies the port impedance multiplier.

UI Access	HFSS > Excitations > Edit Port Impedance Multiplier.		
Parameters	Name	Type	Description
	<MultVal>	Double	New value for the impedance multiplier.
Return Value	None.		

Python Syntax	ChangeImpedanceMult (<MultVal>)
Python Example	oModule.ChangeImpedanceMult (0.5)

VB Syntax	ChangeImpedanceMult <MultVal>
VB Example	oModule.ChangeImpedanceMult 0.5

ConvertNportCircuitElementsToPorts

Converts one or more HFSS Circuit Element to one or more ports.

UI Access	Right-click on element and select Convert to HFSS Ports .		
Parameters	Name	Type	Description
	<NportName>	Array	Array of Nport names.
Return Value	None.		

Python Syntax	ConvertNportCircuitElementToPorts(<NportName>)
Python Example	oModule.ConvertNportCircuitElementToPorts (["Nport2"])

VB Syntax	ConvertNportCircuitElementToPorts <NportName>

VB Example	<code>oModule.ConvertNportCircuitElementToPorts Array("Nport2")</code>

CreateNportCircuitElements

Creates an HFSS Circuit Element from one or more ports.

UI Access	Right-click Circuit Elements > Create Single Port Model... or Circuit Elements > Create Multi-Terminal Model...		
Parameters	Name	Type	Description
	<NPortArray>	Array	Structured array. <code>Array("NAME:<NPortName>", "Definition:=", <string>, <AssignmentArray>)</code>
Return Value	None.		

Python Syntax	<code>CreateNportCircuitElement (<NPortArray>)</code>
Python Example	<pre>oModule.CreateNportCircuitElement(["NAME:Nport1", "Definition:=", "Model", ["NAME:Assignments", ["NAME:Model", "Assign:=", "1"]]</pre>


```
])  
  
oModule.CreateNportCircuitElement(  
  ["NAME:Nport2",  
    "Definition:=", "Model2",  
    ["NAME:Assignments",  
      ["NAME:P01__NET179__T1",  
        "Assign:=", "2"],  
      ["NAME:P02__NET178__T1",  
        "Assign:=", "3"],  
      ["NAME:P03__NET178__T1",  
        "Assign:=", "4"],  
      ["NAME:P04__NET179__T1",  
        "Assign:=", "5"],  
    ]  
  ]  
])
```

VB Syntax	CreateNportCircuitElement <NPortArray>
VB Example	<pre>oModule.CreateNportCircuitElement _ Array("NAME:Nport1", _</pre>

	<pre>"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>

DeleteAllBoundaries

Deletes all boundaries.

UI Access	[product] > Boundaries > Delete All
Parameters	None.
Return Value	None.

Python Syntax	DeleteAllBoundaries()
Python Example	oModule.DeleteAllBoundaries()

VB Syntax	DeleteAllBoundaries
VB Example	<code>oModule.DeleteAllBoundaries</code>

DeleteAllExcitations

Deletes all excitations.

UI Access	[product] > Excitations > Delete All
Parameters	None.
Return Value	None.

Python Syntax	DeleteAllExcitations()
Python Example	<code>oModule.DeleteAllExcitations()</code>

VB Syntax	DeleteAllExcitations
VB Example	<code>oModule.DeleteAllExcitations</code>

DeleteBoundaries

Deletes the specified boundaries and excitations.

UI Access	Delete command in the List dialog box. Click [product] > List to open the List dialog box.		
Parameters	Name	Type	Description

	<table><tr><td><NameArray></td><td>Array</td><td>Array of boundary condition names.</td></tr></table>	<NameArray>	Array	Array of boundary condition names.
<NameArray>	Array	Array of boundary condition names.		
Return Value	None.			

Python Syntax	DeleteBoundaries(<NameArray>)
Python Example	<code>oModule.DeleteBoundaries(["PerfE1", "WavePort1"])</code>

VB Syntax	DeleteBoundaries <NameArray>
VB Example	<code>oModule.DeleteBoundaries Array("PerfE1", "WavePort1")</code>

EditNportCircuitElement

Edits an HFSS Circuit Element.

UI Access	N/A		
Parameters	Name	Type	Description
	<NPortArray>	Array	Structured array. <code>Array("NAME:<NPortName>", "Definition:=", <string>, <AssignmentArray>)</code>
Return Value	None.		

Python Syntax	EditNportCircuitElement(<NPortArray>)
Python Example	<pre>oModule.EditNportCircuitElement(["NAME:Nport1", "Definition:=", "Model", ["NAME:Assignments", ["NAME:Model", "Assign:=", "1"]]])</pre>

VB Syntax	EditNportCircuitElement <NPortArray>
VB Example	<pre>oModule.EditNportCircuitElement _ Array("NAME:Nport1", "Definition:=", "Model", _ Array("NAME:Assignments", _ Array("NAME:Model", "Assign:=", "1")))</pre>

GetBoundaries

Gets boundary names in the current design.

UI Access	N/A
Parameters	None.
Return Value	Array of boundary names.

Python Syntax	GetBoundaries()
Python Example	<code>oModule.GetBoundaries()</code>

VB Syntax	GetBoundaries
VB Example	<code>oModule.GetBoundaries</code>

GetBoundariesOfType

Gets boundary names of the given type.

UI Access	N/A		
Parameters	Name	Type	Description
	<BoundaryType>	String	Name of boundary type.
Return Value	Array of boundary names of the given type.		

Python Syntax	GetBoundariesOfType(<BoundaryType>)
Python Example	<code>oModule.GetBoundariesOfType("PerfectE")</code>

VB Syntax	GetBoundariesOfType < <i>BoundaryType</i> >
VB Example	<code>oModule.GetBoundariesOfType "PerfectE"</code>

GetBoundaryAssignment

Gets a list of face IDs associated with the given boundary or excitation assignment.

UI Access	N/A		
Parameters	Name	Type	Description
	< <i>BoundaryName</i> >	String	Name of the specified boundary or excitation.
Return Value	Array of face IDs or object IDs.		

Python Syntax	GetBoundaryAssignment(< <i>BoundaryName</i> >)
Python Example	<code>oModule.GetBoundaryAssignment("Rad1")</code>

VB Syntax	GetBoundaryAssignment < <i>BoundaryName</i> >
VB Example	<code>oModule.GetBoundaryAssignment "Rad1"</code>

GetDefaultBaseName

Gets the default base name for boundaries for a project.

UI Access	N/A		
Parameters	Name	Type	Description
	< <i>BoundaryType</i> >	String	Name of legal boundary type.
Return Value	String of boundary default base name.		

Python Syntax	GetDefaultBaseName(< <i>BoundaryType</i> >)		
Python Example	oModule.GetDefaultBaseName("Radiation")		

VB Syntax	GetDefaultBaseName < <i>BoundaryType</i> >		
VB Example	oModule.GetDefaultBaseName "Radiation"		

GetDiffPairs

Gets the names of differential pairs defined.

UI Access	N/A		
Parameters	None.		
Return Value	Array of differential pair names.		

Python Syntax	GetDiffPairs()		
Python Example	oModule.GetDiffPairs()		

VB Syntax	GetDiffPairs
VB Example	<code>oModule.GetDiffPairs</code>

GetExcitations

Gets excitation port and terminal names for a model.

UI Access	N/A
Parameters	None.
Return Value	Array of excitation name paired with excitation type.

Python Syntax	GetExcitations()
Python Example	<code>oModule.GetExcitations()</code>

VB Syntax	GetExcitations
VB Example	<code>oModule.GetExcitations</code>

GetExcitationsOfType

Gets excitation names of the given type.

UI Access	N/A		
Parameters	Name	Type	Description
	<ExcitationType>	String	Name of excitation type.
Return Value	Array of excitation names of the given type.		

Python Syntax	GetExcitationsOfType(<ExcitationType>)		
Python Example	oModule.GetExcitationsOfType("Wave Port")		

VB Syntax	GetExcitationsOfType <ExcitationType>		
VB Example	oModule.GetExcitationsOfType "Wave Port"		

GetHybridRegions

Gets hybrid region names for a project.

UI Access	N/A		
Parameters	None.		
Return Value	Array of hybrid region names.		

Python Syntax	GetHybridRegions ()		
Python Example	oModule.GetHybridRegions ()		

VB Syntax	GetHybridRegions
VB Example	<code>oModule.GetHybridRegions</code>

GetHybridRegionsOfType

Gets hybrid region names of the given type.

UI Access	N/A		
Parameters	Name	Type	Description
	<HybridRegionType>	String	Name of legal hybrid region type.
Return Value	Array of hybrid region names of the given type.		

Python Syntax	GetHybridRegionsOfType(<HybridRegionType>)
Python Example	<code>oModule.GetHybridRegionsOfType("FE-BI")</code>

VB Syntax	GetHybridRegionsOfType <HybridRegionType>
VB Example	<code>oModule.GetHybridRegionsOfType "FE-BI"</code>

GetNumBoundaries

Gets the number of boundaries in a design.

UI Access	N/A
Parameters	None.
Return Value	Integer number of boundaries.

Python Syntax	GetNumBoundaries()
Python Example	<code>oModule.GetNumBoundaries()</code>

VB Syntax	GetNumBoundaries
VB Example	<code>oModule.GetNumBoundaries</code>

GetNumBoundariesOfType

Gets the number of boundaries of the given type.

UI Access	N/A		
Parameters	Name	Type	Description
	<BoundaryType>	String	Specified boundary type.
Return Value	Integer number of boundaries.		

Python Syntax	GetNumBoundariesOfType(<BoundaryType>)
Python Example	<code>oModule.GetNumBoundariesOfType("PerfectE")</code>

VB Syntax	GetNumBoundariesOfType < <i>BoundaryType</i> >
VB Example	<code>oModule.GetNumBoundariesOfType "PerfectE"</code>

GetNumExcitations

Gets the number of excitations in a design.

UI Access	N/A
Parameters	None.
Return Value	Integer number of excitations.

Python Syntax	GetNumExcitations()
Python Example	<code>oModule.GetNumExcitations()</code>

VB Syntax	GetNumExcitations
VB Example	<code>oModule.GetNumExcitations</code>

GetNumExcitationsOfType

Gets the number of excitations of the given type, including all defined modes and terminals of ports.

UI Access	N/A		
Parameters	Name	Type	Description
	<ExcitationType>	String	Specified type of excitation.
Return Value	Integer number of excitations.		

Python Syntax	GetNumExcitationsOfType(<ExcitationType>)		
Python Example	oModule.GetNumExcitationsOfType("Voltage")		

VB Syntax	GetNumExcitationsOfType <ExcitationType>		
VB Example	oModule.GetNumExcitationsOfType "Voltage"		

GetNumHybridRegions

Gets number of hybrid regions in a design.

UI Access	N/A		
Parameters	None.		
Return Value	Integer number of hybrid regions.		

Python Syntax	GetNumHybridRegions()		
Python Example	oModule.GetNumHybridRegions()		

VB Syntax	GetNumHybridRegions
VB Example	<code>oModule.GetNumHybridRegions</code>

GetNumHybridRegionsOfType

Gets number of hybrid regions of the given type.

UI Access	placeholder		
Parameters	Name	Type	Description
	<HybridRegionType>	String	Name of legal hybrid region type.
Return Value	Integer number of hybrid regions.		

Python Syntax	GetNumHybridRegionsOfType(<HybridRegionType>)
Python Example	<code>oModule.GetNumHybridRegionsOfType("FE-BI")</code>

VB Syntax	GetNumHybridRegionsOfType <HybridRegionType>
VB Example	<code>oModule.GetNumHybridRegionsOfType "FE-BI"</code>

GetPortExcitationCounts

Gets all port names and corresponding number of modes/terminals for each port excitation.

UI Access	N/A
Parameters	None.
Return Value	Array of port names and corresponding mode/terminal counts.

Python Syntax	GetPortExcitationCounts()
Python Example	<code>oModule.GetPortExcitationCounts()</code>

VB Syntax	GetPortExcitationCounts
VB Example	<code>oModule.GetPortExcitationCounts</code>

ReassignBoundary

Specifies a new geometry assignment for a boundary.

UI Access	Right-click Boundaries > Reassign or Excitations > Reassign		
Parameters	Name	Type	Description
	<AssignmentArray>	Array	Structured array. <pre>Array("Name:<BoundName>", "Objects:=", <AssignmentObjects>, "Faces:=", <AssignmentFaces>)</pre>
Return Value	None.		

Python Syntax	ReassignBoundary(<AssignmentArray>)
Python Example	<pre>oModule.ReassignBoundary(["NAME:PerfH12", "Faces:=", [17]])</pre>

VB Syntax	ReassignBoundary <AssignmentArray>
VB Example	<pre>oModule.ReassignBoundary _ Array("NAME:PerfH12", _ "Faces:=", Array(17))</pre>

RenameBoundary

Renames a boundary or excitation.

UI Access	Right-click a boundary in the project tree, and then click Rename on the shortcut menu.		
Parameters	Name	Type	Description
	<OldName>	String	Name of the boundary to be renamed.
	<NewName>	String	New name for the boundary.
Return Value	None.		

Python Syntax	<code>RenameBoundary(<OldName>, <NewName>)</code>
Python Example	<code>oModule.RenameBoundary("Rad2", "Rad3")</code>

VB Syntax	<code>RenameBoundary <OldName>, <NewName></code>
VB Example	<code>oModule.RenameBoundary "Rad2", "Rad3"</code>

ReprioritizeBoundaries

Specifies the order in which the boundaries and excitations are recognized by the solver. The first boundary in the list has the highest priority.

Note: this command is only valid if all defined boundaries and excitations appear in the list. All ports must be listed before any other boundary type.

UI Access	[product] > Boundaries > Reprioritize		
Parameters	Name	Type	Description
	<NewOrderArray>	Array	Structured array. <code>Array("NAME:NewOrder", <BoundName>, <BoundName>, ...)</code>
Return Value	None.		

Python Syntax	<code>ReprioritizeBoundaries(<NewOrderArray>)</code>
Python Example	<code>oModule.ReprioritizeBoundaries(["NAME:NewOrder",</code>

	<pre>"Imped1", "PerfE1", "PerfH1"])</pre>

VB Syntax	ReprioritizeBoundaries < <i>NewOrderArray</i> >
VB Example	<pre>oModule.ReprioritizeBoundaries Array("NAME:NewOrder", _ "Imped1", "PerfE1", "PerfH1")</pre>

SetDefaultBaseName

Sets the default base name for boundaries for a project.

UI Access	N/A		
Parameters	Name	Type	Description
	< <i>BoundaryType</i> >	String	Name of legal boundary type.
	< <i>DefaultName</i> >	String	Default name for boundaries of specified type.
Return Value	None.		

Python Syntax	SetDefaultBaseName(< <i>BoundaryType</i> >, < <i>DefaultName</i> >)
Python Example	<pre>oModule.SetDefaultBaseName("Radiation", "RadBnd")</pre>

VB Syntax	SetDefaultBaseName <BoundaryType>, <DefaultName>
VB Example	oModule.SetDefaultBaseName "Radiation", "RadBnd"

Script Commands for Creating and Modifying Boundaries

Following are script commands for creating and modifying boundaries that are recognized by the "BoundarySetup" module. In the following commands, all named data can be included or excluded as desired and may appear in any order.

[AssignCircuitPort \[HFSS\]](#)

[AssignCurrent](#)

[AssignDielectricCavity](#)

[AssignFiniteCond](#)

[AssignFloquet](#)

[AssignGradientSurfaceRoughness](#)

[AssignHalfSpace](#)

[AssignHybridRegion](#)

[AssignIERegion](#)

[AssignImpedance](#)

[AssignIncidentWave](#)

[AssignInteriorNearFieldSource](#)

[AssignLayeredImp](#)

[AssignLinkedRegion](#)

[AssignLumpedPort](#)

[AssignLumpedRLC](#)

[AssignMagneticBias](#)

[AssignRFDischargeDCBias](#)

AssignPrimary

[AssignPerfectE](#)

[AssignPerfectH](#)

AssignRadiation

AssignRadiation

[AssignScreeningImpedance](#)

[AssignSymmetry](#)

[AssignTerminal](#)

[AssignVoltage](#)

AssignWavePort

[AutoCreatePECCapForWavePort](#)

[CircuitPortToLumpedPort](#)

[EditCircuitPort \[HFSS\]](#)

EditCurrent

[EditDiffPairs](#)

[EditFiniteCond](#)

[EditGradientSurfaceRoughness](#)

[EditHalfSpace](#)

[EditHybridRegion](#)

[EditImpedance](#)

[EditIncidentWave](#)

[EditInteriorNearFieldSource](#)

[EditLayeredImpedance](#)

[EditPrimary](#)

[EditPerfectE](#)

[EditPerfectH](#)

[EditLumpedPort](#)

[EditLumpedRLC](#)

[EditMagneticBias](#)

[EditNPortCircuitElement](#)

[EditRadiation](#)

[EditRFDischargeDCBias](#)

[EditSymmetry](#)

[EditTerminal](#)

[EditVoltage](#)

[EditWavePort](#)

[LumpedPortToCircuitPort](#)

[SetHybridRegionCoupledGroup](#)

[SetSBRSources](#)

SetSBRSourcesBlockage

SetSBRWedgeSettings

[SetTerminalReferenceImpedances](#)

[SwapCircuitPortAssignment](#)

[UnassignIERegions](#)

AssignCircuitPort

Assigns a circuit port for a driven terminal or driven modal design in HFSS.

UI Access	HFSS > Excitations > Assign > Circuit Port...		
Parameters	Name	Type	Description
	<CircuitPortArray>	Array	Structured array. Array("NAME:<PortName>", "Edges:=", [n1, n2], "Impedance:=", "valueohm", "DoDeembed:=", <boolean> "RenormalizeAllTerminals:=", <boolean> "TerminalIDLit:=", Array())
Return Value	None.		

Python Syntax	AssignCircuitPort (<CircuitPortArray>)
Python Example	<pre>oModule.AssignCircuitPort(["NAME:1", "Edges:=", , [50,56], "Impedance:=", , "50ohm", "DoDeembed:=", , False, "RenormalizeAllTerminals:=", True, "TerminalIDList:=", , []])</pre>

VB Syntax	AssignCircuitPort <CircuitPortArray>
VB Example	<pre>oModule.AssignCircuitPort Array("NAME:1", "Edges:=", Array(50,56), "Impedance:=", "50ohm", "DoDeembed:=", false, "RenormalizeAllTerminals:=", true, "TerminalIDList:=", Array())</pre>

AssignCurrent

Creates a current source.

UI Access	HFSS > Excitations > Assign > Current		
Parameters	Name	Type	Description
	<CurrentArray>	Array	Structured array. Array ("NAME:<BoundName>", "Objects:=", <AssignmentObjects>, "Current:=", <value>, <DirectionArray>, "Faces:=", <AssignmentFaces>) "TerminalIDLit:=", Array())
	<DirectionArray>	Array	Structured array. Array ("NAME:Direction", "Start:=", <LineEndPoint>, "End:=", <LineEndPoint>)
Return Value	None.		

Python Syntax	AssignCurrent (<CurrentArray>)
Python Example	oModule.AssignCurrent (

	<pre>["NAME:Current1", "Current:=", "1000mA", ["NAME:Direction", "Start:=", [-0.4, 0.4, -1.6], "End:=", [-0.4, 0.4, 0]], "Faces:=", [12]])</pre>

VB Syntax	AssignCurrent <CurrentArray>
VB Example	<pre>oModule.AssignCurrent Array("NAME:Current1",_ "Current:=", "1000mA",_ Array("NAME:Direction",_ "Start:=", Array(-0.4, 0.4, -1.6),_ "End:=", Array(-0.4, 0.4, 0)), _ "Faces:=", Array(12))</pre>

AssignCylindricalWave

Creates an incident Cylindrical wave

UI Access	HFSS > Excitations > Assign > Incident Wave > Cylindrical Wave.		
Parameters	Name	Type	Description
	<Parameters>	Array	Structured array. Array("NAME:IncCWave1",

			<pre>"Objects:=" , <Array of objects>, "IsCartesian:=" , <boolean>, "EoX:=" , <string of integer value>, "EoY:=" , <string of integer value>, "EoZ:=" , <string of integer value>, "kX:=" , <string of integer value>, "kY:=" , <string of integer value>, "kZ:=" , <string of integer value>, "OriginX:=" , <string of integer value>, "OriginY:=" , <string of integer value>, "OriginZ:=" , <string of integer value>, "CylinderRadius:=" , <string of float value>)</pre>
Return Value	None.		

Python Syntax	AssignCylindricalWave (<Parameters>)
Python Example	<pre>oModule.AssignCylindricalWave(["NAME:IncCWave1", "Objects:=" , ["Cylinder1"], "IsCartesian:=" , True,</pre>

	<pre> "EoX:=" , "1", "EoY:=" , "0", "EoZ:=" , "0", "kX:=" , "0", "kY:=" , "0", "kZ:=" , "1", "OriginX:=" , "0mm", "OriginY:=" , "0mm", "OriginZ:=" , "0mm", "CylinderRadius:=" , "0.25in" l) </pre>

VB Syntax	AssignCylindricalWave <Parameters>
VB Example	<pre> oModule.AssignCylindricalWave Array("NAME:IncCWave1", "Objects:=", Array("Cylinder1"), "IsCartesian:=", true, "EoX:=", "1", "EoY:=", "0", "EoZ:=", "0", "kX:=", "0", </pre>

	<pre>"kY:=", "0", "kZ:=", "1", "OriginX:=", "0mm", "OriginY:=", "0mm", "OriginZ:=", "0mm", "CylinderRadius:=", "0.25in")</pre>

AssignDielectricCavity

Assigns a Hybrid Region as a Dielectric Cavity.

UI Access	HFSS > Hybrid > Assign Hybrid > Dielectric Cavity.		
Parameters	Name	Type	Description
	<Parameters>	Array	Structured array. Array("NAME:<cavity name>", "Objects:=", <array of objects>)
Return Value	None.		

Python Syntax	AssignDielectricCavity(<Parameters>)
Python Example	<pre>oModule.AssignDielectricCavity(["NAME:Cavity1",</pre>

	<code>"Objects:=", ["Cylinder1"]])</code>

VB Syntax	<code>AssignDielectricCavity <Parameters></code>
VB Example	<pre>oModule.AssignDielectricCavity Array("NAME:Cavity1", "Objects:=", Array("Cylinder1"))</pre>

AssignFEBI

Assigns a Hybrid Region as a FEBI.

UI Access	HFSS > Hybrid > Assign Hybrid > FE-BI....		
Parameters	Name	Type	Description
	<code><Parameters></code>	Array	Structured array. <pre>Array("NAME:<name of the FEBI Hybrid Region.>", "Objects:=", <array of names for the geometry assigned as FEBI Hybrid Region>)</pre>
Return Value	None.		

Python Syntax	<code>AssignFEBI(<Parameters>)</code>
Python Example	<pre>oModule.AssignFEBI(["NAME:FE-BI1", "Objects:=", ["RegularPolyhedron1"]])</pre>

VB Syntax	AssignFEBI <Parameters>
VB Example	<pre>oModule.AssignFEBI Array("NAME:FE-BI1", "Objects:=", Array("RegularPolyhedron1"))</pre>

AssignFiniteCond

Assigns a single finite conductivity boundary on selected edges.

UI Access	2D Extractor > Boundary > Assign > Finite Conductivity.		
Parameters	Name	Type	Description
	<Parameters>	Array	Structured array. Array("NAME:<name of the boundary.>", "Edges:=", <array of edge ids>, "Roughness:=", "<string of double with units of length>",&br/>"UseCoating:=", <boolean>, "LayerThickness:=", "<string of double with units of length>",&br/>"UseMaterial:=", <boolean>, "Material:=", "<string material name for coating>",&br/>"Radius:=", <string of double with units of length>,

	<table><tr><td></td><td></td><td>"Ratio:=", <string of double>)</td></tr></table>					"Ratio:=", <string of double>)
		"Ratio:=", <string of double>)				
Return Value	None.					

Python Syntax	AssignFiniteCond(<Parameters>)
Python Example	Example for the Hammerstad-Jensen surface roughness model. <pre>oModule.AssignFiniteCond ["NAME:FiniteCond1", "Edges:=", [7,9], "Roughness:=", "2um", "UseCoating:=", True, "LayerThickness:=", "1.2um", "UseMaterial:=", True, "Material:=", "Copper"]</pre> Example for the Hurray surface roughness model <pre>oModule.AssignFiniteCond ["NAME:FiniteCond1", "Edges:=", [7], "UseCoating:=", False, "Radius:=", "0.5um", "Ratio:=", "2.9"]</pre>

VB Syntax	AssignFiniteCond <Parameters>
VB Example	Example for the Hammerstad-Jensen surface roughness model. <pre>oModule.AssignFiniteCond Array("NAME:FiniteCond1", "Edges:=", Array(7,9), "Roughness:=", "2um", "UseCoating:=", true, "LayerThickness:=", "1.2um", "UseMaterial:=", true, "Material:=", "Copper")</pre> Example for the Hurray surface roughness model <pre>oModule.AssignFiniteCond Array("NAME:FiniteCond1", "Edges:=", Array(7), "UseCoating:=", false, "Radius:=", "0.5um", "Ratio:=", "2.9")</pre>

AssignFloquet

Creates a Floquet port.

UI Access	HFSS > Excitations > Assign > Floquet.		
Parameters	Name	Type	Description
	<FloquetPortArray>	Array	Structured array. Array("NAME:<BoundName>", "Faces:=", <array of face IDs>, "NumModes:=", <integer>, "RenormalizeAllTerminals:=", <boolean>, "DoDeembed:=", <boolean>, <ModesArray>, "ShowReporterFilter:=", <boolean>, "UseScanAngles:=", <boolean>, "Phi:=", "<numdeg>",&br/> "Theta:=", "<numdeg>",&br/> <LatticeAVector>, <LatticeBVector>, <ModesCalculator>, <ModesList>)
	<ModesArray>	Array	Structured array. Array("NAME:Modes", Array("NAME:<ModeName>",

		<pre>"ModeNum:=", <integer>, "UseIntLine:=", <boolean>, ...)</pre>
<LatticeAVector>	Array	Structured array. <pre>Array("NAME:LatticeAVector", "Start:=", Array("<num><units>", "<num><units>", "<num><units>"), "End:=", Array("<num><units>", "<num><units>", "<num><units>"))</pre>
<LatticeBVector>	Array	Structured array. <pre>Array("NAME:LatticeBVector", "Start:=", Array("<num><units>", "<num><units>", "<num><units>"), "End:=", Array("<num><units>", "<num><units>", "<num><units>"))</pre>
<ModesCalculator>	Array	Structured array. <pre>Array("NAME:ModesCalculator", "Frequency:=", "<Value>GHz", "FrequencyChanged:=", <Boolean>, "PhiStart:=", "<num>deg", "PhiStop:=", "<num>deg", "PhiStep:=", "<num>deg",</pre>

			<code>"ThetaStart:=", "<num>deg",</code> <code>"ThetaStop:=", "<num>deg",</code> <code>"ThetaStep:=", "<num>deg")</code>
	<code><ModesList></code>	Array	Structured array. <code>Array("NAME:ModesList",</code> <code> Array("NAME:Mode",</code> <code> "ModeNumber:=", <ModeID>,</code> <code> "IndexM:=", <integer index>,</code> <code> "IndexN:=", <integer index>,</code> <code> "KC2:=", <integer value>,</code> <code> "PropagationState:=", "Propagating",</code> <code> "Attenuation:=", <integer value>,</code> <code> "PolarizationState:=", <TE or TM>,</code> <code> "AffectsRefinement:=", <boolean>,</code> <code> ...)</code>
Return Value	None.		

Python Syntax	<code>AssignFloquetPort (<FloquetPortArray>)</code>
Python Example	<code>oModule.AssignFloquetPort(["NAME:FloquetPort1",</code> <code>"Faces:=", [7],</code> <code>"NumModes:=", 2,</code>

```
"RenormalizeAllTerminals:=", True,
"DoDeembed:=", False,
["NAME:Modes",
    ["NAME:Model1",
        "ModeNum:=", 1,
        "UseIntLine:=", False],
    ["NAME:Model2",
        "ModeNum:=", 2,
        "UseIntLine:=", False]],
"ShowReporterFilter:=", False,
"UseScanAngles:=", True, "Phi:=", "0deg", "Theta:=", "0deg",
["NAME:LatticeAVector",
    "Start:=", ["0mm", "0mm", "0.8mm"],
    "End:=", ["0mm", "0.6mm", "0.8mm"]],
["NAME:LatticeBVector",
    "Start:=", ["0mm", "0mm", "0.8mm"],
    "End:=", ["0.8mm", "0mm", "0.8mm"]],
["NAME:ModesCalculator",
    "Frequency:=", "1GHz",
    "FrequencyChanged:=", False,
```

```
"PhiStart:=", "0deg",  
"PhiStop:=", "0deg",  
"PhiStep:=", "0deg",  
"ThetaStart:=", "0deg",  
"ThetaStop:=", "0deg",  
"ThetaStep:=", "0deg"],  
["NAME:ModesList",  
  ["NAME:Mode",  
    "ModeNumber:=", 1,  
    "IndexM:=", 0,  
    "IndexN:=", 0,  
    "KC2:=", 0,  
    "PropagationState:=", "Propagating",  
    "Attenuation:=", 0,  
    "PolarizationState:=", "TE",  
    "AffectsRefinement:=", False],  
  ["NAME:Mode",  
    "ModeNumber:=", 2,  
    "IndexM:=", 0,  
    "IndexN:=", 0,  
    "KC2:=", 0,
```

	<pre>"PropagationState:=", "Propagating", "Attenuation:=", 0, "PolarizationState:=", "TM", "AffectsRefinement:=", False]]])</pre>

VB Syntax	AssignFloquetPort < <i>FloquetPortArray</i> >
VB Example	<pre>oModule.AssignFloquetPort Array("NAME:FloquetPort1", "Faces:=", Array(7), "NumModes:=", 2, "RenormalizeAllTerminals:=", true, "DoDeembed:=", false, Array("NAME:Modes", Array("NAME:Mode1", "ModeNum:=", 1, "UseIntLine:=", false), Array("NAME:Mode2", "ModeNum:=", 2, "UseIntLine:=", false)), "ShowReporterFilter:=", false, "UseScanAngles:=", true, "Phi:=", "0deg", "Theta:=", "0deg",</pre>

```

Array("NAME:LatticeAVector", "Start:=", Array("0mm",
"0mm", "0.8mm"), "End:=", Array("0mm", "0.6mm", "0.8mm")),

Array("NAME:LatticeBVector", "Start:=", Array("0mm",
"0mm", "0.8mm"), "End:=", Array("0.8mm", "0mm", "0.8mm")),

Array("NAME:ModesCalculator", "Frequency:=", "1GHz",
"FrequencyChanged:=", false,
"PhiStart:=", "0deg", "PhiStop:=", "0deg", "PhiStep:=",
"0deg", "ThetaStart:=", "0deg", "ThetaStop:=", "0deg", "ThetaStep:=",
"0deg"),

Array("NAME:ModesList", Array("NAME:Mode", "ModeNumber:=",
1, "IndexM:=", 0, "IndexN:=", 0, "KC2:=", 0,
"PropagationState:=", "Propagating",
"Attenuation:=", 0,
"PolarizationState:=", "TE",
"AffectsRefinement:=", false),

Array("NAME:Mode", "ModeNumber:=", 2,
"IndexM:=", 0, "IndexN:=", 0,
"KC2:=", 0,
"PropagationState:=", "Propagating",
"Attenuation:=", 0,
"PolarizationState:=", "TM", "AffectsRefinement:=",
false)))

```


AssignFresnel

Assigns a Fresnel boundary condition, allows specifying either a perfect absorber or to import a Fresnel table file that describes reflection and transmission coefficients.

UI Access	HFSS > Boundaries > Assign > Fresnel (SBR+)...		
Parameters	Name	Type	Description
	<ArgArray>	Array	Structured array. Array("NAME:<FresnelName>", "Faces:=", <array of face IDs>, "Fresnel Boundary Type:=", <PerfectAbsorber or ImportFromTableFile>, "RTTable Path:=", <string path to table file>)
Return Value	None.		

Python Syntax	AssignFresnel(<ArgArray>)
Python Example	<pre>oModule.AssignFresnel(["NAME:Fresnel2", "Faces:=", [8], "Fresnel Boundary Type:=", "ImportFromTableFile", "RTTable Path:=", "C:\\temp\\reflection-transmission-table.rttbl"])</pre>

VB Syntax	AssignFresnel <ArgArray>
VB Example	<pre>oModule.AssignFresnel Array("NAME:Fresnel2", "Faces:=", Array(8), "Fresnel Boundary Type:=", "ImportFromTableFile", "RTTable Path:=", "C:\\temp\\reflection-transmission-table.rttbl")</pre>

AssignGaussianBeam

Assigns a Gaussian Beam type of incident wave excitation.

UI Access	HFSS > Excitations > Assign > Incident Wave > Gaussian Beam...		
Parameters	Name	Type	Description
	<IncidentWaveArray>	Array	Structured array. <pre>Array("NAME:<Name of incident wave>", "Faces:=", <array of face IDs>, "IsCartesian:=", <boolean>, "EoX:=" , "<numeric value>", "EoY:=" , "<numeric value>", "EoZ:=" , "<numeric value>", "kX:=" , "<numeric value>", "kY:=" , "<numeric value>", "kZ:=" , "<numeric value>", "PhiStart:=", <value>,</pre>

			<pre>"PhiStop:=", <value>, "PhiPoints:=", <int>, "ThetaStart:=", <value>, "ThetaStop:=", <value>, "ThetaPoints:=", <int>, "EoPhi:=", <value>, "EoTheta:=", <value>, "OriginX:=" , "<numUnit>", "OriginY:=" , "<numUnit>", "OriginZ:=" , "<numUnit>", "BeamWidth:=" , "<numUnit>") IsCartesian If true, provide the EoX, EoY, EoZ, kX, kY, kZ parameters. If false, provide the PhiStart, PhiStop, PhiPoints, ThetaStart, ThetaStop, ThetaPoints, EoPhi, EoTheta parameters.</pre>
Return Value	None.		

Python Syntax	AssignGaussianBeam(<IncidentWaveArray>)
Python Example	oModule.AssignGaussianBeam(

```

["NAME:IncGBeam1",
  "Faces:="                , [9],
  "IsCartesian:="          , True,
  "EoX:="                  , "1",
  "EoY:="                  , "0",
  "EoZ:="                  , "0",
  "kX:="                   , "0",
  "kY:="                   , "0",
  "kZ:="                   , "1",
  "OriginX:="              , "0mm",
  "OriginY:="              , "0mm",
  "OriginZ:="              , "0mm",
  "BeamWidth:="            , "10mm"
])
oModule.AssignGaussianBeam(
["NAME:IncGBeam2",
  "Faces:="                , [9],
  "IsCartesian:="          , False,
  "PhiStart:="             , "0deg",
  "PhiStop:="              , "0deg",
  "PhiPoints:="            , 1,

```

	<pre> "ThetaStart:=" , "0deg", "ThetaStop:=" , "0deg", "ThetaPoints:=" , 1, "EoPhi:=" , "1", "EoTheta:=" , "0", "OriginX:=" , "0mm", "OriginY:=" , "0mm", "OriginZ:=" , "0mm", "BeamWidth:=" , "10mm" l) </pre>

VB Syntax	AssignGaussianBeam <IncidentWaveArray>
VB Example	<pre> oModule.AssignGaussianBeam Array("NAME:IncGBeam1", "Faces:=", Array(9), "IsCartesian:=", true, "EoX:=" , "1", "EoY:=", "0", "EoZ:=", "0", "kX:=", "0", "kY:=", "0", "kZ:=", "1", "OriginX:=", "0mm", "OriginY:=", "0mm", "OriginZ:=", "0mm", "BeamWidth:=", "10mm") </pre>

	<pre>oModule.AssignGaussianBeam Array("NAME:IncGBeam1", "Faces:=", Array(9), "IsCartesian:=", false, "PhiStart:=", "0deg", "PhiStop:=", "90deg", "PhiPoints:=", 2, "ThetaStart:=", "0deg", "ThetaStop:=", "180deg", "ThetaPoints:=", 3, "EoPhi:=", "1", "EoTheta:=", "0", "OriginX:=", "0mm", "OriginY:=", "0mm", "OriginZ:=", "0mm", "BeamWidth:=", "10mm")</pre>

AssignGradientSurfaceRoughness

Assigns a Gradient Surface Roughness boundary.

UI Access	HFSS > Boundaries > Assign > Gradient Surface Roughness...		
Parameters	Name	Type	Description
	<ArgArray>	Array	Structured array. Array(NAME:<Name of GradientSurfaceBoundary>"," ["NAME:<Name of GradientSurfaceBoundary>"," "Objects:=" , [<geometryID>"], "RMS Roughness:=" , "<value><units>"," "Bulk Conductivity:=" , "<value>"," "Thickness:=" , ""<value><units>"," "Max Frequency:=" , ""<value><units>","

	<table><tr><td></td><td></td><td>"Surface Type:=" , "[Low Profile High Profile]"</td></tr></table>			"Surface Type:=" , "[Low Profile High Profile]"
		"Surface Type:=" , "[Low Profile High Profile]"		
Return Value	None.			

Python Syntax	AssignGradientSurfaceRoughness(<ArgArray>)
Python Example	<pre>oModule = oDesign.GetModule("BoundarySetup") oModule.AssignGradientSurfaceRoughness (["NAME:GradientImped1", "Objects:=" , ["Box1"], "RMS Roughness:=" , "1um", "Bulk Conductivity:=" , "580000000", "Thickness:=" , "15um", "Max Frequency:=" , "100GHz", "Surface Type:=" , "Low Profile"])</pre>

VB Syntax	AssignGradientSurfaceRoughness <ArgArray>
VB Example	<pre>oModule = oDesign.GetModule("BoundarySetup")</pre>

	<pre>oModule.AssignGradientSurfaceRoughness("NAME:GradientImped1", _ "Objects:=" , ["Box1"], "RMS Roughness:=" , "1um", _ "Bulk Conductivity:=" , "58000000", "Thickness:=" , "15um", _ "Max Frequency:=" , "100GHz", "Surface Type:=" , "Low Profile")</pre>

AssignHalfSpace

Assigns a Half Space boundary, dividing the background material at a specified Z axis point. You also assign a material, typically to the lower half.

UI Access	HFSS > Boundaries > Assign > Half Space...		
Parameters	Name	Type	Description
	<ArgArray>	Array	Structured array. <pre>Array("NAME:<Name of Half Space>", "ZLocation:=", "<intUnits>", "Material:=", "<string material name>")</pre>
Return Value	None.		

Python Syntax	AssignHalfSpace(<ArgArray>)
Python Example	<pre>oModule.AssignHalfSpace(["NAME:HalfSpace1", "ZLocation:=", "2mm", "Material:=", "water_sea"])</pre>

VB Syntax	AssignHalfSpace <ArgArray>

VB Example	<pre>oModule.AssignHalfSpace Array("NAME:HalfSpace1", "ZLocation:=", "2mm", "Material:=", "water_sea")</pre>

AssignHertzianDipoleWave

Assigns an incident Hertzian-Dipole wave as excitation.

UI Access	HFSS > Excitations > Assign > Incident Wave > Hertzian-Dipole Wave...		
Parameters	Name	Type	Description
	<ArgArray>	Array	Structured array. <pre>Array("NAME:<Name of incident wave>", "Faces:=", <array of face IDs>, "IsCartesian:=", <boolean>, "EoX:=" , "<numeric value>", "EoY:=" , "<numeric value>", "EoZ:=" , "<numeric value>", "kX:=" , "<numeric value>", "kY:=" , "<numeric value>", "kZ:=" , "<numeric value>", "OriginX:=" , "<numUnit>", "OriginY:=" , "<numUnit>", "OriginZ:=" , "<numUnit>", "SphereRadius:=" , "<numUnit>")</pre>

	<table><tr><td></td><td></td><td>"IsElectricDipole:=" , <boolean>)</td></tr></table>					"IsElectricDipole:=" , <boolean>)
		"IsElectricDipole:=" , <boolean>)				
Return Value	None.					

Python Syntax	AssignHertzianDipoleWave(<ArgArray>)
Python Example	<pre>oModule.AssignHertzianDipoleWave(["NAME:IncHDWave2", "Faces:=" , [9], "IsCartesian:=" , True, "EoX:=" , "0", "EoY:=" , "0", "EoZ:=" , "1", "kX:=" , "0", "kY:=" , "0", "kZ:=" , "1", "OriginX:=" , "0mm", "OriginY:=" , "0mm", "OriginZ:=" , "0mm", "SphereRadius:=" , "10mm", "IsElectricDipole:=" , True</pre>

	l)

VB Syntax	AssignHertzianDipoleWave<ArgArray>
VB Example	<pre>oModule.AssignHertzianDipoleWave Array("NAME:IncHDWave2", "Faces:=", Array(9), "IsCartesian:=", true, "EoX:=" , "0", "EoY:=", "0", "EoZ:=", "1", "kX:=", "0", "kY:=", "0", "kZ:=", "1", "OriginX:=", "0mm", "OriginY:=", "0mm", "OriginZ:=", "0mm", "SphereRadius:=", "10mm", "IsElectricDipole:=", true)</pre>

AssignHybridRegion

Assigns a Hybrid Region to a conductor, one of IE, PO, or SBR.

UI Access	HFSS > Hybrid > Assign Hybrid > IE Region... or PO Region... or SBR+ Region....		
Parameters	Name	Type	Description
	<ArgArray>	Array	Structured array. Array("NAME:<name of hybrid region>", "Objects:=", <array of names for the geometry assigned as Hybrid Region>,

			<pre> "Type:=" , <string one of "IE", "PO" or "SBR">, "IsLinkedRegion:=" , <boolean>, "ConductivityThreshold:=", "<numUnit>") </pre>
Return Value	None.		

Python Syntax	AssignHybridRegion(<ArgArray>)		
Python Example	<pre> oModule.AssignHybridRegion(["NAME:Hybrid1", "Objects:=" , ["Cylinder1"], "Type:=" , "IE", "IsLinkedRegion:=" , False, "ConductivityThreshold:=", "20000S_per_m"]) </pre>		

VB Syntax	AssignHybridRegion <ArgArray>		
VB Example	<pre> oModule.AssignHybridRegion Array("NAME:Hybrid1", "Objects:=", Array("RegularPolyhedron1"), "Type:=", "PO", "IsLinkedRegion:=", false) </pre>		

AssignIERegion

Assigns an IE Region to a conductor contained within a FEBI Radiation boundary.

UI Access	N/A		
Parameters	Name	Type	Description
	<GeometryName>	String	Name of the geometry to be assigned as an IE Region.
Return Value	None.		

Python Syntax	AssignIERegion (<GeometryName>)
Python Example	<code>oModule.AssignIERegion("Box1")</code>

VB Syntax	AssignIERegion <GeometryName>
VB Example	<code>oModule.AssignIERegion "Box1"</code>

AssignInteriorNearFieldSource

Assigns an interior near field source as an external data file and associated coordinate system.

UI Access	HFSS > Boundaries > Assign > Linked Field>Near Field...		
Parameters	Name	Type	Description
	<ArgArray>	Array	Structured array. ("NAME:<Name of Near Field Source data>",

			"Type:=" , "Incident", "ExternalDataFile:=" , "<filePath>.and", "SourceCoordSystem:=" , "<CS_name>"
Return Value	None.		

Python Syntax	AssignInteriorNearFieldSource(<ArgArray>)
Python Example	<pre> oDesign = oProject.SetActiveDesign("internal") oModule = oDesign.GetModule("BoundarySetup") oModule.AssignInteriorNearFieldSource(["NAME:ExtNFDData2", "Type:=" , "Incident", "ExternalDataFile:=" , "D:\\Ansoft\\Sphere_EH.and", "SourceCoordSystem:=" , "Global"]) </pre>

AssignImpedance

Creates an impedance boundary for an HFSS design.

UI Access	HFSS > Boundaries > Assign > Impedance...

	Name	Type	Description
	<i><ImpedanceArray></i>	Array	Structured array. Array ("NAME:<BoundName>", "Resistance:=", <value>, "Reactance:=", <value>, "InfGroundPlane:=", <boolean>, "Objects:=", <AssignmentObjects>, "Faces:=", <AssignmentFaces>))
Parameters			
Return Value	None.		

Python Syntax	AssignImpedance(<i><ImpedanceArray></i>)
Python Example	<pre>oModule.AssignImpedance (["NAME:Imped1", "Resistance:=", "50", "Reactance:=", "50", "InfGroundPlane:=", False, "Faces:=", [12]])</pre>

VB Syntax	AssignImpedance <ImpedanceArray>
VB Example	<pre>oModule.AssignImpedance Array("NAME:Imped1",_ "Resistance:=", "50",_ "Reactance:=", "50",_ "InfGroundPlane:=", false,_ "Faces:=", Array(12))</pre>

AssignIncidentWave

Creates an incident wave excitation.

UI Access	N/A		
Parameters	Name	Type	Description
	<IncidentWaveArray>	Array	Structured array. <pre>Array("NAME:<BoundName>", "Faces:=", <array of face IDs>, "IsCartesian:=", <boolean>, "EoX:=", <value>, "EoY:=", <value>, "EoZ:=", <value>, "kX:=", <value>, "kY:=", <value>, "kZ:=", <value> "PhiStart:=", <value>,</pre>

			<pre>"PhiStop:=", <value>, "PhiPoints:=", <int>, "ThetaStart:=", <value>, "ThetaStop:=", <value>, "ThetaPoints:=", <int>, "EoPhi:=", <value>, "EoTheta:=", <value>, "IsPropagating:=", <boolean>, "IsEvanescent:=", <boolean>, "IsEllipticallyPolarized:=", <boolean>) IsCartesian</pre> If true, provide the EoX, EoY, EoZ, kX, kY, kZ parameters. If false, provide the PhiStart, PhiStop, PhiPoints, ThetaStart, ThetaStop, ThetaPoints, EoPhi, EoTheta parameters.
Return Value	None.		

Python Syntax	AssignIncidentWave (<IncidentWaveArray>)
Python Example	<pre>oModule.AssignIncidentWave(["NAME:IncWave1", "Faces:=", [9], "IsCartesian:=", True,</pre>

```

"EoX:=", "1", "EoY:=", "0", "EoZ:=", "0",
"kX:=", "0", "kY:=", "0", "kZ:=", "1",
"IsPropagating:=", True,
"IsEvanescent:=", False,
"IsEllipticallyPolarized:=", False]
)
oModule.AssignIncidentWave(["NAME:IncWave2",
"Faces:=", [9],
"IsCartesian:=", False,
"PhiStart:=", "0deg",
"PhiStop:=", "90deg",
"PhiPoints:=", 2,
"ThetaStart:=", "0deg",
"ThetaStop:=", "180deg",
"ThetaPoints:=", 3,
"EoPhi:=", "1", "EoTheta:=", "0",
"IsPropagating:=", True,
"IsEvanescent:=", False,
"IsEllipticallyPolarized:=", False
])

```

VB Syntax	AssignIncidentWave <IncidentWaveArray>
VB Example	<pre> oModule.AssignIncidentWave Array("NAME:IncWave1",_ "Faces:=", Array(9),_ "IsCartesian:=", true,_ "EoX:=", "1", "EoY:=", "0", "EoZ:=", "0",_ "kX:=", "0", "kY:=", "0", "kZ:=", "1", "IsPropagating:=", true, _ "IsEvanescent:=", false, _ "IsEllipticallyPolarized:=", false) oModule.AssignIncidentWave Array("NAME:IncWave2",_ "Faces:=", Array(9),_ "IsCartesian:=", false,_ "PhiStart:=", "0deg",_ "PhiStop:=", "90deg",_ "PhiPoints:=", 2,_ "ThetaStart:=", "0deg",_ "ThetaStop:=", "180deg",_ "ThetaPoints:=", 3, _ "EoPhi:=", "1", "EoTheta:=", "0", _ "IsPropagating:=", true, _ </pre>

	<pre>"IsEvanescent:=", false, _ "IsEllipticallyPolarized:=", false)</pre>

AssignLatticePair

Assigns coupled Lattice Pair boundaries.

UI Access	HFSS > Boundaries > Assign > Coupled... > Lattice Pair...		
Parameters	Name	Type	Description
	<BoundaryArray>	Array	Structured array. Array ("NAME:<BoundName>", "Faces:=", <Array of face IDs>, "ReverseV:=", <boolean>, "PhaseDelay:=", <UseScanAngle InputPhaseDelay>, "Phi:=", <numdeg>, "Theta:=", <numdeg>, "Phase:=", <numdeg>)
Return Value	None.		

Python Syntax	AssignLatticePair(<BoundaryArray>)
Python Example	<pre>oModule.AssignLatticePair(["NAME:LatticePair1", "Faces:=" , [9,8],</pre>

	<pre> "ReverseV:=" , False, "PhaseDelay:=" , "UseScanAngle", "Phi:=" , "0deg", "Theta:=" , "0deg"]) oModule.AssignLatticePair(["NAME:LatticePair2", "Faces:=" , [8,9], "ReverseV:=" , False, "PhaseDelay:=" , "InputPhaseDelay", "Phase:=" , "10deg"]) </pre>

VB Syntax	AssignLatticePair <BoundaryArray>
VB Example	<pre> oModule.AssignLatticePair Array("NAME:LatticePair1", "Faces:=", Array(9,8), "ReverseV:=", false, "PhaseDelay:=", "UseScanAngle", "Phi:=", "0deg", </pre>

	<pre>"Theta:=", "0deg") oModule.AssignLatticePair Array("NAME:LatticePair2", "Faces:=", Array(8,9), "ReverseV:=", false, "PhaseDelay:=", "InputPhaseDelay", "Phase:=", "10deg")</pre>

AssignLayeredImp

Creates a layered impedance boundary.

UI Access	HFSS > Boundaries > Assign > Layered Impedance...		
Parameters	Name	Type	Description
	<LayeredImpArray>	Array	Structured array. Array ("NAME:<BoundName>", "Frequency:=", <value>, "Roughness:=", <value>, "IsInternal:=", <bool>, "IsTwoSided:=", <bool>, "IsShellElement:=", <bool>, <LayersArray>, "Objects:=", <AssignmentObjects>, "Faces:=", <AssignmentFaces>, "InfGroundPlane:=", <boolean>)

	<LayersArray>	Array	Structured array. <pre>Array("NAME:Layers", <OneLayerArray>, <OneLayerArray>, ...)</pre>
	<OneLayerArray>	Array	Structured array. <pre>Array("NAME:<LayerName>", "LayerType:=", <LayerType>, "Thickness:=", <value>, "Material:=", <string>)</pre> Thickness Thickness of the layer. Should be specified for all layers except the last layer. Material Material assigned on the layer. For the last layer, do not specify a material if the LayerType is "PerfectE" or "PerfectH".
	<LayerName>	String	Specifies the layer number, such as "Layer1" or "Layer2"
	<LayerType>	String	Should be specified for the last layer only. Possible values: "Infinite", "PerfectE", or "PerfectH"
	Return Value	None.	

Python Syntax	AssignLayeredImp(<LayeredImpArray>)
Python Example	oModule.AssignLayeredImp(

	<pre>["NAME:Layered1", "Objects:=", ["Rectangle1"], "Frequency:=", "0GHz", "Roughness:=", "0um", "IsTwoSided:=", True, "IsShellElement:=", True, ["NAME:Layers", ["NAME:Layer1", "Thickness:=", "1um", "Material:=", "vacuum"]], "InfGroundPlane:=", False]]</pre>

VB Syntax	AssignLayeredImp <LayeredImpArray>
VB Example	<pre>oModule.AssignLayeredImp Array("NAME:Layered1", "Objects:=", Array("Rectangle1"), "Frequency:=", "0GHz", "Roughness:=", "0um", "IsTwoSided:=", true, "IsShellElement:=", true, Array("NAME:Layers", Array("NAME:Layer1", "Thickness:=", "1um", "Material:=", "vacuum"))), "InfGroundPlane:=", false)</pre>

AssignLinearAntennaWave

Creates an incident linear antenna wave excitation.

UI Access	HFSS > Excitations > Assign > Incident Wave > Linear Antenna Wave...		
Parameters	Name	Type	Description
	<IncidentWaveArray>	Array	Structured array. Array("NAME:<Name of incident wave>", "Faces:=", <array of face IDs>, "IsCartesian:=", <boolean>, "EoX:=" , "<numeric value>",&br/>"EoY:=" , "<numeric value>",&br/>"EoZ:=" , "<numeric value>",&br/>"kX:=" , "<numeric value>",&br/>"kY:=" , "<numeric value>",&br/>"kZ:=" , "<numeric value>",&br/>"PhiStart:=",<value>, "PhiStop:=", <value>, "PhiPoints:=", <int>, "ThetaStart:=", <value>, "ThetaStop:=", <value>, "ThetaPoints:=", <int>,

			<pre>"EoPhi:=", <value>, "EoTheta:=", <value>, "OriginX:=", "<numUnit>", "OriginY:=", "<numUnit>", "OriginZ:=", "<numUnit>", "AntennaRadius:=", "<numUnit>", "AntennaLength:=", "<numUnit>") IsCartesian If true, provide the EoX, EoY, EoZ, kX, kY, kZ parameters. If false, provide the PhiStart, PhiStop, PhiPoints, ThetaStart, ThetaStop, ThetaPoints, EoPhi, EoTheta parameters.</pre>
Return Value	None.		

Python Syntax	AssignLinearAntennaWave(<IncidentWaveArray>)		
Python Example	<pre>oModule.AssignLinearAntennaWave (["NAME:InclWave1", "Faces:=", , [9], "IsCartesian:=", , True, "EoX:=", , "1", "EoY:=", , "0", "EoZ:=", , "0",</pre>		

```

"kX:="          , "0",
"kY:="          , "0",
"kZ:="          , "1",
"OriginX:="     , "0mm",
"OriginY:="     , "0mm",
"OriginZ:="     , "0mm",
"AntennaRadius:=" , "10mm",
"AntennaLength:=" , "10mm"
])
oModule.AssignLinearAntennaWave(
["NAME:InclWave2",
"Faces:="          , [9],
"IsCartesian:="    , False,
"PhiStart:="       , "0deg",
"PhiStop:="        , "0deg",
"PhiPoints:="      , 1,
"ThetaStart:="     , "0deg",
"ThetaStop:="      , "0deg",
"ThetaPoints:="    , 1,
"EoPhi:="          , "1",

```

	<pre> "EoTheta:=" , "0", "OriginX:=" , "0mm", "OriginY:=" , "0mm", "OriginZ:=" , "0mm", "AntennaRadius:=" , "10mm", "AntennaLength:=" , "10mm" l) </pre>

VB Syntax	AssignLinearAntennaWave <IncidentWaveArray>
VB Example	<pre> oModule.AssignLinearAntennaWave Array("NAME:IncLWave1", "Faces:=", Array(9), "IsCartesian:=", true, "EoX:=" , "1", "EoY:=", "0", "EoZ:=", "0", "kX:=", "0", "kY:=", "0", "kZ:=", "1", "OriginX:=", "0mm", "OriginY:=", "0mm", "OriginZ:=", "0mm", "AntennaRadius:=", "10mm", "AntennaLength:=", "10mm") oModule.AssignLinearAntennaWave Array("NAME:IncLWave2", "Faces:=", Array(9), "IsCartesian:=", false, "PhiStart:=", "0deg", "PhiStop:=", "90deg", "PhiPoints:=", 2, </pre>

	<pre> "ThetaStart:=", "0deg", "ThetaStop:=", "180deg", "ThetaPoints:=", 3, "EoPhi:=", "1", "EoTheta:=", "0", "OriginX:=", "0mm", "OriginY:=", "0mm", "OriginZ:=", "0mm", "AntennaRadius:=", "10mm", "AntennaLength:=", "10mm") </pre>

AssignLinkedImpedance

Assigns a linked Impedance boundary.

UI Access	HFSS > Boundaries > Assign > Linked Impedance...		
Parameters	Name	Type	Description
	<LinkedImpArray>	Array	Structured array. <pre> Array("NAME:<BoundName>", "Faces:=", <array of face IDs>, "UseInfiniteGroundPlane:=", <boolean>, "UseShellElement:=", <boolean>, <LinkDataArray>) </pre>
	<LinkDataArray>	Array	Structured array. <pre> Array("NAME:<LinkName>", "Project:=", <string file path>, "Product:=", <string product type>, "Design:=", <string linked source design name>, </pre>

			"Soln:=", <string linked source solution name>, <array solution parameters>, "ForceSourceToSolve:=", <boolean>, "PreservePartnerSoln:=" , <boolean>, "PathRelativeTo:=" , <string target project name>)
Return Value	None.		

Python Syntax	AssignLinkedImpedance(<LinkedImpArray>)
Python Example	<pre> oModule.AssignLinkedImpedance (["NAME:Linked1", "Faces:=", [9], "UseInfiniteGroundPlane:=", True, "UseShellElement:=", True, ["NAME:XLink", "Project:=", "C://temp/linkedproject.aedt", "Product:=", "HFSS", "Design:=", "Source_Project_Solver", "Soln:=", "1000MHz : LastAdaptive", ["NAME:Params", "xfactor:=", "1.2", "yfactor:=", "1.6"], "ForceSourceToSolve:=", True, "PreservePartnerSoln:=", False, </pre>

	<pre>"PathRelativeTo:=", "TargetProject"]])</pre>

VB Syntax	AssignLinkedImpedance <LinkedImpArray>
VB Example	<pre>oModule.AssignLinkedImpedance Array("NAME:Linked1", "Faces:=", Array(9), "UseInfiniteGroundPlane:=", true, "UseShellElement:=", true, Array("NAME:XLink", "Project:=", "C://temp/linkedproject.aedt", "Product:=", "HFSS", "Design:=", "Source_Project_Solver", "Soln:=", "1000MHz : LastAdaptive", Array("NAME:Params", "xfactor:=", "1.2", "yfactor:=", "1.6"), "ForceSourceToSolve:=", true, "PreservePartnerSoln:=", false, "PathRelativeTo:=", "TargetProject"))</pre>

AssignLinkedRegion

Assigns a Hybrid Region as a Linked Region.

UI Access	N/A		
Parameters	Name	Type	Description
	<LinkedRegionArray>	Array	Structured array. <pre>Array("NAME:<name of linked region>", "Objects:=", <array, names for the objects assigned as Hybrid Region>, "Type:=" , <string, one of "IE", "PO" or "SBR">, "IsLinkedRegion:=" , <boolean, true for linked region>)</pre>
Return Value	None.		

Python Syntax	AssignLinkedRegion (<LinkedRegionArray>)
Python Example	<pre>oModule.AssignLinkedRegion (["NAME:Linked1", "Objects:=", ["RegularPolyhedron1"], "Type:=", "PO", "IsLinkedRegion:=", True])</pre>

VB Syntax	AssignLinkedRegion <LinkedRegionArray>
VB Example	oModule.AssignLinkedRegion Array("NAME:Linked1",

	<pre>"Objects:=", Array("RegularPolyhedron1"), "Type:=", "PO", "IsLinkedRegion:=", true)</pre>

AssignLumpedPort

Creates a lumped port excitation.

UI Access	HFSS > Excitations > Assign > Port > Lumped Port...		
Parameters	Name	Type	Description
	<LumpedPortArray>	Array	Structured array. Array("NAME:<BoundName>", "Faces:=", <FaceIDArray>, "RenormalizeAllTerminals:=", <boolean>, "DoDeembed:=", <boolean>, <ModesArray>, "ShowReporterFilter:=", <boolean>, "ReporterFilter:=", <array of boolean>, "Impedance:=", <value>)
	<ModesArray>	Array	Structured array. Array("NAME:<ModesArrayName>", <OneModeArray>, <OneModeArray>,...)
	<OneModeArray>	Array	Structured array.

			<pre>Array("NAME:<ModeName>", "ModeNum:=", <integer>, "UseIntLine:=", <boolean>, <IntegerationLineArray>, "AlignmentGroup:=", <integer, group id>, "CharImp:=", <string, characteristic impedance>, "RenormImp:=" , <value, renormalize impedance to>)</pre>
	< <i>IntegerationLineArray</i> >	Array	Structured array. <pre>Array("NAME:<LineName>", "Coordinate System:=", <string, relative coordinate system>, "Start:=" , <array, start location coordinates>, "End:=" , <array, end location coordinates>)</pre>
Return Value	None.		

Python Syntax	AssignLumpedPort (< <i>LumpedPortArray</i> >)
Python Example	<pre>oModule.AssignLumpedPort(["NAME:LumpedPort1", "Faces:=", [9], "DoDeembed:=", False, "RenormalizeAllTerminals:=", True,</pre>

	<pre>["NAME:Modes", ["NAME:Model", "ModeNum:=", 1, "UseIntLine:=", True, ["NAME:IntLine", "Coordinate System:=", "Global", "Start:=", ["-0.4mm", "-1mm", "0.8mm"], "End:=" , ["-0.3mm", "-1.2mm", "0.8mm"]], "AlignmentGroup:=", 0, "CharImp:=", "Zpi", "RenormImp:=", "50ohm"]], "ShowReporterFilter:=", False, "ReporterFilter:=", [True], "Impedance:=", "50ohm"])</pre>

VB Syntax	AssignLumpedPort < <i>LumpedPortArray</i> >
VB Example	oModule.AssignLumpedPort Array("NAME:LumpedPort1",

	<pre>"Faces:=", Array(9), "DoDeembed:=", false, "RenormalizeAllTerminals:=", true, Array("NAME:Modes", Array("NAME:Model", "ModeNum:=", 1, "UseIntLine:=", true, Array("NAME:IntLine", "Coordinate System:=", "Global", "Start:=", Array("-0.4mm", "-1mm", "0.8mm"), "End:=", Array("-0.55mm", "-0.95mm", "0.8mm")), "AlignmentGroup:=", 0, "CharImp:=", "Zpi", "RenormImp:=", "50ohm")), "ShowReporterFilter:=", false, "ReporterFilter:=", Array(true), "Impedance:=", "50ohm")</pre>

AssignLumpedRLC

Creates a lumped RLC boundary.

UI Access	HFSS > Boundaries > Assign > Lumped RLC...		
Parameters	Name	Type	Description
	<LumpedRLCArray>	Array	Structured array. Array("NAME:<BoundName>", "RLC Type:=" , <"Parallel" "Serial" >, "UseResist:=",<boolean>, "Resistance:=", <value>,

			<code>"UseInduct:=", <boolean>, "Inductance:=", <value>, "UseCap:=", <boolean>, "Capacitance:=", <value>, <CurrentLineArray>, "Objects:=", <AssignmentObjects>, "Faces:=", <AssignmentFaces>)</code>
	<code><CurrentLineArray></code>	Array	Structured array. <code>Array("NAME:CurrentLine", _ "Start:=", <LineEndPoint>, "End:=", <LineEndPoint>)</code>
Return Value	None.		

Python Syntax	AssignLumpedRLC (<LumpedRLCArray>)
Python Example	<pre>oModule.AssignLumpedRLC(["NAME:LumpRLC1", "Objects:=", ["Box1"],]</pre>

	<pre> "NAME:CurrentLine", "Start:=", ["0.15mm", "-0.2mm", "0mm"], "End:=" , ["0.15mm", "0.6mm", "0mm"]], "RLC Type:=", "Parallel", "UseResist:=", True, "Resistance:=", "100ohm", "UseInduct:=", True, "Inductance:=", "10nH", "UseCap:=", True, "Capacitance:=", "10pF"]) </pre>

VB Syntax	AssignLumpedRLC < <i>LumpedRLCArray</i> >
VB Example	<pre> oModule.AssignLumpedRLC Array("NAME:LumpRLC1", _ "RLC Type:=" , " Parallel" , "UseResist:=", true, _ "Resistance:=", "100hm", _ "UseInduct:=", true, _ "Inductance:=", "10nH", _ "UseCap:=", true, _ </pre>

	<pre>"Capacitance:=", "10pF", _ Array("NAME:CurrentLine", _ "Start:=", Array(-0.4, -1.2, -1.6), _ "End:=", Array(-0.4, -1.2, 0)), "Faces:=", Array(12))</pre>

AssignMagneticBias

Creates a magnetic bias source.

UI Access	HFSS > Excitations > Assign > Magnetic Bias...		
Parameters	Name	Type	Description
	<MagneticBiasArray>	Array	Structured array. <pre>Array("NAME:<BoundName>", "IsUniformBias:=", <boolean>, "Bias:=", <value>, "XAngle:=", <value>, "YAngle:=", <value>, "ZAngle:=", <value>, "Project:=", <string>, "Objects:=", <AssignmentObjects>)</pre> IsUniformBias If true, supply the Bias, XAngle, YAngle, and ZAngle parameters.

			If false, supply the Project parameter.
Return Value	None.		

Python Syntax	AssignMagneticBias (<MagneticBiasArray>)
Python Example	<pre> oModule.AssignMagneticBias(["NAME:MagBias1", "IsUniformBias:=", True, "Bias:=", "1", "XAngle:=", "10deg", "YAngle:=", "10deg", "ZAngle:=", "10deg", "Objects:=", ["Box2"]]) oModule.AssignMagneticBias(["NAME:MagBias2", "IsUniformBias:=", False, "Project:=", "D:/Maxwell/testing/m3dfs.pjt", "Objects:=", ["Box2"]]) </pre>

VB Syntax	AssignMagneticBias <MagneticBiasArray>
VB Example	oModule.AssignMagneticBias Array("NAME:MagBias1", _

	<pre>"IsUniformBias:=", true,_ "Bias:=", "1",_ "XAngle:=", "10deg",_ "YAngle:=", "10deg",_ "ZAngle:=", "10deg",_ "Objects:=", Array("Box2")) oModule.AssignMagneticBias Array("NAME:MagBias2",_ "IsUniformBias:=", false,_ "Project:=", "D:/Maxwell/testing/m3dfs.pjt",_ "Objects:=", Array("Box2"))</pre>

AssignMultipactionChargeRegion

Creates a Multipaction Charge Region for HFSS multipaction analysis.

UI Access	HFSS > Excitations > Assign > Multipaction Charge Region...		
Parameters	Name	Type	Description
	<ChargeRegionArray>	Array	Structured array. Array("NAME:MultipactionChargeRegion1", "Objects:=", <array of objects>, "Faces:=", <array of face IDs>, "NumParticles:=", "<integer, number of particles>",

			<pre> "ParticleCharge:=", "<num><unit>", "ParticleMass:=", "<num><unit>", "Vx:=", "<num><unit>", "Vy:=", "<num><unit>", "Vz:=", "<num><unit>") </pre>
Return Value	None.		

Python Syntax	AssignMultipactionChargeRegion(<ChargeRegionArray>)
Python Example	<pre> oModule.AssignMultipactionChargeRegion(["NAME:MultipactionChargeRegion1", "Objects:=", , ["Cylinder1"], "NumParticles:=", , "100", "ParticleCharge:=", , "-1.60217662e-19Coulomb", "ParticleMass:=", , "9.10938356e-31kg", "Vx:=", , "0.1m_per_sec", "Vy:=", , "0.3m_per_sec", "Vz:=", , "0.15m_per_sec"]) </pre>

VB Syntax	AssignMultipactionChargeRegion <ChargeRegionArray>
VB Example	<pre> oModule.AssignMultipactionChargeRegion Array("NAME:MultipactionChargeRegion1", _ </pre>

	<pre> "Objects:=", Array("Cylinder1"), _ "NumParticles:=", "100", _ "ParticleCharge:=", "-1.60217662e-19Coulomb", _ "ParticleMass:=", "9.10938356e-31kg", _ "Vx:=", "0.1m_per_sec", "Vy:=", "0.3m_per_sec", "Vz:=", "0.15m_per_sec") </pre>

AssignMultipactionDCBias

Set up DC electric or magnetic bias fields for a multipaction analysis

UI Access	HFSS > Excitations > Assign > Multipaction DC Bias...		
Parameters	Name	Type	Description
	<DCBiasArray>	Array	Structured array. <pre> Array(NAME:MultipactionDCBias1", "Objects:=", <array of objects>, "UniformE:=", <boolean>, "Ex:=", "<value><unit>", "Ey:=", "<value><unit>", "Ez:=", "<value><unit>", <LinkedField>, "UniformH:=" , <boolean>, "Hx:=", "<value><unit>", "Hy:=", "<value><unit>", </pre>

			<code>"Hz:=", "<value><unit>",</code> <code><LinkedField>)</code> UniformE or Uniform H: • True - provide filed values. • False - provide field through linked Maxwell analysis.
	<code><LinkedField></code>	Array	Structured array. <code>Array("NAME:<EField HField>",</code> <code>"Project:=", <string, path to linked project</code> <code>file>,</code> <code>"Product:=", "Maxwell",</code> <code>"Design:=", <string, name of source design>,</code> <code>"Soln:=", <string, name of linked solution>,</code> <code><Parameters for linked solution>,</code> <code>"ForceSourceToSolve:=", <boolean>,</code> <code>"PreservePartnerSoln:=", <boolean>,</code> <code>"PathRelativeTo:=", "TargetProject")</code>
Return Value	None.		

Python Syntax	<code>AssignMultipactionDCBias (<DCBiasArray>)</code>
Python Example	<code>oModule.AssignMultipactionDCBias (</code> <code>["NAME:MultipactionDCBias1",</code>

```
"Objects:=", ["Cylinder1"],
"UniformE:=", False,
["NAME:EField",
    "Project:=", "C://projects/Maxwell/HallSensor.aedt",
    "Product:=", "Maxwell",
    "Design:=", "3_Maxwell3D",
    "Soln:=", "Setup1 : Transient",
    ["NAME:Params",
        "angle:=", "0deg",
        "cell_spacing:=", "2.5mm",
        "die_thickness:=", "2mm",
        "gap:=", "1.5mm",
        "magnet_center_y:=", "0mm",
        "magnet_center_z:=", "0mm",
        "magnet_x:=", "3mm",
        "magnet_y:=", "6mm",
        "magnet_z:=", "5mm",
        "move_x:=", "-1.5mm",
        "sensor_offset_x:=", "0mm",
        "sensor_offset_y:=", "0mm",
```

```

        "sensor_offset_z:=", "0mm",
        "sensor_pitch:=", "0deg",
        "sensor_roll:=", "0deg",
        "sensor_yaw:=", "0deg",
        "target_dia:=", "80mm"

    ],
    "ForceSourceToSolve:=", False,
    "PreservePartnerSoln:=", False,
    "PathRelativeTo:=", "TargetProject"

],
"UniformH:=", True,
"Hx:=", "0A_per_m",
"Hy:=", "0A_per_m",
"Hx:=", "0A_per_m"

])

```

VB Syntax	AssignMultipactionDCBias <DCBiasArray>
VB Example	<pre> oModule.AssignMultipactionDCBias Array("NAME:MultipactionDCBias1", _ "Objects:=", Array("Cylinder1"), _ "UniformE:=", true, _ "Ex:=", "0V_per_meter", "Ey:=", "0V_per_meter", "Ez:=", "0V_per_meter", _ </pre>

	<pre>"UniformH:=", true, _ "Hx:=", "0A_per_m", "Hy:=", "0A_per_m", "Hz:=", "0A_per_m")</pre>

AssignMultipactionSEE

Creates Secondary Electron Emission (SEE) boundaries for a Multipaction analysis.

UI Access	HFSS > Boundaries > Assign > Multipaction SEE...		
Parameters	Name	Type	Description
	<SEEArray>	Array	Structured array. Array("NAME:<SEENAME>", "Faces:=", <array of face IDs>, "AlphaMax:=", <value>, "Alpha0:=", <value>, "E0:=", <value>, "E1:=", <value>, "E2:=", <value>, "Em:=", <value>, "DielectricSurface:=", <boolean>)
Return Value	None.		

Python Syntax	AssignMultipactionSEE (<SEESArray>)
Python Example	<pre>oModule.AssignMultipactionSEE(["NAME:SEE1", "Faces:=", [342, 7, 341, 11, 8, 175, 35], "AlphaMax:=", "2.25", "Alpha0:=", "0", "E0:=", "12.5", "E1:=", "25", "E2:=", "5000", "Em:=", "175", "DielectricSurface:=", false])</pre>

VB Syntax	AssignMultipactionSEE <SEESArray>
VB Example	<pre>oModule.AssignMultipactionSEE Array("NAME:SEE1", _ "Faces:=", Array(342, 7, 341, 11, 8, 175, 35), _ "AlphaMax:=", "2.25", "Alpha0:=", "0", _ "E0:=", "12.5", "E1:=", "25", "E2:=", "5000", _ "Em:=", "175", "DielectricSurface:=", false)</pre>

AssignPerfectE

Creates a perfect E boundary.

UI Access	HFSS > Boundaries > Assign > Perfect E...

Parameters	Name	Type	Description
	<code><PerfectEArray></code>	Array	Structured array. Array("NAME:<BoundName>", "InfGroundPlane:=", <boolean>, "Objects:=", <AssignmentObjects>, "Faces:=", <AssignmentFaces>)
Return Value	None.		

Python Syntax	AssignPerfectE (<PerfectEArray>)
Python Example	<pre>oModule.AssignPerfectE(["NAME:PerfE1", "InfGroundPlane:=", False, "Faces:=", [12]])</pre>

VB Syntax	AssignPerfectE <PerfectEArray>
VB Example	<pre>oModule.AssignPerfectE Array("NAME:PerfE1",_ "InfGroundPlane:=", false,_ "Faces:=", Array(12))</pre>

AssignPerfectH

Creates a perfect H boundary.

UI Access	HFSS > Boundaries > Assign > Perfect H...		
Parameters	Name	Type	Description
	<PerfectHArray>	Array	Structured array. <pre>Array("NAME:<BoundName>", "Objects:=", <AssignmentObjects>, "Faces:=", <AssignmentFaces>)</pre>
Return Value	None.		

Python Syntax	AssignPerfectH (<PerfectHArray>)
Python Example	<pre>oModule.AssignPerfectH(["NAME:PerfH1", "Faces:=", [12]])</pre>

VB Syntax	AssignPerfectH <PerfectHArray>
VB Example	<pre>oModule.AssignPerfectH Array("NAME:PerfH1", _ "Faces:=", Array(12))</pre>

AssignPlaneWave

Creates an incident plane wave excitation.

UI Access	HFSS > Excitations > Assign > Incident Wave > Plane Wave...		
Parameters	Name	Type	Description
	< <i>PlaneWaveArray</i> >	Array	Structured array. Array("NAME:<Name of incident wave>", "Faces:=", <array of face IDs>, "IsCartesian:=", <boolean>, "EoX:=", "<numeric value>",&br/> "EoY:=", "<numeric value>",&br/> "EoZ:=", "<numeric value>",&br/> "kX:=", "<numeric value>",&br/> "kY:=", "<numeric value>",&br/> "kZ:=", "<numeric value>",&br/> "PhiStart:=", <value>, "PhiStop:=", <value>, "PhiPoints:=", <int>, "ThetaStart:=", <value>, "ThetaStop:=", <value>, "ThetaPoints:=", <integer>,

			<pre>"EoPhi:=", <value>, "EoTheta:=", <value>, "OriginX:=", "<numUnit>", "OriginY:=", "<numUnit>", "OriginZ:=", "<numUnit>", "IsPropagating:=", <boolean>, "IsEvanescent:=", <boolean>, "IsEllipticallyPolarized:=", <boolean>, "RealPropConst:=", "<value>", "ImagPropConst:=", "<value>", "PolarizationAngle:=", "<value>deg", "PolarizationRatio:=", " , "<value>")</pre> IsCartesian: • True - provide the EoX, EoY, EoZ, kX, kY, kZ parameters. • False - provide the PhiStart, PhiStop, PhiPoints, ThetaStart, ThetStop, ThetaPoints, EoPhi, EoTheta parameters. IsEvanescent: • True - provide the RealPropConst, ImagPropConst. IsEllipticallyPolarized: • True - provide the PolarizationAngle, PolarizationRatio.
Return Value	None.		

Python Syntax	AssignPlaneWave(<PlaneWaveArray>)
Python Example	<pre>oModule.AssignPlaneWave(["NAME:IncPWave1", "Faces:=" , [9], "IsCartesian:=" , True, "EoX:=" , "1", "EoY:=" , "0", "EoZ:=" , "0", "kX:=" , "0", "kY:=" , "0", "kZ:=" , "1", "OriginX:=" , "0mm", "OriginY:=" , "0mm", "OriginZ:=" , "0mm", "IsPropagating:=" , True, "IsEvanescent:=" , False, "IsEllipticallyPolarized:=", False]) oModule.AssignPlaneWave(</pre>

```
[ "NAME:IncPWave2",  
  "Faces:="                , [8],  
  "IsCartesian:="          , False,  
  "PhiStart:="              , "0deg",  
  "PhiStop:="               , "0deg",  
  "PhiPoints:="             , 1,  
  "ThetaStart:="            , "0deg",  
  "ThetaStop:="             , "0deg",  
  "ThetaPoints:="           , 1,  
  "EoPhi:="                 , "1",  
  "EoTheta:="               , "0",  
  "OriginX:="               , "0mm",  
  "OriginY:="               , "0mm",  
  "OriginZ:="               , "0mm",  
  "IsPropagating:="         , False,  
  "IsEvanescent:="          , True,  
  "IsEllipticallyPolarized:=", False,  
  "RealPropConst:="         , "0",  
  "ImagPropConst:="         , "1"  
]
```

VB Syntax	AssignPlaneWave <PlaneWaveArray>
VB Example	<pre>oModule.AssignPlaneWave Array("NAME:IncPWave3", _ "Faces:=", Array(8), _ "IsCartesian:=", true, _ "EoX:=", "1", "EoY:=", "0", "EoZ:=", "0", _ "kX:=", "0", "kY:=", "0", "kZ:=", "1", _ "OriginX:=", "0mm", "OriginY:=", "0mm", "OriginZ:=", "0mm", _ "IsPropagating:=", false, _ "IsEvanescent:=", false, _ "IsEllipticallyPolarized:=", true, _ "PolarizationAngle:=", "0deg", "PolarizationRatio:=", "1")</pre>

AssignRadiation

Creates a radiation boundary.

UI Access	HFSS > Boundaries > Assign > Radiation...		
Parameters	Name	Type	Description
	<RadiationArray>	Array	Structured array. Array("NAME:<BoundName>", "Objects:=", <AssignmentObjects>, "Faces:=", <AssignmentFaces>,

			<code>"IsFssReference:=", <boolean>,</code> <code>"IsForPML:=", <boolean>)</code> "IsFssReference" and "IsForPML" are only used to support legacy designs. In the current version, these should always be set to "False".
Return Value	None.		

Python Syntax	<code>AssignRadiation(<RadiationArray>)</code>
Python Example	<pre>oModule.AssignRadiation(["NAME:Rad1", "Faces:=", [6, 7], "IsFssReference:=", False, "IsForPML:=", False]) </pre>

VB Syntax	<code>AssignRadiation <RadiationArray></code>
VB Example	<pre>oModule.AssignRadiation Array("NAME:Rad1", _ "Faces:=", Array(6, 7), _ "IsFssReference:=", false, _ "IsForPML:=", false) </pre>

AssignRFDischargeDCBias

Creates the DC bias for an RF Discharge simulation.

UI Access	HFSS > Excitations > Assign > RF Discharge DC Bias...		
Parameters	Name	Type	Description
	<DCBiasParameters>	Array	Structured array. Array("NAME:<DCBiasName>", "UniformH:=", <boolean>, "Hx:=", "<real>A_per_m", "Hy:=", "<real>A_per_m", "Hz:=", "<real>A_per_m")
Return Value	None.		

Python Syntax	AssignRFDischargeDCBias (<DCBiasParameters>)
Python Example	<pre>oModule.AssignRFDischargeDCBias(["NAME:RFDischargeDCBias1", "UniformH:=", True, "Hx:=", "795774.71546A_per_m", "Hy:=", "0A_per_m",</pre>

	<pre>"Hz:=", "0A_per_m"])</pre>

VB Syntax	AssignRFDischargedCBias <DCBiasParameters>
VB Example	<pre>oModule.AssignRFDischargedCBias _ Array("NAME:RFDischargedCBias1", _ "UniformH:=", True, _ "Hx:=", "795774.71546A_per_m", "Hy:=", "0A_per_m", "Hz:=", "0A_per_m")</pre>

AssignScreeningImpedance

Creates a screening impedance boundary.

Command: HFSS>Boundaries>Assign>Screening Impedance

Syntax: AssignScreeningImpedance <ScreeningArray>

Return Value: None.

Parameters: <ScreeningArray>

```
Array("NAME:<name>",
"Objects:=", Array( "<name>"),
"IsAnisotropic:=", <Boolean>,
    If true, you need to specify the coordinate system
"CoordSystem:=", <integer or name>,
"HasExternalLink:=", <Boolean>,
```

true or false. If False, specify XResistance and XReactance values. Also see the first example.

```
"XResistance:=", "<value>",
```

```
"XReactance:=", "<value>"
```

If true, then specify the external link array with the project and solution to use. Also see the second example.

```
Array("NAME:XLink",
```

```
"Project:=", "<projectName>.aedt",
```

```
"Design:=", "<DesignName>",
```

```
"Soln:=", "Setup1 : LastAdaptive",
```

```
Array("NAME:Params", "<variable>:=", "<value>"),
```

```
"ForceSourceToSolve:=", <Boolean>,
```

```
"PreservePartnerSoln:=", <Boolean>,
```

```
"PathRelativeTo:=", "TargetProject"),
```

```
Array("NAME:YLink",
```

```
"Project:=", "<projectName>.aedt",
```

```
"Design:=", "HFSSDesign1",
```

```
"Soln:=", "Setup1 : LastAdaptive",
```

```
Array("NAME:Params", "<variable>:=", "<value>"),
```

```
"ForceSourceToSolve:=", <Boolean>
```

```
"PreservePartnerSoln:=", <Boolean>,
```

```
"PathRelativeTo:=", "TargetProject"))
```

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("Project53")
Set oDesign = oProject.SetActiveDesign("HFSSDesign1")
Set oModule = oDesign.GetModule("BoundarySetup")

oModule.AssignScreeningImpedance Array("NAME:Screening1",
"Faces:=", Array(12),
"IsAnisotropic:=", false,
"HasExternalLink:=", false,
"XResistance:=", "377", "XReactance:=", "0")
```

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("Project53")
Set oDesign = oProject.SetActiveDesign("HFSSDesign1")
Set oModule = oDesign.GetModule("BoundarySetup")

oModule.AssignScreeningImpedance Array("NAME:Screening1", "Objects:=", Array( "Rectangle1"),
    "IsAnisotropic:=", true,
    "CoordSystem:=", 1,
    "HasExternalLink:=", true,
    Array("NAME:XLink",
        "Project:=", "mydesign.aedt",
        "Design:=", "HFSSDesign1",
        "Soln:=", "Setup1 : LastAdaptive",
        Array("NAME:Params", "bend_angle:=", "50deg"),
        "ForceSourceToSolve:=", false,
        "PreservePartnerSoln:=", false,
        "PathRelativeTo:=", "TargetProject"),
    Array("NAME:YLink", "Project:=", "mydesign.aedt",
        "Design:=", "HFSSDesign1",
        "Soln:=", "Setup1 : LastAdaptive",
        Array("NAME:Params", "bend_angle:=", "50deg"),
        "ForceSourceToSolve:=", true,
```

```
"PreservePartnerSoln:=", true,  
"PathRelativeTo:=", "TargetProject"))
```

AssignSymmetry

Creates a symmetry boundary.

UI Access	HFSS > Boundaries > Assign > Symmetry.		
Parameters	Name	Type	Description
	<SymmetryArray>	Array	Structured array. Array("NAME:<BoundName>", "IsPerfectE:=", <boolean>, "Objects:=", <AssignmentObjects>, "Faces:=", <AssignmentFaces>)
Return Value	None.		

Python Syntax	AssignSymmetry (<SymmetryArray>)
Python Example	<pre>oModule.AssignSymmetry(["NAME:Sym1", "IsPerfectE:=", True, "Faces:=", [12]])</pre>

VB Syntax	AssignSymmetry <SymmetryArray>
VB Example	<pre>oModule.AssignSymmetry Array("NAME:Sym1", _ "IsPerfectE=", true, _ "Faces=", Array(12))</pre>

AssignTerminal

UI Access			
Parameters	Name	Type	Description
	<TerminalArray>	Array	Structured array.
Return Value	None.		

Python Syntax	AssignTerminal (<TerminalArray>)
Python Example	

VB Syntax	AssignTerminal <TerminalArray>
VB Example	

AssignVoltage

Creates a voltage source.

UI Access	HFSS > Excitations > Assign > Voltage...		
Parameters	Name	Type	Description
	<VoltageArray>	Array	Structured array. <pre>Array("NAME:<BoundName>", "Voltage:=", <value>, <DirectionArray>, "Objects:=", <AssignmentObjects>, "Faces:=", <AssignmentFaces>)</pre>
	<DirectionArray>	Array	Structured array. <pre>Array("NAME:Direction",_ "Start:=", <LineEndPoint>, "End:=", <LineEndPoint>)</pre>
Return Value	None.		

Python Syntax	AssignVoltage (<VoltageArray>)
Python Example	<pre>oModule.AssignVoltage(["NAME:Voltage1", "Voltage:=", "1000mV", ["NAME:Direction", "Start:=", [-0.4, -1.2, 0], "End:=", [-1.4, -1.2, 0]], "Faces:=", [7])</pre>

VB Syntax	AssignVoltage <VoltageArray>
VB Example	<pre>oModule.AssignVoltage Array("NAME:Voltage1", _ "Voltage:=", "1000mV", _ Array("NAME:Direction", _ "Start:=", Array(-0.4, -1.2, 0), _ "End:=", Array(-1.4, -1.2, 0)), _ "Faces:=", Array(7))</pre>

AutoCreatePECCapforWavePort

Use: If the HFSS Option for HFSS>Boundary Assignment>Automatic PEC cap creation for wave port is enabled, HFSS automatically creates a PEC cap under the following conditions: The general rule is that the dialog for creating wave port PEC backing appears if the wave port is internal to the design and does not touch any non-solve-inside component. If a project has open region defined, all geometries are considered internal.

Command: None.

Syntax: AutoCreatePECCapForWavePort ([<PECCapParameters>])

Return Value: None

Parameters: <PECCapParameters>

```
"NAME:AutoCreatePECCapForWavePort",  
"Wave Port Name:=" , "<string>",  
"Face ID:=" , <intID>,  
"Flip Side:=" , Boolean,
```

```
"Thickness:=" , "<real><units>"
```

VB Example:

```
oModule.AutoCreatePECCapForWavePort Array("NAME:AutoCreatePECCapForWavePort", "Wave Port
Name:=", _
"2", "Face ID:=", 280, "Flip Side:=", false, "Thickness:=", "0.18mm")
```

Python Syntax	AutoCreatePECCapforWavePort(<PECCapParameters>)
Python Example	AutoCreatePECCapForWavePort(["NAME:AutoCreatePECCapForWavePort", "Wave Port Name:=" , "2", "Face ID:=" , 280, "Flip Side:=" , False, "Thickness:=" , "0.18mm")

CircuitPortToLumpedPort

Converts a circuit port to a lumped port for a driven terminal or driven modal design in HFSS.

UI Access	Right-click on a circuit port, then select Convert to Lumped Port .		
Parameters	Name	Type	Description
	<PortName>	String	Name of specified circuit port.

Return Value	None.

Python Syntax	CircuitPortToLumpedPort (<PortName>)
Python Example	<code>oModule.CircuitPortToLumpedPort ("1")</code>

VB Syntax	CircuitPortToLumpedPort <PortName>
VB Example	<code>oModule.CircuitPortToLumpedPort "1"</code>

EditAnisotropicImpedance

Modifies an Anisotropic Impedance boundary condition.

UI Access	Double-click the boundary condition in the project tree to modify its settings.		
Parameters	Name	Type	Description
	<BondName>	String	Name of boundary condition to be edited.
	<Aniso-tropicImpArray>	Array	Structured array. Array("NAME:<string name of anisotropic impedance>", "Faces:=" , <array of face IDs>, "UseInfiniteGroundPlane:=" , <boolean>, "CoordSystem:=" , <string coordinate system name>, "HasExternalLink:=" , <boolean>, "ZxxResistance:=" , <string of an integer value>,

			<pre>"ZxxReactance:=" , <string of an integer value>, "ZxyResistance:=" , <string of an integer value>, "ZxyReactance:=" , <string of an integer value>, "ZyxResistance:=" , <string of an integer value>, "ZyxReactance:=" , <string of an integer value>, "ZyyResistance:=" , <string of an integer value>, "ZyyReactance:=" , <string of an integer value>)</pre>
Return Value	None.		

Python Syntax	EditAnisotropicImpedance (<BondName>, <AnisotropicImpArray>)

Python Example	<pre>oModule.EditAnisotropicImpedance("Anistropic10", ["NAME:Anisotropic1", "Faces:=" , [17215], "UseInfiniteGroundPlane:=", False, "CoordSystem:=" , "Global", "HasExternalLink:=" , False, "ZxxResistance:=" , "377", "ZxxReactance:=" , "0", "ZxyResistance:=" , "0", "ZxyReactance:=" , "0", "ZyxResistance:=" , "0", "ZyxReactance:=" , "0", "ZyyResistance:=" , "377", "ZyyReactance:=" , "0"])</pre>

VB Syntax	<code>EditAnisotropicImpedance <BondName>, <AnisotropicImpArray></code>
VB Example	<pre>oModule.EditAnisotropicImpedance "Anisotropic10", _</pre>

	<pre>Array("NAME:Anisotropic1", "Faces:=", Array(17215), "UseInfiniteGroundPlane:=", false, "CoordSystem:=" , "Global", "HasExternalLink:=" , false, "ZxxResistance:=" , "377", "ZxxReactance:=" , "0", "ZxyResistance:=" , "0", "ZxyReactance:=" , "0", "ZyxResistance:=" , "0", "ZyxReactance:=" , "0", "ZyyResistance:=" , "377", "ZyyReactance:=" , "0")</pre>

EditAperture

Modifies an aperture boundary.

UI Access	Double-click the boundary condition in the project tree to modify its settings.		
Parameters	Name	Type	Description
	<BondName>	String	Name of boundary condition to be edited.
	<ApertureArray>	Array	Array defines selected objects.
Return Value	None.		

Python Syntax	EditAperture(<BondName>, <ApertureArray>)
Python Example	<pre>oModule.EditAperture("Aperture1" ["NAME:Aperture2", "Objects:=", ["Rectangle1"]])</pre>

VB Syntax	EditAperture <BondName>, <ApertureArray>
VB Example	<pre>oModule.EditAperture "Aperture1" Array("NAME:Apertur2", "Objects:=", Array("Rectangle1"))</pre>

EditCircuitPort[HFSS]

Edits a circuit port for a driven terminal or driven modal design in HFSS.

UI Access	Double-click the excitation in the project tree to modify its settings.		
Parameters	Name	Type	Description
	<ExcitationName>	String	Name of the excitation to be edited.
	<CircuitPortArray>	Array	Structured array. Array("NAME:<PortName>", "Impedance:=", "valueohm",

			"DoDeembed:=", <boolean> "RenormalizeAllTerminals:=", <boolean> "TerminalIDList:=", Array())
Return Value	None.		

Python Syntax	EditCircuitPort (<ExcitationName>, <CircuitPortArray>)		
Python Example	<pre>oModule.EditCircuitPort ("5" ["NAME:1", "Impedance:=" , "50ohm", "DoDeembed:=" , False, "RenormalizeAllTerminals:=", True, "TerminalIDList:=" , []])</pre>		

VB Syntax	EditCircuitPort <ExcitationName>, <CircuitPortArray>		
VB Example	<pre>oModule.EditCircuitPort "5" _ Array("NAME:1", _ "Impedance:=", "50ohm", _ "DoDeembed:=", false, _ "RenormalizeAllTerminals:=", true, _</pre>		

	<code>"TerminalIDList:=", Array())</code>

EditDielectricCavity

Modifies definitions of a Dielectric Cavity.

UI Access	Double-click the cavity under Hybrid Regions in the project tree to modify its settings.		
Parameters	Name	Type	Description
	<CavityName>	String	Name of the cavity to be edited.
	<CavityArray>	Array	Structured array. <code>Array("NAME:<new name>")</code>
Return Value	None.		

Python Syntax	<code>EditDielectricCavity(<CavityName>, <CavityArray>)</code>
Python Example	<pre>oModule.EditDielectricCavity("Cavity1" ["NAME:Cavity2"])</pre>

VB Syntax	<code>EditDielectricCavity <CavityName>, <CavityArray></code>
VB Example	<code>oModule.EditDielectricCavity "Cavity1", _</code>

	<code>Array("NAME:Cavity2")</code>

EditDiffPairs

Edits the properties of differential pairs defined from terminal excitations on wave ports.

UI Access	HFSS > Excitations > Differential Pairs.		
Parameters	Name	Type	Description
	<code><DifferentialPairsArray></code>	Array	Structured array. <code>Array("NAME:EditDiffPairs", <OneDiffPairArray>, <OneDiffPairArray>,...)</code>
	<code><OneDiffPairArray></code>	Array	Structured array. <code>Array("NAME:Pair1",_ "PosBoundary:=", <string, name of the terminal to use as the positive terminal.>, "NegBoundary:=", <string, name of the terminal to use as the negative terminal.>, "CommonName:=", <string, name for the common mode.>, "CommonRefZ:=", <value, reference impedance for the common mode.>, "DiffName:=", <string, name for the differential mode.>, "DiffRefZ:=", <value, reference impedance for the differential mode.>, "IsActive:=", <boolean>)</code>

Return Value	None.

Python Syntax	EditDiffPairs(<<i>DifferentialPairsArray</i>>)
Python Example	<pre>oModule.EditDiffPairs(["NAME:EditDiffPairs", ["NAME:Pair1", "PosBoundary:=", "Rectangle1_T1", "NegBoundary:=", "Rectangle2_T1", "CommonName:=", "Comm1", "CommonRefZ:=", "25ohm", "DiffName:=", "Diff1", "DiffRefZ:=", "100ohm", "IsActive:=", True]])</pre>

VB Syntax	EditDiffPairs <<i>DifferentialPairsArray</i>>
VB Example	<pre>oModule.EditDiffPairs Array("NAME:EditDiffPairs", Array("NAME:Pair1", _ "PosBoundary:=", "Rectangle1_T1", "NegBoundary:=", "Rectangle2_T1", _ "CommonName:=", "Comm1", "CommonRefZ:=", "25ohm", _</pre>

	<pre>"DiffName:=", "Diff1", "DiffRefZ:=", "100ohm", _ "IsActive:=", true))</pre>

EditFEBI

Modifies a FE-BI hybrid region.

UI Access	Double-click the FE-BI under Hybrid Regions in the project tree to modify its settings.		
Parameters	Name	Type	Description
	<FEBIName>	String	Name of the FE-BI region to be edited.
	<FEBIArray>	Array	Structured array. Array("NAME:<new name>")
Return Value	None.		

Python Syntax	EditFEBI (<FEBIName>, <FEBIArray>)
Python Example	<pre>oModule.EditFEBI ("FE-BI1" ["NAME:FE-BI2")</pre>

VB Syntax	EditFEBI <FEBIName>, <FEBIArray>
VB Example	<pre>oModule.EditFEBI "FE-BI1", _ Array("NAME:FE-BI2")</pre>

EditFiniteCond

Modifies parameters of single finite conductivity boundary.

UI Access	Double-click finite conductivity boundary in project tree.		
Parameters	Name	Type	Description
	<BoundaryName>	String	Name of the finite conductivity boundary to be edited.
	<FiniteCondArray>	Array	Structured array. Array("NAME:<Name of the boundary>", "Roughness:=", "<string, double value with units of length>", "UseCoating:=", <boolean>, "LayerThickness:=", "<string, double value with units of length>", "UseMaterial:=", <boolean>, "Material:=", "<string, material name for coating.>")
Return Value	None.		

Python Syntax	EditFiniteCond (<BoundaryName>, <FiniteCondArray>)
Python Example	<pre>oModule.EditFiniteCond("FiniteCond1", ["NAME:FiniteCond1",</pre>

	<pre>"Roughness:=", "2um", "UseCoating:=", false])</pre>

VB Syntax	EditFiniteCond <BoundaryName>, <FiniteCondArray>
VB Example	<pre>oModule.EditFiniteCond "FiniteCond1", _ Array("NAME:FiniteCond1", _ "Roughness:=", "2um", _ "UseCoating:=", false)</pre>

EditFloquetPort

Modifies a Floquet port excitation.

UI Access	Double-click on a floquet port under Excitations in history tree.		
Parameters	Name	Type	Description
	<FloquetPortName>	String	Name of the floquet port excitation to be edited.
	<FloquetPortArray>	Array	Structured array.
			<pre>Array("NAME:<BoundName>", "NumModes:=", <integer>, "RenormalizeAllTerminals:=", <boolean>, "DoDeembed:=", <boolean>, <ModesArray>,</pre>

		<pre> "ShowReporterFilter:=", <boolean>, "UseScanAngles:=", <boolean>, "Phi:=", "<numdeg>", "Theta:=", "<numdeg>", <LatticeAVector>, <LatticeBVector>, <ModesCalculator>, <ModesList>) </pre>
<ModesArray>	Array	Structured array. <pre> Array("NAME:Modes", Array("NAME:<ModeName>", "ModeNum:=", <integer>, "UseIntLine:=", <boolean>), ...) </pre>
<LatticeAVector>	Array	Structured array. <pre> Array("NAME:LatticeAVector", "Start:=", Array("<num><units>", "<num><units>", "<num><units>"), "End:=", Array("<num><units>", "<num><units>", "<num><units>")) </pre>
<LatticeBVector>	Array	Structured array. <pre> Array("NAME:LatticeBVector", "Start:=", Array("<num><units>", "<num><units>", </pre>

		<pre>"<num><units>"), "End:=", Array("<num><units>", "<num><units>", "<num><units>"))</pre>
<ModesCalculator>	Array	Structured array. <pre>Array("NAME:ModesCalculator", "Frequency:=", "<Value>GHz", "FrequencyChanged:=", <Boolean>, "PhiStart:=", "<num>deg", "PhiStop:=", "<num>deg", "PhiStep:=", "<num>deg", "ThetaStart:=", "<num>deg", "ThetaStop:=", "<num>deg", "ThetaStep:=", "<num>deg")</pre>
<ModesList>	Array	Structured array. <pre>Array("NAME:ModesList", Array("NAME:Mode", "ModeNumber:=", <ModeID>, "IndexM:=", <integer index>, "IndexN:=", <integer index>, "KC2:=", <integer value>, "PropagationState:=", "Propagating", "Attenuation:=", <integer value>,</pre>

	"PolarizationState:=", <TE or TM>, "AffectsRefinement:=", <boolean>), ...)		
Return Value	None.		

Python Syntax	<code>EditFloquetPort(<FloquetPortName>, <FloquetPortArray>)</code>
Python Example	<pre>oModule.EditFloquetPort("FloquetPort1", ["NAME:FloquetPort1After", "NumModes:=", 2, "RenormalizeAllTerminals:=", True, "DoDeembed:=", False, ["NAME:Modes", ["NAME:Model1", "ModeNum:=", 1, "UseIntLine:=", False], ["NAME:Model2", "ModeNum:=", 2, "UseIntLine:=", False]], "ShowReporterFilter:=", False,</pre>

```

"UseScanAngles:=", True, "Phi:=", "0deg", "Theta:=", "0deg",
["NAME:LatticeAVector",
    "Start:=", ["0mm", "0mm", "0.8mm"],
    "End:=", ["0mm", "0.5mm", "0.8mm"]],
["NAME:LatticeBVector",
    "Start:=", ["0mm", "0mm", "0.8mm"],
    "End:=", ["0.8mm", "0mm", "0.5mm"]],
["NAME:ModesCalculator",
    "Frequency:=", "1GHz",
    "FrequencyChanged:=", False,
    "PhiStart:=", "0deg",
    "PhiStop:=", "0deg",
    "PhiStep:=", "0deg",
    "ThetaStart:=", "0deg",
    "ThetaStop:=", "0deg",
    "ThetaStep:=", "0deg"],
["NAME:ModesList",
    ["NAME:Mode",
        "ModeNumber:=", 1,
        "IndexM:=", 0,
        "IndexN:=", 0,

```

```
        "KC2:=", 0,  
        "PropagationState:=", "Propagating",  
        "Attenuation:=", 0,  
        "PolarizationState:=", "TE",  
        "AffectsRefinement:=", False],  
    ["NAME:Mode",  
        "ModeNumber:=", 2,  
        "IndexM:=", 0,  
        "IndexN:=", 0,  
        "KC2:=", 0,  
        "PropagationState:=", "Propagating",  
        "Attenuation:=", 0,  
        "PolarizationState:=", "TM",  
        "AffectsRefinement:=", False]  
    ]])
```

VB Syntax	EditFloquetPort <FloquetPortName>, <FloquetPortArray>
VB Example	<pre>oModule.AssignFloquetPort "FloquetPort1" Array("NAME:FloquetPort1After",</pre>

```

"NumModes:=", 2,
"RenormalizeAllTerminals:=", true,
"DoDeembed:=", false,
Array("NAME:Modes", Array("NAME:Mode1", "ModeNum:=", 1,
"UseIntLine:=", false),
Array("NAME:Mode2", "ModeNum:=", 2, "UseIntLine:=", false)),
"ShowReporterFilter:=", false,
"UseScanAngles:=", true, "Phi:=", "0deg", "Theta:=", "0deg",
Array("NAME:LatticeAVector", "Start:=", Array("0mm", "0mm", "0.5mm"),
      "End:=", Array("0mm", "0.6mm", "0.8mm")),
Array("NAME:LatticeBVector", "Start:=", Array("0mm", "0mm", "0.5mm"),
      "End:=", Array("0.8mm", "0mm", "0.8mm")),
Array("NAME:ModesCalculator", "Frequency:=", "1GHz", "FrequencyChanged:=", false,
"PhiStart:=", "0deg", "PhiStop:=", "0deg", "PhiStep:=", "0deg",
"ThetaStart:=", "0deg", "ThetaStop:=", "0deg", "ThetaStep:=", "0deg"),
Array("NAME:ModesList",
Array("NAME:Mode", "ModeNumber:=", 1, "IndexM:=", 0, "IndexN:=", 0, "KC2:=", 0,
"PropagationState:=", "Propagating",
"Attenuation:=", 0,
"PolarizationState:=", "TE",
"AffectsRefinement:=", false),

```

	<pre>Array("NAME:Mode", "ModeNumber:=", 2, "IndexM:=", 0, "IndexN:=", 0, "KC2:=", 0, "PropagationState:=", "Propagating", "Attenuation:=", 0, "PolarizationState:=", "TM", "AffectsRefinement:=", false)))</pre>

EditFresnel

Modifies a Fresnel boundary condition.

UI Access	Double-click on a fresnel under Boundaries in project history tree.		
Parameters	Name	Type	Description
	<FresnelName>	String	Name of the fresnel boundary to be edited.
	<FresnelArray>	Array	Structured array. Array("NAME:<FresnelName>", "Fresnel Boundary Type:=", <PerfectAbsorber or ImportFromTableFile>, "RTTable Path:=", <string path to table file>)
Return Value	None.		

Python Syntax	EditFresnel(<FresnelName>, <FresnelArray>)

Python Example	<pre>oModule.AssignFresnel("Fresnel1" ["NAME:Fresnel2", "Fresnel Boundary Type:=", "PerfectAbsorber"])</pre>

VB Syntax	EditFresnel <FresnelName>, <FresnelArray>
VB Example	<pre>oModule.AssignFresnel "Fresnel1" Array("NAME:Fresnel2", _ "Fresnel Boundary Type:=", "PerfectAbsorber")</pre>

EditGlobalMatEnv

Modifies global material environment setting to tell HFSS what material properties to use when calculating far fields.

UI Access	HFSS > Boundaries > Edit Global Material Environment...		
Parameters	Name	Type	Description
	<MatEnvName>	String	Name of the global material environment, default is "vacuum".
Return Value	None.		

Python Syntax	EditGlobalMatEnv(<MatEnvName>)
Python Example	oModule.EditGlobalMatEnv("water_distilled")

VB Syntax	EditGlobalMatEnv <MatEnvName>
VB Example	oModule.EditGlobalMatEnv "water_distilled"

EditGradientSurfaceRoughness

Edit an existing Gradient Surface Roughness boundary.

UI Access	Double-click on a half space under Boundaries in history tree		
Parameters	Name	Type	Description
	<ArgArray>	Array	Structured array. ["NAME:<Name of GradientSurfaceBoundary>", "Objects:=" , ["<geometryID>"], "RMS Roughness:=" , "<value><units>", "Bulk Conductivity:=" , "<value>", "Thickness:=" , ""<value><units>", "Max Frequency:=" , ""<value><units>", "Surface Type:=" , "[Low Profile High Profile]"
Return Value	None.		

Python Syntax	AssignGradientSurfaceRoughness(<ArgArray>)
Python Example	oModule = oDesign.GetModule("BoundarySetup")

	<pre>oModule.AssignGradientSurfaceRoughness (["NAME:GradientImped1", "Objects:=" , ["Box1"], "RMS Roughness:=" , "1um", "Bulk Conductivity:=" , "58000000", "Thickness:=" , "15um", "Max Frequency:=" , "100GHz", "Surface Type:=" , "Low Profile"])</pre>

VB Syntax	AssignGradientSurfaceRoughness <ArgArray>
VB Example	<pre>oModule = oDesign.GetModule("BoundarySetup") oModule.AssignGradientSurfaceRoughness("NAME:GradientImped1", _ "Objects:=" , ["Box1"], "RMS Roughness:=" , "1um", _ "Bulk Conductivity:=" , "58000000", "Thickness:=" , "15um", _ "Max Frequency:=" , "100GHz", "Surface Type:=" , "Low Profile")</pre>

EditHalfSpace

Modifies a Half Space boundary name, Z location, and or materials.

UI Access	Double-click on a half space under Boundaries in history tree.		
Parameters	Name	Type	Description

	<HalfSpaceName>	String	Name of the half space boundary to be edited.
	<HalfSpaceArray>	Array	Structured array. Array("NAME:<Name of Half Space>", "ZLocation:=", "<intUnits>", "Material:=", "<string material name>")
Return Value	None.		

Python Syntax	EditHalfSpace(<HalfSpaceName>, <HalfSpaceArray>)
Python Example	<pre>oModule.EditHalfSpace("HalfSpace1" ["NAME:HalfSpace1After", "ZLocation:=", "2mm", "Material:=", "tungsten"]])</pre>

VB Syntax	EditHalfSpace <HalfSpaceName>, <HalfSpaceArray>
VB Example	<pre>oModule.EditHalfSpace "HalfSpace1", _ Array("NAME:HalfSpace1After", _ "ZLocation:=", "2mm", _ "Material:=", "tungsten")</pre>

EditHybridRegion

Modifies settings of an assigned hybrid region.

UI Access	Double-click on an assignment under Hybrid Regions in project history tree.		
Parameters	Name	Type	Description
	<HybridRegionName>	String	Name of the hybrid region to be edited.
	<HybridRegionArray>	Array	Structured array. <pre>Array("NAME:<new name of hybrid region>", "Type:=", <string one of "IE", "PO" or "SBR">, "IsLinkedRegion:=", <boolean>)</pre>
Return Value	None.		

Python Syntax	EditHybridRegion(<HybridRegionName>, <HybridRegionArray>)		
Python Example	<pre>oModule.EditHybridRegion("Hybrid1" ["NAME:Hybrid1After", "Type:=", "IE", "IsLinkedRegion:=", False])</pre>		

VB Syntax	EditHybridRegion <HybridRegionName>, <HybridRegionArray>		
VB Example	oModule.EditHybridRegion "Hybrid1", _		

	<pre>Array("NAME:Hybrid1After", _ "Type:=", "IE", _ "IsLinkedRegion:=", false)</pre>

EditImpedance

Modifies an impedance boundary definitions.

UI Access	Double-click on an impedance under Boundaries in the project history tree.		
Parameters	Name	Type	Description
	<BoundaryName>	String	Name of the impedance boundary to be edited.
	<ImpedanceArray>	Array	Structured array. Array("NAME:<New name>", "Resistance:=", <value>, "Reactance:=", <value>, "InfGroundPlane:=", <boolean>)
Return Value	None.		

Python Syntax	EditImpedance(<BoundaryName>, <ImpedanceArray>)
Python Example	<pre>oModule.AssignImpedance("Imped1" ["NAME:Imped1After", "Resistance:=", "50",</pre>

	<pre>"Reactance:=", "50", "InfGroundPlane:=", False])</pre>

VB Syntax	EditImpedance <BoundaryName>, <ImpedanceArray>
VB Example	<pre>oModule.AssignImpedance "Imped1", _ Array("NAME:Imped1After", _ "Resistance:=", "50", _ "Reactance:=", "50", _ "InfGroundPlane:=", false)</pre>

EditIncidentWave

Modifies an incident wave excitation.

UI Access	Double-click the excitation in the project tree to modify its settings.		
Parameters	Name	Type	Description
	<BoundaryName>	String	Name of the incident wave excitation to be edited.
	<IncidentWaveArray>	Array	Structured array.
			<pre>Array("NAME:<BoundName>", "IsCartesian:=", <boolean>, "EoX:=", <value>, "EoY:=", <value>,</pre>

			<pre>"EoZ:=", <value>, "kX:=", <value>, "kY:=", <value>, "kZ:=", <value> "PhiStart:=", <value>, "PhiStop:=", <value>, "PhiPoints:=", <int>, "ThetaStart:=", <value>, "ThetaStop:=", <value>, "ThetaPoints:=", <int>, "EoPhi:=", <value>, "EoTheta:=", <value>, "IsPropagating:=", <boolean>, "IsEvanescent:=", <boolean>, "IsEllipticallyPolarized:=", <boolean>) IsCartesian</pre> If true, provide the EoX, EoY, EoZ, kX, kY, kZ parameters. If false, provide the PhiStart, PhiStop, PhiPoints, ThetaStart, ThetaStop, ThetaPoints, EoPhi, EoTheta parameters.
Return Value	None.		

Python Syntax	EditIncidentWave(<BoundaryName>, <IncidentWaveArray>)
Python Example	<pre>oModule.EditIncidentWave("IncWave1", ["NAME:IncWave1After", "IsCartesian:=", True, "EoX:=", "1", "EoY:=", "0", "EoZ:=", "0", "kX:=", "0", "kY:=", "0", "kZ:=", "1", "IsPropagating:=", True, "IsEvanescent:=", False, "IsEllipticallyPolarized:=", False])</pre>

VB Syntax	EditIncidentWave <BoundaryName>, <IncidentWaveArray>
VB Example	<pre>oModule.EditIncidentWave "IncWave1", _ Array("NAME:IncWave1After", _ "IsCartesian:=", true, _ "EoX:=", "1", "EoY:=", "0", "EoZ:=", "0", _ "kX:=", "0", "kY:=", "0", "kZ:=", "1", "IsPropagating:=", true, _ "IsEvanescent:=", false, _ "IsEllipticallyPolarized:=", false)</pre>

EditInteriorNearFieldSource

Edits an interior near field source as an external data file and associated coordinate system.

UI Access	HFSS > Boundaries > Assign > Linked Field>Near Field...		
Parameters	Name	Type	Description
	<ArgArray>	Array	Structured array. ("NAME:<Name of Near Field Source data>", "Type:=" , "Incident", "ExternalDataFile:=" , "<filePath>.and", "SourceCoordSystem:=" , "<CS_name>"
Return Value	None.		

Python Syntax	AssignInteriorNearFieldSource(<ArgArray>)
Python Example	<pre>oDesign = oProject.SetActiveDesign("internal") oModule = oDesign.GetModule("BoundarySetup") oModule.AssignInteriorNearFieldSource(["NAME:ExtNFDData2", "Type:=" , "Incident", "ExternalDataFile:=" , "D:\\Ansoft\\Elliptical_EH.and",</pre>

	<pre>"SourceCoordSystem:=" , "Global"])</pre>

EditLatticePair

Modifies coupled Lattice Pair boundaries.

UI Access	Double-click on a lattice pair under Boundaries in the project history tree.		
Parameters	Name	Type	Description
	<BoundaryName>	String	Name of the lattice pair boundary to be edited.
	<BoundaryArray>	Array	Structured array. Array ("NAME:<BoundName>", "ReverseV:=", <boolean>, "PhaseDelay:=", <UseScanAngle InputPhaseDelay>, "Phi:=", <numdeg>, "Theta:=", <numdeg>, "Phase:=", <numdeg>)
Return Value	None.		

Python Syntax	EditLatticePair(<BoundaryName>, <BoundaryArray>)
Python Example	<pre>oModule.EditLatticePair("LatticePair1", ["NAME:LatticePair1After",</pre>

	<pre>"ReverseV:=", False, "PhaseDelay:=", "UseScanAngle", "Phi:=", "10deg", "Theta:=", "0deg"])</pre>

VB Syntax	EditLatticePair <BoundaryName>, <BoundaryArray>
VB Example	<pre>oModule.EditLatticePair "LatticePair1", _ Array("NAME:LatticePair1After", _ "ReverseV:=", false, _ "PhaseDelay:=", "UseScanAngle", _ "Phi:=", "10deg", _ "Theta:=", "0deg")</pre>

EditLayeredImp

Modifies a layered impedance boundary.

UI Access	Double-click on a layered impedance under Boundaries in the project history tree.		
Parameters	Name	Type	Description
	<BoundaryName>	String	Name of the layered impedance boundary.
	<LayeredImpArray>	Array	Structured array.

		<pre> Array ("NAME:<BoundName>", "Frequency:=", <value>, "Roughness:=", <value>, "IsInternal:=", <bool>, "IsTwoSided:=", <bool>, "IsShellElement:=", <bool>, <LayersArray>, "InfGroundPlane:=", <boolean>) </pre>
<LayersArray>	Array	Structured array. <pre> Array ("NAME:Layers", <OneLayerArray>, <OneLayerArray>, ...) </pre>
<OneLayerArray>	Array	Structured array. <pre> Array ("NAME:<LayerName>", "LayerType:=", <LayerType>, "Thickness:=", <value>, "Material:=", <string>) </pre> Thickness Thickness of the layer. Should be specified for all layers except the last layer. Material Material assigned on the layer. For the last layer, do not specify a material if the LayerType is "PerfectE" or "PerfectH".
<LayerName>	String	Specifies the layer number, such as "Layer1" or "Layer2"

	<table><tr><td><LayerType></td><td>String</td><td>Should be specified for the last layer only. Possible values: "Infinite", "PerfectE", or "PerfectH"</td></tr></table>	<LayerType>	String	Should be specified for the last layer only. Possible values: "Infinite", "PerfectE", or "PerfectH"
<LayerType>	String	Should be specified for the last layer only. Possible values: "Infinite", "PerfectE", or "PerfectH"		
Return Value	None.			

Python Syntax	EditLayeredImp(<BoundaryName>, <LayeredImpArray>)
Python Example	<pre>oModule.EditLayeredImp("Layered1", ["NAME:Layered1After", "Frequency:=", "10GHz", "Roughness:=", "0um", "IsTwoSided:=", True, "IsShellElement:=", True, ["NAME:Layers", ["NAME:Layer1", "Thickness:=", "1um", "Material:=", "vacuum"]], "InfGroundPlane:=", False])</pre>

VB Syntax	EditLayeredImp <BoundaryName>, <LayeredImpArray>
VB Example	<pre>oModule.EditLayeredImp "Layered1", _ Array("NAME:Layered1After", _ "Frequency:=", "10GHz", _</pre>

	<pre>"Roughness:=", "0um", _ "IsTwoSided:=", true, _ "IsShellElement:=", true, _ Array("NAME:Layers", Array("NAME:Layer1", _ "Thickness:=", "1um", "Material:=", "vacuum")), _ "InfGroundPlane:=", false)</pre>

EditLinkedImpedance

Edits a linked Impedance boundary.

UI Access	Double-click on a linked impedance under Boundaries in project history tree.		
Parameters	Name	Type	Description
	<BoundaryName>	String	Name of the linked impedance boundary to be edited.
	<LinkedImpArray>	Array	Structured array. Array("NAME:<BoundName>", "UseInfiniteGroundPlane:=", <boolean>, "UseShellElement:=" , <boolean>, <LinkDataArray>)
	<LinkDataArray>	Array	Structured array. Array("NAME:<LinkName>",&br/>"Project:=", <string file path>, "Product:=", <string product type>, "Design:=", <string linked source design name>,

			<code>"Soln:=", <string linked source solution name>, <array solution parameters>, "ForceSourceToSolve:=", <boolean>, "PreservePartnerSoln:=" , <boolean>, "PathRelativeTo:=" , <string target project name>)</code>
Return Value	None.		

Python Syntax	<code>EditLinkedImpedance(<BoundaryName>, <LinkedImpArray>)</code>
Python Example	<pre>oModule.EditLinkedImpedance("Linked1", ["NAME:Linked1After", "UseInfiniteGroundPlane:=", True, "UseShellElement:=", True, ["NAME:XLink", "Project:=", "C://temp/linkedproject.aedt", "Product:=", "HFSS", "Design:=", "Source_Project_Solver", "Soln:=", "1000MHz : LastAdaptive", ["NAME:Params", "xfactor:=", "1.2", "yfactor:=", "1.6"], "ForceSourceToSolve:=", True, "PreservePartnerSoln:=", False,</pre>

	<pre>"PathRelativeTo:=", "TargetProject"]])</pre>

VB Syntax	EditLinkedImpedance <BoundaryName>, <LinkedImpArray>
VB Example	<pre>oModule.EditLinkedImpedance "Linked1", _ Array("NAME:Linked1After", _ "UseInfiniteGroundPlane:=", true, _ "UseShellElement:=", true, _ Array("NAME:XLink", _ "Project:=", "C://temp/linkedproject.aedt", _ "Product:=", "HFSS", _ "Design:=", "Source_Project_Solver", _ "Soln:=", "1000MHz : LastAdaptive", _ Array("NAME:Params", "xfactor:=", "1.2", "yfactor:=", "1.6"), _ "ForceSourceToSolve:=", true, _ "PreservePartnerSoln:=", false, _ "PathRelativeTo:=", "TargetProject"))</pre>

EditLinkedRegion

Edits a Linked Region.

UI Access	N/A

Parameters	Name	Type	Description
	<BoundaryName>	String	Name of linked region.
	<LinkedRegionArray>	Array	Structured array. Array("NAME:<New name>", "Type:=" , <string, one of "IE", "PO" or "SBR">, "IsLinkedRegion:=" , <boolean, true for linked region>)
Return Value	None.		

Python Syntax	EditLinkedRegion(<BoundaryName>, <LinkedRegionArray>)
Python Example	<pre>oModule.EditLinkedRegion("Linked1", ["NAME:Linked1After", "Type:=", "PO", "IsLinkedRegion:=", True])</pre>

VB Syntax	EditLinkedRegion <BoundaryName>, <LinkedRegionArray>
VB Example	<pre>oModule.EditLinkedRegion "Linked1", _ Array("NAME:Linked1After", _ "Type:=", "PO", _</pre>

	"IsLinkedRegion:=", true)

EditLumpedPort

Modifies a lumped port.

UI Access	Double-click the excitation in the project tree to modify its settings.		
Parameters	Name	Type	Description
	<BoundaryName>	String	Name of lumped port excitation to be edited.
	<LumpedPortArray>	Array	Structured array. Array ("NAME:<BoundName>", "RenormalizeAllTerminals:=", <boolean>, "DoDeembed:=", <boolean>, <ModesArray>, "ShowReporterFilter:=", <boolean>, "ReporterFilter:=", <array of boolean>, "Impedance:=" , <value>)
	<ModesArray>	Array	Structured array. Array ("NAME:<ModesArrayName>", <OneModeArray>, <OneModeArray>,...)
	<OneModeArray>	Array	Structured array. Array ("NAME:<ModeName>", "ModeNum:=", <integer>, "UseIntLine:=", <boolean>, <IntegerationLineArray>,

			"AlignmentGroup:=", <integer, group id>, "CharImp:=", <string, characteristic impedance>, "RenormImp:=", <value, renormalize impedance to>)
	< <i>IntegrationLineArray</i> >	Array	Structured array. Array("NAME:<LineName>", "Coordinate System:=", <string, relative coordinate system>, "Start:=", <array, start location coordinates>, "End:=", <array, end location coordinates>)
Return Value	None.		

Python Syntax	EditLumpedPort(< <i>BoundaryName</i> >, < <i>LumpedPortArray</i> >)
Python Example	<pre>oModule.EditLumpedPort("LumpedPort1", ["NAME:LumpedPort1After", "DoDeembed:=", False, "RenormalizeAllTerminals:=", True, ["NAME:Modes", ["NAME:Model", "ModeNum:=", 1, "UseIntLine:=", True,</pre>

	<pre>["NAME:IntLine", "Coordinate System:=", "Global", "Start:=", ["-0.4mm", "-1mm", "0.8mm"], "End:=" , ["-0.3mm", "-1.2mm", "0.8mm"]], "AlignmentGroup:=", 0, "CharImp:=", "Zpi", "RenormImp:=", "50ohm"]], "ShowReporterFilter:=", False, "ReporterFilter:=", [True], "Impedance:=", "50ohm"])</pre>

VB Syntax	EditLumpedPort <BoundaryName>, <LumpedPortArray>
VB Example	<pre>oModule.EditLumpedPort "lp2", _ Array("NAME:lp2After", "RenormalizeAllTerminals:=", true, "DoDeembed:=", true, _ Array("NAME:Modes", Array("NAME:Model", "ModeNum:=", 1, "UseIntLine:=", true, _ Array("NAME:IntLine", "Start:=", Array("120mm", "50mm", "40mm"), _ "End:=", Array("120mm", "50mm", "120mm")), "CharImp:=", "Zpi")), _ "ShowReporterFilter:=", false, "ReporterFilter:=", Array(true), _ "FullResistance:=", "50ohm", "FullReactance:=", "0ohm")</pre>

EditLumpedRLC

Modifies a lumped RLC boundary.

UI Access	Double-click the boundary in the project tree to modify its settings.		
Parameters	Name	Type	Description
	<BoundaryName>	String	Name of the boundary to be edited.
	<LumpedRLCArray>	Array	Structured array. Array ("NAME:<NewBoundName>", "RLC Type:=", <"Parallel" "Serial" >, "UseResist:=", <boolean>, "Resistance:=", <value>, "UseInduct:=", <boolean>, "Inductance:=", <value>, "UseCap:=", <boolean>, "Capacitance:=", <value>, <CurrentLineArray>)
	<CurrentLineArray>	Array	Structured array. Array ("NAME:CurrentLine", "Start:=", <LineEndPoint>, "End:=", <LineEndPoint>)
Return Value	None.		

Python Syntax	EditLumpedRLC(<BoundaryName>, <LumpedRLCArray>)
Python Example	<pre>oModule.AssignLumpedRLC("LumpRLC1", ["NAME:LumpRLC1After", ["NAME:CurrentLine", "Start:=", ["0.15mm", "-0.2mm", "0mm"], "End:=", ["0.15mm", "0.6mm", "0mm"]], "RLC Type:=", "Parallel", "UseResist:=", True, "Resistance:=", "100ohm", "UseInduct:=", True, "Inductance:=", "10nH", "UseCap:=", True, "Capacitance:=", "10pF"])</pre>

VB Syntax	EditLumpedRLC <BoundaryName>, <LumpedRLCArray>
VB Example	<pre>oModule.EditLumpedRLC "LumpRLC1", _ Array("NAME:LumpRLC1After", _ "RLC Type:=" , " Parallel", _ "UseResist:=", true, _</pre>

	<pre>"Resistance:=", "10Ohm", _ "UseInduct:=", true, _ "Inductance:=", "10nH", _ "UseCap:=", true, _ "Capacitance:=", "10pF", _ Array("NAME:CurrentLine", _ "Start:=", Array(-0.4, -1.2, -1.6), _ "End:=", Array(-0.4, -1.2, 0)))</pre>

EditMagneticBias

Modifies a magnetic bias excitation.

UI Access	Double-click on the magnetic bias under Excitations in the project tree.		
Parameters	Name	Type	Description
	<BoundaryName>	String	Name of the magnetic bias excitation to be edited.
	<MagneticBiasArray>	Array	Structured array. Array("NAME:<NewBoundName>", "IsUniformBias:=", <boolean>, "Bias:=", <value>, "XAngle:=", <value>, "YAngle:=", <value>, "ZAngle:=", <value>,

			"Project:=", <string> IsUniformBias If true, supply the Bias, XAngle, YAngle, and ZAngle parameters. If false, supply the Project parameter.
Return Value	None.		

Python Syntax	EditMagneticBias(<BoundaryName>, <MagneticBiasArray>)
Python Example	<pre>oModule.EditMagneticBias("MagBias1", ["NAME:MagBias1After", "IsUniformBias:=", True, "Bias:=", "1", "XAngle:=", "10deg", "YAngle:=", "10deg", "ZAngle:=", "10deg"])</pre>

VB Syntax	EditMagneticBias <BoundaryName>, <MagneticBiasArray>
VB Example	<pre>oModule.EditMagneticBias "MagBias1", _ Array("NAME:MagBias1After", _ "IsUniformBias:=", true, _ "Bias:=", "1", _</pre>

	<pre>"XAngle:=", "10deg", _ "YAngle:=", "10deg", _ "ZAngle:=", "10deg")</pre>

EditMultipactionChargeRegion

Modifies settings for a Multipaction Charge Region.

UI Access	Double-click on the excitation in project tree to edit its settings.		
Parameters	Name	Type	Description
	<BoundaryName>	String	Name of the multipaction charge region to be edited.
	<ChargeRegionArray>	Array	Structured array. Array("NAME:MultipactionChargeRegion1", "NumParticles:=", "<integer, number of particles>", "ParticleCharge:=", "<num><unit>", "ParticleMass:=", "<num><unit>", "Vx:=", "<num><unit>", "Vy:=", "<num><unit>", "Vz:=", "<num><unit>")
Return Value	None.		

Python Syntax	EditMultipactionChargeRegion (<BoundaryName>, <ChargeRegionArray>)
Python Example	<pre>oModule.EditMultipactionChargeRegion("MultipactionChargeRegion1", ["NAME:MultipactionChargeRegion1After", "NumParticles:=", "100", "ParticleCharge:=", "-1.60217662e-19Coulomb", "ParticleMass:=", "9.10938356e-31kg", "Vx:=", "0.1m_per_sec", "Vy:=", "0.3m_per_sec", "Vz:=", "0.15m_per_sec"])</pre>

VB Syntax	EditMultipactionChargeRegion <BoundaryName>, <ChargeRegionArray>
VB Example	<pre>oModule.EditMultipactionChargeRegion "MultipactionChargeRegion1", _ Array("NAME:MultipactionChargeRegion1After", _ "NumParticles:=", "100", _ "ParticleCharge:=", "-1.60217662e-19Coulomb", _ "ParticleMass:=", "9.10938356e-31kg", _ "Vx:=", "0.1m_per_sec", "Vy:=", "0.3m_per_sec", "Vz:=", "0.15m_per_sec")</pre>

EditMultipactionDCBias

Edits settings for a multipaction DC bias boundary.

UI Access	Double-click on the excitation in the project tree to edit its settings.		
Parameters	Name	Type	Description
	<BoundaryName>	String	Name of the multipaction DC bias to be edited.
	<DCBiasArray>	Array	Structured array. <pre>Array (NAME:MultipactionDCBias1", "UniformE:=", <boolean>, "Ex:=", "<value><unit>", "Ey:=", "<value><unit>", "Ez:=", "<value><unit>", <LinkedField>, "UniformH:=" , <boolean>, "Hx:=", "<value><unit>", "Hy:=", "<value><unit>", "Hz:=", "<value><unit>", <LinkedField>)</pre> UniformE or Uniform H: • True - provide filed values. • False - provide field through linked Maxwell analysis.
	<LinkedField>	Array	Structured array. <pre>Array("NAME:<EField HField>", "Project:=", <string, path to linked project</pre>

			<pre>file>, "Product:=", "Maxwell", "Design:=", <string, name of source design>, "Soln:=", <string, name of linked solution>, <Parameters for linked solution>, "ForceSourceToSolve:=", <boolean>, "PreservePartnerSoln:=", <boolean>, "PathRelativeTo:=" , "TargetProject")</pre>
Return Value	None.		

Python Syntax	EditMultipactionDCBias(<BoundaryName>, <DCBiasArray>)
Python Example	<pre>oModule.EditMultipactionDCBias("MultipactionDCBias1", ["NAME:MultipactionDCBias1After", "UniformE:=", False, ["NAME:EField", "Project:=", "C://projects/Maxwell/HallSensor.aedt", "Product:=", "Maxwell", "Design:=", "3_Maxwell3D", "Soln:=", "Setup1 : Transient", ["NAME:Params",</pre>

```

"angle:=", "0deg",
"cell_spacing:=", "2.5mm",
"die_thickness:=", "2mm",
"gap:=", "1.5mm",
"magnet_center_y:=", "0mm",
"magnet_center_z:=", "0mm",
"magnet_x:=", "3mm",
"magnet_y:=", "6mm",
"magnet_z:=", "5mm",
"move_x:=", "-1.5mm",
"sensor_offset_x:=", "0mm",
"sensor_offset_y:=", "0mm",
"sensor_offset_z:=", "0mm",
"sensor_pitch:=", "0deg",
"sensor_roll:=", "0deg",
"sensor_yaw:=", "0deg",
"target_dia:=", "80mm"

],
"ForceSourceToSolve:=", False,
"PreservePartnerSoln:=", False,

```

	<pre> "PathRelativeTo:=", "TargetProject"], "UniformH:=", True, "Hx:=", "0A_per_m", "Hy:=", "0A_per_m", "Hz:=", "0A_per_m"])</pre>

VB Syntax	EditMultipactionDCBias <BoundaryName>, <DCBiasArray>
VB Example	<pre>oModule.EditMultipactionDCBias "MultipactionDCBias1", _ Array("NAME:MultipactionDCBias1After", _ "UniformE:=", true, _ "Ex:=", "0V_per_meter", "Ey:=", "0V_per_meter", "Ez:=", "0V_per_meter", _ "UniformH:=", true, _ "Hx:=", "0A_per_m", "Hy:=", "0A_per_m", "Hz:=", "0A_per_m")</pre>

EditMultipactionSEE

Edits a Secondary Electron Emission (SEE) boundary settings.

UI Access	Double-click on the boundary in the project tree to edit its settings.		
Parameters	Name	Type	Description
	<BoundaryName>	String	Name of the boundary to be edited.

	<code><SEESArray></code>	Array	Structured array. Array("NAME:<NewSEEName>", "AlphaMax:=", <value>, "Alpha0:=", <value>, "E0:=", <value>, "E1:=", <value>, "E2:=", <value>, "Em:=", <value>, "DielectricSurface:=", <boolean>)
Return Value	None.		

Python Syntax	EditMultipactionSEE(<BoundaryName>, <SEESArray>)
Python Example	<pre>oModule.EditMultipactionSEE("SEE1", ["NAME:SEE1After", "AlphaMax:=", "2.25", "Alpha0:=", "0", "E0:=", "12.5", "E1:=", "25", "E2:=", "5000", "Em:=", "175", "DielectricSurface:=", false])</pre>

VB Syntax	EditMultipactionSEE <BoundaryName>, <SEEArray>
VB Example	<pre>oModule.EditMultipactionSEE "SEE1", _ Array("NAME:SEE1After", _ "AlphaMax:=", "2.25", "Alpha0:=", "0", _ "E0:=", "12.5", "E1:=", "25", "E2:=", "5000", _ "Em:=", "175", "DielectricSurface:=", false)</pre>

EditRFDischargeDCBias

Edits settings for an RF discharge DC bias excitation.

UI Access	Double-click on the excitation in the project tree to edit its settings.		
Parameters	Name	Type	Description
	<BoundaryName>	String	Name of the excitation to be edited.
	<DCBiasParameters>	Array	Structured array. Array("NAME:<DCBiasName>", "UniformH:=", <boolean>, "Hx:=", "<real>A_per_m", "Hy:=", "<real>A_per_m", "Hz:=", "<real>A_per_m")
Return Value	None.		

Python Syntax	EditRFDISchargeDCBias(<BoundaryName>, <DCBiasParameters>)
Python Example	<pre>oModule.EditRFDISchargeDCBias("DCBias1", ["NAME:RFDISchargeDCBias1", "UniformH:=", True, "Hx:=", "795774.71546A_per_m", "Hy:=", "0A_per_m", "Hz:=", "0A_per_m"])</pre>

VB Syntax	EditRFDISchargeDCBias <BoundaryName>, <DCBiasParameters>
VB Example	<pre>oModule.EditRFDISchargeDCBias "DCBias1", _ Array("NAME:RFDISchargeDCBias1", _ "UniformH:=", True, _ "Hx:=", "795774.71546A_per_m", "Hy:=", "0A_per_m", "Hz:=", "0A_per_m")</pre>

EditPerfectE

Modifies a perfect E boundary.

UI Access	Double-click on the boundary in project tree to edit its settings.

Parameters	Name	Type	Description
	<BoundaryName>	String	Name of the boundary to be edited.
	<PerfectEArray>	Array	Structured array. Array("NAME:<NewBoundName>", "InfGroundPlane:=", <boolean>)
Return Value	None.		

Python Syntax	EditPerfectE(<BoundaryName>, <PerfectEArray>)
Python Example	<pre>oModule.EditPerfectE("PerfE1", ["NAME:PerfE1After", "InfGroundPlane:=", False,])</pre>

VB Syntax	EditPerfectE <BoundaryName>, <PerfectEArray>
VB Example	<pre>oModule.EditPerfectE "PerfE1", _ Array("NAME:PerfE1After", _ "InfGroundPlane:=", false)</pre>

EditPerfectH

Modifies a perfect H boundary.

UI Access	Double-click on the boundary in the project tree to edit its settings.		
Parameters	Name	Type	Description
	<BoundaryName>	String	Name of the boundary to be edited.
	<PerfectHArray>	Array	Structured array. Array ("NAME: <NewBoundName>")
Return Value	None.		

Python Syntax	EditPerfectH(<BoundaryName>, <PerfectHArray>)
Python Example	<pre>oModule.EditPerfectH("PerfH1", ["NAME:PerfH1After"])</pre>

VB Syntax	EditPerfectH <BoundaryName>, <PerfectHArray>
VB Example	<pre>oModule.EditPerfectH "PerfH1", _ Array("NAME:PerfH1After")</pre>

EditRadiation

Modifies a radiation boundary.

UI Access	Double-click the boundary in the project tree to modify its settings.		
Parameters	Name	Type	Description
	<BoundaryName>	String	Name of the boundary to be edited.

	<i><RadiationArray></i>	Array	Structured array. <pre> Array("NAME:<BoundName>", "IsIncidentField:=", <boolean>, "IsEnforcedHField:=", <boolean>, "IsEnforcedEField:=", <boolean>, "IsFssReference:=", <boolean>, "IsForPML:=", <boolean>, "UseAdaptiveIE:=", <boolean>, "IncludeInPostproc:=", <boolean>) </pre>
Return Value	None.		

Python Syntax	<code>EditRadiation(<BoundaryName>, <RadiationArray>)</code>
Python Example	<pre> oModule.EditRadiation("Rad1", ["NAME:Rad1After", "IsIncidentField:=", True, "IsEnforcedField:=", False, "IsFssReference:=", False, "IsForPML:=", False, "UseAdaptiveIE:=", False, "IncludeInPostproc:=", True]) </pre>

	)

VB Syntax	EditRadiation <BoundaryName>, <RadiationArray>
VB Example	<pre>oModule.EditRadiation "Rad1", _ Array("NAME:Rad1After", _ "IsIncidentField:=", true, _ "IsEnforcedField:=", false, _ "IsFssReference:=", false, _ "IsForPML:=", false, _ "UseAdaptiveIE:=", false, _ "IncludeInPostproc:=", true)</pre>

EditSymmetry

Modifies a symmetry boundary.

UI Access	Double-click the symmetry boundary in the project tree to edit its settings.		
Parameters	Name	Type	Description
	<BoundaryName>	String	Name of the symmetry boundary to be edited.
	<SymmetryArray>	Array	Structured array. Array("NAME:<BoundName>", "IsPerfectE:=", <boolean>)

Return Value	None.

Python Syntax	EditSymmetry(<BoundaryName>, <SymmetryArray>)
Python Example	<pre>oModule.EditSymmetry("Sym1", ["NAME:Sym1After", "IsPerfectE:=", True])</pre>

VB Syntax	EditSymmetry <BoundaryName>, <SymmetryArray>
VB Example	<pre>oModule.EditSymmetry "Sym1", _ Array("NAME:Sym1After", _ "IsPerfectE:=", true)</pre>

EditTerminal

Modifies properties of a terminal

UI Access	Edit Properties for a selected terminal.		
Parameters	Name	Type	Description
	<TerminalName>	String	Name of the terminal to be edited.
	<TerminalArray>	Array	Structured array. <pre>Array("NAME: <TerminalName>", "ParentBndID:=", "<PortName>",</pre>

	<table><tr><td></td><td></td><td>"TerminalResistance:=," <Value and units of resistance>")</td></tr></table>			"TerminalResistance:=," <Value and units of resistance>")
		"TerminalResistance:=," <Value and units of resistance>")		
Return Value	None.			

Python Syntax	EditTerminal (<TerminalName>, <TerminalArray>)
Python Example	<pre>oModule.EditTerminal("Rectangle2_T1", ["NAME:Rectangle2_T1", "ParentBndID:=", "WavePort1", "TerminalResistance:=", "75ohm"])</pre>

VB Syntax	EditTerminal <TerminalName>, <TerminalArray>
VB Example	<pre>oModule.EditTerminal "Rectangle2_T1", _ Array("NAME:Rectangle2_T1", _ "ParentBndID:=", "WavePort1", "TerminalResistance:=", "75ohm")</pre>

EditVoltage

Modifies a voltage source.

UI Access	Double-click the excitation in the project tree to modify its settings.

Parameters	Name	Type	Description
	<BoundaryName>	String	Name of the voltage source to be edited.
	<VoltageArray>	Array	Structured array. Array("NAME:<BoundName>", "Voltage:=", <value>, <DirectionArray>)
	<DirectionArray>	Array	Structured array. Array("NAME:Direction", "Start:=", <LineEndPoint>, "End:=", <LineEndPoint>)
Return Value	None.		

Python Syntax	EditVoltage(<BoundaryName>, <VoltageArray>)
Python Example	<pre>oModule.EditVoltage("Voltage1", ["NAME:Voltage1After", "Voltage:=", "1000mV", ["NAME:Direction", "Start:=", [-0.4, -1.2, 0], "End:=", [-1.4, -1.2, 0]])</pre>

VB Syntax	EditVoltage <BoundaryName>, <VoltageArray>
VB Example	<pre>oModule.EditVoltage "Voltage1", _ Array("NAME:Voltage1After", _ "Voltage:=", "1000mV", _ Array("NAME:Direction", _ "Start:=", Array(-0.4, -1.2, 0), _ "End:=", Array(-1.4, -1.2, 0)))</pre>

EditVoltageDrop

Use: Edits a voltage drop excitation.

Command: Double-click the excitation in the project tree to edit it.

Syntax:EditVoltageDrop <BoundName> <VoltageDropArray>

Return Value: None

EditWavePort

Modifies a wave port.

UI Access	Double-click the excitation in the project tree to modify its settings.		
Parameters	Name	Type	Description
	<BoundaryName>	String	Name of the wave port to be edited.
	<WavePortArray>	Array	Structured array.

		<pre> Array ("NAME:<BoundName>", "NumModes:=", <integer>, "PolarizeEField:=", <boolean>, "DoDeembed:=", <boolean>, "DeembedDist:=", <value>, "DoRenorm:=", <boolean>, "RenormValue:=", <value>, <ModesArray>, "TerminalIDList:=", <TerminalsArray>) </pre> NumModes Number of modes for modal problems. Number of terminals for terminal problems.
<ModesArray>	Array	Structured array. <pre> Array ("NAME:Modes", <OneModeArray>, <OneModeArray>, ...) </pre>
<OneModeArray>	Array	Structured array. <pre> Array ("NAME:<ModeName>", "ModeNum:=", <integer>, "UseIntLine:=", <boolean>, <IntLineArray>) </pre>
<IntLineArray>	Array	Structured array. <pre> Array ("NAME:IntLine", </pre>

			<code>"Start:=", <LineEndPoint>, "End:=", <LineEndPoint>, "CharImp:=", <string, Characteristic impedance of the mode. Possible values are "Zpi", "Zpv", or "Zvi">)</code>
Return Value	None.		

Python Syntax	<code>EditWavePort(<BoundaryName>, <WavePortArray>)</code>
Python Example	<pre>oModule.EditWavePort("WavePort1", ["NAME:WavePort1After", "NumModes:=", 2, "PolarizeEField:=", False, "DoDeembed:=", True, "DeembedDist:=", "10mil", "DoRenorm:=", True, "RenormValue:=", "50Ohm", ["NAME:Modes", ["NAME:Model", "ModeNum:=", 1, "UseIntLine:=", True,</pre>

	<pre>["NAME:IntLine", "Start:=", [-0.4, -1.2, 0), "End:=", [-1.4, 0.4, 0]], "CharImp:=", "Zpi"), ["NAME:Mode2", "ModeNum:=", 2, "UseIntLine:=", False]]])</pre>

VB Syntax	EditWavePort <BoundaryName>, <WavePortArray>
VB Example	<pre>oModule.EditWavePort "WavePort1", _ Array("NAME:WavePort1After", _ "NumModes:=", 2, _ "PolarizeEField:=", false, _ "DoDeembed:=", true, _ "DeembedDist:=", "10mil", _ "DoRenorm:=", true, _ "RenormValue:=", "50Ohm", Array("NAME:Modes", _ Array("NAME:Model", _</pre>

	<pre>"ModeNum:=", 1, _ "UseIntLine:=", true, _ Array("NAME:IntLine", _ "Start:=", Array(-0.4, -1.2, 0), _ "End:=", Array(-1.4, 0.4, 0)), _ "CharImp:=", "Zpi"), _ Array("NAME:Mode2", _ "ModeNum:=", 2, _ "UseIntLine:=", false)))</pre>

LumpedPortToCircuitPort

Converts a lumped port to a circuit port for a driven terminal or driven modal design in HFSS.

UI Access	Right-click on a lumped port excitation in the project tree, then select Convert to Circuit Port .		
Parameters	Name	Type	Description
	<PortName>	String	Name of the lumped port to be converted.
Return Value	None.		

Python Syntax	LumpedPortToCircuitPort(<PortName>)
Python Example	oModule.LumpedPortToCircuitPort("Port1")

VB Syntax	LumpedPortToCircuitPort <PortName>
VB Example	<code>oModule.LumpedPortToCircuitPort "Port1"</code>

SetAllHybridRegionsToOneWayCoupled

Specifies one-way coupling for all hybrid regions.

UI Access	Right-click Hybrid Regions > Set Regions > All Regions Are One Way Coupled.
Parameters	None.
Return Value	None.

Python Syntax	SetAllHybridRegionsToOneWayCoupled()
Python Example	<code>oModule.SetAllHybridRegionsToOneWayCoupled()</code>

VB Syntax	SetAllHybridRegionsToOneWayCoupled
VB Example	<code>oModule.SetAllHybridRegionsToOneWayCoupled</code>

SetAllHybridRegionsToTwoWayCoupled

Specifies two-way coupling for all hybrid regions.

UI Access	Right-click Hybrid Regions > Set Regions > All Regions Are Two Way Coupled.
Parameters	None.
Return Value	None.

Python Syntax	SetAllHybridRegionsToTwoWayCoupled()
Python Example	<code>oModule.SetAllHybridRegionsToTwoWayCoupled()</code>

VB Syntax	SetAllHybridRegionsToTwoWayCoupled
VB Example	<code>oModule.SetAllHybridRegionsToTwoWayCoupled</code>

SetHybridRegionCoupledGroup

Use: To set coupling for hybrid regions.

Command: **HFSS>HybridRegions>Set Coupling**

Syntax: SetHybridRegionCoupledGroup <value>

Return Value: None

Parameters: <value>

Type: <string>

"OneWayCoupled" or "TwoWayCoupled" for all regions or "Advanced", Array("One Way:= <hybridregionname>, Array(Two Way:=", Array("<hybridregionname>" ,)))

VB Example:

Setting All Hybrid Regions to One Way Coupled

```
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("Project35")
Set oDesign = oProject.SetActiveDesign("HFSSDesign1")
Set oModule = oDesign.GetModule("BoundarySetup")
oModule.SetHybridRegionCoupledGroup "OneWayCoupled"
```

Setting All Hybrid Regions to Two Way Coupled

```
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("Project35")
Set oDesign = oProject.SetActiveDesign("HFSSDesign1")
Set oModule = oDesign.GetModule("BoundarySetup")
oModule.SetHybridRegionCoupledGroup "TwoWayCoupled"
```

Setting Hybrid Region Groupings

```
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("Project35")
Set oDesign = oProject.SetActiveDesign("HFSSDesign1")
Set oModule = oDesign.GetModule("BoundarySetup")
oModule.SetHybridRegionCoupledGroup "Advanced", Array("One Way:=", Array("Two Way:=", Array( _
"Hybrid1", "Hybrid2")))
```

SetHybridRegionsCoupling

Sets coupling for hybrid regions.

UI Access	Right-click Hybrid Regions > Set Couling > Advanced .		
Parameters	Name	Type	Description
	<CouplingArray>	Array	Structured array. <pre>Array("One Way:=", <array of region names>, "Two Way:=", <array of region names>) </pre> Note: At least two regions must be specified in a group.
Return Value	None		

Python Syntax	SetHybridRegionsCoupling(<CouplingArray>)
Python Example	<pre>oModule.SetHybridRegionsCoupling(["One Way:=", ["FE-BI1", "FE-BI2"], "Two Way:=", ["FE-BI3", "FE-BI4", "FE-BI5"]]) </pre>

VB Syntax	SetHybridRegionsCoupling <CouplingArray>
VB Example	<pre>oModule.SetHybridRegionsCoupling Array(_ </pre>

	<pre>"One Way:=", Array("FE-BI1", "FE-BI2"), _ "Two Way:=", Array("FE-BI3", "FE-BI4", "FE-BI5"))</pre>

SetScatteredFieldFormulation

Sets incident wave to scattered field formulation. **Note:** Incident field formulation is only applicable in a driven terminal/modal design with incident wave.

UI Access	Right-click on Excitations , then select Set Incident Field Formulation > Scattered .
Parameters	None.
Return Value	None.

Python Syntax	SetScatteredFieldFormulation()
Python Example	<code>oModule.SetScatteredFieldFormulation()</code>

VB Syntax	SetScatteredFieldFormulation
VB Example	<code>oModule.SetScatteredFieldFormulation</code>

SetSBRCreepingWaveSettings

Sets creeping wave settings for SBR+ solutions.

UI Access	Right-click on Hybrid Regions , then select Creeping Wave Settings...

	Name	Type	Description
	<CWaveArray>	Array	Structured array. <pre>Array("NAME:SBRCreepingWaveSettings", "CWRaySampleDensity:=" , <value>, "CWRayCutoffDb:=" , <value>, "CWCurvatureSensitivity:=", <value>, "CWAngularRayInterval:=", <value>)</pre>
Parameters			
Return Value	None.		

Python Syntax	SetSBRCreepingWaveSettings(<CWaveArray>)
Python Example	<pre>oModule.SetSBRCreepingWaveSettings(["NAME:SBRCreepingWaveSettings", "CWRaySampleDensity:=" , 10, "CWRayCutoffDb:=" , 40, "CWCurvatureSensitivity:=", 50, "CWAngularRayInterval:=", 2])</pre>

VB Syntax	SetSBRCreepingWaveSettings <CWaveArray>
VB Example	oModule.SetSBRCreepingWaveSettings _

	<pre>Array("NAME:SBRCreepingWaveSettings", _ "CWRaySampleDensity:=", 10, _ "CWRayCutoffDb:=", 40, _ "CWCurvatureSensitivity:=", 50, _ "CWAngularRayInterval:=", 2)</pre>

SetSBRTxRxSettings

Assigns Transmit and Receive assignments for antennas in SBR+ solutions.

UI Access	Right-click on Excitations > Select Tx/Rx...		
Parameters	Name	Type	Description
	<TxRxArray>	Array	Structured array. Array("NAME:SBRTxRxSettings", <AssignmentList>, <AssignmentList>,...)
	<AssignmentList>	Array	Structured array. Array("NAME:<ListName>", "Tx Antenna:=", <string, antenna components>, "Rx Antennas:=", <string, antenna components>)
Return Value	None.		

Python Syntax	SetSBRTxRxSettings(<TxRxArray>)

Python Example	<pre>oModule.SetSBRTxRxSettings(["NAME:SBRTxRxSettings", ["NAME:Tx/Rx List 0", "Tx Antenna:=", "cHorn1_1_p1", "Rx Antennas:=", "Beam1_1_p1,pHorn1_1_p1,sDipole1_1_p1"], ["NAME:Tx/Rx List 1", "Tx Antenna:=", "Beam1_1_p1", "Rx Antennas:=", "Beam1_1_p1,pHorn1_1_p1,sDipole1_1_p1"], ["NAME:Tx/Rx List 2", "Tx Antenna:=", "sDipole1_1_p1", "Rx Antennas:=", "Beam1_1_p1,pHorn1_1_p1,sDipole1_1_p1"]])</pre>

VB Syntax	SetSBRTxRxSettings <TxRxArray>
VB Example	oModule.SetSBRTxRxSettings Array("NAME:SBRTxRxSettings", _

	<pre>Array("NAME:Tx/Rx List 0", _ "Tx Antenna:=", "cHorn1_1_p1", _ "Rx Antennas:=", "Beam1_1_p1,pHorn1_1_p1,sDipole1_1_p1"), _ Array("NAME:Tx/Rx List 1", _ "Tx Antenna:=", "Beam1_1_p1", _ "Rx Antennas:=", "Beam1_1_p1,pHorn1_1_p1,sDipole1_1_p1"), _ Array("NAME:Tx/Rx List 2", _ "Tx Antenna:=", "sDipole1_1_p1", _ "Rx Antennas:=", "Beam1_1_p1,pHorn1_1_p1,sDipole1_1_p1"))</pre>

SetTerminalReferenceImpedances

Sets the reference impedance for all terminals within a specified port.

UI Access	HFSS > Excitations > Set Terminal Renormalizing Impedances...		
Parameters	Name	Type	Description
	<RefImpValue>	String	Impedance value with unit.
	<PortName>	String	Optional. Name of the port.
	<RenormalizeTerminals>	Boolean	Optional. • True - renormalize terminals. • False - do not renormalize terminals.
Return Value	None.		

Python Syntax	<code>SetTerminalReferenceImpedances(<RefImpValue>, <PortName>, <RenormalizeTerminals>)</code>
Python Example	<code>oModule.SetTerminalReferenceImpedances("50ohm", "", False)</code>

VB Syntax	<code>SetTerminalReferenceImpedances <RefImpValue>, <PortName>, <RenormalizeTerminals></code>
VB Example	<code>oModule.SetTerminalReferenceImpedances "50ohm", "", false</code>

SetTotalFieldFormulation

Sets incident wave to total field formulation. **Note:** Incident field formulation is only applicable in a driven terminal/modal design with incident wave.

UI Access	Right-click on Excitations , then select Set Incident Field Formulation > Total .
Parameters	None.
Return Value	None.

Python Syntax	<code>SetTotalFieldFormulation()</code>
Python Example	<code>oModule.SetTotalFieldFormulation()</code>

VB Syntax	<code>SetTotalFieldFormulation</code>
VB Example	<code>oModule.SetTotalFieldFormulation</code>

SwapCircuitPortDirection

Swaps the direction of a circuit port.

UI Access	N/A		
Parameters	Name	Type	Description
	<PortName>	String	Name of specified port.
Return Value	None.		

Python Syntax	SwapCircuitPortDirection(<PortName>)		
Python Example	oModule.SwapCircuitPortDirection("1")		

VB Syntax	SwapCircuitPortDirection <PortName>		
VB Example	oModule.SwapCircuitPortDirection "1"		

SwapLatticePair

Swaps the faces defined in a lattice pair.

UI Access	Right-click on a lattice pair in the project tree, then select Swap Faces .		
Parameters	Name	Type	Description
	<LatticePairName>	String	Name of specified lattice pair boundary to be edited.
Return Value	None.		

Python Syntax	SwapLatticePair(<LatticePairName>)		

Python Example	<code>oModule.SwapLatticePair("LatticePair1")</code>

VB Syntax	<code>SwapLatticePair <LatticePairName></code>
VB Example	<code>oModule.SwapLatticePair "LatticePair1"</code>

UnassignIERegions

Use: Unassign one or more IE Regions assigned to conducting objects.

Command: **Unassign IE Regions**

Syntax: UnassignIERegion Array (<"geometryName">)

Return Value: None

Parameters: <GeometryName>

Type: String

Name of one or more geometries assigned as an IE Region.

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("Project58")
Set oDesign = oProject.SetActiveDesign("HFSSDesign1")
Set oModule = oDesign.GetModule("BoundarySetup")
oModule.UnassignIERegions Array("Box1")
```

Mechanical Boundary, Contact, & Excitation Commands

Boundary and excitation commands for Mechanical designs are dependent upon the solution type (Modal, Thermal, or Structural). Mechanical boundary and excitation commands should be executed by the oModule object. For example, use oModule.AssignFrictionlessSupport to assign a frictionless support boundary to a model for a Mechanical – Modal solution.

See the following sections to view the parameters used in the assignment and editing of each boundary and excitation:

Modal Boundary Commands:

- [AssignFixedSupport and EditFixedSupport](#)
- [AssignFrictionlessSupport and EditFrictionlessSupport](#)
- [AssignCylindricalSupport and EditCylindricalSupport](#)

Thermal Boundary Commands:

- [AssignConvection and EditConvection](#)
- [AssignTemperature and EditTemperature](#)
- [AssignRotatingFluid and EditRotatingFluid](#)

Thermal Contact Commands:

- [AssignContact and EditContact](#)

Thermal Excitation Commands:

- [AssignEMLoss and EditEMLoss](#)
- [AssignHeatFlux and EditHeatFlux](#)
- [AssignHeatGeneration and EditHeatGeneration](#)

Thermal Initial Condition Commands:

- [AssignInitialTemperature and EditInitialTemperature](#) (Transient Thermal solutions only)

Structural Boundary Commands:

- [AssignFixedSupport and EditFixedSupport](#)
- [AssignFrictionlessSupport and EditFrictionlessSupport](#)
- [AssignCylindricalSupport and EditCylindricalSupport](#)

Structural Excitation Commands:

- [AssignDisplacement and EditDisplacement](#)
- [AssignForce and EditForce](#)
- [AssignPressure and EditPressure](#)
- [AssignThermalCondition and EditThermalCondition](#)

AssignContact and EditContact**AssignContact**

This command creates contact in a Mechanical–Thermal design for the purpose of defining thermal resistance between adjacent objects.

UI Access	Mechanical > Contacts > Assign > Contact

Parameters	Name	Type	Description
	<NAME>	string	Contact name
	<Faces>	list	Faces included in the boundary condition assignment (integer list)
	<Objects>	list	Shell objects included in the boundary condition assignment (string list)
	<Resistance Type>	string	Specifies which one of the following five resistance parameters (*) to use:
	* <Thermal Conductance>	string	Total thermal contact conductance for selected assignment faces/objects (with units)
	* <Thermal Conductance per Area>	string	Thermal contact conductance per unit area (with units)
	* <Thermal Resistance>	string	Total thermal contact resistance for selected assignment faces/objects (with units)
	* <Thermal Impedance>	string	Total area-based thermal contact conductance for selected assignment faces/objects (with units)
	* <Thickness>	string	Thickness of the <Material> used to determine thermal contact conductivity (with unit)
	<Material>	string	Name of material (additional parameter for <Resistance Type> = "Thickness" only)
Return Value	None		

Python Syntax	AssignContact([<NAME>, <Faces>, <Objects>, <Resistance Type>, <Thermal Conductance Thermal Conductance per Area Thermal Resistance Thermal Impedance Thickness>, <Material>])
Python Example	oModule.AssignConvection([

Thermal Impedance	<pre> "NAME:Contact1" "Objects:=" "Faces:=" "Resistance Type:=" "Thermal Impedance:=" </pre>
Python Example Thickness	<pre> oModule.AssignConvection(["NAME:Contact1" "Objects:=" "Faces:=" "Resistance Type:=" "Thickness:=" "Material:=" </pre>

VB Syntax	<i>AssignContact(<NAME>, <Faces>, <Objects>, <Resistance Type>, <Thermal Conductance Thermal Conductance per Area Thermal Resistance Thermal Impedance Thickness>, <Material>)</i>
VB Example Thermal Conductance per Area	<pre> oModule.AssignContact Array("NAME:Contact1", "Objects:=", Array("Shell1"), _ "Faces:=" Array(40,63), "Resistance Type:=", "Thermal Conductance per Area", _ "Thermal Conductance per Area:=", "2.5e-4w_per_mm2_per_cel") </pre>

EditContact

Note:

In the above script examples, you can substitute "*Edit*" for "*Assign*" to modify a pre-existing contact instead of creating a new one. The script must identify the **NAME** of the existing contact within the *EditContact* command line. Then, you can change any of the setup parameters with the exception of the assignment *Faces* or *Objects*.

To change the assignment *Faces* or *Objects*, you must use the *ReassignBoundary* command, provide the excitation NAME, and specify the new Faces and/or Objects.

Script examples for editing a contact are given below. In these examples, the name, resistance type, and associated parameter value are being modified.

- **UI Access:** Double-click the contact in the Project Manager or right-click it and choose **Properties** from the shortcut menu.
- **Python Example:**

```
oModule.EditContact("Contact1",  
    [  
        "NAME:Contact_Top",  
        "Resistance Type:=" , "Thickness"  
        "Thickness:="      , "2.5mil",  
        "Material:="       , "polyethylene"  
    ] )
```

- **VB Example:**

```
oModule.EditContact "Contact1", Array("NAME:Contact_Top", "Resistance Type:=", _  
    "Thickness", "Thickness:=", "3mil", "Material:=", "polyethylene")
```

AssignConvection and EditConvection

AssignConvection

This command creates a convection boundary in a Mechanical–Thermal design and optionally sets up a link to import heat transfer coefficients from an Icepak design.

UI Access	Mechanical > Boundaries > Assign > Convection		
Parameters	Name	Type	Description
	<NAME>	string	Boundary condition name
	<Objects>	list	Objects included in the boundary condition assignment
	<Faces>	list	Faces included in the boundary condition assignment
	<Temperature>	string	Temperature of the ambient environment
	<Uniform>	bool	True or False: Type of film coefficient – Uniform when True, Non-Uniform (imported from Icepak) when False
	<FilmCoeff>	string	Uniform convective heat transfer coefficient at surfaces (applicable only when <Uniform> is True)
	The following eight parameters (*) are applicable only when <Uniform> is False:		
	* <Project>	string	Source project
	* <Product>	string	Source design product ("ElectronicsDesktop")
	* <Design>	string	Source design name
	* <Soln>	string	Source solution name
	* <NAME:Params>	string	Parameters array identifier (mapped variables, if any, and their values are included in this array)
	* <ForceSourceToSolve>	bool	True or False – simulate source design as needed
	* <PreservePartnerSoln>	bool	True or False – preserve source design solution
	* <PathRelativeTo>	string	Source path location relative to ("SourceProject" or "TargetProject")
Return Value	None		

Python Syntax	AssignConvection ([<NAME>, <Objects>, <Faces>, <Temperature>, <Uniform>, <FilmCoeff>])
Python Example (Uniform Film Coefficient)	<pre>oModule.AssignConvection(["NAME:Convection1", "Objects:=" , ["Coil","HeatSink"], "Faces:=" , [104,121], "Temperature:=" , "AmbientTemp", "Uniform:=" , True, "FilmCoeff:=" , "2.5w_per_m2kel"])</pre>

Python Syntax	AssignConvection ([<NAME>, <Objects>, <Faces>, <Temperature>, <Uniform>, <Project>, <Product>, <Design>, <Soln>, [<NAME:Params>], <ForceSourceToSolve>, <PreservePartnerSoln>, <PathRelativeTo>])
Python Example (Non-Uniform Film Coefficient)	<pre>oModule.AssignConvection(["NAME:Convection1" , "Objects:=" , ["Coil","HeatSink"], "Faces:=" , [104,121], "Temperature:=" , "AmbientTemp", "Uniform:=" , False, "Project:=" , "This Project*", "Product:=" , "ElectronicsDesktop", "Design:=" , "IcepakDesign1", "Soln:=" , "Setup1 : SteadyState", ["NAME:Params" , # NOTE: The highlighted line is a mapped va "Pwr:=" , "0.625W" # <-- source design and its associated va]])</pre> name from the mapped to the

	<pre>], # source design (only present when mapped variables exist). "ForceSourceToSolve:=" , True, "PreservePartnerSoln:=" , True, "PathRelativeTo:=" , "TargetProject"]) </pre>

VB Syntax	AssignConvection (<NAME>, <Objects>, <Faces>, <Temperature>, <Uniform>, <FilmCoeff>)
VB Example Uniform Film Coefficient	<pre> oModule.AssignConvection Array("NAME:Convection1", "Objects:=", Array("Coil", "HeatSink"), _ "Faces:=" Array(104,121), "Temperature:=", "AmbientTemp", "Uniform:=", True, _ "FilmCoeff:=", "2.5w_per_m2kel") </pre>

VB Syntax	AssignConvection (<NAME>, <Objects>, <Faces>, <Temperature>, <Uniform>, <Project>, <Product>, <Design>, <Soln>, (<NAME:Params>), <ForceSourceToSolve>, <PreservePartnerSoln>, <PathRelativeTo>)
VB Example Non-Uniform Film Coefficient	<pre> oModule.AssignConvection Array("NAME:Convection1", "Objects:=", Array("Coil", "HeatSink"), _ "Faces:=" Array(104,121), "Temperature:=", "AmbientTemp", "Uniform:=", False, "Pro- ject:=", _ "This Project*", "Product:=", "ElectronicsDesktop", "Design:=", "IcepakDesign1", "Soln:=", _ "Setup1 : SteadyState", Array("NAME:Params", "Pwr:=", "0.625W"), "For- ceSourceToSolve:=", _ True, "PreservePartnerSoln:=", True, "PathRelativeTo:=", "TargetProject") </pre>
Note	Yellow highlighted text entries above are a mapped variable from the source design and its associated value mapped to the source design. (These entries are only present when mapped variables exist.)

EditConvection

Note:

In the above script examples, you can substitute "*Edit*" for "*Assign*" to modify a pre-existing convection boundary instead of creating a new one. The script must identify the **NAME** of the existing boundary within the *EditConvection* command line. Then, you can change any of the setup parameters with the exception of the assignment *Faces* or *Objects*.

To change the assignment *Faces* or *Objects*, you must use the *ReassignBoundary* command, provide the excitation NAME, and specify the new Faces and/or Objects.

Script examples for editing a convection boundary with a uniform film coefficient are given below. In these examples, the name, ambient temperature, and film coefficient values are being modified.

- **UI Access:** Double-click the boundary in the Project Manager or right-click it and choose **Properties** from the shortcut menu.
- **Python Example:**

```
oModule.EditConvection("Convection1",  
    [  
        "NAME:Convection_Enclosure",  
        "Temperature:=", "75fah",  
        "FilmCoeff:=", "5w_per_m2kel",  
    ])
```

- **VB Example:**

```
oModule.EditConvection "Convection1", Array("NAME:Convection_Enclosure", _  
    "Temperature:=", "75fah", "FilmCoeff:=", "5w_per_m2kel")
```

AssignCylindricalSupport* and *EditCylindricalSupport

AssignCylindricalSupport

This command creates a Cylindrical Support boundary in a Mechanical–Modal or Mechanical-Structural design.

UI Access	Mechanical > Boundaries > Assign > Cylindrical Support		
Parameters	Name	Type	Description
	<NAME>	string	Boundary condition name
	<Faces>	list	Faces included in the boundary condition assignment
	<Axial>	string	State of axial direction constraint ("Fixed" or "Free")
	<Radial>	string	State of radial direction constraint ("Fixed" or "Free")
	<Tangential>	string	State of tangential direction constraint ("Fixed" or "Free")
Return Value	None		

Python Syntax	AssignCylindricalSupport (<NAME>, <Faces>, <Axial>, <Radial>, <Tangential>)
Python Example	<pre>oModule.AssignCylindricalSupport (["NAME:CylindricalSupport1", "Faces:=", "Axial:=", "Radial:=", "Tangential:="], [27], "Free", "Fixed", "Free",)</pre>

VB Syntax	AssignCylindricalSupport (<NAME>, <Faces>, <Axial>, <Radial>, <Tangential>)
VB Example	<pre>oModule.AssignCylindricalSupport Array("NAME:CylindricalSupport1", "Faces:=", _</pre>

<pre>Array(27), "Radial:=", "Fixed", "Axial:=", "Free", "Tangential:=", "Free")</pre>

EditCylindricalSupport

Note:

In the above script examples, you can substitute "*Edit*" for "*Assign*" to modify a pre-existing cylindrical support boundary instead of creating a new one. The script must identify the **Name** of the existing support within the *Edit* command line. Then, you can change the **Name** and the status of the **Axial**, **Radial**, and/or **Tangential** constraint directions (*Free* or *Fixed*). You cannot change the assignment *Faces* in this manner.

Script examples for editing a cylindrical support boundary are given below.

- **UI Access:** Double-click the boundary in the Project Manager or right-click it and choose **Properties** from the shortcut menu.
- **Python Example:**

```
oModule.EditCylindricalSupport("CylindricalSupport1",  
    [  
        "NAME:CylindricalSupport_Left",  
        "Axial:="                        , "Fixed",  
        "Radial:="                      , "Fixed",  
        "Tangential:="                  , "Free",  
    ])
```

- **VB Example:**

```
oModule.EditCylindricalSupport "CylindricalSupport1", Array("NAME:CylindricalSupport_Left",  
—
```

```
"Radial:=", "Fixed", "Axial:=", "Fixed", "Tangential:=", "Free")
```

AssignDisplacement and EditDisplacement

AssignDisplacement

This command creates a Displacement excitation in a Mechanical–Structural design.

UI Access	Mechanical > Excitations > Assign > Displacement		
Parameters	Name	Type	Description
	<NAME>	string	Excitation name
	<Faces>	list	Faces included in the excitation assignment
	<Edges>	list	Edges included in the excitation assignment
	<DefinedBy>	string	Method of defining displacement direction (for faces only): "NormalTo" or "Component" – Omit parameter when the assignment selection includes edges (always defined by components)
			The following parameter is only applicable when "DefinedBy" = "NormalTo":
	<Displacement>	string	Magnitude of displacement normal to face (positive value pulls outward on face, negative pushes inward)
			The following seven parameters are only applicable when "DefinedBy" = "Component":
	<Coordinate System>	string	Coordinate system name for displacement components
	<FreeX>	bool	Free in X direction (True or False)
	<FreeY>	bool	Free in Y direction (True or False)
	<FreeZ>	list	Free in Z direction (True or False)
	<DisplacementX>	string	Displacement in X direction (omit when "FreeX" = True)
	<DisplacementY>	list	Displacement in Y direction (omit when "FreeY" = True)
	<DisplacementZ>	list	Displacement in Z direction (omit when "FreeZ" = True)
Return Value	None		

Python Syntax	For displacements normal to faces: AssignDisplacement ([<NAME>, <Faces>, <DefinedBy>, <Displacement>]) For component-based displacement of faces: AssignDisplacement ([<NAME>, <Faces>, <DefinedBy>, <Coordinate System>, <FreeX>, <FreeY>, <FreeZ>, <DisplacementX>, <DisplacementY>, <DisplacementZ>]) For displacement of edges or mixed faces and edges: AssignDisplacement ([<NAME>, <Faces>, <Edges>, <Coordinate System>, <FreeX>, <FreeY>, <FreeZ>, <DisplacementX>, <DisplacementY>, <DisplacementZ>])
Python Example	<pre>oModule.AssignDisplacement (["NAME:Displacement1", "Faces:=", "DefinedBy:=", "Coordinate System:=", "FreeX:=", "FreeY:=", "FreeZ:=", "DisplacementX:=", "DisplacementZ:="], ["11"], "Component", "Global", False, True, False, "0mm", "-0.05mm",])</pre>

VB Syntax	For displacements normal to faces: AssignDisplacement (<NAME>, <Faces>, <DefinedBy>, <Displacement>) For component-based displacement of faces: AssignDisplacement (<NAME>, <Faces>, <DefinedBy>, <Coordinate System>, <FreeX>, <FreeY>, <FreeZ>, <DisplacementX>, <DisplacementY>, <DisplacementZ>) For displacement of edges or mixed faces and edges:

	<code>AssignDisplacement ([<NAME>, <Faces>, <Edges>, <Coordinate System>, <FreeX>, <FreeY>, <FreeZ>, <DisplacementX>, <DisplacementY>, <DisplacementZ>])</code>
VB Example	<code>oModule.AssignDisplacement Array("NAME:Displacement1", "Edges=", _ Array("24"), "Coordinate System=", "RelCS1", "FreeX=", false, _ "FreeY=" true, "FreeZ", true, "DisplacementX", "0.1mm")</code>

EditDisplacement

Note:

In the above script examples, you can substitute *Edit* for *Assign* to modify a pre-existing Displacement excitation instead of creating a new one. The script must identify the **Name** of the existing excitation within the *Edit* command line. Then, you can change any of the parameters except for the assignment *Objects*, which cannot be changed in this manner.

Script examples for editing a Displacement excitation are given below.

- **UI Access:** Double-click the excitation in the Project Manager or right-click it and choose **Properties** from the shortcut menu.
- **Python Example:**

```
oModule.EditDisplacement("Displacement1",
[
    "NAME:Displacement_Top",
    "DefinedBy:"           , "Component",
    "Coordinate System:"   , "Global",
    "FreeX:"               , True,
    "FreeY:"               , False,
    "FreeZ:"               , False,
    "DisplacementY:"       , "0mm",
    "DisplacementZ:"       , "-0.05mm",
```

```
] )
```

- **VB Example:**

```
oModule.EditDisplacementwnt "Displacement1", Array("NAME:Displacement_Top", _  
    "Coordinate System:=", "RelCS2", "FreeX:=", true, "FreeY:=", true, _  
    "FreeZ:=", false, "DisplacementZ:=", "-0.05mm")
```

AssignEMLoss and EditEMLoss

AssignEMLoss

This command creates an EM Loss excitation in a Mechanical–Thermal design.

UI Access	Mechanical > Excitations > Assign > EM Loss		
Parameters	Name	Type	Description
	<NAME>	string	Boundary condition name
	<Objects>	list	Objects included in the boundary condition assignment
	<Project>	string	Source design's project name
	<Product>	string	Source design's design type
	<Design>	int	Source design's name
	<Soln>	bool	Source design's solution name
	<NAME:Params>	string	Parameters array identifier. Mapped variables, if any, and their values are included in this array
	<ForceSourceToSolve>	bool	True or False
	<PreservePartnerSoln>	bool	True or False
	<PathRelativeTo>	string	TargetProject or SourceProduct
	<Intrinsics>	list	List of frequencies
	<SurfaceOnly>	list	List of surfaces
Return Value	None		

Python Syntax	AssignEMLoss ([<NAME>, <Objects>, <Project>, <Product>, <Design>, <Soln>, [<NAME:Params>], <ForceSourceToSolve>, <PreservePartnerSoln>, <PathRelativeTo>, <Intrinsics>, <SurfaceOnly>])
Python Example	<pre>oModule.AssignEMLoss (["NAME:EMLoss1", "Objects:=", , ["Bus3", "Bus2", "Bus1"], "Project:=", , "This Project*", "Product:=", , "ElectronicsDesktop", "Design:=", , "Bus_AC", "Soln:=", , "Setup1 : LastAdaptive", ["NAME:Params", # NOTE: The following two lines are mapped "appv:=", , "0", # <-- source design and their associated val "current:=", , "current" # <-- source design (only present when mapped], "ForceSourceToSolve:=", True, "PreservePartnerSoln:=", True, "PathRelativeTo:=", , "TargetProject", "Intrinsics:=", , ["60Hz"], "SurfaceOnly:=", , ["Bus1"]])</pre>

VB Syntax	AssignEMLoss (<NAME>, <Objects>, <Project>, <Product>, <Design>, <Soln>, (<NAME:Params>), <ForceSourceToSolve>, <PreservePartnerSoln>, <PathRelativeTo>, <Intrinsics>, <SurfaceOnly>)
VB Example	<pre>oModule.AssignEMLoss Array("NAME:EMLoss1", "Objects:=", Array("Bus3", "Bus2", "Bus1"),</pre>

	<pre> - "Project:=", "This Project*", "Product:=", "ElectronicsDesktop", "Design:=", "Bus_AC", "Soln:=", "Setup1 : LastAdaptive", Array("NAME:Params", "appv:=", "0", "current:=", "current"), "ForceSourceToSolve:=", true, "PreservePartnerSoln:=", true, "PathRelativeTo:=", "TargetProject", "Intrinsics:=", Array("60Hz"), "SurfaceOnly:=", Array("Bus1")) </pre>
Note	Yellow highlighted text entries above are mapped variables from the source design and their associated values mapped to the source design. (These entries are only present when mapped variables exist.)

EditEMLoss

Note:

In the above script examples, you can substitute *Edit* for *Assign* to modify a pre-existing EM loss excitation instead of creating a new one. The script must identify the **NAME** of the existing excitation within the *Edit* command line. Then, you can change any of the parameters except for the assignment *Objects*.

To change the assignment *Objects*, you must use the ReassignBoundary command, provide the excitation NAME, and specify the new Objects.

Script examples for editing an EM loss excitation are given below.

- **UI Access:** Double-click the excitation in the Project Manager or right-click it and choose **Properties** from the shortcut menu.
- **Python Example:**

```

oModule.EditEMLoss("EMLoss1",
[
    "NAME:EMLoss_250A",

```

```

    "Project:="                , "This Project*",
    "Product:="                , "ElectronicsDesktop",
    "Design:="                 , "Bus_AC2",
    "Soln:="                   , "Setup1 : LastAdaptive",
    [
        "NAME:Params",
        "appv:="                , "0",
        "current:="             , "250A"
    ],
    "ForceSourceToSolve:="     , True,
    "PreservePartnerSoln:="    , True,
    "PathRelativeTo:="         , "TargetProject",
    "Intrinsics:="             , ["50Hz"],
    "SurfaceOnly:="            , ["Bus1"]
])
```

• **VB Example:**

```
oModule.EditEMLoss "EMLoss1", Array("NAME:EMLoss_250A", "Project:=", _
    "This Project*", "Product:=", "ElectronicsDesktop", "Design:=", "Bus_AC2", _
    "Soln:=", "Setup1 : LastAdaptive", Array("NAME:Params", "appv:=", "0", _
    "current:=", "250A"), "ForceSourceToSolve:=", true, "PreservePartnerSoln:=", _
    true, "PathRelativeTo:=", "TargetProject", "Intrinsics:=", Array("50Hz"), _
    "SurfaceOnly:=", Array("Bus1"))
```

AssignFixedSupport and EditFixedSupport

AssignFixedSupport

This command creates a Fixed Support boundary in a Mechanical–Modal or Mechanical-Structural design.

UI Access	Mechanical > Boundaries > Assign > Fixed Support		
Parameters	Name	Type	Description

	<NAME>	string	Boundary condition name
	<Faces>	list	Faces included in the boundary condition assignment
Return Value	None		

Python Syntax	AssignFixedSupport (<NAME>, <Faces>)		
Python Example	<pre>oModule.AssignFixedSupport (["NAME:FixedSupport1", "Faces:=",], [273])</pre>		

VB Syntax	AssignFixedSupport (<NAME>, <Faces>)		
VB Example	<pre>oModule.AssignFixedSupport Array("NAME:FixedSupport1", "Faces:=", Array(273))</pre>		

EditFixedSupport

Note:

In the above script examples, you can substitute "*Edit*" for "*Assign*" to modify a pre-existing fixed support boundary instead of creating a new one. The script must identify the **Name** of the existing support within the *Edit* command line. Then, you can change the **Name** of the boundary. You cannot change the assignment *Faces* in this manner.

Script examples for editing a fixed support boundary are given below.

- **UI Access:** Double-click the boundary in the Project Manager or right-click it and choose **Properties** from the shortcut menu.
- **Python Example:**

```
oModule.EditFixedSupport ("FixedSupport1",  
    [  
        "NAME:FixedSupport_Left"  
    ])  

```

- **VB Example:**

```
oModule.EditFixedSupport "FixedSupport1", Array("NAME:FixedSupport_Left")
```

AssignForce and EditForce

AssignForce

This command creates a Force excitation in a Mechanical–Structural design.

UI Access	Mechanical > Excitations > Assign > Force		
Parameters	Name	Type	Description
	<NAME>	string	Excitation name
	<Faces>	list	Faces included in the excitation assignment
	<Objects>	list	Objects included in the excitation assignment (for Maxwell 3D source designs only)
	<Uniform>	bool	True or False
	The following four parameters (*) are applicable only when <Uniform> is True:		
	* <Coordinate System>	string	Name of the coordinate system on which force components are based
	* <ForceX>	string	Force component in X-direction (with units) assigned to each specified face
	* <ForceY>	string	Force component in Y-direction (with units) assigned to each specified face
	* <ForceZ>	string	Force component in Z-direction (with units) assigned to each

			specified face
	The following nine parameters (!) are applicable only when <i><Uniform></i> is False:		
	! <i><Project></i>	string	Source design's project name
	! <i><Product></i>	string	Source design's design type
	! <i><Design></i>	int	Source design's name
	! <i><Soln></i>	string	Source design's solution name
	! <i><NAME:Params></i>	string	Parameters array identifier. Mapped variables, if any, and their values are included in this array
	! <i><ForceSourceToSolve></i>	bool	True or False
	! <i><PreservePartnerSoln></i>	bool	True or False
	! <i><PathRelativeTo></i>	string	TargetProject or SourceProduct
	! <i><Intrinsics></i>	list	List of frequencies
Return Value	None		

Python Syntax	AssignForce (<NAME>, <Faces>, <Uniform>, <Coordinate System>, <ForceX>, <ForceY>, <ForceZ>)
Python Example (Uniform Force)	<pre>oModule.AssignForce(["NAME:Force1" , "Faces:=" , [12,19], "Uniform:=" , True, "Coordinate System:=" , "Global", "ForceX:=" , "0.5kNewton", "ForceY:=" , "0newton", "ForceZ:=" , "-5kNewton"]) </pre>

Python Syntax	AssignForce(<NAME>, <Faces>, <Objects>, <Uniform>, <Project>, <Product>, <Design>, <Soln>, <NAME:Params>, <ForceSourceToSolve>, <PreservePartnerSoln>, <PathRelativeTo>, <Intrinsics>)

Python Example (Non-Uniform Force)	<pre>oModule.AssignForce(["NAME:Forcel", "Faces:=", "Objects:=", "Uniform:=", "Project:=", "Product:=", "Design:=", "Soln:="], [63], ["Bar1","Bar2"], False, "This Project*", "ElectronicsDesktop", "HFSSDesign1", "Setup1 : Sweep", ["NAME:Params", "BarDia:=", "BarLength:=", "Pwr:="], # NOTE: Highlighted lines are mapped variable # <-- names from the source design and # <-- their associated values mapped # <-- to the source design (only present # <-- when mapped variables exist). "ForceSourceToSolve:=", True, "PreservePartnerSoln:=", True, "PathRelativeTo:=", "TargetProject", "Intrinsics:=", ["1.25GHz"]])</pre>

VB Syntax	AssignForce (<NAME>, <Faces>, <Coordinate System>, <ForceX>, <ForceY>, <ForceZ>)
VB Example (Uniform Force)	<pre>oModule.AssignForce Array("NAME:Forcel", "Faces:=", Array(12,19), _ "Uniform:=", true, "Coordinate System:=", "Global", "ForceX:=", _ "0.5kNewton", "ForceY:=", "0newton", "ForceZ:=", "-5kNewton")</pre>
VB Syntax	AssignForce (<NAME>, <Faces>, <Objects>, <Uniform>, <Project>, <Product>, <Design>, <Soln>, <NAME:Params>, <ForceSourceToSolve>, <PreservePartnerSoln>, <PathRelativeTo>, <Intrinsics>)
VB Example	<pre>oModule.AssignForce Array("NAME:Forcel", "Faces:=", Array(63), _</pre>

(Non-Uniform Force)	<pre>"Objects:=", Array("Bar1", "Bar2"), "Uniform:=", false, _ "Project:=", "This Project*", "Product:=", "ElectronicsDesktop", _ "Design:=", "HFSSDesign1", "Soln:=", "Setup1 : Sweep", _ Array("NAME:Params", "BarDia:=", "2.5mm", "BarLength:=", _ "100mm", "Pwr:=", "35kW"), "ForceSourceToSolve:=", true, _ "PreservePartnerSoln:=", true, "PathRelativeTo:=", _ "TargetProject", "Intrinsics:=", Array("1.25GHz"))</pre>
Note	Yellow highlighted text entries above are mapped variables from the source design and their associated values mapped to the source design. (These entries are only present when mapped variables exist.)

EditForce

Note:

In the above script examples, you can substitute *Edit* for *Assign* to modify a pre-existing force excitation instead of creating a new one. The script must identify the **NAME** of the existing excitation within the *Edit* command line. Then, you can change any of the setup parameters with the exception of the assigned *Faces* or *Objects*.

To change the assignment *Faces* or *Objects*, you must use the *ReassignBoundary* command, provide the excitation NAME, and specify the new Faces and/or Objects.

Script examples for editing a Force excitation are given below. Both editing examples are for a uniform force excitation.

- **UI Access:** Double-click the excitation in the Project Manager or right-click it and choose **Properties** from the shortcut menu.
- **Python Example:**

```
oModule.EditForce("Force1",
[
    "NAME:Force_Clamp"      ,
    "Coordinate System:=", "RelativeCS1",
    "ForceX:="              , "0.6kNewton",
    "ForceY:="              , "0newton",
    "ForceZ:="              , "-6kNewton"
])
```

- **VB Example:**

```
oModule.EditForce "Force1", Array("NAME:Force_Clamp", _
    "Coordinate System:=", "RelativeCS1", "ForceX:=", "0.6kNewton", _
    "ForceY:=", "0newton", "ForceZ:=", "-6kNewton")
```

AssignFrictionlessSupport and EditFrictionlessSupport

AssignFrictionlessSupport

This command creates a Frictionless Support boundary in a Mechanical–Modal or Mechanical-Structural design.

UI Access	Mechanical > Boundaries > Assign > Frictionless Support		
Parameters	Name	Type	Description
	<NAME>	string	Boundary condition name
	<Faces>	list	Faces included in the boundary condition assignment
Return Value	None		

Python Syntax	AssignFrictionlessSupport(<NAME>, <Faces>)
Python Example	<pre>oModule.AssignFrictionlessSupport(["NAME:FrictionlessSupport1", "Faces:=" , [34,56]])</pre>

VB Syntax	AssignFrictionlessSupport (<NAME>, <Faces>)
VB Example	<pre>oModule.AssignFrictionlessSupport Array("NAME:FrictionlessSupport1", "Faces:=", Array (34, 56))</pre>

EditFrictionlessSupport

Note:

In the above script examples, you can substitute "*Edit*" for "*Assign*" to modify a pre-existing frictionless support boundary instead of creating a new one. The script must identify the **Name** of the existing support within the *Edit* command line. Then, you can change the **Name** of the boundary. You cannot change the assignment *Faces* in this manner.

Script examples for editing a frictionless support boundary are given below.

- **UI Access:** Double-click the boundary in the Project Manager or right-click it and choose **Properties** from the shortcut menu.
- **Python Example:**

```
oModule.EditFrictionlessSupport("FrictionlessSupport1",  
    [  
        "NAME:FrictionlessSupports_Front"  
    ]  
)
```

- **VB Example:**

```
oModule.EditFrictionlessSupport "FrictionlessSupport1", Array("NAME:FrictionlessSupports_  
Front")
```

AssignHeatFlux and EditHeatFlux

AssignHeatFlux

This command creates a Heat Flux excitation in a Mechanical–Thermal design.

UI Access	Mechanical > Excitations > Assign > Heat Flux		
Parameters	Name	Type	Description
	<NAME>	string	Excitation name
	<Faces>	list	Faces included in the excitation assignment
	<Objects>	list	Objects included in the excitation assignment. Assigning heat flux to an object is equivalent to assigning it to each of the object's faces.
	<SourceType>	string	Choices are "Total Power" and "Surface Flux" with the latter representing uniform distributed power per unit surface area. This parameter only applies to the GetProperty, SetProperty, and ChangeProperty commands. When creating a heat flux excitation, you do not have to set this parameter. You only have to define either a TotalPower or SurfaceFlux value (below).
	<TotalPower>	string	Total thermal power applied to each of the specified faces (for SourceType = "Total Power").

	<table><tr><td><SurfaceFlux></td><td>string</td><td>Distributed thermal power per unit surface area (for SourceType="SurfaceFlux").</td></tr></table>	<SurfaceFlux>	string	Distributed thermal power per unit surface area (for SourceType="SurfaceFlux").
<SurfaceFlux>	string	Distributed thermal power per unit surface area (for SourceType="SurfaceFlux").		
Return Value	None			

Python Syntax	<code>AssignHeatFlux (<NAME>, <Faces>, <Objects>, <TotalPower>)</code> or <code>AssignHeatFlux (<NAME>, <Faces>, <Objects>, <SurfaceFlux>)</code>
Python Example (Total Power)	<pre>oModule.AssignHeatFlux(["NAME:HeatFlux1", "Faces:=" , [92,51,8], "Objects:=" , ["Box1"], "TotalPower:=" , "1.5W"])</pre>
Python Example (Surface Flux)	<pre>oModule.AssignHeatFlux(["NAME:HeatFlux1", "Faces:=" , [92,51,8], "Objects:=" , ["Box1"], "SurfaceFlux:=" , "50KW_per_m2"])</pre>

VB Syntax	AssignHeatFlux (<NAME>, <Faces>, <Objects>, <TotalPower>) or AssignHeatFlux (<NAME>, <Faces>, <Objects>, <TotalPower>)
VB Example (Total Power)	<pre>oModule.AssignHeatFlux Array("NAME:HeatFlux1", "Faces:=", Array(8,17), _ "Objects:=", Array("Box1"), TotalPower:=", "1.5W")</pre>
VB Example (Surface Flux)	<pre>oModule.AssignHeatFlux Array("NAME:HeatFlux1", "Faces:=", Array(8,17), _ "Objects:=", Array("Box1"), SurfaceFlux:=", "50KW_per_m2")</pre>

EditHeatFlux

Note:

In the above script examples, you can substitute "*Edit*" for "*Assign*" to modify a pre-existing heat flux excitation instead of creating a new one. The script must identify the **Name** of the existing excitation within the *Edit* command line. Then, you can change the **Name**, **TotalPower**, and/or **Surface Flux**. If you specify a *Surface Flux* value when editing an existing heat flux excitation of the Total Power type, the excitation is changed to the Surface Flux type. Conversely, specifying a *Total Power* value will change an excitation of the Surface Flux type to the Total Power type. You do not have to use *ChangeProperty* to reset the *SourceType* value.

You cannot change the assignment *Faces* in this manner. Instead, you must use the *ReassignBoundary* command, provide the excitation *NAME*, and specify the new *Objects* or *Faces*.

Script examples for editing a heat flux excitation are given below.

- **UI Access:** Double-click the excitation in the Project Manager or right-click it and choose **Properties** from the shortcut menu.
- **Python Example:**

```
oModule.EditHeatFlux("HeatFlux1",  
    [  
        "NAME:HeatFlux_ChipFaces",  
        "TotalPower:=" , "12W"  
    ])
```

- **VB Example:**

```
oModule.EditHeatFlux "HeatFlux1", Array("NAME:HeatFlux_ChipFaces", _  
    "TotalPower:=", "12W")
```

AssignHeatGeneration and EditHeatGeneration

AssignHeatGeneration

This command creates a Heat Generation excitation in a Mechanical–Thermal design.

UI Access	Mechanical > Excitations > Assign > Heat Generation		
Parameters	Name	Type	Description
	<NAME>	string	Excitation name
	<Objects>	list	Objects included in the excitation assignment
	<TotalPower>	string	Total thermal power distributed among the specified objects
Return Value	None		

Python Syntax	AssignHeatGeneration (<NAME>, <Objects>, <TotalPower>)

Python Example	<pre>oModule.AssignHeatGeneration(["NAME:HeatGeneration1", "Objects:=", "Ring", "TotalPower:=", "75W"])</pre>

VB Syntax	AssignHeatGeneration (<NAME>, <Objects>, <TotalPower>)
VB Example	<pre>oModule.AssignHeatGeneration Array("NAME:HeatGeneration1", "Objects:=", Array("Ring"), - TotalPower:=", "75W")</pre>

EditHeatGeneration

Note:

In the above script examples, you can substitute *Edit* for *Assign* to modify a pre-existing heat generation excitation instead of creating a new one. The script must identify the **Name** of the existing excitation within the *Edit* command line. Then, you can change the **Name** and/or **TotalPower**. You cannot change the assignment *Objects* in this manner.

Script examples for editing a heat generation excitation are given below.

- **UI Access:** Double-click the excitation in the Project Manager or right-click it and choose **Properties** from the shortcut menu.
- **Python Example:**

```
oModule.EditHeatGeneration("HeatGeneration1",  
    [  
        "NAME:HeatGeneration1",  
        "Objects:=", "Ring",  
        "TotalPower:=", "75W"  
    ]  
)
```

```
        "NAME:HeatGeneration_Ring",  
        "TotalPower:=" , "0.125HP"  
    ] )
```

• **VB Example:**

```
oModule.EditHeatGeneration "HeatGeneration1", Array("NAME:HeatGeneration_Ring", _  
    "TotalPower:=", "0.125HP")
```

AssignInitialTemperature and EditInitialTemperature

AssignInitialTemperature

Assigns an initial temperature patch to a object (for Transient Thermal solutions only).

UI Access	Mechanical > Initial Temperature > Assign		
Parameters	Name	Type	Description
	<NAME>	string	Temperature patch name
	<Objects>	list	Objects included in the temperature patch assignment
	<Initial Temperature>	string	"AmbientTemp" or initial temperature value and unit
Return Value	None		

Python Syntax	AssignInitialTemperature (<NAME>, <Objects>, <Initial Temperature>)
Python Example	<pre>oModule.AssignInitialTemperature(["NAME:InitTemp1", "Objects:=" , ["Box1"], "Initial Temperature:=" , "AmbientTemp"</pre>

	])

VB Syntax	AssignInitialTemperature (<NAME>, <Objects>, <Initial Temperature>)
VB Example	<pre>oModule.AssignInitialTemperature Array(_ "NAME:InitTemp1", "Objects:=", Array("Box1"), _ "Initial Temperature:=", "AmbientTemp")</pre>

EditInitialTemperature

Note:

In the above script examples, you can substitute *Edit* for *Assign* to modify a pre-existing initial temperature assignment instead of creating a new one. The script must identify the **NAME** of the existing initial temperature within the *EditInitialTemperature* command line. Then, you can change any of the setup parameters with the exception of the assignment *Objects*.

To change the assignment *Objects*, you must use the *ReassignInitialTemperature*, *AddAssignmentToInitialTemperature*, or *RemoveAssignmentFromInitialTemperature* command. Provide the excitation NAME, and specify the new Object or Objects.

Additionally, the *DeleteInitialTemperature*, *DeleteAllInitialTemperatures*, and *RenameInitialTemperature* commands are available to delete or rename previously assigned initial temperatures.

Script examples for editing an initial temperature are given below. In these examples, the name and initial temperature parameters are both being modified.

- **UI Access:** Double-click the initial temperature in the Project Manager or right-click it and choose **Properties** from the shortcut menu.

- **Python Example:**

```
oModule.EditInitialTemperature("InitTemp1",
    [
        "NAME:InitTemp_Rod",
        "Initial Temperature:=" , "185cel"
    ])

```

- **VB Example:**

```
oModule.EditInitialTemperature "InitTemp1", _
    Array("NAME:InitTemp_Rod", _
        "Initial Temperature:=", "185cel")

```

AssignPressure and EditPressure

AssignPressure

This command creates a Pressure excitation in a Mechanical–Structural design.

UI Access	Mechanical > Excitations > Assign > Pressure		
Parameters	Name	Type	Description
	<NAME>	string	Excitation name
	<Faces>	list	Faces included in the excitation assignment
	<Faces>	list	Faces included in the excitation assignment
	<DefinedBy>	list	Specifies the type of pressure definition: "NormalTo" or "Component"
			The following parameter is applicable when "DefinedBy" = "NormalTo":

	<i><Pressure></i>	string	Normal pressure magnitude (with units) assigned to each specified face
			The following four parameters are applicable when "DefinedBy" = "Component":
	<i><Coordinate System></i>	string	Name of the coordinate system on which the pressure components are based
	<i><PressureX></i>	string	Pressure component in X-direction (with units) assigned to each specified face
	<i><PressureY></i>	string	Pressure component in Y-direction (with units) assigned to each specified face
	<i><PressureZ></i>	string	Pressure component in Z-direction (with units) assigned to each specified face
Return Value	None		

Python Syntax	AssignPressure (<NAME>, <Faces>, <DefinedBy>, <Pressure>) or AssignPressure (<NAME>, <Faces>, <DefinedBy>, <Coordinate System>, <PressureX>, <PressureY>, <PressureZ>)
Python Example (NormalTo)	<pre>oModule.AssignPressure (["NAME:Pressure1", "Faces:=", [12,19], "DefinedBy:=", "NormalTo", "Pressure:=" , "60psi"]) </pre>
Python Example (Component)	<pre>oModule.AssignPressure (["NAME:Pressure1" , "Faces:=" , [12,19], </pre>

	<pre> "DefinedBy:=" , "Component", "Coordinate System:=", "Global", "PressureX:=" , "10kPascal", "PressureX:=" , "0pascal", "PressureX:=" , "-50kPascal"]) </pre>

VB Syntax	AssignPressure (<NAME>, <Faces>, <DefinedBy>, <Pressure>) or AssignPressure (<NAME>, <Faces>, <DefinedBy>, <Coordinate System>, <PressureX>, <PressureY>, <PressureZ>)
VB Example (NormalTo)	<pre> oModule.AssignPressure Array("NAME:Pressure1", "Faces:=", Array(12,19), _ "DefinedBy:=", "NormalTo", "Pressure:=", "60psi") </pre>
VB Example (Component)	<pre> oModule.AssignPressure Array("NAME:Pressure1", "Faces:=", Array(12,19), _ "DefinedBy:=", "Component", "Coordinate System:=", "Global", "PressureX:=", _ "0.5kNewton", "PressureY:=", "0newton", "PressureZ:=", "-5kNewton") </pre>

EditPressure

Note:

In the above script examples, you can substitute *Edit* for *Assign* to modify a pre-existing pressure excitation instead of creating a new one. The script must identify the **Name** of the existing excitation within the *Edit* command line. Then, you can change the **Name**, **DefinedBy**, **Coordinate System**, **Pressure**, **PressureX**, **PressureY**, and/or **PressureZ** parameters.

You cannot change the assignment *Faces* in this manner. Instead, you must use the *ReassignBoundary* command, provide the excitation NAME, and specify the new Faces.

Script examples for editing a Pressure excitation are given below.

- **UI Access:** Double-click the excitation in the Project Manager or right-click it and choose **Properties** from the shortcut menu.
- **Python Example:**

```
oModule.EditPressure("Pressure1",
[
    "NAME:Pressure_Clamp",
    "DefinedBy:=", "Component",
    "Coordinate System:=", "RelativeCS1",
    "PressureX:=", "0.6kNewton",
    "PressureY:=", "0newton",
    "PressureZ:=", "-6kNewton"
])
```

- **VB Example:**

```
oModule.EditPressure "Pressure1", Array("NAME:Pressure_Clamp", "DefinedBy:=", _
    "Component", "Coordinate System:=", "RelativeCS1", "PressureX:=", _
```

`"0.6kNewton", "PressureY:=", "0newton", "PressureZ:=", "-6kNewton")`

AssignRotatingFluid and EditRotatingFluid

AssignRotatingFluid

This command creates a Rotating Fluid boundary in a Mechanical–Thermal design.

UI Access	Mechanical > Boundaries > Assign > RotatingFluid		
Parameters	Name	Type	Description
	<NAME>	string	Boundary name
	<Objects>	list	Objects included in the boundary assignment
	<BandObject>	list	Object indicating rotating portion of machine, extending from mid gap and encompassing the rotor
	<GapThickness>	string	Thickness of the air gap between the rotor and stator
	<Speed>	string	Rotation rate of the rotor
	<AxisDirection>	string	Orientation of the axis of rotation
	<RotorPosition>	string	Indicates the design of the rotating machine ("Inner" or "Outer" rotor)
Return Value	None		

Python Syntax	AssignRotatingFluid (<NAME>, <Objects>, <BandObject>, <GapThickness>, <Speed>, <AxisDirection>, <RotorPosition>)
Python Example	<pre>oModule.AssignRotatingFluid(["NAME:RotatingFluid1", "Objects:=", , ["Band", "Cylinder1", "InnerRegion"], "BandObject:=" , "Band"</pre>

	<pre>"GapThickness:=" , "1.5mm" "Speed:=" , "720rpm" "AxisDirection:=" , "Global::Z" "RotorPosition:=" , "Inner"])</pre>

VB Syntax	AssignRotatingFluid (<NAME>, <Objects>, <BandObject>, <GapThickness>, <Speed>, <AxisDirection>, <RotorPosition>))
VB Example	<pre>oModule.AssignRotatingFluid Array("NAME:RotatingFluid1", "Objects:=", Array("Band", _ "Cylinder1", "InnerRegion"), BandObject:=", "Band", "GapThickness:=" "1.5mm" _ "Speed:=", "720rpm", "AxisDirection:=", "Global::Z", "RotorPosition:=", "Inner")</pre>

EditRotatingFluid

Note:

In the above script examples, you can substitute *Edit* for *Assign* to modify a pre-existing rotating fluid boundary instead of creating a new one. The script must identify the **Name** of the existing boundary within the *Edit* command line. Then, you can change the **Name**, **BandObject**, **GapThickness**, **Speed**, **AxisDirection**, and/or **RotorPosition**. You cannot change the assignment *Objects* in this manner.

Script examples for editing a rotating fluid boundary are given below.

- **UI Access:** Double-click the boundary in the Project Manager or right-click it and choose **Properties** from the shortcut menu.
- **Python Example:**

```
oModule.EditRotatingFluid("RotatingFluid1",  
    [
```

```
        "NAME:RotatingFluid_8PercentSlip",  
        "BandObject:=", "Band_1"  
        "GapThickness:=", "1.75mm"  
        "Speed:=", "662rpm"  
        "AxisDirection:=", "Global::X"  
        "RotorPosition:=", "Outer"  
    ])
```

• **VB Example:**

```
oModule.EditRotatingFluid "RotatingFluid1", Array("NAME:RotatingFluid_8PercentSlip", "BandOb-  
ject:=", "Band_1"  
    "GapThickness:=", "1.75mm", "Speed:=", "662rpm", "AxisDirection:=", "Global::X", "RotorPos-  
ition:=", "Outer")
```

AssignTemperature and EditTemperature

AssignTemperature

This command creates a Temperature boundary in a Mechanical–Thermal design.

UI Access	Mechanical > Boundaries > Assign > Temperature		
Parameters	Name	Type	Description
	<NAME>	string	Boundary condition name
	<Faces>	list	Faces included in the boundary condition assignment
	<Temperature>	string	Temperature assigned to specified faces
Return Value	None		

Python Syntax	AssignTemperature (<NAME>, <Faces>, <Temperature>)
Python Example	<pre>oModule.AssignTemperature(["NAME:Temperature1", "Faces:=" , [8,17], "Temperature:=" , "90cel"])</pre>

VB Syntax	AssignTemperature (<NAME>, <Faces>, <Temperature>)
VB Example	<pre>oModule.AssignTemperature Array("NAME:Temperature1", "Faces:=", Array(8,17), _ Temperature:=", "90cel")</pre>

EditTemperature

Note:

In the above script examples, you can substitute *Edit* for *Assign* to modify a pre-existing temperature boundary instead of creating a new one. The script must identify the **Name** of the existing boundary within the *Edit* command line. Then, you can change the **Name** and/or **Temperature**. You cannot change the assignment *Faces* in this manner.

Script examples for editing a temperature boundary are given below.

- **UI Access:** Double-click the boundary in the Project Manager or right-click it and choose **Properties** from the shortcut menu.
- **Python Example:**

```
oModule.EditTemperature("Temperature1",  
    [  
        "NAME:Temperature1",  
        "Faces:="          , [8,17],  
        "Temperature:="    , "90cel"  
    ])
```



```

        "NAME:Temperature_ChipFace",
        "Temperature:="                , "180fah"
    ] )

```

- **VB Example:**

```

oModule.EditTemperature "Temperature1", Array("NAME:Temperature_ChipFace", _
    "Temperature:=", "180fah")

```

AssignThermalCondition* and *EditThermalCondition

AssignThermalCondition

This command creates a Thermal Condition excitation in a Mechanical–Structural design.

UI Access	Mechanical > Excitations > Assign > Thermal Condition		
Parameters	Name	Type	Description
	<NAME>	string	Boundary condition name
	<Objects>	list	Objects included in the boundary condition assignment
	<Uniform>	bool	True or False
	The following parameter ⁽¹⁾ is only applicable when <Uniform> is True:		
	¹ <ThermalCondition>	string	User-specified uniform temperature (variable name, expression, or numerical value with units)
	The following eight parameters ⁽²⁾ are only applicable when <Uniform> is False:		
	² <Project>	string	Source project
	² <Product>	string	Source design product ("ElectronicsDesktop")
	² <Design>	string	Source design name ("IcepakDesignX")
	² <Soln>	string	Source solution name
	² <NAME:Params>	string	Parameters array identifier (mapped variables, if any, and their

			values are included in this array)
	² <ForceSourceToSolve>	bool	True or False (simulate source design as needed)
	² <PreservePartnerSoln>	bool	True or False (preserve source design solution)
	² <PathRelativeTo>	string	Source path location relative to ("SourceProject" or "TargetProject")
Return Value	None		

Python Syntax	AssignThermalCondition ([<NAME>, <Objects>, <Uniform>, <ThermalCondition>]) or AssignThermalCondition ([<NAME>, <Objects>, <Uniform>, <Project>, <Product>, <Design>, <Soln>, <NAME:Params>], <ForceSourceToSolve>, <PreservePartnerSoln>, <PathRelativeTo>)		
Python Example [Uniform]	<pre>oModule.AssignThermalCondition(["NAME:ThermalCondition1", "Objects:=", "Uniform:=", "ThermalCondition:="], ["Q1", "Q2", "U1"], True, "EnvTemp")</pre>		
Python Example [Non-Uniform, with mapped variables]	<pre>oModule.AssignThermalCondition(["NAME:ThermalCondition1", "Objects:=", "Uniform:=", "Project:=", "Product:=", "Design:=", "Soln:="], ["Q1", "Q2", "U1"], False, "This Project*", "ElectronicsDesktop", "IcepakDesign1", "Setup1 : SteadyState", [</pre>		

	<pre> "NAME:Params", "Q1Pwr:=" , "50mW", # <-- NOTE: These three lines are vari- ables in the source "RPwr:=" , "25mW", # <-- design and their associated values mapped to the source "U1Pwr:=" , "40mW", # <-- design (only present when mapped vari- ables exist).] "ForceSourceToSolve:=" , False, "PreservePartnerSoln:=" , False, "PathRelativeTo:=" , "TargetProject"]) </pre>

VB Syntax	AssignThermalCondition Array(<NAME>, <Objects>, <Uniform>, <ThermalCondition>) or AssignThermalCondition (<NAME>, <Objects>, <Uniform>, <Project>, <Product>, <Design>, <Soln>, Array (<NAME"Params">, <ForceSourceToSolve>, <Project>, <PreservePartnerSoln>, <PathRelativeTo>)
VB Example [Uniform]	<pre> oModule.AssignThermalCondition Array("NAME:ThermalCondition1", "Objects:=", _ Array("Q1", "Q2", "U1"), "Uniform:=", True, "ThermalCondition:=", "EnvTemp") </pre>
VB Example [Non-Uniform, with mapped variables]	<pre> oModule.AssignThermalCondition Array("NAME:ThermalCondition2", "Objects:=", Array(_ "Q1", "Q2", "U1"), "Uniform:=", false, "Project:=", "This Project*", "Product:=", _ "ElectronicsDesktop", "Design:=", "IcepakDesign1", "Soln:=", "Setup1 : SteadyState", _ Array("NAME:Params", "Q1Pwr:=", "50mW", "RPwr:=", "25mW", "U1Pwr:=", "40mW"), _ "ForceSourceToSolve:=", true, "PreservePartnerSoln:=", true, _ </pre>

	<code>"PathRelativeTo:=", "TargetProject")</code>
Note	Yellow highlighted text entries above are mapped variables from the source design and their associated values mapped to the source design. (These entries are only present when mapped variables exist.)

EditThermalCondition

Note:

In the above script examples, you can substitute *Edit* for *Assign* to modify a pre-existing Thermal Condition excitation instead of creating a new one. The script must identify the **Name** of the existing excitation within the *Edit* command line. Then, you can change any of the parameters except for the assignment *Objects*, which cannot be changed in this manner.

Script examples for editing a Thermal Condition excitation are given below.

- **UI Access:** Double-click the excitation in the Project Manager or right-click it and choose **Properties** from the shortcut menu.
- **Python Example:**

```
oModule.EditThermalCondition("ThermalCondition1",
    [
        "NAME:ThermalCondition_95C",
        "Uniform:="                , True,
        "ThermalCondition="        , "95cel"
    ])

```

- **VB Example:**

```
oModule.EditThermalCondition "ThermalCondition1", Array("NAME:ThermalCondition_95C", _
    "Uniform:=", True, "ThermalCondition:=", "95cel")

```

This page intentionally
left blank.

12 - Thermal Monitor Script Commands

For *Mechanical* designs of the *Transient Thermal* solution type, a **Monitor** is available:

```
oModule = oDesign.GetModule("Monitor")
```

The thermal monitor is used for capturing and plotting temperatures at points assigned prior to solving the analysis setup. The following scripting commands are available:

[AssignPointMonitor](#)

[ReassignPointMonitor](#)

Note:

Displaying the *Thermal Monitor* tab of the *Solutions* window, where the monitor's results appear, is strictly controlled through the user interface. There are no script commands associated with viewing the thermal monitor, only for assigning or reassigning points that you want to monitor.

AssignPointMonitor

This command assigns one or more points at which temperature results will be monitored (for Mechanical–Transient Thermal designs only). Monitor points are defined according to the type of entity specified (at a vertex, midpoint of an edge, centroid of a flat face, or centroid of an object).

One point assignment is created for each specified entity, and the specified name is incremented automatically when more than one entity is specified. If the name you specify ends with a number, that number is incremented by 1 for subsequent points. If the name does not include a number, one is added to the second assignment point name, and that number is incremented for subsequent points.

UI Access	Mechanical > Monitor > Assign > Point		
Parameters	Name	Type	Description
	<NAME>	string	Point monitor name

	<Quantities>	string list	Result type to monitor – currently, only "Temperature" is supported
	<Objects>	string list	Solid, shell, or polyline objects used to define monitor points
	<Faces>	integer list	Faces (identified by number) used to define monitor points
	<Edges>	integer list	Edges (identified by number) used to define monitor points
	<Vertices>	integer list	Vertices (identified by number) used to define monitor points
Return Value	None		

Python Syntax	AssignPointMonitor([<NAME>, <Quantities>, <Objects>, <Faces>, <Edges>, <Vertices>])
Python Example	<pre>oModule.AssignPointMonitor(["NAME:PointMonitor1", "Quantities:=" , ["Temperature"], "Objects:=" , ["Box1", "Cylinder1"], "Faces:=" , [15, 43], "Edges:=" , [28, 61], "Vertices:=" , [12, 20]]) </pre>

VB Syntax	AssignPointMonitor(<NAME>, <Quantities>, <Objects>, <Faces>, <Edges>, <Vertices>)
VB Example	<pre>oModule.AssignPointMonitor Array("NAME:PointMonitor1", "Quantities:=", _ Array("Temperature"), "Objects:=", Array("Box1", "Cylinder1"), "Faces:=", _ Array(15, 43), "Edges:=", Array(28, 61), "Vertices:=", Array(12, 20)) </pre>

ReassignPointMonitor

This command reassigns a previously created thermal monitor point to a new entity and is applicable only to Mechanical–Transient Thermal designs. A monitor point is defined according to the type of entity specified (at a vertex, midpoint of an edge, centroid of a flat face, or centroid of an object).

You cannot reassign multiple points to an existing monitor point. Specify a single entity.

UI Access	Mechanical > Monitor > Reassign		
Parameters	Name	Type	Description
	<NAME>	string	Point monitor name
	<Objects>	string list	Solid, shell, or polyline object used to define the reassigned monitor point *
	<Faces>	integer list	Face (identified by number) used to define the reassigned monitor point *
	<Edges>	integer list	Edge (identified by number) used to define the reassigned monitor point *
	<Vertices>	integer list	Vertex (identified by number) used to define the reassigned monitor point *
			* Note: Despite <i>Type = list</i> , specify only a single entity from one category (object, face, edge, or vertex).
Return Value	None		

Python Syntax	ReassignPointMonitor([<NAME>, <Objects> <Faces> <Edges> <Vertices>])
Python Example	<pre>oModule.ReassignPointMonitor(["NAME:PointMonitor1", "Faces:" , [19]])</pre>

VB Syntax	ReassignPointMonitor(<NAME>, <Objects> <Faces> <Edges> <Vertices>)
VB Example	<code>oModule.ReassignPointMonitor Array("NAME:PointMonitor1", "Faces:=", Array(19))</code>

13 - Analysis Setup Module Script Commands

Mechanical analysis setup commands should be executed by the Analysis module, referred to in Mechanical scripts as the "AnalysisSetup" module.

Set oModule = oDesign.GetModule("AnalysisSetup")

[AddTwoWayCoupling](#)

[ClearLinkedData](#)

[DeleteSetups](#)

[DeleteTwoWayCoupling](#)

[EditSetup](#)

[EditTwoWayCoupling](#)

[GetSetupCount](#)

[GetSetups](#)

[InsertSetup \[Mechanical\]](#)

[PasteSetup](#)

[RenameSetup](#)

[RevertAllToInitial](#)

[RevertCoupledSolution](#)

[RevertSetupToInitial](#)

[RevertSetupToInitialCondition](#)

AddTwoWayCoupling

Add a 2-Way Coupling setup to the specified analysis setup.

Note:

The only parameter you specify as part of adding a two-way coupling setup is the number of coupling iterations to run. The link setup (between source and target designs) is specified as part of an EM Loss excitation assignment in the target Mechanical design.

Access via Project Manager shortcut menu only.

UI Access	Under <i>Analysis</i> , right-click <SetupName> and choose Add 2-Way Coupling from the shortcut menu.		
Parameters	Name	Type	Description
	NumCouplingIters	string	Number of two-way coupling iterations to run. String represents an integer value (such as "3").
Return Value	None		

Python Syntax	AddTwoWayCoupling(<SetupName>, [<NAME>, <NumCouplingIters>])
Python Example	<pre>oModule.AddTwoWayCoupling("Setup1", ["NAME:Options", "NumCouplingIters:=" , 3])</pre>

VB Syntax	AddTwoWayCoupling <SetupName>, Array(<NAME>, <NumCouplingIters>)
VB Example	oModule.AddTwoWayCoupling "Setup1", Array("NAME:Options", "NumCouplingIters:=", 3)

ClearLinkedData (Module)

Clear the linked data of the specified solution setups. (This command is similar to the ClearLinkedData command for the *design* level, which clears the linked data for all solution setups in the design.)

UI Access	Project Manager > {Design name} > Analysis > right-click {Setup name} > Clear Linked Data or, with a solution setup selected in the Project Manager: Mechanical > Analysis > Clear Linked Data								
Parameters	<table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><SetupNameArray></td><td>Array</td><td>Specify the name of the setups whose linked data are to be cleaned</td></tr></table>			Name	Type	Description	<SetupNameArray>	Array	Specify the name of the setups whose linked data are to be cleaned
Name	Type	Description							
<SetupNameArray>	Array	Specify the name of the setups whose linked data are to be cleaned							
Return Value	None								

Python Syntax	ClearLinkedData(<SetupNameArray>)
Python Example	oModule.ClearLinkedData(["setup1"])

VB Syntax	ClearLinkedData <SetupNameArray>
VB Example	oModule.ClearLinkedData Array("setup1")

CopySetup

Copy the specified Optimetrics setup.

UI Access	NA		
Parameters	Name	Type	Description
	<SetupName>	String	Name of the setup.
Return Value	None.		

Python Syntax	CopySetup (<SetupName>)		
Python Example	oModule.CopySetup ("OptimizationSetup1")		

VB Syntax	CopySetup <SetupName>		
VB Example	oModule.CopySetup "OptimizationSetup1"		

CopyDrivenSetup

Copies a driven setup.

UI Access	N/A		
Parameters	Name	Type	Description
	<SetupName>	String	Name of the setup.
Return Value	None.		

Python Syntax	<code>CopyDrivenSetup(<SetupName>)</code>
Python Example	<code>oModule.CopyDrivenSetup("Setup1")</code>

VB Syntax	<code>CopyDrivenSetup <SetupName></code>
VB Example	<code>oModule.CopyDrivenSetup "Setup1"</code>

CopyEigenSetup

Copies an eigen-analysis setup.

UI Access	N/A		
Parameters	Name	Type	Description
	<SetupName>	String	Name of the setup.
Return Value	None.		

Python Syntax	<code>CopyEigenSetup(<SetupName>)</code>
Python Example	<code>oModule.CopyEigenSetup("Setup1")</code>

VB Syntax	<code>CopyEigenSetup <SetupName></code>
VB Example	<code>oModule.CopyEigenSetup "Setup1"</code>

DeleteSetups

Deletes one or more solution setups, which are specified by an array of solution setup names.

UI Access	Right-click a solution setup in the project tree and then click Delete on the shortcut menu, or delete selected solution setups in the List dialog box.		
Parameters	Name	Type	Description
	<SetupArray>	Array	Array of solution setup names.
Return Value	None.		

Python Syntax	DeleteSetups (<SetupArray>)
Python Example	<code>oModule.DeleteSetups (["Setup1", "Setup2"])</code>

VB Syntax	DeleteSetups <SetupArray>
VB Example	<code>oModule.DeleteSetups Array("Setup1", "Setup2")</code>

DeleteTwoWayCoupling

Delete the current 2-Way Coupling setup.

Access via Project Manager only.

UI Access	Under <i>Analysis</i> > <SetupName> in the Project Manager, right-click 2-Way Coupling and choose Delete from the shortcut menu.

	Alternatively, under <i>Analysis</i> > <i><SetupName></i> in the Project Manager, select 2-Way Coupling and press Delete .		
Parameters	Name	Type	Description
	None		
Return Value	None		

Python Syntax	DeleteTwoWayCoupling()		
Python Example	<code>oModule.DeleteTwoWayCoupling()</code>		

VB Syntax	DeleteTwoWayCoupling		
VB Example	<code>oModule.DeleteTwoWayCoupling</code>		

EditSetup

Modifies an existing solution setup.

UI Access	Double-click a solution setup in the project tree to modify its settings.		
Parameters	Name	Type	Description
	<i><SetupName></i>	String	Name of the solve setup being edited.
	<i><Attributes></i>	Array	Structured array.
			<code>Array("NAME:<NewSetupName>", <NamedParameters>)</code> See the <code>InsertSetup</code> command for details and examples.
Return Value	None.		

Python Syntax	EditSetup (<SetupName>, <Attributes>)
Python Example	<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,</pre>

```
[ "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.EditSetup("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.EditSetup("EddyCurrent",  
[  
    "NAME:EddyCurrent",  
    "Enabled:="                , True,  
    [  

```

```

    "NAME:MeshLink",
    "ImportMesh:="          , False
],
"MaximumPasses:="          , 4,
"MinimumPasses:="          , 2,
"MinimumConvergedPasses:=" , 1,
"PercentRefinement:="      , 30,
"SolveFieldOnly:="         , False,
"PercentError:="           , 0.1,
"SolveMatrixAtLast:="      , True,
"UseNonLinearIterNum:="    , False,
[
    "NAME:ExpressionCache",
    [
        "NAME:CacheItem",
        "Title:="            , "eddy_loss1",
        "Expression:="        , "eddy_loss",
        "Intrinsics:="        , "Phase='\0deg\'",
        "ReportType:="        , "Fields",
        [
            "NAME:ExpressionContext"

```

```

    ]
  ]
],
"UseCacheFor:="      , ["Pass"],
"UseIterativeSolver:=" , False,
"RelativeResidual:="  , 0.0001,
"NonLinearResidual:=" , 0.0001,
"SmoothBHCurve:="     , False,
"Frequency:="         , "200Hz",
"HasSweepSetup:="     , False,
"UseHighOrderShapeFunc:=", False,
"UseMuLink:="         , False,
"LossAdaptiveCtrl:="   , "0.3"
])
oModule.EditSetup("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",
        "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,
"EnableCWRays:="           , 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"
    ]
]

```

	<pre>]], ["NAME:Frequencies", "10GHz"], "Minimum Power:=" , "0.01", "Maximum Power:=" , "1000000", "Minimum Pressure:=" , "100pascal", "Maximum Pressure:=" , "101325pascal", "Postproc Sampling:=" , 500, "Temperature:=" , "0cel", "BuiltInGas:=" , "Helium"]) </pre>

VB Syntax	EditSetup <SetupName>, <Attributes>
VB Example	<pre> oModule.EditSetup "Setup1", Array("NAME:NewSetup", "AdaptiveFreq:=", "1GHz", </pre>

```
"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,
        "MinPass:=", 1,
        "MinConvPass:=", 1,
        "PerError:=", 1,
        "PerRefine:=", 30),
    Array("NAME:Mult",
```

	<pre> "MaxPass:=", 1, "MinPass:=", 1, "MinConvPass:=", 1, "PerError:=", 1, "PerRefine:=", 30)), Array("NAME:AC", "MaxPass:=", 10, "MinPass:=", 1, "MinConvPass:=", 2, "PerError:=", 1, "PerRefine:=", 30))</pre>

EditTwoWayCoupling

Edit the properties of the current 2-Way Coupling setup.

Access via Project Manager only.

UI Access	Under <i>Analysis</i> >, <SetupName> in the Project Manager, right-click 2-Way Coupling and choose Properties from the shortcut menu. Alternatively, under <i>Analysis</i> >, <SetupName> in the Project Manager, select 2-Way Coupling and edit settings in the <i>Mechanical</i> tab of the docked <i>Properties</i> window.		
Parameters	Name	Type	Description

	NumCouplingIters	string	Number of two-way coupling iterations to run. String represents an integer value (such as "5").
Return Value	None		

Python Syntax	EditTwoWayCoupling ([<NAME>, <NumCouplingIters>])
Python Example	<pre>oModule.EditTwoWayCoupling("Setup1", ["NAME:Options", "NumCouplingIters:=" , 5]) </pre>

VB Syntax	EditTwoWayCoupling, (<NAME>, <NumCouplingIters>)
VB Example	oModule.EditTwoWayCoupling, Array("NAME:Options", "NumCouplingIters:=", 5)

GetSetupCount

Gets the number of analysis setups in a design.

UI Access	N/A
Parameters	None.
Return Value	Integer containing number of setups.

Python Syntax	GetSetupCount ()
Python Example	<code>oModule.GetSetupCount ()</code>

VB Syntax	GetSetupCount
VB Example	<code>oModule.GetSetupCount</code>

GetSetups

Gets the names of analysis setups in a design.

UI Access	N/A
Parameters	None.
Return Value	Array of analysis setup names

Python Syntax	GetSetups ()
Python Example	<code>oModule.GetSetups ()</code>

VB Syntax	GetSetups
VB Example	<code>oModule.GetSetups</code>

GetSweeps

Gets the names of all sweeps in a given analysis setup.

UI Access	N/A		
Parameters	Name	Type	Description
	<SetupName>	String	Name of specified setup.
Return Value	Array of sweep names.		

Python Syntax	GetSweeps (<SetupName>)
Python Example	<code>oModule.GetSweeps ("Setup1")</code>

VB Syntax	GetSweeps <SetupName>
VB Example	<code>oModule.GetSweeps "Setup1"</code>

InsertSetup [Mechanical]

This command adds a new solution setup. The information on this page is specific to the implementation of the InsertSetup command for Mechanical designs (all solution types: Modal, Steady-State Thermal, Transient Thermal, and Structural).

The parameters depend on the following model characteristics:

- whether you are setting up a *Modal*, *Steady-State Thermal*, *Transient Thermal*, or *Structural* solution (*SetupType* = **MechModal**, **MechSteadyStateThermal**, **MechTransientThermal**, or **MechStructural**, respectively)
- whether you are importing the mesh from a source design
- whether there are any variables mapped between the source and target designs (when importing a mesh)

Certain options, when selected, activate additional parameters. For example, stepping parameters (**Initial**, **Min**, and **Max**) are only applicable if the *Stepping* option = **On**.

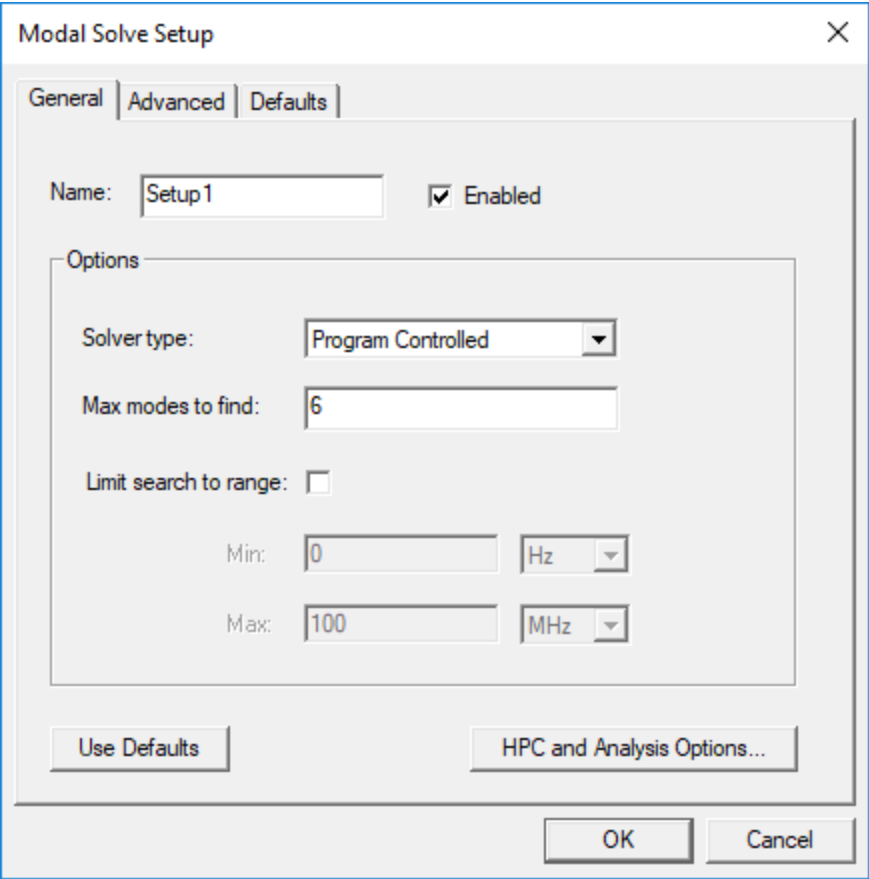
The following settings are common to all mechanical solution types and appear in the various tabs of the **Modal Solve Setup**, **Thermal Solve Setup**, or **Structural Solve Setup** dialog boxes or in the **Setup Link** dialog box. The *Setup Link* command is located under the *Advanced* tab of all of the mechanical ...**Solve Setup** dialog boxes.

UI Access	Mechanical > Analysis > Add Solution Setup		
Parameters [Common]	Name	Type	Description
	<NAME>	string	Analysis setup name
	<Enabled>	bool	True False (analysis setup enabled)
	<ImportMesh>	bool	True False
	<NAME:MeshLink>	array name	Identifies the import mesh setup array, applicable when <ImportMesh> = True
	The following nine parameters (*) are only applicable when either <ImportMesh> = True (any solution type) or <HasRestartLink> = True (transient solutions only):		
	* <Project>	string	Source project
	* <Product>	string	Source design product ("ElectronicsDesktop")
	* <Design>	string	Source design name
	* <Soln>	string	Source solution name
	* <NAME:Params>	string	Parameters array identifier (mapped variables, if any, and their values are included in this array)
	* <ForceSourceToSolve>	bool	True False – Simulate source design as needed
	* <PreservePartnerSoln>	bool	True False – Preserve source design solution
	* <PathRelativeTo>	string	Source path location relative to ("SourceProject" "TargetProject")
	* <ApplyMeshOp>	bool	True False – Apply mesh operations in target design on the imported mesh (not applicable to transient restart analyses)

Return Value	None

Parameters For Modal Solutions Only

The following image shows the default settings under the *General* tab of the **Modal Solve Setup** dialog box:



The following additional parameters correspond to Modal solutions:

UI Access	Mechanical > Analysis > Add Solution Setup

Parameters [Modal Solutions]	Name	Type	Description
	<SetupType>	string	"MechModal" for the Modal solution type
	<Max Modes>	integer	Number of requested modes for which to solve
	<Limit Search>	bool	True False
	<Solver>	string	"Program Controlled" "Direct" "Iterative" "Subspace" "Super-node"
	The following two parameters (*) are only applicable when <Limit Search> is True:		
	* <Range Max>	string	Frequency upper limit of modes to solve
	* <Range Min>	string	Frequency lower limit of modes to solve
Return Value	None		

Note:

Only the settings specific to *Modal* solutions are listed in the preceding table. See the [\[Common\] parameters](#) section for other applicable settings.

Modal InsertSetup Script Examples

Python Syntax [Modal Solution]	InsertSetup (<SetupType>, [<NAME>, <Enabled>, [<NAME:MeshLink>, <ImportMesh>, <Project>, <Product>, <Design>, <Soln>, [<NAME:Params>], <ForceSourceToSolve>, <PreservePartnerSoln>, <PathRelativeTo>, <ApplyMeshOps>], <Solver>, <Stepping>, <Initial>, <Min>, <Max>])
Python Example [Modal Solution]	<pre>oModule.InsertSetup("MechModal", ["NAME:Setup1", "Enabled:=", , True,</pre>

	<pre> ["ImportMesh:=" , True, "Project:=" , "This Project*", "Product:=" , "ElectronicsDesktop", "Design:=" , "HFSSDesign1", "Soln:=" , "Setup1 : LastAdaptive", ["NAME:Params", # NOTE: The following two lines are variables "Width:=" , "6in", # <--- in the source design and their associated "Height:=" , "38in" # <--- values mapped to the source design (only], # present when mapped variables exist). "ForceSourceToSolve:=" , True, "PreservePartnerSoln:=" , True, "PathRelativeTo:=" , "TargetProject", "ApplyMeshOp:=" , True] "Max Modes:=" , 6, "Limit Search:=" , True, "Range Max:=" , "20kHz", "Range Min:=" , "1Hz", "Solver:=" , "Program Controlled"]) </pre>
VB Syntax [Modal Solution]	InsertSetup <SetupType>, (<NAME>, <Enabled>, (<NAME:MeshLink>, <ImportMesh>, <Project>, <Product>, <Design>, <Soln>, (<NAME:Params>), <ForceSourceToSolve>, <PreservePartnerSoln>, <PathRelativeTo>, <ApplyMeshOps>), <Solver>, <Stepping>, <Initial>, <Min>, <Max>)
VB Example [Modal Solution]	<pre> oModule.InsertSetup "MechModal", Array("NAME:Setup1", "Enabled:=", true, _ Array("NAME:MeshLink", "ImportMesh:=", true, "Project:=", "This Project*", _ "Product:=", "ElectronicsDesktop", "Design:=", "HFSSDesign1", "Soln:=", _ "Setup1 : LastAdaptive", Array("NAME:Params", "Width:=", "6in", _ </pre>

	<pre>"Height:=", "38in"), "ForceSourceToSolve:=", true, _ "PreservePartnerSoln:=", true, "PathRelativeTo:=", "TargetProject", _ "ApplyMeshOp:=" true), "Max Modes:=", 10, "Limit Search:=", true, _ "Range Max:=", "20kHz", "Range Min:=", "1Hz", "Solver:=", "Program Controlled")</pre>
Note	<u>Yellow highlighted and underlined parameters</u> above are variables in the source design and their associated values mapped to the source design. (These parameters are only present when mesh linking is specified and mapped variables exist.)

Parameters For Thermal Solutions Only

The following image shows the default settings under the *General* tab of the **Thermal Solve Setup** dialog box for **Steady-State** and **Transient** solutions, respectively:

Thermal Solve Setup **(Steady-State)**

General | Advanced | Convergence | Initial Conditions | Defaults

Name ☒ Enabled

Solver Type ▼

Stepping ▼

Thermal Solve Setup

(Transient)

General | Advanced | Convergence | Initial Conditions | Save Fields | Defaults

Name

Setup1

☒ Enabled

Solver Type

Program Controlled

Transient Setup

Start

0

s

Stop

20

s

Time Step

1

s

Time variation preview

Stepping

Program Controlled

The following parameters correspond to *both* thermal solution types unless otherwise noted:

UI Access	Mechanical > Analysis > Add Solution Setup		
Parameters [Steady-State and Transient Thermal]	Name	Type	Description
	<SetupType>	string	"MechThermal" for the Thermal solution type
	<Solver>	string	"Program Controlled" "Direct" "Iterative"

Solutions]	<Stepping>	string	"Program Controlled" "Off" "On"
	<TemperatureConvergenceCondition>	string	"Program Controlled" "On"
	<HeatConvergenceCondition>	string	"Program Controlled" "Off" "On"
	<Initial Temperature>	string	Global initial temperature assumed at start of solution (where not overridden by assigned temperature boundaries)
	<Start Time> (*)	string	Start time, with units (for transient solutions only)
	<Stop Time> (*)	string	Stop time, with units (for transient solutions only)
	<Time Step> (*)	string	Time step, with units (for transient solutions only)
	<SaveFieldsType> (*)	string	"None" "Every N Steps" (for transient solutions only)
	<"N Steps"> (*)	string	How frequently to save fields (for transient solutions only when <SaveFieldsType> = "Every N Steps")
	<"HasRestartLink"> (*)	bool	True False – Indicates if a link to a source design is specified for a restart analysis (for transient solutions only)
	<NAME:RestartSoln> (*)	array name	Identifies the restart solution setup array (for transient solutions only when <HasRestartLink> = True) (see " Parameters [Common]" in the first table above for the properties included in this array)
	The following three parameters ⁽¹⁾ are applicable when <Stepping> is "Off":		
	¹ <Stepping Define By> (*)	string	"Substeps" "Time"
	¹ <NumberOfSubSteps>	string	Number of substeps to run
	¹ <TimeSubStep> (*)	string	Substep time, with units (for transient solutions when <Stepping Define By> = "Time")
	The following three parameters ⁽²⁾ are applicable when <Stepping> is "On":		

	² <Stepping Define By> (*)	string	"Substeps" "Time" (for transient solutions only)
	² <Initial>	string	Initial number of substeps to run (or initial time for transient solutions when <SteppingDefineBy> = "Time")
	² <Min>	string	Minimum number of substeps to solve (or minimum time for transient solutions when <SteppingDefineBy> = "Time")
	² <Max>	string	Maximum number of substeps to solve (or maximum time for transient solutions when <SteppingDefineBy> = "Time")
	The following two parameters (³) are applicable when <TemperatureConvergenceCondition> is "On":		
	³ <TemperatureConvergenceTolerance>	string	Percentage difference between adjacent iteration temperature results for the solution to be considered converged
	³ <TemperatureConvergenceMinRef>	string	Decimal numeric difference between adjacent iteration temperature results for the solution to be considered converged
	The following two parameters (⁴) are applicable when <HeatConvergenceCondition> is "On":		
	⁴ <HeatConvergenceTolerance>	string	Percentage difference between adjacent iteration heat flux results for the solution to be considered converged
	⁴ <HeatConvergenceMinRef>	string	Decimal numeric difference between adjacent iteration heat flux results for the solution to be considered converged
Note:	An asterisk (*) after the Name and cyan-highlighted rows or text, indicate parameters applicable only to Transient Thermal solutions.		
Return Value	None		

Note:

Only the settings specific to thermal solutions are listed in the preceding table. See the common parameters section for other applicable settings.

Thermal InsertSetup Script Examples

Python Syntax [Steady-State Thermal Solution]	InsertSetup(<SetupType>, [<NAME>, <Enabled>, [<NAME:MeshLink>, <ImportMesh>, <Project>, <Product>, <Design>, <Soln>, [<NAME:Params>], <ForceSourceToSolve>, <PreservePartnerSoln>, <PathRelativeTo>, <ApplyMeshOps>], <Solver>, <Stepping>, <Initial>, <Min>, <Max>, <TemperatureConvergenceCondition>, <TemperatureConvergenceTolerance>, <TemperatureConvergenceMinRef>, <HeatConvergenceCondition>, <HeatConvergenceTolerance>, <HeatConvergenceMinRef>, <Initial Temperature>])
Python Example [Steady-State Thermal Solution]	<pre>oModule.InsertSetup("MechSteadyStateThermal", ["NAME:Setup1", "Enabled:=" ["NAME:MeshLink", "ImportMesh:=" "Project:=" "Product:=" "Design:=" "Soln:=" ["NAME:Params", # NOTE: The following two lines are variables]]], True, , True, , "This Project*", , "ElectronicsDesktop", , "HFSSDesign1", , "Setup1 : LastAdaptive",]</pre>

```

    "appv:="                                , "0", # <--- in the source design and their
associated
    "current:="                             , "15A" # <--- values mapped to the source
design (only
    ],                                     # present when mapped variables
exist).
    "ForceSourceToSolve:="                  , True,
    "PreservePartnerSoln:="                , True,
    "PathRelativeTo:="                      , "TargetProject",
    "ApplyMeshOp:="                        , True
]
"Solver:="                                , "Program Controlled",
"Stepping:="                              , "On",
"Initial:="                                , "1",
"Min:="                                    , "1",
"Max:="                                    , "10",
"TemperatureConvergenceCondition:="        , "On",
"TemperatureConvergenceTolerance:="        , "0.25",
"TemperatureConvergenceMinRef:="           , "0.01cel",
"HeatConvergenceCondition:="               , "Program Controlled",
"HeatConvergenceTolerance:="               , "0.5",
"HeatConvergenceMinRef:="                   , "1e-6W",
"Initial Temperature:="                     , "AmbientTemp"
])

```

Python Syntax
[Transient Thermal
Solution]

```

InsertSetup(<SetupType>, [<NAME>, <Enabled>, [<NAME:MeshLink>, <ImportMesh>, <Project>,
<Product>, <Design>, <Soln>, [<NAME:Params>], <ForceSourceToSolve>, <PreservePartnerSoln>,
<PathRelativeTo>, <ApplyMeshOps>], <Solver>, <Stepping>, <Stepping Define By>, <Initial>, <Min>,
<Max>, <TemperatureConvergenceCondition>, <TemperatureConvergenceTolerance>, <Tem-
peratureConvergenceMinRef>, <HeatConvergenceCondition>, <HeatConvergenceTolerance>, <HeatCon-
vergenceMinRef>, <Initial Temperature>, <Start Time>, <Stop Time>, <Time Step>, <SaveFieldsType>, <N
Steps>, <HasRestartLink>, [<NAME:RestartSoln>, <Project>, <Product>, <Design>, <Soln>,
<NAME:Params>], <ForceSourceToSolve>, <PreservePartnerSoln>, <PathRelativeTo>], <Copy Fields

```

	<i>from Source>])</i>
Python Example [Transient Thermal Solution]	<pre> oModule.InsertSetup("MechTransientThermal", ["NAME:Setup1", "Enabled:=" , True, ["NAME:MeshLink", "ImportMesh:=" , False,] "Solver:=" , "Program Controlled", "Stepping:=" , "On", "Stepping Define By:=" , "Time", "Initial:=" , "0.01s", "Min:=" , "0.003s", "Max:=" , "0.1s", "TemperatureConvergenceCondition:=" , "On", "TemperatureConvergenceTolerance:=" , "0.25", "TemperatureConvergenceMinRef:=" , "0.01cel", "HeatConvergenceCondition:=" , "Program Controlled", "HeatConvergenceTolerance:=" , "0.5", "HeatConvergenceMinRef:=" , "1e-6W", "Initial Temperature:=" , "AmbientTemp", "Start Time:=" , "0s", "Stop Time:=" , "20s", "Time Step:=" , "1s", "SaveFieldsType:=" , "Every N Steps", "N Steps:=" , "2", "HasRestartLink:=" , True, ["NAME:RestartSoln", "Project:=" , "This Project*", </pre>

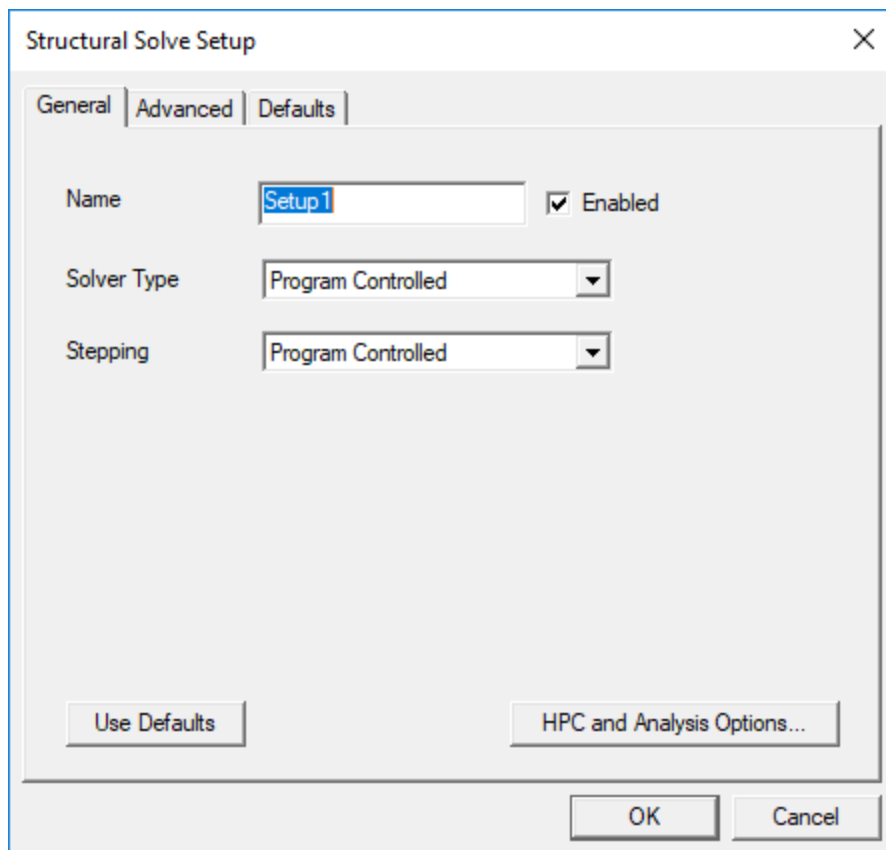
	<pre> "Product:=" , "ElectronicsDesktop", "Design:=" , "MechanicalDesign1", "Soln:=" , "Setup1 : Solution", ["NAME:Params"], "ForceSourceToSolve:=" , False, "PreservePartnerSoln:=" , True, "PathRelativeTo:=" , "TargetProject"], "Copy Fields From Source:=" , True]) </pre>

VB Syntax [Steady-State Thermal Solution]	InsertSetup <SetupType>, (<NAME>, <Enabled>, (<NAME:MeshLink>, <ImportMesh>, <Project>, <Product>, <Design>, <Soln>, (<NAME:Params>), <ForceSourceToSolve>, <PreservePartnerSoln>, <PathRelativeTo>, <ApplyMeshOps>), <Solver>, <Stepping>, <Stepping Define By>, <Initial>, <Min>, <Max>, <TemperatureConvergenceCondition>, <TemperatureConvergenceTolerance>, <TemperatureConvergenceMinRef>, <HeatConvergenceCondition>, <HeatConvergenceTolerance>, <HeatConvergenceMinRef>, <Initial Temperature>)
VB Example [Steady-State Thermal Solution]	<pre> oModule.InsertSetup "MechSteadyStateThermal", Array("NAME:Setup1", "Enabled:=", _ true, Array("NAME:MeshLink", "ImportMesh:=", true, "Project:=", "This Project*", _ "Product:=", "ElectronicsDesktop", "Design:=", "HFSSDesign1", "Soln:=", "Setup1 : _ LastAdaptive", Array("NAME:Params", "appv:=", "0", "current=", "15A"), _ "ForceSourceToSolve:=", true, "PreservePartnerSoln:=", true, "PathRelativeTo:=", _ "TargetProject", "ApplyMeshOp:=" true), "Solver:=", "Program Controlled", _ "Stepping:=", "On", "Initial:=", "1", "Min:=", "1", "Max:=", "10", _ "TemperatureConvergenceCondition:=", "On", "TemperatureConvergenceTolerance:=", _ "0.25", "TemperatureConvergenceMinRef:=", "0.01cel", "HeatConvergenceCondition:=", _ "Program Controlled", "HeatConvergenceTolerance:=", "0.5", _ "HeatConvergenceMinRef:=", "1e-6W", "Initial Temperature:=", "AmbientTemp") </pre>

Note	Yellow highlighted and underlined parameters above are variables in the source design and their associated values mapped to the source design. (These parameters are only present when mesh linking is specified and mapped variables exist.)
VB Syntax [Transient Thermal Solution]	InsertSetup <SetupType>, (<NAME>, <Enabled>, (<NAME:MeshLink>, <ImportMesh>, <Project>, <Product>, <Design>, <Soln>, (<NAME:Params>), <ForceSourceToSolve>, <PreservePartnerSoln>, <PathRelativeTo>, <ApplyMeshOps>), <Solver>, <Stepping>, <Stepping Define By>, <Initial>, <Min>, <Max>, <TemperatureConvergenceCondition>, <TemperatureConvergenceTolerance>, <TemperatureConvergenceMinRef>, <HeatConvergenceCondition>, <HeatConvergenceTolerance>, <HeatConvergenceMinRef>, <Initial Temperature>, <Start Time>, <Stop Time>, <Time Step>, <SaveFieldsType>, <N Steps>, <HasRestartLink>, (<NAME:RestartSoln>, <Project>, <Product>, <Design>, <Soln>, (<NAME:Params>), <ForceSourceToSolve>, <PreservePartnerSoln>, <PathRelativeTo>), <Copy Fields from Source>)
VB Example [Transient Thermal Solution]	oModule.InsertSetup "MechTransientThermal", Array("NAME:Setup1", "Enabled:=", true, _ Array("NAME:MeshLink", "ImportMesh:=", false), "Solver:=", "Program Controlled", _ "Stepping:=", "On", "Stepping Define By:=", "Time", "Initial:=", "0.01s", _ "Min:=", "0.003s", "Max:=", "0.1s", "TemperatureConvergenceCondition:=", "On", _ "TemperatureConvergenceTolerance:=", "0.25", "TemperatureConvergenceMinRef:=", _ "0.01cel", "HeatConvergenceCondition:=", "Program Controlled", _ "HeatConvergenceTolerance:=", "0.5", "HeatConvergenceMinRef:=", "1e-6W", _ "Initial Temperature:=", "AmbientTemp", "Start Time:=", "0s", "Stop Time:=", "20s" _ "Time Step:=", "1s", "SaveFieldsType:=", "Every N Steps", "N Steps:=", "2" _ "HasRestartLink:=", true, ("NAME:RestartSoln", "Project:=", "This Project*", _ "Product:=", "ElectronicsDesktop", "Design:=", "MechanicalDesign1", _ "Soln:=", "Setup1 : Solution", ("NAME:Params"), "ForceSourceToSolve:=", false, _ "PreservePartnerSoln:=", true, "PathRelativeTo:=", "TargetProject"), _ "Copy Fields From Source:=", true)

Parameters For Structural Solutions Only

The following image shows the default settings under the *General* tab of the **Structural Solve Setup** dialog box:



The following parameters correspond to the fields contained under this tab:

UI Access	Mechanical > Analysis > Add Solution Setup		
Parameters [Structural Solutions]	Name	Type	Description
	<SetupType>	string	"MechStructural" for the Structural solution type
	<Solver>	string	"Program Controlled" "Direct" "Iterative"
	<Stepping>	string	"Program Controlled" "Off" "On"
	The following parameter (¹) is only applicable when <Stepping> is "Off":		
	¹ <NumSubSteps>	string	Number of substeps to run
	The following three parameters (²) are only applicable when <Stepping> is "On":		
	² <Initial>	string	Initial number of substeps to run
	² <Min>	string	Minimum number of substeps to solve
	² <Max>	string	Maximum number of substeps to solve
Return Value	None		

Note:

Only the settings specific to Structural solutions are listed in the preceding table. See the common parameters section for other applicable settings.

Structural InsertSetup Script Examples

Python Syntax [Structural]	InsertSetup(<SetupType>, [<NAME>, <Enabled>, [<NAME:MeshLink>, <ImportMesh>, <Project>, <Product>, <Design>, <Soln>, [<NAME:Params>], <ForceSourceToSolve>, <PreservePartnerSoln>, <PathRelativeTo>, <ApplyMeshOps>], <Solver>, <Stepping>, <Initial>, <Min>, <Max>])

Solution]	
Python Example [Structural Solution]	<pre>oModule.InsertSetup("MechStructural", ["NAME:Setup1", "Enabled:=" , True, ["NAME:MeshLink", "ImportMesh:=" , True, "Project:=" , "This Project*", "Product:=" , "ElectronicsDesktop", "Design:=" , "HFSSDesign1", "Soln:=" , "Setup1 : LastAdaptive", ["NAME:Params", "appv:=" , "0", "current:=" , "15A"] # NOTE: The following two lines are variables # <--- in the source design and their asso- # <--- values mapped to the source design # present when mapped variables exist).], "ForceSourceToSolve:=" , True, "PreservePartnerSoln:=" , True, "PathRelativeTo:=" , "TargetProject", "ApplyMeshOp:=" , True] "Solver:=" , "Program Controlled", "Stepping:=" , "On", "Initial:=" , "1", "Min:=" , "1", "Max:=" , "10"]])</pre>
VB Syntax	InsertSetup <SetupType>, (<NAME>, <Enabled>, (<NAME:MeshLink>, <ImportMesh>, <Project>, <Product>

[Structural Solution]	<Design>, <Soln>, (<NAME:Params>), <ForceSourceToSolve>, <PreservePartnerSoln>, <PathRelativeTo>, <ApplyMeshOps>, <Solver>, <Stepping>, <Initial>, <Min>, <Max>
VB Example [Structural Solution]	<pre>oModule.InsertSetup "MechStructural", Array("NAME:Setup1", "Enabled:=", true, Array(- "NAME:MeshLink", "ImportMesh:=", true, "Project:=", "This Project*", "Product:=", _ "ElectronicsDesktop", "Design:=", "HFSSDesign1", "Soln:=", "Setup1 : LastAdaptive", - Array("NAME:Params", "appv:=", "0", "current=", "15A", "ForceSourceToSolve:=", _ true, "PreservePartnerSoln:=", true, "PathRelativeTo:=", "TargetProject", _ "ApplyMeshOp:=" true), "Solver:=", "Program Controlled", "Stepping:=", "On", _ "Initial:=", "1", "Min:=", "1", "Max:=", "10")</pre>
Note	Yellow highlighted and underlined parameters above are variables in the source design and their associated values mapped to the source design. (These parameters are only present when mesh linking is specified and mapped variables exist.)

PasteDrivenSetup

Pastes a driven setup.

UI Access	N/A
Parameters	None.
Return Value	None.

Python Syntax	PasteDrivenSetup()

Python Example	<code>oModule.PasteDrivenSetup()</code>

VB Syntax	PasteDrivenSetup
VB Example	<code>oModule.PasteDrivenSetup</code>

PasteEigenSetup

Pastes an eigen-analysis setup.

UI Access	N/A
Parameters	None.
Return Value	None.

Python Syntax	PasteEigenSetup()
Python Example	<code>oModule.PasteEigenSetup()</code>

VB Syntax	PasteEigenSetup
VB Example	<code>oModule.PasteEigenSetup</code>

PasteSetup

Use: Paste a solve setup.

Syntax: PasteSetup

Return Value: None

VB Example: `oModule.PasteSetup`

RenameSetup

Renames an existing solution setup.

UI Access	Right-click a solution setup in the Project Manager and then click Rename on the shortcut menu.		
Parameters	Name	Type	Description
	<OldSetupName>	String	Name of the solution setup being renamed.
	<NewSetupName>	String	New name for the solution setup.
Return Value	None.		

Python Syntax	RenameSetup (<OldSetupName>, <NewSetupName>)
Python Example	<code>oModule.RenameSetup ("Setup1", "MySetup")</code>

VB Syntax	RenameSetup <OldSetupName>, <NewSetupName>
VB Example	<code>oModule.RenameSetup "Setup1", "MySetup"</code>

ResetToTimeZero

Resets a simulation to time zero.

UI Access	CleanStop when running Electronics Desktop in Batchmode.		
Parameters	Name	Type	Description
	<setupName>	String	Name of the simulation setup to be reset.
Return Value	None.		

Python Syntax	ResetToTimeZero(<setupName>)		
Python Example	oModule.ResetToTimeZero("Setup1")		

VB Syntax	ResetToTimeZero <setupName>		
VB Example	oModule.ResetToTimeZero "Setup1"		

RevertAllToInitial

Marks the current mesh for all solution setups as invalid. This will force the next simulation to begin with the initial mesh.

UI Access	> Analysis Setup > Revert to Initial Mesh.		
Parameters	None.		
Return Value	None.		

Python Syntax	RevertAllToInitial ()		
Python Example	oModule.RevertAllToInitial ()		

VB Syntax	RevertAllToInitial
VB Example	<code>oModule.RevertAllToInitial</code>

RevertAllToZeroDisplacement

Reverts the displacement values of all objects to zero.

UI Access	N/A
Parameters	None.
Return Value	None.

Python Syntax	RevertAllToZeroDisplacement()
Python Example	<code>oModule.RevertAllToZeroDisplacement()</code>

VB Syntax	RevertAllToZeroDisplacement
VB Example	<code>oModule.RevertAllToZeroDisplacement</code>

RevertCoupledSolution

Invalidate results of previously run iterations and begin at iteration 1 when solved again. This command is useful if you wish to reduce the number of iterations solved to fewer than the number of previously solved iterations.

Access via the menu bar or Project Manager.

UI Access	From menu bar: Mechanical> Analysis> Revert Coupled Solution Right-click Analysis in the Project Manager and choose Revert Coupled Solution from the shortcut menu. Alternatively, under <i>Analysis</i> in the Project Manager, right-click <SetupName> and choose Revert Coupled Solution .								
Parameters	<table><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									
Return Value	None								

Python Syntax	RevertCoupledSolution()		
Python Example	<code>oModule.RevertCoupledSolution()</code>		

VB Syntax	RevertCoupledSolution		
VB Example	<code>oModule.RevertCoupledSolution</code>		

RevertSetupToInitial

Marks the current mesh for a solution setup as invalid. This will force the next simulation to begin with the initial mesh.

UI Access	Right-click a solution setup in the Project Manager and then click Rename on the shortcut menu.		
Parameters	Name	Type	Description
	<SetupName>	String	Name of specified setup.
Return Value	None.		

Python Syntax	<code>RevertSetupToInitial (<SetupName>)</code>
Python Example	<code>oModule.RevertSetupToInitial ("Setup1")</code>

VB Syntax	<code>RevertSetupToInitial <SetupName></code>
VB Example	<code>oModule.RevertSetupToInitial "Setup1"</code>

RevertSetupToInitialCondition

Reverts one or more specified setups to their initial condition. Used for linked thermal designs to revert the target solution (clearing restart, thermal monitor, and field results).

UI Access	Project Manager > Analysis > right-click <i>SetupName</i> > Revert to Initial Condition. or, with a specific setup selected under Analysis in the Project Manager: Mechanical > Analysis > Revert to Initial Condition (from the menu bar)								
Parameters	<table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><<i>SetupName</i>></td><td>String</td><td>Name of solution setup.</td></tr></table>	Name	Type	Description	< <i>SetupName</i> >	String	Name of solution setup.		
Name	Type	Description							
< <i>SetupName</i> >	String	Name of solution setup.							
Return Value	None								

Python Syntax	<code>RevertSetupToInitialCondition(<setupName>)</code>
Python Example	<code>oModule.RevertSetupToInitialCondition('Setup1')</code>

VB Syntax	RevertSetupToInitialCondition <setupName>
VB Example	<code>oModule.RevertSetupToInitialCondition "Setup1"</code>

RevertSetupToZeroDisplacement

Reverts the displacement values of objects for a specified setup to zero.

UI Access	N/A		
Parameters	Name	Type	Description
	<SetupName>	String	Name of specified setup.
Return Value	None.		

Python Syntax	RevertSetupToZeroDisplacement (<SetupName>)
Python Example	<code>oModule.RevertSetupToZeroDisplacement ("Setup1")</code>

VB Syntax	RevertSetupToZeroDisplacement <SetupName>
VB Example	<code>oModule.RevertSetupToZeroDisplacement "Setup1"</code>

SolveSetup

Solves the specified setup.

UI Access	Right-click the setup in the project tree, and then click Analyze on the shortcut menu.		
Parameters	Name	Type	Description
	<SetupName>	String	Name of the setup to be solved
Return Value	None		

Python Syntax	SolveSetup (<SetupName>)		
Python Example	oModule.SolveSetup ("Setup1")		

VB Syntax	SolveSetup <SetupName>		
VB Example	oModule.SolveSetup "Setup1"		

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.

[EditSetup](#)

[EditSetup \[Optimization\]](#)

[EditSetup \[Sensitivity\]](#)

[EditSetup \[Statistical\]](#)

[GetPropNames \[Optimetrics\]](#)

[GetPropValue \[Optimetrics\]](#)

[GetSetupNames \[Optimetrics\]](#)

[GetSetupNamesByType \[Optimetrics\]](#)

[InsertSetup \[Parametric\]](#)

[InsertSetup \[Optimization\]](#)

[InsertSetup \[Sensitivity\]](#)

[InsertSetup \[Statistical\]](#)

[PasteSetup \[Optimetrics\]](#)

[RenameSetup \[Optimetrics\]](#)

[SetPropValue \[Optimetrics\]](#)

[SolveSetup \[Optimetrics\]](#)

The topics for this section include:

[General Commands Recognized by the Optimetrics Module](#)

[Parametric Script Commands](#)

[Optimization Script Commands](#)

[Sensitivity Script Commands](#)

[Statistical Script Commands](#)

CopySetup

Copy the specified Optimetrics setup.

UI Access	NA		
Parameters	Name	Type	Description
	<SetupName>	String	Name of the setup.
Return Value	None.		

Python Syntax	CopySetup (<SetupName>)
Python Example	oModule.CopySetup ("OptimizationSetup1")

VB Syntax	CopySetup <SetupName>

VB Example	<code>oModule.CopySetup "OptimizationSetup1"</code>

DeleteSetups [Optimetrics]

Deletes the specified Optimetrics setups.

UI Access	Right-click the setup in the project tree, and then click Delete on the shortcut menu		
Parameters	Name	Type	Description
	<NameArray>	Array of Strings	An Array of Setup Names
Return Value	None		

Python Syntax	<code>DeleteSetups (<NameArray>)</code>		
Python Example	<code>oModule.DeleteSetups (["OptimizationSetup1"])</code>		

VB Syntax	<code>DeleteSetups <NameArray></code>		
VB Example	<code>oModule.DeleteSetups Array("OptimizationSetup1")</code>		

DistributedAnalyzeSetup

Distributes all variable value instances within a parametric sweep to different machines already specified from within the user interface

UI Access	Right-click the parametric setup name in the project tree and select Distribute Analysis.		
Parameters	Name	Type	Description

	<ParametricSetupName>	String	Name of the Setup
	isBlocking	Boolean	An optional arg that defaults to <code>true</code> . If it's <code>false</code> , the command immediately returns while the analysis or analyses run in the background. The status of the analyses can be checked with the <code>AreThereSimulationsRunning</code> command.
Return Value	None		

Python Syntax	DistributedAnalyzeSetup (<ParametricSetupName>)		
Python Example	<code>oModule.DistributedAnalyzeSetup ("ParametricSetup1")</code>		

VB Syntax	DistributedAnalyzeSetup <ParametricSetupName>		
VB Example	<code>oModule.DistributedAnalyzeSetup "ParametricSetup1"</code>		

EditSetup

Modifies an existing solution setup.

UI Access	Double-click a solution setup in the project tree to modify its settings.		
Parameters	Name	Type	Description
	<SetupName>	String	Name of the solve setup being edited.
	<Attributes>	Array	Structured array. <code>Array ("NAME:<NewSetupName>", <NamedParameters>)</code>

	See the InsertSetup command for details and examples.
Return Value	None.

Python Syntax	EditSetup (<SetupName>, <Attributes>)
Python Example	<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",</pre>

```

"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.EditSetup("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.EditSetup("EddyCurrent",  
[  
  "NAME:EddyCurrent",
```

```

"Enabled:="                , True,

[
    "NAME:MeshLink",
    "ImportMesh:="          , False
],

"MaximumPasses:="          , 4,
"MinimumPasses:="          , 2,
"MinimumConvergedPasses:=" , 1,
"PercentRefinement:="       , 30,
"SolveFieldOnly:="         , False,
"PercentError:="           , 0.1,
"SolveMatrixAtLast:="      , True,
"UseNonLinearIterNum:="    , False,

[
    "NAME:ExpressionCache",
    [
        "NAME:CacheItem",
        "Title:="           , "eddy_loss1",
        "Expression:="       , "eddy_loss",
        "Intrinsics:="       , "Phase='\0deg\'",
    ]
]

```

```
        "ReportType:="          , "Fields",
    [
        "NAME:ExpressionContext"
    ]
]
],
"UseCacheFor:="                , ["Pass"],
"UseIterativeSolver:="        , False,
"RelativeResidual:="          , 0.0001,
"NonLinearResidual:="          , 0.0001,
"SmoothBHCurve:="              , False,
"Frequency:="                  , "200Hz",
"HasSweepSetup:="              , False,
"UseHighOrderShapeFunc:="      , False,
"UseMuLink:="                  , False,
"LossAdaptiveCtrl:="           , "0.3"
])
oModule.EditSetup("HfssDriven",
["NAME:Setup3",
    "AdaptMultipleFreqs:=", False,
    "Frequency:=", "5GHz",
```


	<pre> "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, </pre>

```

"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",
"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,
    "EnableCWRays:="                 , 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",

```

	<pre> "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"])</pre>

VB Syntax	EditSetup <SetupName>, <Attributes>
VB Example	<pre>oModule.EditSetup "Setup1", Array("NAME:NewSetup", "AdaptiveFreq:=", "1GHz",</pre>

```
"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,
        "MinPass:=", 1,
        "MinConvPass:=", 1,
        "PerError:=", 1,
        "PerRefine:=", 30),
```

	<pre>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))</pre>

EnableSetup

Enables and disables a defined optimetrics analysis setup.

UI Access	Right-click on a setup in the project tree, select Enable Setup or Disable Setup		
Parameters	Name	Type	Description
	<SetupName>	String	Name of specified setup.
	<Enable>	Boolean	Determines whether enable or disable a setup. • True - enable setup. • False - disable setup.
Return Value	None.		

Python Syntax	<code>EnableSetup(<SetupName>, <Enable>)</code>
Python Example	<code>oModule.EnableSetup("OptimizationSetup1", True)</code>

VB Syntax	<code>EnableSetup <SetupName>, <Enable></code>
VB Example	<code>oModule.EnableSetup "OptimizationSetup1", true</code>

ExportDXConfigFile

Create an xml file with the setup information for Design Xplorer

UI Access	Right click on the Design Xplorer setup in the project tree and choose Export External Connector Addin Configuration...		
Parameters	Name	Type	Description
	<SetupName>	String	Must be one of existing DesignExplorer setup names
	<FileName>	String	Must be a valid file path and name
Return Value	None		

Python Syntax	<code>ExportDXConfigFile (<SetupName>, <FileName>)</code>

Python Example	<code>oModule.ExportDXConfigFile ("DesignXplorerSetup1", "c:/exportdir/DXSetup1.xml")</code>

VB Syntax	<code>ExportDXConfigFile <SetupName>, <FileName></code>
VB Example	<code>oModule.ExportDXConfigFile ("DesignXplorerSetup1", "c:/exportdir/DXSetup1.xml")</code>

ExportOptimetricsProfile

Export Optimetrics profile data

UI Access	Right click on the Optimetrics setup in the project tree and choose View Analysis Result... On the Post Analysis Display dialog box, click the Profile tab and click on the Export button.		
Parameters	Name	Type	Description
	<SetupName>	String	Must be one of the existing Parametric, Optimization, Sensitivity, Statistical or DesignXplorer setup names
	<FileName>	String	Must be a valid file path and name with extension of csv, tab, dat, or txt
	[profileNum]	String	Must be a numeric string. Optional: defaulted to last profile number. It should be a zero indexed profile number.
Return Value	None		

Python Syntax	<code>ExportOptimetricsProfile (<SetupName>, <FileName>, [profileNum])</code>
Python Example	<code>oModule.ExportOptimetricsProfile</code>

	("StatisticalSetup1", "c:/exportdir/test.csv")

VB Syntax	ExportOptimetricsProfile <SetupName>, <FileName>, [profileNum]
VB Example	<pre>oModule.ExportOptimetricsProfile "StatisticalSetup1", "c:/exportdir/test.csv"</pre>

ExportOptimetricsResult

Export an existing Optimization, Sensitivity, Statistical or DesignXplorer result. (Does not export Parametric results.)

UI Access	Right click on the desired Optimetrics setup in the project tree and choose View Analysis Result... On the Post Analysis Display dialog box, click the Result tab, then select Table view, and click on the Export button		
Parameters	Name	Type	Description
	<SetupName>	String	Must be one of the existing Optimization, Sensitivity, Statistical, or DesignXplorer setup names
	<FileName>	String	Must be a valid file path and name with extension of csv, tab, dat, or txt..
	[useFullOutputName]	Boolean	Optional: defaulted to false. If set to true values will be printed with units. This parameter is ignored for Optimization and Statistical results.
Return Value	None		

Python Syntax	ExportOptimetricsResult (<SetupName>, <FileName>, [useFullOutputName])
Python Example	<pre>oModule.ExportOptimetricsResult (</pre>

	<code>"StatisticalSetup1", "c:/exportdir/test.csv", false)</code>

VB Syntax	<code>ExportOptimetricsResult <SetupName>, <FileName>, [useFullOutputName]</code>
VB Example	<code>oModule.ExportOptimetricsResult "StatisticalSetup1", "c:/exportdir/test.csv", false</code>

ExportParametricResults

Export existing Parametric results.

UI Access	Right click on the desired Parametric setup in the project tree and choose View Analysis Result... On the Post Analysis Display dialog box, click the Result tab, then select Table view, and click on the Export button.		
Parameters	Name	Type	Description
	<SetupName>	String	Must be one of the existing Parametric setup names
	<FileName>	String	Must be a valid file path and name with extension of csv, tab, dat or txt
	<bOutputUnits>	Boolean	If set to true, values will be printed with units
Return Value	None		

Python Syntax	<code>ExportParametricResults (<SetupName>, <FileName>, <bOutputUnits>)</code>
Python Example	<code>oModule.ExportParametricResults ("ParametricSetup1", "c:/exportdir/test.csv", False)</code>

VB Syntax	<code>ExportParametricResults <SetupName>, <FileName>, <bOutputUnits></code>

VB Example	<pre>oModule.ExportParametricResults "ParametricSetup1", "c:/exportdir/test.csv",false</pre>

ExportParametricSetupTable

Exports the parametric setup table as a CSV file.

UI Access	Double-click parametric setup. Select Table tab. Click Export .		
Parameters	Name	Type	Description
	<SetupName>	String	Name of the setup.
	<filePath>	String	Full path for file export.
Return Value	None		

Python Syntax	ExportParametricSetupTable (<SetupName>, <filePath>)
Python Example	<pre>oModule.ExportParametricSetupTable('ParametricSetup1', 'E:/Files/ParametricSetup1_ Table.csv')</pre>

VB Syntax	ExportParametricSetupTable <SetupName>, <filePath>
VB Example	<pre>obj.ExportParametricSetupTable "ParametricSetup1", "E:/Files/ParametricSetup1_ Table.csv"</pre>

ExportRespSurfaceMinMaxTable

Exports min-max table from a response surface to a file

UI Access	Click on Export... From the Response Surface tab of the Design of Experiments Post Analysis Display dialog.		
Parameters	Name	Type	Description
	<DOEName>	String	Name of the Design of Experiments (DOE) setup.
	<FileName>	String	Output file name with path.
Return Value	None.		

Python Syntax	ExportRespSurfaceMinMaxTable(<DOEName>, <FileName>)		
Python Example	<pre>oModule.ExportRespSurfaceMinMaxTable ("DesignOfExperimentsSetup1", "C:/temp/DesignOfExperimentsSetup1_Min-Max_Search.csv")</pre>		

VB Syntax	ExportRespSurfaceMinMaxTable <DOEName>, <FileName>		
VB Example	<pre>oModule.ExportRespSurfaceMinMaxTable _ "DesignOfExperimentsSetup1", _ "C:/temp/DesignOfExperimentsSetup1_Min-Max_Search.csv"</pre>		

ExportRespSurfaceRefinePoints

Exports refinement points table to a file

UI Access	From the Response Surface tab of the Design of Experiments Post Analysis Display dialog box, select Refinement Points option under View , Click on Export....		
Parameters	Name	Type	Description
	<DOEName>	String	Name of the Design of Experiments (DOE) setup.
	<FileName>	String	Output file name with path.
Return Value	None.		

Python Syntax	ExportRespSurfaceRefinePoints(<DOEName>, <FileName>)		
Python Example	<pre>oModule.ExportRespSurfaceRefinePoints("DesignOfExperimentsSetup1", "C:/temp/DesignOfExperimentsSetup1_Refine_Points.csv")</pre>		

VB Syntax	ExportRespSurfaceRefinePoints <DOEName>, <FileName>		
VB Example	<pre>oModule.ExportRespSurfaceRefinePoints_ "DesignOfExperimentsSetup1", _ "C:/temp/DesignOfExperimentsSetup1_Refine_Points.csv"</pre>		

ExportRespSurfaceResponsePoints

Exports response points table to a file

UI Access	From the Response Surface tab of the Design of Experiments Post Analysis Display dialog box, select Response Points option under View , Click on Export....

Parameters	Name	Type	Description
	<DOEName>	String	Name of the Design of Experiments (DOE) setup.
	<FileName>	String	Output file name with path.
Return Value	None.		

Python Syntax	ExportRespSurfaceResponsePoints (<DOEName>, <FileName>)
Python Example	<pre>oModule.ExportRespSurfaceResponsePoints("DesignOfExperimentsSetup1", "C:/temp/DesignOfExperimentsSetup1_Response_Points.csv")</pre>

VB Syntax	ExportRespSurfaceResponsePoints <DOEName>, <FileName>
VB Example	<pre>oModule.ExportRespSurfaceResponsePoints_ "DesignOfExperimentsSetup1", _ "C:/temp/DesignOfExperimentsSetup1_Response_Points.csv"</pre>

ExportRespSurfaceVerificationPoints

Exports verification points table to a file

UI Access	From the Response Surface tab of the Design of Experiments Post Analysis Display dialog box, select Verification Points option under View , Click on Export...		
Parameters	Name	Type	Description
	<DOEName>	String	Name of the Design of Experiments (DOE) setup.
	<FileName>	String	Output file name with path.

Return Value	None.

Python Syntax	<code>ExportRespSurfaceVerificationPoints (<DOEName>, <FileName>)</code>
Python Example	<pre>oModule.ExportRespSurfaceVerificationPoints("DesignOfExperimentsSetup1", "C:/temp/DesignOfExperimentsSetup1_Veri_Points.csv")</pre>

VB Syntax	<code>ExportRespSurfaceVerificationPoints <DOEName>, <FileName></code>
VB Example	<pre>oModule.ExportRespSurfaceVerificationPoints_ "DesignOfExperimentsSetup1", _ "C:/temp/DesignOfExperimentsSetup1_Veri_Points.csv"</pre>

GenerateVariationData [Parametric]

Generate variation data before parametric solve for CAD integrated project

Command: Right click on the parametric setup in the project tree and choose "Generate Variation Data"

Syntax: GenerateVariationData <SetupName>

Return Value: None

Parameters: <SetupName>

Name of the setup.

VB Example:

```
oModule.GenerateVariationData "ParametricSetup1"
```

GetChildNames [Optimetrics]

If used without a specific optimization setup name, gets a list of all setups for all types. If a with a specific setup name, returns names for that optimization setup.

UI Access	NA		
Parameters	Name	Type	Description
	typeName	text string	Optional, default to get all types of setup names. Or one of type name return in GetChildTypes(). Also, the type name can be used without the prefix "Opti".
Return Value	Array of setup names.		

Python Syntax	GetChildNames()
Python Example	<pre>oOptimModule = oDesign.GetChildObject("Optimetrics") arrAllSetup = oOptimModule.GetChildNames() arrParmSetup = oOptimModule.GetChildNames("'OptiParametric'") arrOptimizeSetup = oOptimModule.GetChildNames("'Optimization'")</pre>

GetChildObject [Optimetrics]

Gets a Setup Object of the Optimetrics module

UI Access	NA

Parameters	Name	Type	Description
	Setup Name	text string	A optimetrics setup name, names returned by the GetChildNames().
Return Value	A script object for the setup See discussion of Optimetrics Setup Objects in Object Script Property Function Summary .		

Python Syntax	GetChildObject()
Python Example	<pre>oParamSetup = oOptModule.GetChildObject('ParametricSetup1') oOptSetup = oOptModule.GetChildObject('OptimizationSetup1')</pre>

GetChildTypes [Optimetrics]

Use: Gets child types of queried Optimetrics module.

Syntax: GetChildTypes()

Return Value: Array of text string, it can be an empty array if there is no setup is defined. There are six types of setup, they are ['OptiParametric', 'OptiOptimization', 'OptiSensitivity', 'OptiStatistical', 'OptiDesignExplorer', 'OptiDXDOE'].

Python Syntax	GetChildTypes ()
Python Example	<pre>oOptimModule = oDesign.GetChildObject("Optimetrics") arrSetupTypes = oOptimModule.GetChildTypes()</pre>

GetName

Returns the design name of the active design, in that order separated by a semicolon.

UI Access	N/A
Parameters	None.
Return Value	String indicating the name of the active design.

Python Syntax	GetName()
Python Example	<code>design_name = oDesign.GetName()</code>

VB Syntax	GetName
VB Example	<code>design_name = oDesign.GetName</code>

GetObjPath [Design]

Obtains the path to the design.

UI Access	N/A
Parameters	None.
Return Value	String containing the path to the design.

Python Syntax	GetObjPath()
Python Example	<code>oDesign.GetObjPath()</code>

VB Syntax	GetObjPath
VB Example	<code>oDesign.GetObjPath</code>

GetOptimetricResult

Returns an Optimetric calculation. The specific calculation is determined by the setup.

UI Access	N/A		
Parameters	Name	Type	Description
	<SetupName>	String	Optimetrics setup name.
	<vars>	Array	Array containing string variable names. Use the Sweep Definitions tab in the UI or the <SweepDefs> parameter in the InsertSetup script to determine appropriate inputs.
	<values>	Array	<i>Optional.</i> Array containing string values. When multiple variables and values are provided, the order must be the same in both the <vars> and <values> arrays. The first variable is paired with the first value, the second variable is paired with the second value, and so on.
Return Value	Calculation result. If the setup contains more than one calculation, the output will be an array of values.		

Python Syntax	GetOptimetricResult(<SetupName>, <vars>, <values>)
Python Example	oModule.GetOptimetricResult('ParametricSetup1', ['AR', 'Re'], ['4.64', '6e+04'])

VB Syntax	GetOptimetricResult <SetupName>, <vars>, <values>
VB Example	<pre>dim vars(1) vars(0) = "AR" vars(1) = "Re" dim values(1) values(0) = "4.64" values(1) = "6e+04" oModule.GetOptimetricResult "ParametricSetup1", vars, values</pre>

GetOptimetricsResult

Get an existing Optimization, Sensitivity, Statistical, Parametric or DesignXplorer result.

UI Access	Right click on the desired Optimetrics setup in the project tree and choose View Analysis Result... On the Post Analysis Display dialog box, click the Result tab, then select Table view, and click on the Export button		
Parameters	Name	Type	Description
	<SetupName>	String	Must be one of the existing Optimization, Sensitivity, Statistical, or DesignXplorer setup names
	<FileName>	String	Must be a valid file path and name with extension of csv, tab, dat, or txt..
	[useFullOutputName]	Boolean	Optional: defaulted to false. If set to true values will be printed with units. This parameter is ignored for Optimization and Statistical results.

Return Value	None

Python Syntax	GetOptimetricsResult (<SetupName>, <FileName>, [useFullOutputName])
Python Example	<pre>oModule.GetOptimetricsResult ("StatisticalSetup1", "c:/exportdir/test.csv", false)</pre>

VB Syntax	GetOptimetricsResult <SetupName>, <FileName>, [useFullOutputName]
VB Example	<pre>oModule.ExportOptimetricsResult "StatisticalSetup1", "c:/exportdir/test.csv", false</pre>

GetPropNames [Optimetrics]

Use: Always returns the empty set for Optimetrics objects since they do not have properties.

Syntax: GetPropNames(bIncludeReadOnly)

Return Value: Returns empty set.

Parameters: bIncludeReadOnly—optional, default to True.

Python Syntax	GetPropNames ()
Python Example	<pre>oOptModule.GetPropNames() oOptModule.GetPropNames(True) oOptModule.GetPropNames(False)</pre>

GetPropValue [Optimetrics]

Returns the property value for a setup property.

UI Access	NA		
Parameters	Name	Type	Description
	property-path		a child object's property path. See property path discussion here .
Return Value	Returns the value of an setup property.		

Python Syntax	GetPropValue(propPath)		
Python Example	<pre>oOptModule.GetPropValue("OptimizationSetup1\Optimizer") //get the optimizer name for OptimizationSetup1 oOptModule.GetPropValue("OptimizationSetup1\Optimizer\Choices") //Get the menu property's menu items. In this case all Optimizer names.</pre>		

VB Syntax	GetPropValue()		
VB Example	<pre>GetPropValue("Optimetrics/OptimizationSetup1/Enabled") Returns True if Enabled, or False if disabled.</pre>		

GetSetupNames [Optimetrics]

Gets a list of Optimetrics setup names

UI Access	NA		

Parameters	Name	Type	Description
	None		
Return Value	IAnsoftCollectionObj – a collection of Optimetrics setup names		

Python Syntax	GetSetupNames()
Python Example	<pre>oModule = oDesign.GetModule("Optimetrics") setupNames = oModule.GetSetupNames()</pre>

VB Syntax	GetSetupNames()
VB Example	<pre>Set oModule_opt = oDesign.GetModule("Optimetrics") Set opt_setup_list = oModule.GetSetupNames() numsetups = setupnames.Count</pre>

GetSetupNamesByType [Optimetrics]

Gets a list of Optimetrics setup names by type.

UI Access	NA		
Parameters	Name	Type	Description
	<Optimetrics type>	String	Examples: parametric, optimization, statistical, sensitivity

Return Value	Array of Optimetrics setup names of the given type.

Python Syntax	GetSetupNamesByType (<Optimetrics type>)
Python Example	<pre>for name in oModule.GetSetupNamesByType("optimization") AddInfoMessage(str(name))</pre>

VB Syntax	GetSetupNamesByType <Optimetrics type>
VB Example	<pre>For each name in oModule.GetSetupNamesByType("optimization") MsgBox name Next</pre>

ImportSetup

Import an Optimetric setup from a file.

UI Access	NA		
Parameters	Name	Type	Description
	<SetupTypeName>	String	Must be one of "OptiParametric" , "OptiOptimization" , "OptiSensitivity" , "OptiStatistical" , or "OptiDesignExplorer".
	<SetupInfo>	Array	Array("NAME:<SetupName>", "FilePath") <SetupName> Type: <string>

			Name of the setup. <FilePath> Type : <string: file path> Must be a valid file path and name.
Return Value	None		

Python Syntax	ImportSetup (<SetupTypeName>, <SetupInfo>)
Python Example	<pre>oModule.ImportSetup ("OptiStatistical", ["NAME:StatisticalSetup1", "c:/importdir/mySetupInfoFile"])</pre>

VB Syntax	ImportSetup <SetupTypeName>, <SetupInfo>
VB Example	<pre>oModule.ImportSetup "OptiStatistical", Array("NAME:StatisticalSetup1", "c:/importdir/mySetupInfoFile")</pre>

PasteSetup [Optimetrics]

Pastes the specified Optimetrics setup.

UI Access	NA		
Parameters	Name	Type	Description
	<SetupName>	String	Name of the Setup
Return Value	None		

Python Syntax	PasteSetup (<SetupName>)		
Python Example	<code>oModule.PasteSetup ("OptimizationSetup1")</code>		

VB Syntax	PasteSetup <SetupName>		
VB Example	<code>oModule.PasteSetup "OptimizationSetup1"</code>		

RenameSetup [Optimetrics]

Renames the specified Optimetrics setup.

UI Access	Right-click the setup in the project tree, and then click Rename on the shortcut menu.		
Parameters	Name	Type	Description
	<OldName>	String	The name that needs to be replaced
	<NewName>	String	Replacement name
Return Value	None		

Python Syntax	RenameSetup (<OldName> <NewName>)
Python Example	<code>oModule.RenameSetup ("OptimizationSetup1" "MyOptimization")</code>

VB Syntax	RenameSetup <OldName> <NewName>
VB Example	<code>oModule.RenameSetup "OptimizationSetup1" "MyOptimization"</code>

SetPropValue [Optimetrics]

Sets the property value for the active Optimetrics setup.

UI Access	Set Property value on Optimetrics objects		
Parameters	Name	Type	Description
	Property path	text string	Setup property path. See discussion of Property Path
	new Value	Text String, Number, or Boolean	New value data type is depending on the property type,
Return Value	True if the property is found and the new value is valid. Otherwise return False.		

Python Syntax	SetPropValue(propPath, newValue)
Python Example	<code>oOptModule.SetPropValue("ParametricSetup1\Enabled", False) //disable ParametricSetup1</code> <code>oOptModule.SetPropValue("OptimizationSetup1/Optimizer","Quasi Newton")</code>

VB Syntax	SetPropValue(propPath, newValue)
VB Example	SetPropValue("ParametricSetup1\Enabled", False)

SolveAllSetup

Solves all Optimetrics setups

UI Access	Right-click on Optimetrics in Project Manager and select Analyze>All from context menu		
Parameters	Name	Type	Description
	isBlocking	Boolean	An optional arg that defaults to <code>true</code> . If it's <code>false</code> , the command immediately returns while the analysis or analyses run in the background. The status of the analyses can be checked with the <code>AreThereSimulationsRunning</code> command.
Return Value	None		

Python Syntax	SolveAllSetup()
Python Example	<code>oModule.SolveAllSetup()</code>

VB Syntax	SolveAllSetup
VB Example	<code>oModule.SolveAllSetup</code>

SolveSetup [Optimetrics]

Solves the specified Optimetrics setup.

UI Access	Right-click the setup in the project tree, and then click Analyze on the shortcut menu.		
Parameters	Name	Type	Description
	<SetupName>	String	Name of the setup to be solved
	isBlocking	Boolean	An optional arg that defaults to <code>true</code> . If it's <code>false</code> , the command immediately returns while the analysis or analyses run in the background. The status of the analyses can be checked with the <code>AreThereSimulationsRunning</code> command.
Return Value	None		

Python Syntax	SolveSetup (<SetupName>)
Python Example	<code>oModule.SolveSetup ("OptimizationSetup1")</code>

VB Syntax	SolveSetup <SetupName>
VB Example	<code>oModule.SolveSetup "OptimizationSetup1"</code>

General Commands Recognized by the Optimetrics Module

Following are general script commands recognized by the **Optimetrics** module:

[CopySetup](#)

- [DistributedAnalyzeSetup](#)
- [EditSetup](#)
- [ExportDXConfigFile](#)
- [ExportOptimetricsProfile](#)
- [ExportOptimetricsResult](#)
- [ExportParametricResults](#)
- [GetOptimetricResult](#)
- [GetPropNames \[Optimetrics\]](#)
- [GetPropValue \[Optimetrics\]](#)
- [GetSetupNames \[Optimetrics\]](#)
- [GetSetupNamesByType \[Optimetrics\]](#)
- [ImportSetup](#)
- [PasteSetup \[Optimetrics\]](#)
- [RenameSetup \[Optimetrics\]](#)
- [SetPropValue \[Optimetrics\]](#)
- [SolveSetup \[Optimetrics\]](#)
- [SolveAllSetup](#)

CopySetup

Copy the specified Optimetrics setup.

UI Access	NA

Parameters	Name	Type	Description
	<SetupName>	String	Name of the setup.
Return Value	None.		

Python Syntax	CopySetup (<SetupName>)
Python Example	oModule.CopySetup ("OptimizationSetup1")

VB Syntax	CopySetup <SetupName>
VB Example	oModule.CopySetup "OptimizationSetup1"

DeleteSetups [Optimetrics]

Deletes the specified Optimetrics setups.

UI Access	Right-click the setup in the project tree, and then click Delete on the shortcut menu		
Parameters	Name	Type	Description
	<NameArray>	Array of Strings	An Array of Setup Names
Return Value	None		

Python Syntax	DeleteSetups (<NameArray>)
Python Example	oModule.DeleteSetups (["OptimizationSetup1"])

VB Syntax	DeleteSetups <NameArray>
VB Example	<code>oModule.DeleteSetups Array("OptimizationSetup1")</code>

DistributedAnalyzeSetup

Distributes all variable value instances within a parametric sweep to different machines already specified from within the user interface

UI Access	Right-click the parametric setup name in the project tree and select Distribute Analysis.		
Parameters	Name	Type	Description
	<ParametricSetupName>	String	Name of the Setup
	isBlocking	Boolean	An optional arg that defaults to <code>true</code> . If it's <code>false</code> , the command immediately returns while the analysis or analyses run in the background. The status of the analyses can be checked with the <code>AreThereSimulationsRunning</code> command.
Return Value	None		

Python Syntax	DistributedAnalyzeSetup (<ParametricSetupName>)
Python Example	<code>oModule.DistributedAnalyzeSetup ("ParametricSetup1")</code>

VB Syntax	DistributedAnalyzeSetup <ParametricSetupName>

VB Example	<code>oModule.DistributedAnalyzeSetup "ParametricSetup1"</code>

EditSetup

Modifies an existing solution setup.

UI Access	Double-click a solution setup in the project tree to modify its settings.		
Parameters	Name	Type	Description
	<code><SetupName></code>	String	Name of the solve setup being edited.
	<code><Attributes></code>	Array	Structured array.
			<code>Array("NAME:<NewSetupName>", <NamedParameters>)</code> See the <code>InsertSetup</code> command for details and examples.
Return Value	None.		

Python Syntax	<code>EditSetup (<SetupName>, <Attributes>)</code>
Python Example	<pre>oModule.EditSetup("Setup1", ["NAME:NewSetup", "AdaptiveFreq:=", "1GHz", "EnableDistribProbTypeOption:=", false, "SaveFields:=", "true", "Enabled:=", true,</pre>

```
["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,
        "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.EditSetup("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.EditSetup("EddyCurrent",  
[  
  "NAME:EddyCurrent",  
  "Enabled:="                , True,  
  [  
    "NAME:MeshLink",  
    "ImportMesh:="          , False  
  ],  
  "MaximumPasses:="        , 4,  
  "MinimumPasses:="        , 2,  
  "MinimumConvergedPasses:=", 1,  
  "PercentRefinement:="    , 30,  
  "SolveFieldOnly:="       , False,  
  "PercentError:="         , 0.1,  
  "SolveMatrixAtLast:="    , True,
```

```

"UseNonLinearIterNum:=" , False,
[
  "NAME:ExpressionCache",
  [
    "NAME:CacheItem",
    "Title:=" , "eddy_loss1",
    "Expression:=" , "eddy_loss",
    "Intrinsics:=" , "Phase='\0deg\'",
    "ReportType:=" , "Fields",
    [
      "NAME:ExpressionContext"
    ]
  ]
],
"UseCacheFor:=" , ["Pass"],
"UseIterativeSolver:=" , False,
"RelativeResidual:=" , 0.0001,
"NonLinearResidual:=" , 0.0001,
"SmoothBHCurve:=" , False,
"Frequency:=" , "200Hz",

```

```
"HasSweepSetup:="      , False,
"UseHighOrderShapeFunc:=", False,
"UseMuLink:="          , False,
"LossAdaptiveCtrl:="    , "0.3"
])
oModule.EditSetup("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",
"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,
        "EnableCWRays:="         , 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",

```

	<pre> "Postproc Sampling:=" , 500, "Temperature:=" , "0cel", "BuiltInGas:=" , "Helium"]) </pre>

VB Syntax	EditSetup <SetupName>, <Attributes>
VB Example	<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", </pre>

```

"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.

UI Access	Right-click on a setup in the project tree, select Enable Setup or Disable Setup		
Parameters	Name	Type	Description
	<SetupName>	String	Name of specified setup.
	<Enable>	Boolean	Determines whether enable or disable a setup. • True - enable setup. • False - disable setup.
Return Value	None.		

Python Syntax	EnableSetup(<SetupName>, <Enable>)
Python Example	<code>oModule.EnableSetup("OptimizationSetup1", True)</code>

VB Syntax	EnableSetup <SetupName>, <Enable>
VB Example	<code>oModule.EnableSetup "OptimizationSetup1", true</code>

ExportDXConfigFile

Create an xml file with the setup information for Design Xplorer

UI Access	Right click on the Design Xplorer setup in the project tree and choose Export External Connector Addin Configuration...

Parameters	Name	Type	Description
	<SetupName>	String	Must be one of existing DesignExplorer setup names
	<FileName>	String	Must be a valid file path and name
Return Value	None		

Python Syntax	ExportDXConfigFile (<SetupName>, <FileName>)
Python Example	<pre>oModule.ExportDXConfigFile ("DesignXplorerSetup1", "c:/exportdir/DXSetup1.xml")</pre>

VB Syntax	ExportDXConfigFile <SetupName>, <FileName>
VB Example	<pre>oModule.ExportDXConfigFile ("DesignXplorerSetup1", "c:/exportdir/DXSetup1.xml")</pre>

ExportOptimetricsProfile

Export Optimetrics profile data

UI Access	Right click on the Optimetrics setup in the project tree and choose View Analysis Result... On the Post Analysis Display dialog box, click the Profile tab and click on the Export button.		
Parameters	Name	Type	Description
	<SetupName>	String	Must be one of the existing Parametric, Optimization, Sensitivity, Statistical or DesignXplorer setup names

	<FileName>	String	Must be a valid file path and name with extension of csv, tab, dat, or txt
	[profileNum]	String	Must be a numeric string. Optional: defaulted to last profile number. It should be a zero indexed profile number.
Return Value	None		

Python Syntax	ExportOptimetricsProfile (<SetupName>, <FileName>, [profileNum])		
Python Example	<pre>oModule.ExportOptimetricsProfile ("StatisticalSetup1", "c:/exportdir/test.csv")</pre>		

VB Syntax	ExportOptimetricsProfile <SetupName>, <FileName>, [profileNum]		
VB Example	<pre>oModule.ExportOptimetricsProfile "StatisticalSetup1", "c:/exportdir/test.csv"</pre>		

ExportOptimetricsResult

Export an existing Optimization, Sensitivity, Statistical or DesignXplorer result. (Does not export Parametric results.)

UI Access	Right click on the desired Optimetrics setup in the project tree and choose View Analysis Result... On the Post Analysis Display dialog box, click the Result tab, then select Table view, and click on the Export button		
Parameters	Name	Type	Description
	<SetupName>	String	Must be one of the existing Optimization, Sensitivity, Statistical, or DesignXplorer setup names
	<FileName>	String	Must be a valid file path and name with extension of csv, tab, dat, or txt..
	[useFullOutputName]	Boolean	Optional: defaulted to false. If set to true values will be printed with units.

	<table><tr><td></td><td></td><td>This parameter is ignored for Optimization and Statistical results.</td></tr></table>					This parameter is ignored for Optimization and Statistical results.
		This parameter is ignored for Optimization and Statistical results.				
Return Value	None					

Python Syntax	ExportOptimetricsResult (<SetupName>, <FileName>, [useFullOutputName])
Python Example	<pre>oModule.ExportOptimetricsResult ("StatisticalSetup1", "c:/exportdir/test.csv", false)</pre>

VB Syntax	ExportOptimetricsResult <SetupName>, <FileName>, [useFullOutputName]
VB Example	<pre>oModule.ExportOptimetricsResult "StatisticalSetup1", "c:/exportdir/test.csv", false</pre>

ExportParametricResults

Export existing Parametric results.

UI Access	Right click on the desired Parametric setup in the project tree and choose View Analysis Result... On the Post Analysis Display dialog box, click the Result tab, then select Table view, and click on the Export button.		
Parameters	Name	Type	Description
	<SetupName>	String	Must be one of the existing Parametric setup names
	<FileName>	String	Must be a valid file path and name with extension of csv, tab, dat or txt
	<bOutputUnits>	Boolean	If set to true, values will be printed with units
Return Value	None		

Python Syntax	ExportParametricResults (<SetupName>, <FileName>, <bOutputUnits>)
Python Example	<pre>oModule.ExportParametricResults ("ParametricSetup1", "c:/exportdir/test.csv", False)</pre>

VB Syntax	ExportParametricResults <SetupName>, <FileName>, <bOutputUnits>
VB Example	<pre>oModule.ExportParametricResults "ParametricSetup1", "c:/exportdir/test.csv", false</pre>

ExportParametricSetupTable

Exports the parametric setup table as a CSV file.

UI Access	Double-click parametric setup. Select Table tab. Click Export .		
Parameters	Name	Type	Description
	<SetupName>	String	Name of the setup.
	<filePath>	String	Full path for file export.
Return Value	None		

Python Syntax	ExportParametricSetupTable (<SetupName>, <filePath>)
Python Example	<pre>oModule.ExportParametricSetupTable('ParametricSetup1', 'E:/Files/ParametricSetup1_ Table.csv')</pre>

VB Syntax	ExportParametricSetupTable <SetupName>, <filePath>
VB Example	obj.ExportParametricSetupTable "ParametricSetup1", "E:/Files/ParametricSetup1_Table.csv"

ExportRespSurfaceMinMaxTable

Exports min-max table from a response surface to a file

UI Access	Click on Export... From the Response Surface tab of the Design of Experiments Post Analysis Display dialog.		
Parameters	Name	Type	Description
	<DOEName>	String	Name of the Design of Experiments (DOE) setup.
	<FileName>	String	Output file name with path.
Return Value	None.		

Python Syntax	ExportRespSurfaceMinMaxTable(<DOEName>, <FileName>)
Python Example	oModule.ExportRespSurfaceMinMaxTable ("DesignOfExperimentsSetup1", "C:/temp/DesignOfExperimentsSetup1_Min-Max_Search.csv")

VB Syntax	<code>ExportRespSurfaceMinMaxTable <DOEName>, <FileName></code>
VB Example	<pre>oModule.ExportRespSurfaceMinMaxTable _ "DesignOfExperimentsSetup1", _ "C:/temp/DesignOfExperimentsSetup1_Min-Max_Search.csv"</pre>

ExportRespSurfaceRefinePoints

Exports refinement points table to a file

UI Access	From the Response Surface tab of the Design of Experiments Post Analysis Display dialog box, select Refinement Points option under View , Click on Export...		
Parameters	Name	Type	Description
	<DOEName>	String	Name of the Design of Experiments (DOE) setup.
	<FileName>	String	Output file name with path.
Return Value	None.		

Python Syntax	<code>ExportRespSurfaceRefinePoints(<DOEName>, <FileName>)</code>
Python Example	<pre>oModule.ExportRespSurfaceRefinePoints("DesignOfExperimentsSetup1", "C:/temp/DesignOfExperimentsSetup1_Refine_Points.csv")</pre>

VB Syntax	<code>ExportRespSurfaceRefinePoints <DOEName>, <FileName></code>
VB Example	<pre>oModule.ExportRespSurfaceRefinePoints_ "DesignOfExperimentsSetup1", _</pre>

	"C:/temp/DesignOfExperimentsSetup1_Refine_Points.csv"

ExportRespSurfaceResponsePoints

Exports response points table to a file

UI Access	From the Response Surface tab of the Design of Experiments Post Analysis Display dialog box, select Response Points option under View , Click on Export....		
Parameters	Name	Type	Description
	<DOEName>	String	Name of the Design of Experiments (DOE) setup.
	<FileName>	String	Output file name with path.
Return Value	None.		

Python Syntax	ExportRespSurfaceResponsePoints (<DOEName>, <FileName>)
Python Example	<pre>oModule.ExportRespSurfaceResponsePoints("DesignOfExperimentsSetup1", "C:/temp/DesignOfExperimentsSetup1_Response_Points.csv")</pre>

VB Syntax	ExportRespSurfaceResponsePoints <DOEName>, <FileName>
VB Example	<pre>oModule.ExportRespSurfaceResponsePoints_ "DesignOfExperimentsSetup1", _ "C:/temp/DesignOfExperimentsSetup1_Response_Points.csv"</pre>

ExportRespSurfaceVerificationPoints

Exports verification points table to a file

UI Access	From the Response Surface tab of the Design of Experiments Post Analysis Display dialog box, select Verification Points option under View , Click on Export...		
Parameters	Name	Type	Description
	<DOEName>	String	Name of the Design of Experiments (DOE) setup.
	<FileName>	String	Output file name with path.
Return Value	None.		

Python Syntax	ExportRespSurfaceVerificationPoints (<DOEName>, <FileName>)		
Python Example	<pre>oModule.ExportRespSurfaceVerificationPoints("DesignOfExperimentsSetup1", "C:/temp/DesignOfExperimentsSetup1_Veri_Points.csv")</pre>		

VB Syntax	ExportRespSurfaceVerificationPoints <DOEName>, <FileName>		
VB Example	<pre>oModule.ExportRespSurfaceVerificationPoints_ "DesignOfExperimentsSetup1", _ "C:/temp/DesignOfExperimentsSetup1_Veri_Points.csv"</pre>		

ExportDOEResponseCurve

Exports response curve from a response surface to a file.

UI Access	Click on Export... From the Response Surface tab of the Design of Experiments Post Analysis Display dia-

	log.		
Parameters	Name	Type	Description
	<DOEName>	String	Name of the Design of Experiments (DOE) setup.
	<FileName>	String	Output file name with path.
Return Value	None		

Python Syntax	<i>DOEName (<FileName>)</i>
Python Example	<pre>oModule.ExportDOEResponseCurve("DesignOfExperimentsSetup1", "C:/Users/AEDT/R24.1/D/DesignOfExperimentsSetup1_Response_Points.csv")</pre>

VB Syntax	<i>DOEName (<FileName>)</i>
VB Example	<pre>oModule.ExportDOEResponseCurve "DesignOfExperimentsSetup1", Array(_ "\$CPU_Power=25W", "\$Fan_X=2m_per_sec", "\$Memory_Power=5W"), "\$CPU_Power", 50_ "CPU_Monitor1.Temperature", _C:/Users/AEDT/R24.1/D/DesignOfExperimentsSetup1_responsecurve.csv"</pre>

ExportDOEResponseCurveSlices

Exports response curve slices from a response surface to a file.

UI Access	Click on Export... From the Response Surface tab of the Design of Experiments Post Analysis Display dialog.

Parameters	Name	Type	Description
	<DOEName>	String	Name of the Design of Experiments (DOE) setup.
	<FileName>	String	Output file name with path.
Return Value	None		

Python Syntax	<i>DOEName (<FileName>)</i>
Python Example	<pre>oModule.ExportDOEResponseCurveSlices("DesignOfExperimentsSetup1", "C:/Users/AEDT/R24.1/D/DesignOfExperimentsSetup1_Response_Points.csv")</pre>

VB Syntax	<i>DOEName (<FileName>)</i>
VB Example	<pre>oModule.ExportDOEResponseCurveSlices "DesignOfExperimentsSetup1", Array("\$CPU_Power=25W", "\$Fan_X=2m_per_sec", "\$Memory_Power=5W"), "\$CPU_Power", 50, _"\$Fan_X", 50, "CPU_Monitor1.Temperature", _C:/Users/AEDT/R24.1/D/DesignOfExperimentsSetup1.csv"</pre>

ExportDOEResponseSurface

Exports response surface to a file.

UI Access	Click on Export... From the Response Surface tab of the Design of Experiments Post Analysis Display dialog.		
Parameters	Name	Type	Description
	<DOEName>	String	Name of the Design of Experiments (DOE) setup.

	<table><tr><td><FileName></td><td>String</td><td>Output file name with path.</td></tr></table>	<FileName>	String	Output file name with path.
<FileName>	String	Output file name with path.		
Return Value	None			

Python Syntax	<i>DOEName (<FileName>)</i>
Python Example	<pre>oModule.ExportDOEResponseSurface("DesignOfExperimentsSetup1", "C:/Users/AEDT/R24.1/D/DesignOfExperimentsSetup1_Response_Points.csv")</pre>

VB Syntax	<i>DOEName (<FileName>)</i>
VB Example	<pre>oModule.ExportDOEResponseCurveSurface "DesignOfExperimentsSetup1", Array("\$CPU_Power=25W", "\$Fan_X=2m_per_sec", "\$Memory_Power=5W"), "\$CPU_Power", 50, _"\$Fan_X", 50, "CPU_Monitor1.Temperature", _C:/Users/AEDT/R24.1/D/DesignOfExperimentsSetup1.csv"</pre>

ExportDOELocalSensitivity

Exports local sensitivity to a file.

UI Access	Click on Export... From the Response Surface tab of the Design of Experiments Post Analysis Display dialog.		
Parameters	Name	Type	Description
	<DOEName>	String	Name of the Design of Experiments (DOE) setup.
	<FileName>	String	Output file name with path.

Return Value	None

Python Syntax	<i>DOEName</i> (<FileName>)
Python Example	<pre>oModule.ExportDOELocalSensitivity("DesignOfExperimentsSetup1", "C:/Users/AEDT/R24.1/D/DesignOfExperimentsSetup1_Response_Points.csv")</pre>

VB Syntax	<i>DOEName</i> (<FileName>)
VB Example	<pre>oModule.ExportDOELocalSensitivity "DesignOfExperimentsSetup1", Array(_ "\$CPU_Power=25W", "\$Fan_X=2m_per_sec", "\$Memory_Power=5W"), Array("\$CPU_Power", _"\$Fan_X", "\$Memory_Power"), "CPU_Monitor1.Temperature", true, _"C:/Users/AEDT/R24.1/D/DesignOfExperimentsSetup1_localsensitivity.csv"</pre>

ExportDOELocalSensitivityCurve

Exports local sensitivity curves to a file.

UI Access	Click on Export... From the Response Surface tab of the Design of Experiments Post Analysis Display dialog.		
Parameters	Name	Type	Description
	<DOEName>	String	Name of the Design of Experiments (DOE) setup.
	<FileName>	String	Output file name with path.
Return Value	None		

Python Syntax	<i>DOEName (<FileName>)</i>
Python Example	<pre>oModule.ExportDOELocalSensitivityCurve("DesignOfExperimentsSetup1", "C:/Users/AEDT/R24.1/D/DesignOfExperimentsSetup1_Response_Points.csv")</pre>

VB Syntax	<i>DOEName (<FileName>)</i>
VB Example	<pre>oModule.ExportDOELocalSensitivityCurve "DesignOfExperimentsSetup1", Array(_ "\$CPU_Power=25W", "\$Fan_X=2m_per_sec", "\$Memory_Power=5W"), Array("\$CPU_Power", _"\$Fan_X", "\$Memory_Power"), "Input Parameters", 50, "CPU_Monitor1.Temperature", _"C:/Users/AEDT/R24.1/D/DesignOfExperimentsSetup1_localsensitivitycurves.csv"</pre>

GenerateVariationData [Parametric]

Generate variation data before parametric solve for CAD integrated project

Command: Right click on the parametric setup in the project tree and choose "Generate Variation Data"

Syntax: GenerateVariationData <SetupName>

Return Value: None

Parameters: <SetupName>

 Name of the setup.

VB Example:

```
oModule.GenerateVariationData "ParametricSetup1"
```

GetChildNames [Optimetrics]

If used without a specific optimization setup name, gets a list of all setups for all types. If a with a specific setup name, returns names for that optimization setup.

UI Access	NA		
Parameters	Name	Type	Description
	typeName	text string	Optional, default to get all types of setup names. Or one of type name return in GetChildTypes(). Also, the type name can be used without the prefix "Opti".
Return Value	Array of setup names.		

Python Syntax	GetChildNames()		
Python Example	<pre>oOptimModule = oDesign.GetChildObject("Optimetrics") arrAllSetup = oOptimModule.GetChildNames() arrParmSetup = oOptimModule.GetChildNames("'OptiParametric'") arrOptimizeSetup = oOptimModule.GetChildNames("'Optimization'")</pre>		

GetChildObject [Optimetrics]

Gets a Setup Object of the Optimetrics module

UI Access	NA		
Parameters	Name	Type	Description
	Setup Name	text string	A optimetrics setup name, names returned by the GetChildNames().

Return Value	A script object for the setup See discussion of Optimetrics Setup Objects in Object Script Property Function Summary .

Python Syntax	<code>GetChildObject()</code>
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	<code>GetChildTypes ()</code>
Python Example	<pre>oOptimModule = oDesign.GetChildObject("Optimetrics") arrSetupTypes = oOptimModule.GetChildTypes()</pre>

GetName

Returns the design name of the active design, in that order separated by a semicolon.

UI Access	N/A
Parameters	None.
Return Value	String indicating the name of the active design.

Python Syntax	GetName()
Python Example	<code>design_name = oDesign.GetName()</code>

VB Syntax	GetName
VB Example	<code>design_name = oDesign.GetName</code>

GetObjPath [Design]

Obtains the path to the design.

UI Access	N/A
Parameters	None.
Return Value	String containing the path to the design.

Python Syntax	GetObjPath()
Python Example	<code>oDesign.GetObjPath()</code>

VB Syntax	GetObjPath
VB Example	<code>oDesign.GetObjPath</code>

GetOptimetricResult

Returns an Optimetric calculation. The specific calculation is determined by the setup.

UI Access	N/A		
Parameters	Name	Type	Description
	<SetupName>	String	Optimetrics setup name.
	<vars>	Array	Array containing string variable names. Use the Sweep Definitions tab in the UI or the <SweepDefs> parameter in the InsertSetup script to determine appropriate inputs.
	<values>	Array	<i>Optional.</i> Array containing string values. When multiple variables and values are provided, the order must be the same in both the <vars> and <values> arrays. The first variable is paired with the first value, the second variable is paired with the second value, and so on.
Return Value	Calculation result. If the setup contains more than one calculation, the output will be an array of values.		

Python Syntax	GetOptimetricResult(<SetupName>, <vars>, <values>)
Python Example	<code>oModule.GetOptimetricResult('ParametricSetup1', ['AR', 'Re'], ['4.64', '6e+04'])</code>

VB Syntax	GetOptimetricResult <SetupName>, <vars>, <values>
VB Example	<pre> dim vars(1) vars(0) = "AR" vars(1) = "Re" dim values(1) values(0) = "4.64" values(1) = "6e+04" oModule.GetOptimetricResult "ParametricSetup1", vars, values </pre>

GetPropNames [Optimetrics]

Use: Always returns the empty set for Optimetrics objects since they do not have properties.

Syntax: GetPropNames(bIncludeReadOnly)

Return Value: Returns empty set.

Parameters: bIncludeReadOnly—optional, default to True.

Python Syntax	GetPropNames ()
Python Example	<pre> oOptModule.GetPropNames () oOptModule.GetPropNames (True) oOptModule.GetPropNames (False) </pre>

GetPropValue [Optimetrics]

Returns the property value for a setup property.

UI Access	NA		
Parameters	Name	Type	Description
	property-path		a child object's property path. See property path discussion here .
Return Value	Returns the value of an setup property.		

Python Syntax	GetPropValue(propPath)		
Python Example	<code>oOptModule.GetPropValue("OptimizationSetup1\Optimizer") //get the optimizer name for OptimizationSetup1</code>		
	<code>oOptModule.GetPropValue("OptimizationSetup1\Optimizer\Choices") //Get the menu property's menu items. In this case all Optimizer names.</code>		

VB Syntax	GetPropValue()		
VB Example	<code>GetPropValue("Optimetrics/OptimizationSetup1/Enabled")</code>		
	Returns True if Enabled, or False if disabled.		

GetSetupNames [Optimetrics]

Gets a list of Optimetrics setup names

UI Access	NA		
Parameters	Name	Type	Description
	None		

Return Value	IAnsoftCollectionObj – a collection of Optimetrics setup names

Python Syntax	GetSetupNames()
Python Example	<pre>oModule = oDesign.GetModule("Optimetrics") setupNames = oModule.GetSetupNames()</pre>

VB Syntax	GetSetupNames()
VB Example	<pre>Set oModule_opt = oDesign.GetModule("Optimetrics") Set opt_setup_list = oModule.GetSetupNames() numsetups = setupnames.Count</pre>

GetSetupNamesByType [Optimetrics]

Gets a list of Optimetrics setup names by type.

UI Access	NA		
Parameters	Name	Type	Description
	<Optimetrics type>	String	Examples: parametric, optimization, statistical, sensitivity
Return Value	Array of Optimetrics setup names of the given type.		

Python Syntax	GetSetupNamesByType (<Optimetrics type>)

Python Example	<pre>for name in oModule.GetSetupNamesByType("optimization") AddInfoMessage(str(name))</pre>

VB Syntax	GetSetupNamesByType <Optimetrics type>
VB Example	<pre>For each name in oModule.GetSetupNamesByType("optimization") MsgBox name Next</pre>

ImportSetup

Import an Optimetric setup from a file.

UI Access	NA		
Parameters	Name	Type	Description
	<SetupTypeName>	String	Must be one of "OptiParametric", "OptiOptimization", "OptiSensitivity", "OptiStatistical", or "OptiDesignExplorer".
	<SetupInfo>	Array	Array("NAME:<SetupName>", "FilePath") <SetupName> Type: <string> Name of the setup. <FilePath> Type : <string: file path>

			Must be a valid file path and name.
Return Value	None		

Python Syntax	ImportSetup (<SetupTypeName>, <SetupInfo>)		
Python Example	<pre>oModule.ImportSetup ("OptiStatistical", ["NAME:StatisticalSetup1", "c:/importdir/mySetupInfoFile"])</pre>		

VB Syntax	ImportSetup <SetupTypeName>, <SetupInfo>		
VB Example	<pre>oModule.ImportSetup "OptiStatistical", Array("NAME:StatisticalSetup1", "c:/importdir/mySetupInfoFile")</pre>		

PasteSetup [Optimetrics]

Pastes the specified Optimetrics setup.

UI Access	NA		
Parameters	Name	Type	Description
	<SetupName>	String	Name of the Setup
Return Value	None		

Python Syntax	PasteSetup (<SetupName>)
Python Example	<code>oModule.PasteSetup ("OptimizationSetup1")</code>

VB Syntax	PasteSetup <SetupName>
VB Example	<code>oModule.PasteSetup "OptimizationSetup1"</code>

RenameSetup [Optimetrics]

Renames the specified Optimetrics setup.

UI Access	Right-click the setup in the project tree, and then click Rename on the shortcut menu.		
Parameters	Name	Type	Description
	<OldName>	String	The name that needs to be replaced
	<NewName>	String	Replacement name
Return Value	None		

Python Syntax	RenameSetup (<OldName> <NewName>)
Python Example	<code>oModule.RenameSetup ("OptimizationSetup1" "MyOptimization")</code>

VB Syntax	RenameSetup <OldName> <NewName>
VB Example	<code>oModule.RenameSetup "OptimizationSetup1" "MyOptimization"</code>

SetPropValue [Optimetrics]

Sets the property value for the active Optimetrics setup.

UI Access	Set Property value on Optimetrics objects		
Parameters	Name	Type	Description
	Property path	text string	Setup property path. See discussion of Property Path
	new Value	Text String, Number, or Boolean	New value data type is depending on the property type,
Return Value	True if the property is found and the new value is valid. Otherwise return False.		

Python Syntax	SetPropValue(propPath, newValue)
Python Example	<pre>oOptModule.SetPropValue("ParametricSetup1\Enabled", False) //disable ParametricSetup1 oOptModule.SetPropValue("OptimizationSetup1/Optimizer","Quasi Newton")</pre>

VB Syntax	SetPropValue(propPath, newValue)
VB Example	<code>SetPropValue("ParametricSetup1\Enabled", False)</code>

SolveAllSetup

Solves all Optimetrics setups

UI Access	Right-click on Optimetrics in Project Manager and select Analyze>All from context menu		
Parameters	Name	Type	Description
	isBlocking	Boolean	An optional arg that defaults to <code>true</code> . If it's <code>false</code> , the command immediately returns while the analysis or analyses run in the background. The status of the analyses can be checked with the <code>AreThereSimulationsRunning</code> command.
Return Value	None		

Python Syntax	<code>SolveAllSetup()</code>
Python Example	<code>oModule.SolveAllSetup()</code>

VB Syntax	<code>SolveAllSetup</code>
VB Example	<code>oModule.SolveAllSetup</code>

SolveSetup [Optimetrics]

Solves the specified Optimetrics setup.

UI Access	Right-click the setup in the project tree, and then click Analyze on the shortcut menu.		
Parameters	Name	Type	Description
	<SetupName>	String	Name of the setup to be solved
	isBlocking	Boolean	An optional arg that defaults to <code>true</code> . If it's <code>false</code> , the command immediately returns while the analysis or analyses run in the background. The status of the

			analyses can be checked with the <code>AreThereSimulationsRunning</code> command.
Return Value	None		

Python Syntax	<code>SolveSetup (<SetupName>)</code>		
Python Example	<code>oModule.SolveSetup ("OptimizationSetup1")</code>		

VB Syntax	<code>SolveSetup <SetupName></code>		
VB Example	<code>oModule.SolveSetup "OptimizationSetup1"</code>		

Parametric Script Commands

[EditSetup \[Parametric\]](#)

[ExportParametricSetupTable](#)

[GenerateVariationData \[Parametric\]](#)

[InsertSetup \[Parametric\]](#)

EditSetup [Parametric]

Modifies an existing parametric setup

UI Access	Right-click the setup in the project tree, and then click Properties on the shortcut menu.		
Parameters	Name	Type	Description

	<SetupName>	String	Name of the Setup
	<ParametricParams>	List	List that defines the parameters of the parametric setup; examples are listed below.
Return Value	None		

Python Syntax	EditSetup (<SetupName>, <ParametricParams>)		
Python Example	See EditSetup [Optimization]		

VB Syntax	EditSetup <SetupName>, <ParametricParams>		
VB Example	See EditSetup [Optimization]		

ExportParametricSetupTable

Exports the parametric setup table as a CSV file.

UI Access	Double-click parametric setup. Select Table tab. Click Export .		
Parameters	Name	Type	Description
	<SetupName>	String	Name of the setup.
	<filePath>	String	Full path for file export.
Return Value	None		

Python Syntax	ExportParametricSetupTable (<SetupName>, <filePath>)
Python Example	<pre>oModule.ExportParametricSetupTable('ParametricSetup1', 'E:/Files/ParametricSetup1_ Table.csv')</pre>

VB Syntax	ExportParametricSetupTable <SetupName>, <filePath>
VB Example	<pre>obj.ExportParametricSetupTable "ParametricSetup1", "E:/Files/ParametricSetup1_ Table.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"
```

InsertSetup [Parametric]

Inserts a new parametric setup.

UI Access	Right-click the Optimetrics folder in the project tree, and then click Add> Parametric on the shortcut menu.

Parameters	Name	Type	Description
	<Parametric Params>	Array	Array("NAME:<SetupName>", "SaveFields:=", <SaveField>, <StartingPoint>, "Sim. Setups:=", <SimSetups>, <SweepDefs>, <SweepOps>, Array("NAME:Goals", Array("NAME:Goal", <OptiGoalSpec>), ... Array("NAME:Goal", <OptiGoalSpec>)))
	<SetupName>	String	Name of the parametric setup.
	<SimSetups>	Array of Strings	An array of Twin Builder solution setup names.
	<SweepDefs>	Array	Array("NAME:Sweeps", Array("NAME:SweepDefinition", "Variable:=", <VarName>, "Data:=", <SweepData>, "Synchronize:=", <SyncNum>), ... Array("NAME:SweepDefinition", "Variable:=", <VarName>, "Data:=", <SweepData>, "Synchronize:=", <SyncNum>))
	<SweepData>	String	"<SweepType>, <StartV>, <StopV>, <StepV>"
	<SweepType>	String	The type of sweep data.
	<SyncNum>	Integer	<i>SweepData</i> s with the same value are synchronized.
	<SweepOps>		Array("NAME:Sweep Operations", "<OpType>:=, Array(<VarValue>, ..., <VarValue>), ...

			<OpType>:=, Array(<VarValue>, ..., <VarValue>))
	<OpType>	String	The sweep operation type.
Return Value	None		

Python Syntax	InsertSetup ("OptiParametric", <ParametricParams>)
Python Example	<pre>oModule.InsertSetup("OptiParametric", ["NAME:ParametricSetup1", "SaveFields:=", true, ["NAME:StartingPoint"], "Sim. Setups:=", ["Setup1"], ["NAME:Sweeps", ["NAME:SweepDefinition", "Variable:=", "\$width", "Data:=", "LIN 12mm 17mm 2.5mm", "OffsetF1:=", false, "Synchronize:=", 0</pre>

```
],  
[  
  "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"
```


VB Syntax	InsertSetup "OptiParametric", <ParametricParams>
VB Example	<pre>oModule.InsertSetup "OptiParametric", Array("NAME:ParametricSetup1", _ "SaveFields:=", true, _ Array("NAME:StartingPoint"), _ "Sim. Setups:=", Array("Setup1"), _ Array("NAME:Sweeps", _ Array("NAME:SweepDefinition", _ "Variable:=", "\$width", _ "Data:=", "LIN 12mm 17mm 2.5mm", _ "OffsetF1:=", false, _</pre>

```

"Synchronize:=", 0),
Array("NAME:SweepDefinition", _
"Variable:=", "$length", _
"Data:=", "LIN 8mm 12mm 2mm", _
"OffsetFl:=", false, _
"Synchronize:=", 0)),
Array("NAME:Sweep Operations"), _
Array("NAME:Goals", _
Array("NAME:Goal", _
"Solution:=", "Setup1 : LastAdaptive", _
"Calculation:=", "returnloss", _
"Context:=", "", _
    Array("NAME:Ranges", _
        "Range:=", Array("Var:=", "Freq", "Type:=", "s", _
            "Start:=", "8GHz", "Stop:=", "8GHz"))), _
Array("NAME:Goal", _
"Solution:=", "Setup1 : LastAdaptive", _
"Calculation:=", "reflect", _
"Context:=", "", _
    Array("NAME:Ranges", _

```

	<pre>"Range:=", Array("Var:=", "Freq", "Type:=", "s", _ "Start:=", "8GHz", "Stop:=", "8GHz"))))</pre>

Optimization Script Commands

[EditSetup \[Optimization\]](#)

[InsertSetup \[Optimization\]](#)

EditSetup [Optimization]

Modifies an existing optimization setup.

UI Access	Right-click the setup in the Project Manager tree, and then click Properties from the shortcut menu		
Parameters	Name	Type	Description
	<SetupName>	String	Name of the Setup
	<OptimizationParams>	List	Parameters that define the setup; examples are listed below.
Return Value	None		

Python Syntax	EditSetup (<SetupName>, <OptimizationParams>)
Python Example	<pre>oModule.EditSetup ("OptimizationSetup1", ["NAME:OptimizationSetup1", "UseFastCalculationUpdateAlgo:=", False, "FastCalcOptCtrledByUser:=", False,</pre>

```
"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.EditSetup "OptimizationSetup1", Array("NAME:OptimizationSetup1", "UseFastCalculationUpdateAlgo:=", _ false, "FastCalcOptCtrledByUser:=", false, "IsEnabled:=", true, "SaveSolutions:=", _ false, Array("NAME:StartingPoint"), "Optimizer:=", "Quasi Newton", Array("NAME:AnalysisStopOptions", "StopForNumIteration:=", _ true, "StopForElapsTime:=", false, "StopForSlowImprovement:=", false, "StopForGrdTolerance:=", _ false, "MaxNumIteration:=", 1000, "MaxSolTimeInSec:=", 3600, "RelGradientTolerance:=", _ 0, "MinNumIteration:=", 10), "CostFuncNormType:=", "L2", "PriorPSetup:=", "", "PreSolvePSetup:=", _ true, Array("NAME:Variables"), Array("NAME:LCS"), Array("NAME:Goals", Array("NAME:Goal", "ReportType:=", _ "Standard", "Solution:=", "TR", Array("NAME:SimValueContext", "SimValueContext:=", Array(_ 1, 0, 2, 0, false, false, -1, 1, 0, 1, 1, "", 0, 0)), "Calculation:=", "Time", "Name:=", _ "Time", Array("NAME:Ranges", "Range:=", </pre>

	<pre>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)</pre>

InsertSetup [Optimization]

Use: Inserts a new optimization setup.

UI Access	Right-click the Optimetrics folder in the project tree, and then click Add > Optimization on the shortcut menu.		
Parameters	Name	Type	Description
	<OptimizationParams>	Array	<pre>Array("NAME:<SetupName>", "SaveFields:=", <SaveField>, <StartingPoint>, "Optimizer:=", <Optimizer>, "MaxIterations:=", <MaxIter>, "PriorPSetup:=", <PriorSetup>, "PreSolvePSetup:=", <Preceed>, <OptimizationVars>, <Constraint>, Array("NAME:Goals", Array("NAME:Goal", <OptiGoalSpec>, <OptimizationGoalSpec>), ... Array("NAME:Goal", <OptiGoalSpec>,</pre>

		<code><OptimizationGoalSpec>)),</code> <code>"Acceptable_Cost:=", <AcceptableCost>, "Noise:=",</code> <code><Noise>, "UpdateDesignWhenDone:=", <UpdateDesign></code>
<code><OptimizationVars></code>	Array	<code>Array("NAME:Variables", "VarName:=", Array("i:=",</code> <code><IncludeVar>, "Min:=", <MinV>, "Max:=", <MaxV>,</code> <code>"MinStep:=", <MinStepV>, "MaxStep:=", <MaxStepV>),</code> <code>..... "VarName:=", Array("i:=", <IncludeVar>,</code> <code>"Min:=", <MinV>, "Max:=", <MaxV>,</code> <code>"MinStep:=", <MinStepV>, "MaxStep:=", <MaxStepV>))</code>
<code><MinStepV></code>	VarValue	The minimum step of the variable.
<code><MaxStepV></code>	VarValue	The maximum step of the variable.
<code><AcceptableCost></code>	Double	The acceptable cost value for the optimizer to stop.
<code><Noise></code>	Double	The noise of the design.
<code><UpdateDesign></code>	Boolean	Specifies whether or not to apply the optimal variation to the design after the optimization is done.
<code><OptimizationGoalSpec></code>	Array	<code>"Condition:=", <OptimizationCond>,</code> <code>Array("NAME:GoalValue", "GoalValeType:=",</code> <code><GoalValueType>,</code> <code>"Format:=", <GoalValueFormat>, "bG:=",</code> <code>Array("v:=", <GoalValue>)), "Weight:=", <Weight>)</code>
<code><OptimizationCond></code>	String	Either " <code><=</code> ", " <code>=</code> ", or " <code>>=</code> "
<code><GoalValueType></code>	String	Either "Independent" or "Dependent"
<code><GoalValueFormat></code>	String	Either "Real/Imag" or "Mag/Ang".
<code><GoalValue></code>	String	Value in string. Value can be a real number, complex number, or expression.

Return Value	None

<MinStepV>

Type : <VarValue>

The minimum step of the variable.

<MaxStepV>

Type: <VarValue>

The maximum step of the variable.

<AcceptableCost>

Type: <double>

The acceptable cost value for the optimizer to stop.

<Noise>

Type: <double>

The noise of the design.

<UpdateDesign>

Type: <bool>

Specifies whether or not to apply the optimal variation to the design after the optimization is done.

<OptimizationGoalSpec>

"Condition:=", <OptimizationCond>,

Array("NAME:GoalValue", "GoalValueType:=",

<GoalValueType>,

```
"Format:=", <GoalValueFormat>, "bG:=",  
Array("v:=", <GoalValue>)), "Weight:=", <Weight>)  
<OptimizationCond>  
    Type: <string>  
    Either "<=", "==", or ">="
```

<GoalValueType>

Type: <string>

Either "Independent" or "Dependent"

<GoalValueFormat>

Type:<string>

Either "Real/Imag" or "Mag/Ang".

<GoalValue>

Type: <string>

Value in string. Value can be a real number, complex number, or expression.

Python Syntax	InsertSetup ("OptiOptimization", <OptimizationParams>)
Python Example	<pre>oModule.InsertSetup("OptiOptimization", ["NAME:OptimizationSetup1", "SaveFields:=", false, _ ["NAME:StartingPoint", "\$length:=", "8mm",</pre>

```

"$width:=", "14.5mm"],
"Optimizer:=", "Quasi Newton",
"MaxIterations:=", 100,
"PriorPSetup:=", "ParametricSetup1",
"PreSolvePSetup:=", true,
["NAME:Variables",
"$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:=", "",

```

	<pre> ["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]) </pre>

VB Syntax	InsertSetup "OptiOptimization", <OptimizationParams>

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, _

```

	<code>"UpdateDesignWhenDone:=", true)</code>

Sensitivity Script Commands

[EditSetup \[Sensitivity\]](#)

[InsertSetup \[Sensitivity\]](#)

EditSetup [Sensitivity]

Modifies an existing sensitivity setup.

UI Access	Right-click the setup in the project tree, and then click Properties on the shortcut menu		
Parameters	Name	Type	Description
	<SetupName>	String	Name of the Setup
Return Value	None		

Python Syntax	<code>EditSetup (<SetupName>, <SensitivityParams>)</code>
Python Example	<pre>oModule.EditSetup("OptimizationSetup1", ["NAME:OptimizationSetup1", "UseFastCalculationUpdateAlgo:=", False, "FastCalcOptCtrledByUser:=", False, "IsEnabled:=", True, "SaveSolutions:=", False,</pre>


```
[  
  "NAME:StartingPoint"  
],  
  "Optimizer:=", "Quasi Newton",  
  [  
    "NAME:AnalysisStopOptions",  
    "StopForNumIteration:=", True,  
    "StopForElapsTime:=", False,  
    "StopForSlowImprovement:=", False,  
    "StopForGrdTolerance:="          , False,  
    "MaxNumIteration:=", 1001,  
    "MaxSolTimeInSec:=", 3600,  
    "RelGradientTolerance:=", 0,  
    "MinNumIteration:=", 10  
  ],  
  "CostFuncNormType:=", "L2",  
  "PriorPSetup:=", "",  
  "PreSolvePSetup:=", True,  
  [  
    ]
```

```
"NAME:Variables"

],

[
  "NAME:LCS"
],

[
  "NAME:Goals",
  [
    "NAME:Goal",
    "ReportType:=", "Standard",
    "Solution:=", "TR3",
    [
      "NAME:SimValueContext",
      "SimValueContext:=", [1,0,2,0,False,False,-1,1,0,1,1,"",0,0]
    ],
    "Calculation:=", "mag(DIFF1.VAL) ",
    "Name:=", "DIFF1.VAL",
    [
      "NAME:Ranges",
      "Range:=", [
        "Var:=", "Time","Type:=", "a"]
```

```

],
"Condition:=", "==",
[
"NAME:GoalValue",
"GoalValueType:=", "Independent",
"Format:=", "Real/Imag",
"bG:=", ["v:=", "[1;]"]
],
"Weight:=", "[1;]"
]
],
"Acceptable_Cost:=", 0,
"Noise:=", 0.0001,
"UpdateDesign:=", False,
"UpdateIteration:=", 5,
"KeepReportAxis:=", True,
"UpdateDesignWhenDone:=", True
])

```

VB Syntax	EditSetup <SetupName>, <SensitivityParams>
VB Example	<pre> oModule.EditSetup("OptimizationSetup1", Array("NAME:OptimizationSetup1", "UseFastCalculationUpdateAlgo:=", False, "FastCalcOptCtrledByUser:=", False, "IsEnabled:=", True, "SaveSolutions:=", False, Array("NAME:StartingPoint"), "Optimizer:=", "Quasi Newton", Array("NAME:AnalysisStopOptions", "StopForNumIteration:=", True, "StopForElapsTime:=", False, "StopForSlowImprovement:=", False, "StopForGrdTolerance:=", False, "MaxNumIteration:=", 1001, "MaxSolTimeInSec:=", 3600, "RelGradientTolerance:=", 0, </pre>

```
"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,
```

	<pre>"UpdateDesign:=", False, "UpdateIteration:=", 5, "KeepReportAxis:=", True, "UpdateDesignWhenDone:=", True))</pre>

InsertSetup [Sensitivity]

Inserts a new sensitivity setup.

UI Access	Right-click Optimetrics in the project tree, and then click Add>Sensitivity on the shortcut menu.		
Parameters	Name	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>)
	<SensitivityVars>	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.
Return Value	None		

Python Syntax	InsertSetup ("OptiSensitivity", <SensitivityParams>)
Python Example	<pre>oModule.InsertSetup("OptiSensitivity", _ ["NAME:SensitivitySetup1",_ "SaveFields:=", true,_ ["NAME:StartingPoint"], _ "MaxIterations:=", 20,_ "PriorPSetup:=", "",_ "PreSolvePSetup:=", true, _ ["NAME:Variables"], _ ["NAME:LCS"],_ "NAME:Goals",_</pre>


```
[ "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"]]]], _
"Primary Goal:=", 1, _
"PrimaryError:=", 0.001])
```

VB Syntax	InsertSetup "OptiSensitivity", <SensitivityParams>
VB Example	<pre> 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"))), _ </pre>

	<pre>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)</pre>

Statistical Script Commands

[EditSetup \[Statistical\]](#)

[InsertSetup Statistical](#)

EditSetup [Statistical]

Modifies an existing statistical setup.

UI Access	Right-click the setup in the project tree, and clickProperties on the shortcut menu.		
Parameters	Name	Type	Description
	<SetupName>	String	Name of the Setup

Return Value	None

Python Syntax	EditSetup (<SetupName>, <StatisticalParams>)
Python Example	See EditSetup [Optimization]

VB Syntax	EditSetup <SetupName>, <StatisticalParams>
VB Example	See EditSetup [Optimization]

Example:

```

Dim oAnsoftApp
Dim oDesktop
Dim oProject
Dim oDesign
Dim oEditor
Dim oModule

Set oAnsoftApp = CreateObject("Ansoft.ElectronicsDesktop")
Set oDesktop = oAnsoftApp.GetAppDesktop()
oDesktop.RestoreWindow

Set oProject = oDesktop.SetActiveProject("optiguide")
Set oDesign = oProject.SetActiveDesign("HFSSModel1")

```

```
Set oModule = oDesign.GetModule("Optimetrics")

oModule.EditSetup "StatisticalSetup1", Array("NAME:StatisticalSetup1",
Array("NAME:ProdOptiSetupData",
"SaveFields:=", true, "CopyMesh:=", false),
Array("NAME:StartingPoint", "$length:=", "7.824547736mm",
"$width:=", "14.8570192mm"),
"MaxIterations:=", 50,
"PriorPSetup:=", "",

Array("NAME:Variables",
"$length:=", Array("i:=", true,
"int:=", false,
"Dist:=", "Uniform",
"Tol:=", "10%", "StdD:=", "0.2mm", "Min:=", "-3",
"Max:=", "3", "Shape:=", "1", "Scale:=", "0.04mm",
"Location:=", "0.4mm",
"Dataset:=", "", "LatinHypercube:=", "true", "VarMin:=", "0.2mm", "VarMax:=", "0.6mm", "Prob:=",
"0.01",
"Mean:=", "0.4mm"),

"$width:=", Array("i:=", true,
"int:=", false,
"Dist:=", "Gaussian",
"Tol:=", "10%",
"StdD:=", "0.2mm",
"Min:=", "-3", "Max:=", "3",
"Shape:=", "1",
"Scale:=", "0.04mm",
"Location:=", "0.4mm",
"Dataset:=", "",
"LatinHypercube:=", "true",
"VarMin:=", "0.2mm", "VarMax:=", "0.6mm",
"Prob:=", "0.02",
```

```
"Mean:=", "0.4mm"))),  
  
Array("NAME:Goals", Array("NAME:Goal",  
"ReportType:=", "Modal Solution Data",  
"Solution:=", "Setup1 : PortOnly",  
Array("NAME:SimValueContext", "Domain:=", "Sweep"),  
"Calculation:=", "returnloss",  
"Name:=", "returnloss",  
  
Array("NAME:Ranges",  
"Range:=", Array("Var:=", "Freq",  
"Type:=", "s",  
"Start:=", "8.2GHz", "Stop:=", "0")))),  
  
Array("NAME:Goal",  
"ReportType:=", "Modal Solution Data",  
"Solution:=", "Setup1 : PortOnly",  
Array("NAME:SimValueContext",  
"Domain:=", "Sweep"),  
"Calculation:=", "reflect",  
"Name:=", "reflect",  
Array("NAME:Ranges",  
"Range:=", Array("Var:=", "Freq",  
"Type:=", "s",  
"Start:=", "8.2GHz", "Stop:=", "0")))))))
```

InsertSetup [Statistical]

Inserts a new statistical setup.

UI Access	Right-click Optimetrics in the project tree, and then click Add>Statistical on the shortcut menu.		
Parameters	Name	Type	Description
	<StatisticalParams>	Array	Array("NAME:<SetupName>", "SaveFields:=", <SaveField>, <StartingPoint>, "MaxIterations:=",

			<code><MaxIter>, "PriorPSetup=", <PriorSetup>, "PreSolvePSetup=", <Preceed>, <StatisticalVars>, Array("NAME:Goals", Array("NAME:Goal", <OptiGoalSpec>), ..., Array("NAME:Goal", <OptiGoalSpec>)))</code>
	<StatisticalVars>	Array	<code>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>))</code>
	<DistType>	String	Distribution can be "Gaussian" or "Uniform".
	<Tolerance>	VarValue	The tolerance for the variable when distribution is Uniform
	<StdD>	VarValue	The standard deviation for the variable when distribution is Gaussian.
	<MinCutoff>	Double	The minimum cut-off for the variable when distribution is Gaussian.
	<MaxCutoff>	Double	The maximum cut-off for the variable when distribution is Gaussian.
Return Value	None		

Python Syntax	InsertSetup ("OptiStatistical", <StatisticalParams>)
Python Example	<pre> oModule.InsertSetup("OptiStatistical", _ ["NAME:StatisticalSetup1", _ "SaveFields:=", true, _ ["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", _ </pre>

	<pre> "Calculation:=", "reflect", _ "Context:=", "", _ ["NAME:Ranges", _ "Range:=", ["Var:=", "Freq", "Type:=", _ "s", "Start:=", "8GHz", "Stop:=", "8GHz"]]]]) </pre>

VB Syntax	InsertSetup "OptiStatistical", <StatisticalParams>
VB Example	<pre> oModule.InsertSetup "OptiStatistical", _ Array("NAME:StatisticalSetup1", _ "SaveFields:=", true, _ Array("NAME:StartingPoint"), _ "MaxIterations:=", 50, _ "PriorPSetup:=", "", _ Array("NAME:Variables"), _ Array("NAME:Goals", _ Array("NAME:Goal", _ "Solution:=", "Setup1 : LastAdaptive", _ "Calculation:=", "returnloss", _ </pre>

```

"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"))))

```

Example:

```

Dim oAnsoftApp
Dim oDesktop
Dim oProject
Dim oDesign
Dim oEditor
Dim oModule

```

```
Set oAnsoftApp = CreateObject("Ansoft.ElectronicsDesktop")

Set oDesktop = oAnsoftApp.GetAppDesktop()

oDesktop.RestoreWindow

Set oProject = oDesktop.SetActiveProject("OptimTee")

Set oDesign = oProject.SetActiveDesign("TeeModel")

oDesign.ChangeProperty Array("NAME:AllTabs",
Array("NAME:LocalVariableTab",
Array("NAME:PropServers", "LocalVariables"),
Array("NAME:ChangedProps",
Array("NAME:offset",
Array("NAME:Statistical", "Included:=", true))))))

Set oModule = oDesign.GetModule("Optimetrics")

oModule.InsertSetup "OptiStatistical", Array("NAME:StatisticalSetup1",
Array("NAME:ProdOptiSetupData",
"SaveFields:=", false, "CopyMesh:=", false),
Array("NAME:StartingPoint", "offset:=", "0in"),
"MaxIterations:=", 50, "PriorPSetup:=", "",

Array("NAME:Variables",
"offset:=", Array("i:=", true,
"int:=", false,
"Dist:=", "Gaussian",
"Tol:=", "10%",
"StdD:=", ".5in",
"Min:=", "-3",
"Max:=", "3",
"Shape:=", "1",
"Scale:=", "0in",
```

```
"Location:=", "0in",
"Dataset:=", "",
"LatinHypercube:=", "true",
"VarMin:=", "-1in",
"VarMax:=", "1in",
"Prob:=", "0.01",
"Mean:=", "0in")),

Array("NAME:Goals", Array("NAME:Goal",
"ReportType:=", "Modal Solution Data",
"Solution:=", "Setup1 : LastAdaptive", Array("NAME:SimValueContext"),

"Calculation:=", "Power11",
"Name:=", "Power11",
Array("NAME:Ranges",
"Range:=", Array("Var:=", "Freq", "Type:=", "d",
"DiscreteValues:=", "10GHz")))))
```

For Q3D Extractor and Circuit the command details are as follows:

Inserts a new statistical setup.

Command: Right-click **Optimetrics** in the project tree, and then click **Add>Statistical** on the shortcut menu.

Syntax: InsertSetup "OptiStatistical", <StatisticalParams>

Return Value: None

Parameters: <StatisticalParams>

```
Array("NAME:<SetupName>", "SaveFields:=",
<SaveField>, <StartingPoint>, "MaxIterations:=",
<MaxIter>, "PriorPSetup:=", <PriorSetup>,
"PreSolvePSetup:=", <Preceed>, <StatisticalVars>,
```

```
Array("NAME:Goals", Array("NAME:Goal",  
<OptiGoalSpec>), ..., Array("NAME:Goal",  
<OptiGoalSpec>))),  
<StatisticalVars>  
Array("NAME:Variables",  
  "VarName:=", Array("i:=", <IncludeVar>, "Dist:=",  
    <DistType>, "Tol:=", <Tolerance>,  
    "StdD:=", <StdD>, "Min:=", <MinCutoff>, "Max:=",  
    <MaxCutoff>, ...  
  "VarName:=", Array("i:=", <IncludeVar>, "Dist:=",  
    <DistType>, "Tol:=", <Tolerance>, "StdD:=",  
    <StdD>, "Min:=", <MinCutoff>, "Max:=",  
    <MaxCutoff>))
```

Parameters:

<DistType>

Type : <string>

Distribution can be "Gaussian" or "Uniform".

<Tolerance>

Type: <VarValue>

The tolerance for the variable when distribution is Uniform.

<StdD>

Type: <VarValue>

The standard deviation for the variable when distribution is Gaussian.

<MinCutoff>

Type: <double>

The minimum cut-off for the variable when distribution is Gaussian.

<MaxCutoff>

Type: <double>

The maximum cut-off for the variable when distribution is Gaussian.

Example: oModule.InsertSetup "OptiStatistical", _

Array("NAME:StatisticalSetup1", _

"SaveFields:=", true, _

Array("NAME:StartingPoint"), _

"MaxIterations:=", 50, _

"PriorPSetup:=", "", _

Array("NAME:Variables"), _

Array("NAME:Goals", _

Array("NAME:Goal", _

"Solution:=", "Setup1 : LastAdaptive", _

"Calculation:=", "returnloss", _

"Context:=", "", _

```
Array("NAME:Ranges", _  
  "Range:=", Array("Var:=", "Freq", _  
    "Type:=", "s", _  
    "Start:=", "8GHz", "Stop:=", "8GHz"))), _  
  Array("NAME:Goal", _  
    "Solution:=", "Setup1 : LastAdaptive", _  
    "Calculation:=", "reflect", _  
    "Context:=", "", _  
    Array("NAME:Ranges", _  
      "Range:=", Array("Var:=", "Freq", "Type:=", _  
        "s", "Start:=", "8GHz", "Stop:=", "8GHz")))))
```

15 - Field Overlays Module Script Commands

Field overlay commands should be executed by the Field Overlays module, which is called "FieldsReporter" in scripts.

Set oModule = oDesign.GetModule("FieldsReporter")

oModule.CommandName <args>

[AddMarkerToPlot](#)

[CreateFieldPlot](#)

[DeleteFieldPlot](#)

[ExportMarkerTable](#)

[ExportPlotImageWithViewToFile](#)

[GetFieldPlotName](#)

[ModifyFieldPlot](#)

[RenameFieldPlot](#)

[RenamePlotFolder](#)

[SetFieldPlotSettings](#)

[SetPlotFolderSettings](#)

[UpdateAllFieldsPlots](#)

[UpdateQuantityFieldsPlots](#)

AddMarker[Fields Reporter]

Adds a marker to the current fields plot.

UI Access	Right-click Field Overlays > Plot Fields > Marker > Add Marker .		
Parameters	Name	Type	Description
	<Position>	Array	Array containing X, Y and Z coordinates.
Return Value	None.		

Python Syntax	AddMarker(<Position>)
Python Example	<pre>oModule.AddMarker(["-267.007756778494mil", "-640.898759461403mil", "9.09494701772928e-13mil"])</pre>

VB Syntax	AddMarker <Position>
VB Example	<pre>oModule.AddMarker _ Array("-267.007756778494mil", _ "-640.898759461403mil", _ "9.09494701772928e-13mil")</pre>

AddMarkerToPlot

Adds a marker to a trace on a named field plot.

UI Access	N/A		
Parameters	Name	Type	Description
	<Location>	Array	Array of strings containing X.Y, and Z coordinates for the marker.
	<PlotName>	String	Name of the field plot.
Return Value	None.		

Python Syntax	AddMarkerToPlot(<Location>, <PlotName>)		
Python Example	<pre>oModule.AddMarkerToPlot (["0.290455877780914in", "-0.616900205612183in", "1.77635683940025e-015in"], "Mag_H1") oModule.AddMarkerToPlot (["-0.317279517650604in", "1.22481322288513in", "0in"], "Mag_H1")</pre>		

VB Syntax	AddMarkerToPlot <Location>, <PlotName>		
VB Example	<pre>oModule.AddMarkerToPlot _ Array("0.290455877780914in", _ "-0.616900205612183in", "1.77635683940025e-015in"), _</pre>		

	<pre>"Mag_E1" oModule.AddMarkerToPlot _ Array("-0.317279517650604in", _ "1.22481322288513in", "0in"), _ "Mag_E1"</pre>

AddNamedExpression

Creates a named expression using the expression at the top of the stack.

UI Access	Click Add in the Fields Calculator window.		
Parameters	Name	Type	Description
	<ExprName>	String	Name for the new named expression.
	<FieldType>	String	Type of field.
Return Value	None		

Python Syntax	AddNamedExpression(<ExprName>, <FieldType>)
Python Example	<pre>oModule.AddNamedExpression("Mag_JxE", "Fields")</pre>

VB Syntax	AddNamedExpression <ExprName>, <FieldType>
VB Example	<pre>oModule.AddNamedExpression "Mag_JxE", "Fields"</pre>

CalcOp

Performs a calculator operation.

UI Access	Operation commands like Mag , + , etc.		
Parameters	Name	Type	Description
	<OpString>	String	The text on the corresponding calculator button.
Return Value	None.		

Python Syntax	CalcOp(<OpString>)
Python Example	oModule.CalcOp ("+")

VB Syntax	CalcOp <OpString>
VB Example	oModule.CalcOp "+"

CalcStack

Performs an operation on the stack.

UI Access	Stack operation buttons such as Push and Pop .		
Parameters	Name	Type	Description
	<OpString>	String	The text on the corresponding calculator button.
Return Value	None.		

Python Syntax	<code>CalcStack(<OpString>)</code>
Python Example	<code>oModule.CalcStack("push")</code>

VB Syntax	<code>CalcStack <OpString></code>
VB Example	<code>oModule.CalcStack "push"</code>

CalculatorRead

Gets a register file and applies it to the calculator stack.

UI Access	Click Read... in the Fields Calculator dialog.		
Parameters	Name	Type	Description
	<FileName>	String	Path to and including name of input register file.
	<SoluName>	String	Specified solution to read in.
	<FieldType>	String	Type of specified field.
	<VarArray>	Array	Array of variable names, value pairs.
Return Value	None.		

Python Syntax	<code>CalculatorRead(<FileName>, <SoluName>, <FieldType>, <VarArray>)</code>
Python Example	<code>oModule.CalculatorRead(</code>

	<pre>"c:\example.reg", "Setup1: LastAdaptive", "Fields", ["Freq=", "10GHz", "Phase=", "0deg"])</pre>

VB Syntax	CalculatorRead <FileName>, <SoluName>, <FieldType>, <VarArray>
VB Example	<pre>oModule.CalculatorRead _ "c:\example.reg", _ "Setup1: LastAdaptive", "Fields", _ Array("Freq=", "10GHz", "Phase=", "0deg")</pre>

CalculatorWrite

Writes contents of top register to file.

UI Access	Click Write... in the Fields Calculator window.		
Parameters	Name	Type	Description
	<OutputFilePath>	String	Path to and including name of output register file.
	<SoluNameArray>	Array	Array of strings containing solution names.
	<VarArray>	Array	Array of variable names, value pairs.
Return Value	None		

Python Syntax	CalculatorWrite(<OutputFilePath>, <SoluNameArray>, <VarArray>)
Python Example	oModule.CalculatorWrite("C:\test.reg",

	<pre>["Solution:=", "Setup1 : LastAdaptive"], ["Freq:=", "1GHz", "Phase:=", "0deg"])</pre>

VB Syntax	<code>CalculatorWrite <OutputFilePath>, <SoluNameArray>, <VarArray></code>
VB Example	<pre>oModule.CalculatorWrite "C:\test.reg", _ Array("Solution:=", "Setup1 : LastAdaptive"), _ Array("Freq:=", "1GHz", "Phase:=", "0deg")</pre>

ChangeProperty

Changes the properties of a field plot marker.

UI Access	Select a field plot marker in the marker table.		
Parameters	Name	Type	Description
	<propertyArgs>	Array	Structured array.
Return Value	None.		

Python Syntax	<code>ChangeProperty(<propertyArgs>)</code>

Python Example

```
oDesign.ChangeProperty(  
    [  
        "NAME:AllTabs",  
        [  
            "NAME:FieldPlotMarkerDataTab",  
            [  
                "NAME:PropServers",  
                "FieldsReporter:FieldsPlotMarker:m1"  
            ],  
            [  
                "NAME:ChangedProps",  
                [  
                    "NAME:Name",  
                    "Value:=" , "Temp"  
                ]  
            ]  
        ]  
    ])  
oDesign.ChangeProperty(  
    [  
        "NAME:AllTabs",  
        [  
            "NAME:FieldPlotMarkerDataTab",  
            [  
                "NAME:PropServers",  
                "FieldsReporter:FieldsPlotMarker:Temp"  
            ],  
            [  
                "NAME:ChangedProps",
```



```

                                [
                                    "NAME:Display Value",
                                    "Value:="                                , True
                                ]
                            ]
                        ])
oDesign.ChangeProperty(
    [
        "NAME:AllTabs",
        [
            "NAME:FieldPlotMarkerDataTab",
            [
                "NAME:PropServers",
                "FieldsReporter:FieldsPlotMarker:Temp"
            ],
            [
                "NAME:ChangedProps",
                [
                    "NAME:Display Name",
                    "Value:="                                , False
                ]
            ]
        ]
    ]
)
oDesign.ChangeProperty(
    [
        "NAME:AllTabs",
        [

```

```

Value:="                                , True
                                ]
                                ]
                                ]
    ])
oDesign.ChangeProperty(
[
    "NAME:AllTabs",
    [
        "NAME:FieldPlotMarkerDataTab",
        [
            "NAME:PropServers",
            "FieldsReporter:FieldsPlotMarker:Temp"
        ],
        [
            "NAME:ChangedProps",
            [
                "NAME:Color",
                "R:="                                , 255,
                "G:="                                , 255,
                "B:="                                , 255
            ]
        ]
    ]
])
oDesign.ChangeProperty(
[
    "NAME:AllTabs",
    [
        "NAME:FieldPlotMarkerDataTab",
        [
            "NAME:PropServers",
            "FieldsReporter:FieldsPlotMarker:Temp"
        ],
    ],

```


VB Syntax	ChangeProperty <propertyArgs>
VB Example	<pre> oDesign.ChangeProperty Array("NAME:AllTabs", Array("NAME:FieldPlotMarkerDataTab", Array("NAME:PropServers", _ "FieldsReporter:FieldsPlotMarker:m1"), Array("NAME:ChangedProps", Array("NAME:Display Value", "Value:=", _ true)))) oDesign.ChangeProperty Array("NAME:AllTabs", Array("NAME:FieldPlotMarkerDataTab", Array("NAME:PropServers", _ "FieldsReporter:FieldsPlotMarker:m1"), Array("NAME:ChangedProps", Array("NAME:Name", "Value:=", _ "Temp")))) oDesign.ChangeProperty Array("NAME:AllTabs", Array("NAME:FieldPlotMarkerDataTab", Array("NAME:PropServers", _ "FieldsReporter:FieldsPlotMarker:Temp"), Array("NAME:ChangedProps", Array("NAME:Display Name", "Value:=", _ false)))) oDesign.ChangeProperty Array("NAME:AllTabs", Array("NAME:FieldPlotMarkerDataTab", Array("NAME:PropServers", _ "FieldsReporter:FieldsPlotMarker:Temp"), Array("NAME:ChangedProps", Array("NAME:Display Name", "Value:=", _ true)))) oDesign.ChangeProperty Array("NAME:AllTabs", Array("NAME:FieldPlotMarkerDataTab", Array("NAME:PropServers", _ "FieldsReporter:FieldsPlotMarker:Temp"), Array("NAME:ChangedProps", Array("NAME:Color", "R:=", _ 255, "G:=", 255, "B:=", 255)))) oDesign.ChangeProperty Array("NAME:AllTabs", Array("NAME:FieldPlotMarkerDataTab", Array("NAME:PropServers", _ "FieldsReporter:FieldsPlotMarker:Temp"), Array("NAME:ChangedProps", Array("NAME:Symbol Size Scale", "Value:=", _ "1")))) </pre>

ChangeGeomSettings

Changes the line discretization setting.

UI Access	In the Fields Calculator dialog box, click on Geom Settings...		
Parameters	Name	Type	Description
	<LineDiscr>	Integer	Line discretization value.
Return Value	None.		

Python Syntax	ChangeGeomSettings(<LineDiscr>)
Python Example	oModule.ChangeGeomSettings(500)

VB Syntax	ChangeGeomSettings <LineDiscr>
VB Example	oModule.ChangeGeomSettings 500

ClcEval

Evaluates the expression at the top of the stack using the provided solution name and variable values.

UI Access	Click Eval in the Fields Calculator dialog.		
Parameters	Name	Type	Description
	<SoluName>	String	Name of specified solution.
	<VarArray>	Array	Array of variable name, value pairs.
	<FieldType>	String	Optional. Type of specified field.

Return Value	None

Python Syntax	<code>ClcEval(<SoluName>, <VarArray>, [Optional <FieldType>])</code>
Python Example	<pre>oModule.ClcEval("Setup1: LastAdaptive", ["Freq:", "10GHz", "Phase:", "0deg"])</pre>

VB Syntax	<code>ClcEval <SoluName>, <VarArray>, [Optional <FieldType>]</code>
VB Example	<pre>oModule.ClcEval "Setup1: LastAdaptive", _ Array ("Freq:", "10GHz", _ "Phase:", "0deg")</pre>

ClcMaterial

Performs a material operation on the top stack element.

UI Access	Click Matl... in the Fields Calculator dialog.		
Parameters	Name	Type	Description
	<MatString>	String	The material property to apply.
	<OpString>	String	Name of operation. Possible values are "mult", or "div".
Return Value	None.		

Python Syntax	<code>ClcMaterial(<MatString>, <OpString>)</code>
Python Example	<code>oModule.ClcMaterial("Permeability (mu)" "mult")</code>

VB Syntax	<code>ClcMaterial <MatString>, <OpString></code>
VB Example	<code>oModule.ClcMaterial "Permeability (mu)" "mult"</code>

ClcMaterialValue

Shows the value of the material property without performing any operation.

UI Access	Select None in the Material Operation dialog.		
Parameters	Name	Type	Description
	<MaterialName>	String	Name of specified material property.
Return Value	None.		

Python Syntax	<code>ClcMaterialValue(<MaterialName>)</code>
Python Example	<code>oModule.ClcMaterialValue("Mass Density")</code>

VB Syntax	<code>ClcMaterialValue <MaterialName></code>
VB Example	<code>oModule.ClcMaterialValue "Mass Density"</code>

ClearAllMarkers[Fields Reporter]

Clears all markers in the current fields overlay plot.

UI Access	Right-click Field Overlays > Plot Fields > Marker > Clear All .
Parameters	None.
Return Value	None.

Python Syntax	ClearAllMarkers()
Python Example	<code>oModule.ClearAllMarkers()</code>

VB Syntax	ClearAllMarkers
VB Example	<code>oModule.ClearAllMarkers</code>

ClearAllNamedExpr

Clears all user-defined named expressions from the list.

UI Access	Click ClearAll in the Fields Calculator dialog.
Parameters	None.
Return Value	None.

Python Syntax	ClearAllNamedExpr()
Python Example	<code>oModule.ClearAllNamedExpr()</code>

VB Syntax	ClearAllNamedExpr
VB Example	<code>oModule.ClearAllNamedExpr</code>

CopyNamedExprToStack

Copies the named expression selected to the calculator stack.

UI Access	Select a named expression and then click Copy to stack .		
Parameters	Name	Type	Description
	<ExprName>	String	Name of the expression to be copied to the top of the stack.
Return Value	None.		

Python Syntax	CopyNamedExprToStack(<ExprName>)
Python Example	<code>oModule.CopyNamedExprToStack("Mag_JxE")</code>

VB Syntax	CopyNamedExprToStack <ExprName>
VB Example	<code>oModule.CopyNamedExprToStack "Mag_JxE"</code>

CreateFieldPlot

Note:

Use in conjunction with [GetGeometryIdsForNetLayerCombinations](#) and `GetGeometryIdsForAllNetLayerCombinations`.

- `GetGeometryIdsForNetLayerCombinations` returns ID numbers of all faces and edges of the active design related to the current target combination of nets and layers.
- `GetGeometryIdsForAllNetLayerCombinations` returns ID numbers for all possible combinations of nets and layers it is possible to choose.

Use: Creates a field/mesh plot "Field" or a visual ray trace ("VRT") plot.

Command: **Mechanical > 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"

A space is required on either side of the colon (:) character. If omitted, the plot will not be created.

QuantityName:

Type of plot to create. Possible values vary according to the solver.

Mesh plots: "Mesh"

***Mechanical* Field plots include**

(dependent on the solution type and whether there is trace mapping):

```
"Mag_Displacement", "Displacement_Vector", "Temperature",  
"HeatFlux", "Mag_HeatFlux", "Surface Loss Density",  
"Volume Loss Density", "Linked Heat Transfer Coefficient",  
"Equivalent Stress", "Equivalent Strain",  
"Thermal Conductivity X", "Thermal Conductivity Y",  
"Thermal Conductivity Z", "Youngs Modulus X",  
"Youngs Modulus Y", "Youngs Modulus Z", "Poisson Ratio XY",  
"Poisson Ratio YZ", "Poisson Ratio XZ", "Thermal Expansion X",  
"Thermal Expansion Y", "Thermal Expansion Z",  
"Shear Modulus X", "Shear Modulus Y", "Shear Modulus Z"
```

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", "Jsrf", "SAR Field",  
"Jc", "Surface-Loss", "QSurf", "Temperature", "Energy",  
"Average-Surface-Loss-Density", "Dielectric_Loss",  
"MeshPlots", "Heat Flux", and "Displacement"
```

UserSpecifyName

0 if default name for plot is used, 1 otherwise.

Not needed. <PlotName> will be respected regardless of whether this flag is set.

UserSpecifyFolder

0 if default folder for plot is used, 1 otherwise.

Not needed. The specified PlotFolder will be respected regardless of whether this flag is set.

IntrinsicVar:

Formatted string that specifies the frequency and phase at which to make the plot.

For example: "Freq='1GHz' Phase='30deg' "

PlotGeomInfo:

Creating field plot on selected layer-net pairs:

For example, Maxwell: "PlotGeomInfo:=", [1, "Volume", "ObjList",
2, "LC1_1:Top", "LC1_1:Top#L1"]

HFSS example with "Plot on surface" option is not checked:

```
"PlotGeomInfo:=", [1, "Volume", "LayerNets", 2, "Top", 2,  
"net1", "net2", "Ground", 1, "GND"]
```

The command creates field plot on two layer-nets combinations. It starts with type "Volume", and subtype "LayerNets", followed by the number of layer-nets combinations which is 2 in this example.

The two layer-nets combinations are (1) "Top" layer, two nets, net names are "net1" and "net2", i.e., [Top, net1] pair, and [Top, net2] pair. (2) "Ground" layer and one net, net name is "GND", i.e., [Ground, GND] pair.

HFSS example with "Plot on surface" option checked:

```
"PlotGeomInfo:=", [1,"Surface","LayerNetsExtFace",2,
"Top",2,"net1","net2","_Ground",1,"GND"]
```

The command creates field plot on two layer-nets combinations. It starts with type "Surface", and subtype "LayerNetsExtFace", followed by the number of layer-nets combinations which is 2 in this example.

The two layer-nets combinations are (1) "Top" layer, two nets, net names are "net1" and "net2", i.e., [Top, net1] pair, and [Top, net2] pair. (2) "Ground" layer and one net, net name is "GND", i.e., [Ground, GND] pair.

Limitations:

- No Layer/Nets name is supported for field plot command recording should be able to add it based on this change.
- No Layer/Nets name support for pure Layout design field plot command.

<PlotGeomArray>

```
Array(<NumGeomTypes>, <GeomTypeData>, <GeomTypeData>, ...)
```

For example: Array(4, "Volume", "ObjList", 1, "Box1",
"Surface", "FacesList", 1, "12", "Line", 1,
"Polyline1", "Point", 2, "Point1", "Point2")

<NumGeomTypes>

Type: <int>

Number of different geometry types (volume, surface, line, point) plotted on at the same time.

<GeomTypeData>

```
<GeomType>, <ListType>, <NumIDs>, <ID>, <ID>, ...)
```

<GeomType>

Type: <string>

Possible values are "Volume", "Surface", "Line", "Point".

<ListType>

Type: <string>

Possible values are "ObjList", or "FacesList".

These are used for the `GeomType` of "Line" or "Point".

<NumIDs>

Type: <int>

Number of IDs or object names that will follow.

<ID>

Type: <int> or <string>

ID of a face or name of an object, line, or point on which to plot.

<FilterBoxArray>

Array of names of objects to use to restrict the plot range.

`Array(<NumFilters>, <ObjName>, <ObjName>, ...)`

Example: `Array(1, "Box1")`

Example: `Array(0)` *no filtering*

<PlotOnPointSettings>

`Array("NAME:PlotOnPointSettings",
"PlotMarker:=", <bool>,
"PlotArrow:=", <bool>)`

<PlotOnLineSettings>

```
Array("NAME:PlotOnLineSettings",  
      Array("NAME:LineSettingsID",  
            "Width:=", <int>,  
            "Style:=", <string>),  
      "IsoValType:=", <string>,  
      "ArrowUniform:=", <bool>,  
      "NumofArrow:=", <int>)
```

Style

Possible values are: "Cylinder", "Solid", "Dashdash", "Dotdot", "Dotdash"

IsoValType

Possible values are "Tone", "Fringe", "Gourard"

<PlotOnSurfaceSettings>

```
Array("NAME:PlotOnSurfaceSettings",  
      "Filled:=", <bool>,  
      "IsoValType:=", <string>,  
      "SmoothShade:=", <bool>,  
      "AddGrid:=", <bool>,  
      "MapTransparency:=", <bool>,  
      "Transparency:=", <double>,  
      "ArrowUniform:=", <bool>,  
      "ArrowSpacing:=", <double>,  
      "GridColor:=", Array(<int>, <int>, <int>))
```

IsoValType

Possible values are "Tone", "Line", "Fringe", "Gourard"

GridColor

Array containing the R, G, B components of the color. Components should be in the range 0 to 255.

<PlotOnVolumeSettings>

```
Array("NAME:PlotOnVolumeSettings",  
"PlotIsoSurface:=", <bool>,  
"CloudDensity:=", <double>,  
"PointSize:=", <int>,  
"ArrowUniform:=", <bool>,  
"ArrowSpacing:=", <double>)
```

Python Example – E Magnitude Field Plot:

```
oModule.CreateFieldPlot(  
[  
    "NAME:Mag_E1",  
    "SolutionName:="      , "Setup1 : LastAdaptive",  
    "QuantityName:="      , "Mag_E",  
    "PlotFolder:="        , "E Field1",  
    "UserSpecifyName:="   , 0,  
    "UserSpecifyFolder:=" , 0,  
    "IntrinsicVar:="      , "Freq='1GHz' Phase='0deg'",  
    "PlotGeomInfo:=", [1 , "Surface", "FacesList", 1, "7"],  
    "FilterBoxes:=", [0],  
    [  
        "NAME:PlotOnSurfaceSettings",  
        "Filled:="          , False,  
        "IsoValType:="      , "Fringe",  
        "SmoothShade:="     , True,  
        "AddGrid:="         , False,  
        "MapTransparency:=", True,  
        "Transparency:="    , 0,  
        "ArrowUniform:="    , True,  
        "ArrowSpacing:="    , 0.100000001490116,  
        "GridColor:="       , [255, 255, 255]    ]  
]
```

```
]
], "Field")
```

VB Example – Demag_Coef Field Plot:

```
oModule.CreateFieldPlot Array("NAME:Demag_Coef2", _
    "SolutionName:=", "Setup1 : Transient", _
    "UserSpecifyName:=", 0, "UserSpecifyFolder:=", 0, _
    "QuantityName:=", "Demag_Coef", "PlotFolder:=", "Demag-Coef", _
    "StreamlinePlot:=", false, "AdjacentSidePlot:=", false, _
    "FullModelPlot:=", false, "IntrinsicVar:=", _
    "Time=" & Chr(39) & "0s" & Chr(39) & "", "PlotGeomInfo:=", _
    Array(1, "Volume", "ObjList", 1, "Mag2_0"), "FilterBoxes:=", _
    Array(1, "Mag1_0"), _
    Array("NAME:PlotOnVolumeSettings", "PlotIsoSurface:=", _
    true, "PointSize:=", 1, "Refinement:=", 0, _
    "CloudSpacing:=", 0.5, "CloudMinSpacing:=", -1, _
    "CloudMaxSpacing:=", -1, _
    Array("NAME:Arrow3DSpacingSettings", _
    "ArrowUniform:=", true, "ArrowSpacing:=", 0, _
    "MinArrowSpacing:=", 0, "MaxArrowSpacing:=", 0)), _
    "EnableGaussianSmoothing:=", false), "Field"
```

Parameters: <PlotParameterArray> "VRT" (VB)

```
Array("NAME:<PlotName>", _
    "SolutionName:=", <string>, _
    "QuantityName:=", <string>, _
    "PlotFolder:=", <string>, _
    "UserSpecifyName:=", <int>, _
    "UserSpecifyFolder:=", <int>, _
    "IntrinsicVar:=", <string>, _
```



```
"MaxFrequency:=", "<Number><FreqUnits>", _  
"LaunchFrom:=", ["Launch from custom", | _  
"LaunchFromPointID:=", ] 18007, "RayDensity:=", <int>, _  
"NumberBounces:=", <int>, _  
"Multi-Bounce Ray Density Control:=", <Boolean>, _  
"MBRD Max sub divisions:=", <int>, _  
"ShootFilterType:=", ["All Rays" | "Rays by index", _  
"start index:=", <int>, "stop index:=", <int> | _  
"index step:=", <int> "Rays in box", "FilterBoxID:=", _  
<objID> | "Single ray", "Ray elevation:=", _  
"<real> deg", "Ray azimuth:=", "<real>deg"]), "VRT"
```

VB Example – VRT Plot:

```
Dim oAnsoftApp  
Dim oDesktop  
Dim oProject  
Dim oDesign  
Dim oEditor  
Dim oModule  
Set oAnsoftApp = CreateObject("Ansoft.ElectronicsDesktop")  
Set oDesktop = oAnsoftApp.GetAppDesktop()  
oDesktop.RestoreWindow  
Set oProject = oDesktop.SetActiveProject("abrams_1p5")  
Set oDesign = oProject.SetActiveDesign("UseSbr")  
Set oModule = oDesign.GetModule("FieldsReporter")  
oModule.CreateFieldPlot _  
    Array("NAME:VRT_Plot1", "SolutionName:=", _  
    "Setup1 : LastAdaptive", "QuantityName:=", _  
    "QuantityName_SBR", "PlotFolder:=", _
```

```

"Visual Ray Trace SBR", "UserSpecifyName:=", 0, _
"UserSpecifyFolder:=", 0, "IntrinsicVar:=", "", _
"MaxFrequency:=", "1GHz", "LaunchFrom:=", _
"Launch from custom", "LaunchFromPointID:=", _
18007, "RayDensity:=", 2, "NumberBounces:=", _
5, "Multi-Bounce Ray Density Control:=", _
false, "MBRD Max sub divisions:=", 2, _
"ShootFilterType:=", "All Rays"), "VRT"

```

VB Example – E Magnitude Field Plot:

```

oModule.CreateFieldPlot Array("NAME:Mag_E1", _
    "SolutionName:=", "Setup1 : LastAdaptive", _
    "QuantityName:=", "Mag_E", _
    "PlotFolder:=", "E Field1", _
    "UserSpecifyName:=", 0, _
    "UserSpecifyFolder:=", 0, _
    "IntrinsicVar:=", "Freq='1GHz' Phase='0deg'", _
    "PlotGeomInfo:=", Array( 1, "Surface", "FacesList", 1, "7"), _
    "FilterBoxes:=", Array(0), _
    Array("NAME:PlotOnSurfaceSettings", _
        "Filled:=", false, "IsoValType:=", "Fringe", _
        "SmoothShade:=", true, _
        "AddGrid:=", false, _
        "MapTransparency:=", true, _
        "Transparency:=", 0, _
        "ArrowUniform:=", true, _
        "ArrowSpacing:=", 0.100000001490116, _
        "GridColor:=", Array(255, 255, 255))), "Field"

```

Python Example – Electron Density Field Plot:

```
oModule = oDesign.GetModule("FieldsReporter")
oModule.CreateFieldPlot(
    [
        "NAME:Electron_Density3",
        "SolutionName:=", "AIR : RFDischarge",
        "UserSpecifyName:=", 0,
        "UserSpecifyFolder:=", 0,
        "QuantityName:=", "Electron_Density",
        "PlotFolder:=", "RF Discharge Fields",
        "StreamlinePlot:=", False,
        "AdjacentSidePlot:=", False,
        "FullModelPlot:=", False,
        "IntrinsicVar:=", "Freq=\'0.20000000000000001GHz\'",
        "GasPressure=\'0.02kPascal\'",
        "PlotGeomInfo:=", [1, "Surface", "FacesList", 1, "48"],
        "FilterBoxes:=", [0],
        [
            "NAME:PlotOnSurfaceSettings",
            "Filled:=", False,
            "IsoValType:=", "Tone",
            "AddGrid:=", False,
            "MapTransparency:=", True,
            "Refinement:=", 0,
            "Transparency:=", 0,
            "SmoothingLevel:=", 0,
            "ShadingType:=", 0,
            [
                "NAME:Arrow3DSpacingSettings",
                "ArrowUniform:=", True,
```

```

        "ArrowSpacing:="      , 0,
        "MinArrowSpacing:="   , 0,
        "MaxArrowSpacing:="   , 0
        "GridColor:="         , [255,255,255]
    ],
    ],
    "EnableGaussianSmoothing:=" , False,
    "SurfaceOnly:="             , False
], "Field")

```

VB Example – Mesh Plot:

```

Dim oAnsoftApp
Dim oDesktop
Dim oProject
Dim oDesign
Dim oEditor
Dim oModule
Set oAnsoftApp = CreateObject("Ansoft.ElectronicsDesktop")
Set oDesktop = oAnsoftApp.GetAppDesktop()
oDesktop.RestoreWindow
Set oProject = oDesktop.SetActiveProject("abrams_1p5")
Set oDesign = oProject.SetActiveDesign("UseSbr")
Set oModule = oDesign.GetModule("FieldsReporter")
oModule.CreateFieldPlot Array("NAME:Mesh1", "SolutionName:=", _
    "Setup1 : LastAdaptive", "QuantityName:=", "Mesh", _
    "PlotFolder:=", "MeshPlots", "FieldType:=", "Fields", _
    "UserSpecifyName:=", 0, "UserSpecifyFolder:=", 0, _
    "StreamlinePlot:=", false, "IntrinsicVar:=", "Freq=" _
    & Chr(39) & "2GHz" & Chr(39) & " Phase=" _
    & Chr(39) & "0deg" & Chr(39) & "" & "", _
    "PlotGeomInfo:=", Array(1, "Surface", "FacesList", 1, _

```

```
"10568"), "FilterBoxes:=", Array(0), "Real time mode:=", _
true, Array("NAME:MeshSettings", "Scale factor:=", 100, _
  "Transparency:=", 0, "Mesh type:=", "Shaded", _
  "Surface only:=", true, "Add grid:=", true, _
  "Refinement:=", 0, "Use geometry color:=", true, _
  "Mesh line color:=", Array(0, 0, 255), "Filled color:=", _
  Array( 255, 255, 255))), _
"Field"
```

DeleteFieldPlot

Deletes one or more field plots.

UI Access	Right-click on one filed plot, select Delete .		
Parameters	Name	Type	Description
	<NameArray>	Array	Array of strings containing the names of the plots to delete.
Return Value	None.		

Python Syntax	DeleteFieldPlot(<NameArray>)
Python Example	oModule.DeleteFieldPlot(["Mag_E1", "Vector_E1"])

VB Syntax	DeleteFieldPlot <NameArray>
VB Example	oModule.DeleteFieldPlot Array("Mag_E1", "Vector_E1")

DeleteMarker[Fields Reporter]

Deletes one or more marker in the current field plot.

UI Access	Right-click Field Overlays > Plot Fields > Marker > Delete Marker .		
Parameters	Name	Type	Description
	<MarkerNames>	Array	Array of strings containing marker names.
Return Value	None.		

Python Syntax	DeleteMarker(<MarkerNames>)
Python Example	oModule.DeleteMarker(["m1", "m2"])

VB Syntax	DeleteMarker <MarkerNames>
VB Example	oModule.DeleteMarker Array("m1", "m2")

DeleteNamedExpr

Deletes the selected named expression from the list.

UI Access	Select a named expression and then click Delete .		
Parameters	Name	Type	Description

	<table><tr><td><ExprName></td><td>String</td><td>Name of specified named expression.</td></tr></table>	<ExprName>	String	Name of specified named expression.
<ExprName>	String	Name of specified named expression.		
Return Value	None.			

Python Syntax	DeleteNamedExpr(<ExprName>)
Python Example	<code>oModule.DeleteNamedExpr("Mag_JxE")</code>

VB Syntax	DeleteNamedExpr <ExprName>
VB Example	<code>oModule.DeleteNamedExpr "Mag_JxE"</code>

DeleteUneditablePlot

Deletes one or more uneditable plots.

UI Access	N/A		
Parameters	Name	Type	Description
	<NameArray>	Array	Array of the plot names to delete.
Return Value	None.		

Python Syntax	DeleteUneditablePlot(<NameArray>)
Python Example	<code>oModule.DeleteUneditablePlot(["Mag_E1", "Vector_E1"])</code>

VB Syntax	DeleteUneditablePlot <NameArray>
VB Example	<code>oModule.DeleteUneditablePlot Array("Mag_E1", "Vector_E1")</code>

DoesNamedExpressionExists

Determines whether specified named expression exists.

UI Access	N/A		
Parameters	Name	Type	Description
	<ExpName>	String	Name of the specified named expression.
Return Value	Boolean: • 1 - named expression exists. • 0 - named expression does not exist.		

Python Syntax	DoesNamedExpressionExists(<ExpName>)
Python Example	<code>oModule.DoesNamedExpressionExists("Mag_JxE")</code>

VB Syntax	DoesNamedExpressionExists <ExpName>
VB Example	<code>oModule.DoesNamedExpressionExists "Mag_JxE"</code>

EnterComplex

Enters a complex number onto the stack.

UI Access	Click Number , and then click Scalar. Complex option is selected.		
Parameters	Name	Type	Description
	<ComplexNum>	String	String of complex value. Ex. "1 + 2j".
Return Value	None.		

Python Syntax	EnterComplex(<ComplexNum>)
Python Example	<code>oModule.EnterComplex("1 + 2 j")</code>

VB Syntax	EnterComplex <ComplexNum>
VB Example	<code>oModule.EnterComplex "1 + 2 j"</code>

EnterComplexVector

Enters a complex vector onto the stack.

UI Access	Click Number , and then click Vector. Complex option is selected.		
Parameters	Name	Type	Description
	<ComplexVector>	Array	Array of strings containing X, Y and Z complex values.

Return Value	None.

Python Syntax	EnterComplexVector(<ComplexVector>)
Python Example	<pre>oModule.EnterComplexVector (["1 + 2 j", "1 + 2 j", "1 + 2 j"])</pre>

VB Syntax	EnterComplexVector <ComplexVector>
VB Example	<pre>oModule.EnterComplexVector _ Array("1 + 2 j", _ "1 + 2 j", _ "1 + 2 j")</pre>

EnterCoord

Enters a coordinate system defined in the 3D Modeler editor.

UI Access	Click Geometry and then select Coord .		
Parameters	Name	Type	Description

	<table><tr><td><CoordName></td><td>String</td><td>Name of a coordinate system defined in the 3D Modeler editor.</td></tr></table>	<CoordName>	String	Name of a coordinate system defined in the 3D Modeler editor.
<CoordName>	String	Name of a coordinate system defined in the 3D Modeler editor.		
Return Value	None.			

Python Syntax	EnterCoord(<CoordName>)
Python Example	<code>oModule.EnterCoord("Global")</code>

VB Syntax	EnterCoord <CoordName>
VB Example	<code>oModule.EnterCoord "Global"</code>

EnterEdge

Enters an edge defined in the 3D Modeler editor.

UI Access	N/A		
Parameters	Name	Type	Description
	<EdgeName>	String	Name of an edge defined in the 3D Modeler editor.
Return Value	None.		

Python Syntax	EnterEdge(<EdgeName>)
Python Example	<code>oModule.EnterPoint("Edge_1")</code>

VB Syntax	EnterEdge <EdgeName>
VB Example	oModule.EnterPoint "Edge_1"

EnterLine

Enters a line defined in the 3D Modeler editor.

UI Access	Click Geometry and then select Line		
Parameters	Name	Type	Description
	<LineName>	String	Name of a line defined in the 3D Modeler editor.
Return Value	None.		

Python Syntax	EnterLine(<LineName>)
Python Example	oModule.EnterLine("Line1")

VB Syntax	EnterLine <LineName>
VB Example	oModule.EnterLine "Line1"

EnterOutputVar

Enters Output Vars, only valid for Eigenmode problems.

UI Access	Click Geometry and then select Output Vars .		
Parameters	Name	Type	Description
	<VarName>	String	Name of the output vars. Only 'freq' is supported.
	<VarType>	String	Type of the output vars. 'Complex' type.
Return Value	None.		

Python Syntax	EnterOutputVar(<VarName>, <VarType>)		
Python Example	oModule.EnterOutputVar("Freq", "Complex")		

VB Syntax	EnterOutputVar <VarName>, <VarType>		
VB Example	oModule.EnterOutputVar "Freq", "Complex"		

EnterPoint

Enters a point defined in the 3D Modeler editor.

UI Access	Click Geometry and then select Point .		
Parameters	Name	Type	Description
	<PointName>	String	Name of a point defined in the 3D Modeler editor.
Return Value	None.		

Python Syntax	EnterPoint(<PointName>)
Python Example	oModule.EnterPoint("Point1")

VB Syntax	EnterPoint <PointName>
VB Example	oModule.EnterPoint "Point1"

EnterQty

Enters a field quantity.

UI Access	Click Quantity , and then select from the list.		
Parameters	Name	Type	Description
	<FieldQty>	String	The field quantity to be entered onto the stack.
Return Value	None.		

Python Syntax	EnterQty(<FieldQty>)
Python Example	oModule.EnterQty("E")

VB Syntax	EnterQty <FieldQty>
VB Example	oModule.EnterQty "E"

EnterScalar

Enters a scalar onto the stack.

UI Access	Click Number and then click Scalar . Complex option not selected.		
Parameters	Name	Type	Description
	<Scalar>	Double	The real number to enter onto the stack.
Return Value	None.		

Python Syntax	EnterScalar(<Scalar>)
Python Example	<code>oModule.EnterScalar(3.14159265358979)</code>

VB Syntax	EnterScalar <Scalar>
VB Example	<code>oModule.EnterScalar 3.14159265358979</code>

EnterScalarFunc

Enters a scalar function.

UI Access	Click Function and then select Scalar .		
Parameters	Name	Type	Description
	<VarName>	String	Name of a variable to enter as a scalar function onto the stack.

Return Value	None.

Python Syntax	EnterScalarFunc(<VarName>)
Python Example	<code>oModule.EnterScalarFunc("Phase")</code>

VB Syntax	EnterScalarFunc <VarName>
VB Example	<code>oModule.EnterScalarFunc "Phase"</code>

EnterSurf

Enters a surface defined in the 3D Modeler editor.

UI Access	Click Geometry and then select Surface .		
Parameters	Name	Type	Description
	<SurfName>	String	Name of a surface defined in the 3D Modeler editor.
Return Value	None.		

Python Syntax	EnterSurf(<SurfName>)
Python Example	<code>oModule.EnterSurf("Rectangle1")</code>

VB Syntax	EnterSurf <SurfName>

VB Example	<code>oModule.EnterSurf "Rectangle1"</code>

EnterVector

Enters a vector onto the stack.

UI Access	Click Number , and then click Vector . Complex option not selected.		
Parameters	Name	Type	Description
	<VectorArray>	Array	Array of strings containing X, Y and Z components of the vector.
Return Value	None.		

Python Syntax	<code>EnterVector(<VectorArray>)</code>
Python Example	<code>oModule.EnterVector([1.0, 1.0, 1.0])</code>

VB Syntax	<code>EnterVector <VectorArray></code>
VB Example	<code>oModule.EnterVector Array(1.0, 1.0, 1.0)</code>

EnterVectorFunc

Enters a vector function.

UI Access	Click Function and then select Vector .		
Parameters	Name	Type	Description

	<table><tr><td><VarNames></td><td>Array</td><td>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.</td></tr></table>	<VarNames>	Array	Array of strings containing names of a variable for the X, Y, and Z coordinates, respectively, to enter as a vector function on the stack.
<VarNames>	Array	Array of strings containing names of a variable for the X, Y, and Z coordinates, respectively, to enter as a vector function on the stack.		
Return Value	None.			

Python Syntax	EnterVectorFunc(<VarNames>)
Python Example	oModuleEnterVectorFunc(["X", "Y", "Z"])

VB Syntax	EnterVectorFunc <VarNames>
VB Example	oModuleEnterVectorFunc Array("X", "Y", "Z")

EnterVol

Enters a volume defined in the 3D Modeler editor.

UI Access	Click Geometry and then select Volume .		
Parameters	Name	Type	Description
	<VolumeName>	String	Name of a volume defined in the 3D Modeler editor.
Return Value	None.		

Python Syntax	EnterVol(<VolumeName>)
Python Example	oModule.EnterVol("Box1")

VB Syntax	EnterVol <VolumeName>
VB Example	oModule.EnterVol "Box1"

ExportFieldPlot

Exports a field plot to a file.

UI Access	N/A		
Parameters	Name	Type	Description
	<PlotName>	String	Name of the field plot to export.
	<ShowHeader>	Boolean	True - export field plot header to a file. False - export field plot.
	<FileName>	String	Name of file to save as, including the file path.
Return Value	None.		

Python Syntax	ExportFieldPlot(<PlotName>, <ShowHeader>, <FileName>)
Python Example	oModule.ExportFieldPlot("Mag_E1", True, "C:/field_report.dsp")

VB Syntax	ExportFieldPlot <PlotName>, <ShowHeader>, <FileName>
VB Example	oModule.ExportFieldPlot "Mag_E1", _

	<code>true, "C:/field_report.dsp"</code>

ExportMarkerTable

Exports the marker table to a .csv or .tab file.

UI Access	[product] > Fields > Plot Fields > Marker > Export Marker Table .		
Parameters	Name	Type	Description
	<FileName>	String	Name of export file include path.
Return Value	None.		

Python Syntax	<code>ExportMarkerTable(<FileName>)</code>
Python Example	<code>oModule.ExportMarkerTable("C:/work/FieldMarkerTable.csv")</code>

VB Syntax	<code>ExportMarkerTable <FileName></code>
VB Example	<code>oModule.ExportMarkerTable "C:/work/FieldMarkerTable.csv"</code>

ExportOnGrid [Field Overlays]

Evaluates the top stack element at a set of points specified by a grid and exports the data to a file.

UI Access	Click Export , and then click On Grid .		
Parameters	Name	Type	Description
	<OutputFile>	String	Name of the output file.
	<Min>	Array	Minimum values for the coordinate components of the grid system

	<Max>	Array	Maximum values for the coordinate components of the grid system
	<Spacing>	Array	Spacing values for the coordinate components of the grid system
	<SolnName>	String	Name of the simulation setup
	<VarVals>	Array	Array of strings containing setup definitions.
	<IncludePoints>	Boolean	Optional. Specifies whether include points in the output file.
	<CSType>	String	Optional. Type of coordinate system. "Cartesian" (default) "Cylindrical" "Spherical"
	<Offsets>	Array	Optional. Origin for the offset coordinate system. For Cartesian, x, y, z, for Cylindrical, R, Phi, Z, for Spherical, Rho, Theta, Phi.
	<ByCount>	Boolean	Optional.
Return Value	None.		

Note: Regarding the **ExportOnGrid** legacy script which only has “IncludePtInOutput” argument (the last Boolean one), AEDT can still read it and assign other new arguments as default values. Those default values are RefCSName = “global”, PtInSI = “True”, FieldInRefCS = “False”

Python Syntax	ExportOnGrid(<OutputFile>, <Min>, <Max>, <Spacing>, <SolnName>, <SolnParameters>, [<ExportOption>, "IncludePointsInOutput:=", <boolean>], "RefCSName:=", <CSName>, "PtsInSI:=", <boolean>, "FieldRefCS:=", <boolean>], "<CSType>", [<Offsets>, <ByCount>])
Python Example	<pre>oModule.ExportOnGrid("C:\offset_grid_model_unit_ref.fld", ["-1mm", "16mm", "0mm"], ["1mm", "18mm", "1mm"], ["2mm", "2mm", "1mm"], "4500MHz : LastAdaptive", ["Freq:=", "4.5GHz", "Phase:=", "0deg"], [</pre>

	<pre> "NAME:ExportOption", "IncludePtInOutput:=" , True, "RefCSName:=" , "offset", "PtInSI:=" , False, "FieldInRefCS:=" , True], "Cartesian", ["0mm", "0mm", "0mm"], False) </pre>

VB Syntax	ExportOnGrid(<OutputFile>, <Min>, <Max>, <Spacing>, <SolnName>, <SolnParameters>, [<ExportOption>, "IncludePointsInOutput:=", <boolean>], "RefCSName:=", <CSName>, "PtsInSI:=", <boolean>, "FieldRefCS:=", <boolean>], "<CSType>", [<Offsets>, <ByCount>])
VB Example	<pre> oModule.ExportOnGrid "C:/Grid.fld", _ Array("0mm", "0deg", "-25mm"), _ Array("20mm", "90deg", "125mm"), _ Array("10mm", "45deg", "50mm"), _ "Setup1 : LastAdaptive", _ Array("Freq:=", "10000Hz", "Phase:=", "0deg"), _ true, "Cylindrical", _ Array("0mm", "0mm", "0mm") </pre>

	<pre>oModule.ExportOnGrid("C:\offset_grid_model_unit_ref.fld", Array("-1mm", "16mm", "0mm"), Array("1mm", "18mm", "1mm"), Array("2mm", "2mm", "1mm"), "4500MHz : LastAdaptive", Array("Freq:=", "4.5GHz","Phase:=" , "0deg"), ["NAME:ExportOption", "IncludePtInOutput:=" , True, "RefCSName:=" , "offset", "PtInSI:=" , False, "FieldInRefCS:=" , True], "Cartesian", Array("0mm", "0mm", "0mm"), False)</pre>

ExportPlotImageToFile

Deprecated. Use [ExportPlotImageWithViewToFile](#).

Creates field plot exports of existing field plots from a given view points, and with the model being auto-sized automatically for each view.

UI Access	N/A

Parameters	Name	Type	Description
	<FileName>	String	Full path plus file name.
	<FolderName>	String	Plot folder name.
	<ItemName>	String	Name of fields to plot.
	<SetViewTopDownDirectionByRCS>	String	Optional. Name of relative coordinate system to use for the field plot.
Return Value	None.		

Python Syntax	ExportPlotImageToFile(<FileName>, <FolderName>, <ItemName>, <SetViewTopDownDirectionByRCS>)
Python Example	<pre>oModule.ExportPlotImageToFile ("C:\TestEPITF2.jpg", "", "Mag_E2", "RelativeCS1")</pre>

VB Syntax	ExportPlotImageToFile <FileName>, <FolderName>, <ItemName>, <SetViewTopDownDirectionByRCS>
VB Example	<pre>oModule.ExportPlotImageToFile _ "C:\TestEPITF2.jpg", "", _ "Mag_E2", "RelativeCS1"</pre>

ExportPlotImageWithViewToFile [Reporter]

Exports a field plot image to a specified file.

Note:

This script replaces ExportPlotImageToFile, which is deprecated.

UI Access	N/A		
Parameters	Name	Type	Description
	<FileName>	String	Full path of file to export.
	<PlotQuantityName>	String	Name of quantity to plot.
	<PlotItemName>	String	Name of fields to plot.
	<PixelSizeX>	Integer	Desired image width, in pixels.
	<PixelSizeY>	Integer	Desired image height, in pixels.
	<ViewOrientation>	String	<i>Optional.</i> Name of orientation to use for plot.
Return Value	Image file is exported to the specified path.		

Python Syntax	ExportPlotImageWithViewToFile(<FileName>, <PlotQuantityName>, <PlotItemName>, <PixelSizeX>, <PixelSizeY>, <ViewOrientation>)
Python Example	oModule.ExportPlotImageWithViewToFile("E:/MyDir/OptimToutput/magE2.gif", "E Field", "Mag_E2", 1920, 1080, "newot2")

VB Syntax	ExportPlotImageWithViewToFile <FileName>, <PlotQuantityName>, <PlotItemName>, <PixelSizeX>, <PixelSizeY>, <ViewOrientation>
VB	oModule.ExportPlotImageWithViewToFile "E:/MyDir/OptimToutput/magE2.gif", "E Field", _

Example	"Mag_E2", 1920, 1080, "newot2"

ExportToFile

Note:

The ExportToFile script command has replaced the script command [ExportReport](#). ExportReport remains in order to retain backward compatibility for existing scripts, but it is strongly recommended that you now use ExportToFile.

From a data table or plot, generates text format, comma delimited, tab delimited, or .dat type output files.

UI Access	Right-click on report name in the project tree and select Export Data .		
Parameters	Name	Type	Description
	<ReportName>	String	Name of report to be exported.
	<FileName>	String	Full path of the exported image file name; with extension of .txt - Post processor format file .csv - Comma-delimited data file .tab - Tab-separated file .dat - Ansys plot data file
	<UnitSpec>	String	For example, "kV, Mhz, yd"
	<UseTraceNumberFormat>	Boolean	"True", "False"
Return Value	None.		

Python Syntax	ExportToFile(<ReportName>, <FileName>)
Python Example	<pre>oModule.ExportToFile("rETotal", "C:/Users/Documents/rETotal.csv") oModule.ExportToFile("S Parameter Table 1", "D:/Users/Documents/cfft.csv", False, " kV, MHz, yd ", True)</pre>

VB Syntax	ExportToFile <ReportName>, <FileName>
VB Example	<pre>oModule.ExportToFile "rETotal", "C:/Documents/rETotal.csv"</pre>

GetFieldFolderNames

Gets the folder names of the field plots.

UI Access	N/A
Parameters	None.
Return Value	Array of folder names.

Python Syntax	GetFieldFolderNames()
Python Example	<pre>oModule.GetFieldFolderNames()</pre>

VB Syntax	GetFieldFolderNames
VB Example	<pre>oModule.GetFieldFolderNames</pre>

GetFieldPlotNames

Gets the names of field overlay plots defined in a design.

UI Access	N/A
Parameters	None.
Return Value	Array of strings containing field plots names.

Python Syntax	GetFieldPlotNames()
Python Example	<code>oModule.GetFieldPlotNames()</code>

VB Syntax	GetFieldPlotNames
VB Example	<code>oModule.GetFieldPlotNames</code>

GetFieldPlotQuantityName

Gets the quantity name of a specified field plot.

UI Access	N/A		
Parameters	Name	Type	Description
	<PlotName>	String	Name of specified plot.
Return Value	String plot quantity name.		

Python Syntax	GetFieldPlotQuantityName(<PlotName>)
Python Example	<code>oModule.GetFieldPlotQuantityName("Mag_H1")</code>

VB Syntax	GetFieldPlotQuantityName <PlotName>
VB Example	<code>oModule.GetFieldPlotQuantityName "Mag_H1"</code>

GetMeshPlotNames

Gets the names of mesh plots defined in a design.

UI Access	N/A
Parameters	None..
Return Value	Array of plot name.

Python Syntax	GetMeshPlotNames
Python Example	<code>oModule.GetMeshPlotNames()</code>

VB Syntax	GetMeshPlotNames
VB Example	<code>oModule.GetMeshPlotNames</code>

GetTopEntryValue

Gets the value of the top entry of the calculator stack.

UI Access	N/A		
Parameters	Name	Type	Description
	<SolnName>	String	Name of specified solution.
	<VarVals>	Array	Array of variable name, value pairs.
Return Value	Array of strings containing top entry values.		

Python Syntax	GetTopEntryValue(<SolnName>, <VarVals>)
Python Example	<pre>oModule.GetTopEntryValue("Setup1:LastAdaptive", ["Freq:=", "1GHz", "Phase:=", "0deg", "x_size:=", "2mm"])</pre>

VB Syntax	GetTopEntryValue <SolnName>, <VarVals>
VB Example	<pre>oModule.GetTopEntryValue "Setup1:LastAdaptive", _ Array("Freq:=", "1GHz", "Phase:=", "0deg", _ "x_size:=", "2mm")</pre>

LoadNamedExpressions

Loads a named expression definition from a saved file.

UI Access	In the Fields Calculator, click Load From... in the Library area.		
Parameters	Name	Type	Description
	<FileName>	String	Filename and full path to the file to hold the named expression definition.
	<FieldType>	String	For products with just one filed type, it is set to "Fields".
	<NamedExpr>	Array	rray of strings containing the names of expression definitions to load from the file.
Return Value	None.		

Python Syntax	LoadNamedExpressions(<FileName>, <FieldType>, <NamedExpr>)		
Python Example	<pre>oModule.LoadNamedExpressions ("C:\Ansoft\PersonalLib\smth.clc", "Fields", ["smoothedtemp"])</pre>		

VB Syntax	LoadNamedExpressions <FileName>, <FieldType>, <NamedExpr>		
VB Example	<pre>oModule.LoadNamedExpressions _ "C:\Ansoft\PersonalLib\smth.clc", _ "Fields", Array("smoothedtemp")</pre>		

ModifyFieldPlot

Modifies a plot definition.

UI Access	[product] > Fields > Modify Plot		
Parameters	Name	Type	Description
	<OriginalName>	String	Name of the plot to be modified.
	<PlotParams>	Array	Array containing modified settings.
Return Value	None.		

Python Syntax	ModifyFieldPlot(<OriginalName>, <PlotParams>)
Python Example	<pre>oModule.ModifyFieldPlot("Vector_E1" , ["NAME:Vector_E2", "SolutionName:=", "Setup1 : LastAdaptive", "QuantityName:=", "Vector_E", "PlotFolder:=", "E Field1", "UserSpecifyName:=", 0, "UserSpecifyFolder:=", 0, "IntrinsicVar:=", "Freq='1GHz' Phase='30deg'", "PlotGeomInfo:=", [1, "Surface","FacesList", 1, "7"], "FilterBoxes:=", [0],</pre>

	<pre>["NAME:PlotOnSurfaceSettings", "Filled:=", False, "IsoValType:=", "Fringe", "SmoothShade:=", True, "AddGrid:=", False, "MapTransparency:=", True, "Transparency:=", 0, _ "ArrowUniform:=", True, "ArrowSpacing:=", 0.100000001490116, "GridColor:=", [255, 255, 255]]])</pre>

VB Syntax	ModifyFieldPlot <OriginalName>, <PlotParams>
VB Example	<pre>oModule.ModifyFieldPlot "Vector_E1" ,_ Array("NAME:Vector_E2", _ "SolutionName:=", "Setup1 : LastAdaptive", _ "QuantityName:=", "Vector_E", _ "PlotFolder:=", "E Field1", _ "UserSpecifyName:=", 0, _</pre>

	<pre>"UserSpecifyFolder:=", 0, _ "IntrinsicVar:=", "Freq='1GHz' Phase='30deg'", _ "PlotGeomInfo:=", Array(1, "Surface", "FacesList", 1, "7"), _ "FilterBoxes:=", Array(0), _ Array("NAME:PlotOnSurfaceSettings", _ "Filled:=", false, _ "IsoValType:=", "Fringe", _ "SmoothShade:=", true, _ "AddGrid:=", false, _ "MapTransparency:=", true, _ "Transparency:=", 0, _ "ArrowUniform:=", true, _ "ArrowSpacing:=", 0.100000001490116, _ "GridColor:=", Array(255, 255, 255)))</pre>

ReassignFieldPlot

Reassigns a field plot.

UI Access	Right-click on a field plot > Reassign .		
Parameters	Name	Type	Description
	<PlotName>	String	Name of specified plot to reassign.
	<PlotParameterArray>	Array	See CreateFieldPlot
Return Value	None.		

Python Syntax	ReassignFieldPlot(<PlotName>, <PlotParameterArray>)
Python Example	<pre>oModule.ReassignFieldPlot "Mag_H1", Array("NAME:Mag_H1", _ "SolutionName:=", "Setup1 : LastAdaptive", _ "UserSpecifyName:=", 0, _ "UserSpecifyFolder:=", 0, _ "QuantityName:=", "Mag_H", _ "PlotFolder:=", "H Field", _ "StreamlinePlot:=", False, "AdjacentSidePlot:=", False, "FullModelPlot:=", False, "IntrinsicVar:=", "Freq=\'20GHz\' Phase=\'0deg\'", "PlotGeomInfo:=", [1,"Surface","FacesList",1,"117"], "FilterBoxes:=", [0], ["NAME:PlotOnSurfaceSettings", "Filled:=", False, "IsoValType:=", "Fringe", "AddGrid:=", False, "MapTransparency:=", True,</pre>

	<pre> "Refinement:=", 0, "Transparency:=", 0, "SmoothingLevel:=", 0, "ShadingType:=", 0, ["NAME:Arrow3DSpacingSettings", "ArrowUniform:=", True, "ArrowSpacing:=", 0, "MinArrowSpacing:=", 0, "MaxArrowSpacing:=", 0], "GridColor:=", [255,255,255]], "EnableGaussianSmoothing:=", False]) </pre>

VB Syntax	ReassignFieldPlot <PlotName>, <PlotParameterArray>
VB Example	<pre> oModule.ReassignFieldPlot "Mag_E1", _ Array("NAME:Mag_E1", "SolutionName:=", _ "Setup1 : LastAdaptive", "UserSpecifyName:=", 0, _ "UserSpecifyFolder:=", 0, "QuantityName:=", _ "Mag_E", "PlotFolder:=", "E Field", _ </pre>

	<pre>"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)</pre>

RenameFieldPlot

Renames a plot.

UI Access	Right-click the plot you want to rename in the project tree, and then click Rename on the shortcut menu.		
Parameters	Name	Type	Description
	<OldName>	String	Original name of the plot.
	<NewName>	String	New name of the plot.

Return Value	None.

Python Syntax	<code>RenameFieldPlot(<OldName>, <NewName>)</code>
Python Example	<pre>oModule.RenameFieldPlot("Vector_E1", "Vector_E2")</pre>

VB Syntax	<code>RenameFieldPlot <OldName>, <NewName></code>
VB Example	<pre>oModule.RenameFieldPlot _ "Vector_E1", "Vector_E2"</pre>

RenamePlotFolder

Renames a plot folder.

UI Access	Right-click a plot folder in the project tree, and then click Rename on the shortcut menu.		
Parameters	Name	Type	Description
	<code><OldName></code>	String	Original name of the folder.
	<code><NewName></code>	String	New name of the folder.
Return Value	None.		

Python Syntax	<code>RenamePlotFolder(<OldName>, <NewName>)</code>
Python Example	<pre>oModule.RenamePlotFolder(</pre>

	"E Field", "Surface Plots")

VB Syntax	RenamePlotFolder <OldName>, <NewName>
VB Example	<pre>oModule.RenamePlotFolder _ "E Field", "Surface Plots"</pre>

SaveFieldsPlots

Saves field plot(s) to a file.

UI Access	HFSS > Fields > Save As...		
Parameters	Name	Type	Description
	<PlotNames>	Array	Array of plot names.
	<FileName>	String	Name of the file to save as, including the file path.
Return Value	None.		

Python Syntax	SaveFieldsPlots(<PlotNames>, <FileName>)
Python Example	<pre>oModule.SaveFieldsPlots(["Mag_E1", "Mag_E2"], "C:/field_report.dsp")</pre>

VB Syntax	SaveFieldsPlots <PlotNames>, <FileName>

VB Example	<pre>oModule.SaveFieldsPlots Array("Mag_E1", "Mag_E2"), _ "C:/field_report.dsp"</pre>

SaveNamedExpressions

Saves a named expression definition to a file.

UI Access	In the Fields Calculator, click Save To... in the Library area.		
Parameters	Name	Type	Description
	<FileName>	String	Filename and full path to the file to hold the named expression definition.
	<NamedExprs>	Array	Array of strings containing the names of expression definitions to load from the file.
	<Overwrite>	Boolean	Specifies whether to overwrite the file.
Return Value	None.		

Python Syntax	SaveNamedExpressions(<FileName>, <NamedExprs>, <Overwrite>)
Python Example	<pre>oModule.SaveNamedExpressions("C:\Ansoft\PersonalLib\smth.clc", ["smoothedtemp"], True)</pre>

VB Syntax	SaveNamedExpressions <FileName>, <NamedExprs>, <Overwrite>
VB Example	<pre>oModule.SaveNamedExpressions _ "C:\Ansoft\PersonalLib\smth.clc", _</pre>

	<code>Array("smoothedtemp"), true</code>

SetFieldPlotSettings

Sets plot attributes.

UI Access	[product] > Fields > Modify Plot Attributes		
Parameters	Name	Type	Description
	<PlotName>	String	Name of the plot to modify.
	<PlotItemAttributes>	Array	Structured array. <code>Array("NAME:FieldsPlotItemSettings", <PlotOnPointsSettings>, <PlotOnLineSettings>, <PlotOnSurfaceSettings>, <PlotOnVolumeSettings>)</code> <i>See description of CreateFieldPlot command for details.</i>
Return Value	None.		

Python Syntax	<code>SetFieldPlotSettings(<PlotName> <PlotItemAttributes>)</code>
Python Example	<pre>oModule.SetFieldPlotSettings("Mag_E2", ["NAME:FieldsPlotItemSettings", ["NAME:PlotOnLineSettings",</pre>

	<pre>["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>

VB Syntax	SetFieldPlotSettings <PlotName> <PlotItemAttributes>
VB Example	<pre>oModule.SetFieldPlotSettings "Mag_E2", _ Array("NAME:FieldsPlotItemSettings", _</pre>

```

Array("NAME:PlotOnLineSettings", _
Array("NAME:LineSettingsID", _
"Width:=", 4, _
"Style:=", "Cylinder"), _
"IsoValType:=", "Tone", _
"ArrowUniform:=", true, _
"NumofArrow:=", 100), _
Array("NAME:PlotOnSurfaceSettings", _
"Filled:=", false, _
"IsoValType:=", "Tone", _
"SmoothShade:=", true, _
"AddGrid:=", false, _
"MapTransparency:=", true, _
"Transparency:=", 0, _
"ArrowUniform:=", true, _
"ArrowSpacing:=", 0.100000001490116, _
"GridColor:=", Array(255, 255, 255)))

```

SetPlotFolderSettings

Sets the attributes of all plots in the specified folder.

UI Access	[product] > Fields > Modify Plot Attributes		
Parameters	Name	Type	Description
	<PlotFolderName>	String	Name of the folder with the attributes to modify.
	<PlotFolderAttributes>	Array	Structured array. <pre>Array("NAME:FieldsPlotSettings", "Real time mode:=", <bool>, <ColorMapSettings>, <Scale3DSettings>, <Marker3DSettings>, <Arrow3DSettings>)</pre>
	<ColorMapSettings>	Array	Structured array. <pre>Array("NAME:ColorMapSettings", "ColorMapType:=", <string Possible values are "Uniform", "Ramp", "Spectrum">, "SpectrumType:=", <string Possible values are "Rainbow", "Temperature", "Magenta", "Gray">, "UniformColor:=", <array containing the R, G, B components of the color. Components should be in the range 0 to 255.>, "RampColor:=", <array containing the R, G, B com- ponents of the color. Components should be in the range 0 to 255.>)</pre>
	<Scale3DSettings>	Array	Structured array. <pre>Array("NAME:Scale3DSettings", "m_nLevels:=", <int>,</pre>

		<pre>"m_autoScale:=", <bool>, "minvalue:=", <double>, "maxvalue:=", <double>, "log:=", <bool>, "IntrinsicMin:=", <double>, "IntrinsicMax:=", <double>)</pre>
<Marker3DSettings>	Array	Structured array. <pre>Array("NAME:Marker3DSettings", "MarkerType:=", <integer 9: Sphere 10: Box 11: Tetrahedron 12: Octahedron default: Sphere>, "MarkerMapSize:=", <bool>, "MarkerMapColor:=", <bool>, "MarkerSize:=", <double>)</pre>
<Arrow3DSettings>	Array	Structured array. <pre>Array("NAME:Arrow3DSettings", "ArrowType:=", <integer 0: Line</pre>

			1: Cylinder 2: Umbrella default: Line>, "ArrowMapSize:=", <bool>, "ArrowMapColor:=", <bool>, "ShowArrowTail:=", <bool>, "ArrowSize:=", <double>
Return Value	None.		

Python Syntax	SetPlotFolderSettings(<PlotFolderName>, <PlotFolderAttributes>)
Python Example	<pre>oModule.SetPlotFolderSettings("E Field1", ["NAME:FieldsPlotSettings", "Real time mode:=", True, ["NAME:ColorMapSettings", "ColorMapType:=", "Spectrum", "SpectrumType:=", "Rainbow", "UniformColor:=", [127, 255, 255], "RampColor:=", [255, 127, 127]], ["NAME:Scale3DSettings", "m_nLevels:=", 27,</pre>

	<pre> "m_autoScale:=", True, "minvalue:=", 9.34379863739014, "maxvalue:=", 13683.755859375, "log:=", False, "IntrinsicMin:=", 9.34379863739014, "IntrinsicMax:=", 13683.755859375), ["NAME:Marker3DSettings", "MarkerType:=", 0, "MarkerMapSize:=", True, "MarkerMapColor:=", False, "MarkerSize:=", 0.25], ["NAME:Arrow3DSettings", "ArrowType:=", 1, "ArrowMapSize:=", True, "ArrowMapColor:=", True, "ShowArrowTail:=", True, "ArrowSize:=", 0.25]]) </pre>

VB Syntax	SetPlotFolderSettings <PlotFolderName>, <PlotFolderAttributes>
VB Example	<pre> oModule.SetPlotFolderSettings "E Field1", _ Array("NAME:FieldsPlotSettings", _ "Real time mode:=", true, _ Array("NAME:ColorMapSettings", _ "ColorMapType:=", "Spectrum", _ "SpectrumType:=", "Rainbow", _ "UniformColor:=", Array(127, 255, 255), _ "RampColor:=", Array(255, 127, 127)), _ Array("NAME:Scale3DSettings", _ "m_nLevels:=", 27, _ "m_autoScale:=", true, _ "minvalue:=", 9.34379863739014, _ "maxvalue:=", 13683.755859375, _ "log:=", false, _ "IntrinsicMin:=", 9.34379863739014, _ "IntrinsicMax:=", 13683.755859375), _ Array("NAME:Marker3DSettings", _ "MarkerType:=", 0, _ "MarkerMapSize:=", true, _ "MarkerMapColor:=", false, _ </pre>

	<pre>"MarkerSize:=", 0.25), _ Array("NAME:Arrow3DSettings", _ "ArrowType:=", 1, _ "ArrowMapSize:=", true, _ "ArrowMapColor:=", true, _ "ShowArrowTail:=", true, _ "ArrowSize:=", 0.25))</pre>

UpdateAllFieldsPlots

Updates the All Fields Plots.

UI Access	N/A
Parameters	None.
Return Value	None.

Python Syntax	UpdateAllFieldsPlots()
Python Example	<code>oModule.UpdateAllFieldsPlots()</code>

VB Syntax	UpdateAllFieldsPlots
VB Example	<code>oModule.UpdateAllFieldsPlots</code>

16 - Fields Summary Script Commands

Fields Summary commands should be executed by the Field Overlays module, which is called "FieldsReporter" in scripts.

```
Set oModule = oDesign.GetModule("FieldsReporter")
```

```
oModule.CommandName <args>
```

[EditFieldsSummarySetting](#)

[ExportFieldsSummary](#)

EditFieldsSummarySetting

Creates a fields summary report in a Mechanical design (Thermal or Structural solutions).

UI Access	Mechanical > Fields > Create Fields Summary		
Parameters	Name	Type	Description
	<Calculation>	list	"<Entity Type>", "<Geometry Type>", "<Selected Geometry>", "<Selected Quantity>", " <Normal>", "<Side>"
	<Entity Type>	string	"Boundary" or "Object"
	<Geometry Type>	string	"Surface" or "Volume"
	<Selected Geometry>	string	Name of selected geometry
	<Selected Quantity>	string	Name of selected quantity
	<Normal>	string	Coordinate values of normal direction (for example, "0.00, 0.00, 1.00"), empty string ("") when not applicable
	<Side>	string	"Default", "Adjacent", or "Combined"
Return Value	None		

Python Syntax	EditFieldsSummarySetting(<Calculation>, <Calculation>, ... , <Calculation>)
Python Example (Structural)	<pre>oModule.EditFieldsSummarySetting(["Calculation:=" , ["Object", "Volume", "EndCap", "Equivalent Stress", "", "Default"], "Calculation:=" , ["Object", "Volume", "EndCap_1", "Equivalent Stress", "", "Default"], "Calculation:=" , ["Object", "Volume", "ResistorBody", "Equivalent Stress", "", "Default"],</pre>

	<pre> "Calculation:=" , ["Object", "Volume", "Solder", "Equivalent Stress", "", "Default"], "Calculation:=" , ["Object", "Volume", "Solder_1", "Equivalent Stress", "", "Default"], "Calculation:=" , ["Object", "Volume", "Wire", "Equivalent Stress", "", "Default"], "Calculation:=" , ["Object", "Volume", "Wire_1", "Equivalent Stress", "", "Default"]] </pre>

VB Syntax	<code>EditFieldsSummarySetting(<Calculation>, <Calculation>, ... , <Calculation>)</code>
VB Example (Thermal)	<pre> oModule.EditFieldsSummarySetting Array("Calculation:=", Array("Object", "Surface", "Facelist1", "Mag_HeatFlux", "0.00,0.00,1.00", "Default"), "Calculation:=", Array("Object", "Volume", "HeatSink", "Temperature", "", "Default"), "Calculation:=", Array("Object", "Volume", "Transistor", "Temperature", "", "Default"), "Calculation:=", Array("Boundary", "Surface", "Temperature1", "Heat Flow Rate", "0.00,1.00,0.00", "Default")) </pre>

ExportFieldsSummary

Exports a fields summary report as a CSV file.

UI Access	Mechanical > Fields > Create Fields Summary > [Apply and Export Export]		
Parameters	Name	Type	Description
	<SolutionName>	string	Setup name : SteadyState or Setup name : Transient
	<DesignVariationKey>	string	Nominal
	<ExportFileName>	string	File path to and name of CSV file
	<IntrinsicValue>	string	Intrinsic variable
Return Value	None		

Python Syntax	<code>ExportFieldsSummary(<SolutionName>, <DesignVariationKey>, <ExportFileName>, <IntrinsicValue>)</code>

Python Example	<pre>oModule.ExportFieldsSummary(["SolutionName:=", "Setup1 : Solution", "DesignVariationKey:=", "Nominal", "ExportFileName:=", "D:\\Docu- ments\\Ansoft\\Structural\\CSV\\SummaryReport.csv", "IntrinsicValue:=", ""]) </pre>

VB Syntax	ExportFieldsSummary(<SolutionName>, <DesignVariationKey>, <ExportFileName>, <IntrinsicValue>)
VB Example	<pre>oModule.ExportFieldsSummary Array("SolutionName:=", "Setup1 : Solution", _ "DesignVariationKey:=", "Nominal", "ExportFileName:=", _ "D:\\Documents\\Thermal\\CSV\\SummaryReport2.csv", "Intrins- icValue:=", "") </pre>

This page intentionally
left blank.

17 - User Defined Document Script Commands

The product has to implement the [GetModule](#) call to create the UserDefinedDocument scripting object (e.g., Check AltraSimDesign.cpp (function GetMgrIDispatch())). To access the UserDefinedDocuments scripting object, use:

```
Set oModule = oDesign.GetModule("UserDefinedDocuments")
```

Once you have the scripting object, you can use the following methods:

- [AddDocument](#)
- [EditDocument](#)
- [RenameDocument](#)
- [DeleteDocument](#)
- [UpdateDocument](#)
- [ViewHtmlDocument](#)
- [ViewPdfDocument](#)
- [SaveHtmlDocumentAs](#)
- [SavePdfDocumentAs](#)
- [GetDocumentDefinitionNames](#)
- [DeleteAllDocuments](#)
- [UpdateAllDocuments](#)

You can find examples of how to use these methods on each of the methods' pages. A complete [example of a Python script](#) is also available, as is an [example with a line by line explanation](#).

AddDocument

Creates a new document based on provided data and traces. The document names come from UserDefinedDocument folder under syslib, userlib, and personallib. This creates a document and places it in the Project Manager under **Results > Documents**.

UI Access	Right click on Results > Create Document . Choose a document name.		
Parameters	Name	Type	Description
	<data>	Array	Data the defines the document.
	<traces>	Array	trace data for the inputs in the document.
Return Value	None		

Python Syntax	AddDocument (<data>, <traces>)
Python Example	<pre>oModule.AddDocument (["NAME:Design Summary", "", "SysLib", "DesignSummary", ["NAME:Inputs"]], ["NAME:DocTraces"])</pre>

VB Syntax	AddDocument <data>, <traces>

**VB
Example**

In this example, the document names come from the UserDefinedDocument folder in the syslib, userlib, and personallib folders. This creates a document and places it in the Documents folder under results.

```
oModule.AddDocument _
Array("NAME:Design Summary", _
"", "SysLib", "DesignSummary", _
Array("NAME:Inputs")), _
Array("NAME:DocTraces")
```

The following example is explained in [Explication of a Sample UDD Script](#).

```
oModule.AddDocument Array("NAME:Test Report1", "Test Report", "SysLib", _
"Examples/TestUDDInputs", Array("NAME:Inputs", Array("NAME:DLMetrics", "Solution", _
"Data Line Metrics", -1, -1), Array("NAME:DQ0", "Trace", "DQ0", -1, -1), _
Array("NAME:DQS", "Trace", "DQS", -1, -1), _
Array("NAME:Name", "Text", "User Name", Array("Sita Ramesh")), _
Array("NAME:Summary", "Bool", "Display Summary", Array(true)), _
Array("NAME:Version", "Number", "Script Version")), _
Array("NAME:DocTraces", Array("NAME:DLMetrics", _
Array("User Defined", "", "DDR3 AC-Timing 4-DQ1", Array("Context:=", ""), _
Array("Index:=", Array("All"), "Trise:=", Array("Nominal"), "Tfall:=", _
Array("Nominal"), "Pulse_Width:=", Array("Nominal"), _
"Data_Rate:=", Array("Nominal"), "Length:=", Array("Nominal")), _
Array("Probe Component:=", Array("")), Array()), _
Array("NAME:DQ0", Array( _
```


	<pre>"Standard", "DQ0", "NexximTransient", Array("NAME:Context", "SimValueContext:=", Array(_ 1, 0, 2, 0, false, false, -1, 1, 0, 1, 1, "", 0, 0, _ "DE", false, "0", "DP", false, "20000000", "DT", false, "0.001", "WE", _ false, "100ns", "WM", false, "100ns", "WN", false, "0ps", "WS", false, "0ps")), _ Array("Time:=", Array("All"), "Trise:=", Array("Nominal"), "Tfall:=", Array("Nom- inal"), _ "Pulse_Width:=", Array("Nominal"), "Data_Rate:=", Array("Nominal"), _ "Length:=", Array("Nominal")), Array("Probe Component:=", Array("DQ0")), Array()))</pre>

DeleteAllDocuments

Deletes all documents in the object.

UI Access	Right click on Documents in the Project Manager and click Delete All Documents .
Parameters	None.
Return Value	None.

Python Syntax	DeleteAllDocuments()
Python Example	oModule.DeleteAllDocuments()

VB Syntax	DeleteAllDocuments

VB Example	<code>oModule.DeleteAllDocuments</code>

DeleteDocument

Deletes a specified document.

UI Access	Right click on the created document in the Project Manager under Results > Documents and click Delete .		
Parameters	Name	Type	Description
	<name>	String	Name of the document to be deleted.
Return Value	None.		

Python Syntax	<code>DeleteDocument(<names>)</code>
Python Example	<code>oModule.DeleteDocument("Project Design Summary")</code>

VB Syntax	<code>DeleteDocument <names></code>
VB Example	<code>oModule.DeleteDocument "Project Design Summary"</code>

EditDocument

Edits specified documents. If the document pops up a dialog box, the user can make a change the inputs for the document. The document is regenerated and updated. A new one is *not* created.

UI Access	Right click on the created document in the Project Manager under Results > Documents and click Modify document .

Parameters	Name	Type	Description
	<originalName>	String	Name of the original document
	<modifiedData>	Array	New data to add to or modify the document
	<modifiedtraces>	Array	Trace data for the inputs of the document
Return Value	None.		

Python Syntax	EditDocument(<name>,<data>,<traces>)
Python Example	<pre>oModule.EditDocument("Design Summary", ["NAME:Design Summary", "", "SysLib", "DesignSummary", ["NAME:Inputs"]], ["NAME:DocTraces"])</pre>

VB Syntax	EditDocument <name>,<data>,<traces>

VB Example	<pre>oModule.EditDocument "Design Summary", _ Array("NAME:Design Summary", "", "SysLib", _ "DesignSummary", Array("NAME:Inputs")), Array("NAME:DocTraces")</pre>

GetDocumentDefinitionNames

Document definition names are the list of names that can be used to create a document. They appear when you click on **Create document**. This method returns the filenames of document definitions according to the files in the installation directories:

- syslib/UserDefinedDocuments
- userlib/UserDefinedDocuments
- personallib/UserDefinedDocuments

UI Access	NA		
Parameters	Name	Type	Description
	<separator>	String	Separator used to convey the directory "level"
Return Value	None.		

Python Syntax	GetDocumentDefinitionNames(<separator>)
Python Example	oModule.GetDocumentDefinitionNames("/")

VB Syntax	GetDocumentDefinitionNames <separator>
VB Example	oModule.GetDocumentDefinitionNames "/"

GetDocumentNames

Retrieves the names for all documents.

UI Access	N/A
Parameters	None.
Return Value	Array of strings containing document names.

Python Syntax	GetDocumentNames()
Python Example	<code>oModule.GetDocumentNames()</code>

VB Syntax	GetDocumentNames
VB Example	<code>oModule.GetDocumentNames</code>

RenameDocument

Changes the name of a document.

UI Access	Right click on the created document in the Project Manager under Results> Documents and click Rename .		
Parameters	Name	Type	Description
	<oldName>	String	Current name of the document
	<newName>	String	New name of the document

Return Value	None.

Python Syntax	<code>RenameDocument(<oldName>, <newName>)</code>
Python Example	<code>oModule.RenameDocument("Design Summary", "Project Design Summary")</code>

VB Syntax	<code>RenameDocument <oldName>, <newName></code>
VB Example	<code>oModule.RenameDocument "Design Summary", "Project Design Summary"</code>

SaveHtmlDocumentAs

Saves a pre-existing HTML file to a different name and/or location.

UI Access	Right click on the created document in the Project Manager under Results > Documents and click Save As > HTML .		
Parameters	Name	Type	Description
	<code><name></code>	String	Name of the document to be saved.
	<code><saveTo></code>	String	File path to save the document to.
Return Value	none		

Python Syntax	<code>SaveHtmlDocumentAs(<name>, <saveTo>)</code>
Python Example	<code>oModule.SaveHtmlDocumentAs("Design Summary 1", "DS1.html")</code>

VB Syntax	SaveHtmlDocumentAs <name> <saveTo>
VB Example	<code>oModule.SaveHtmlDocumentAs "Design Summary 1" "DS1.html"</code>

SavePdfDocumentAs

Saves a pre-existing PDF file to a different name and/or location.

UI Access	Right click on the created document in the Project Manager under Results > Documents and click Save As > PDF .		
Parameters	Name	Type	Description
	<name>	String	Name of the document to be saved.
	<saveTo>	String	File path to save the document at.
Return Value	None.		

Python Syntax	SavePdfDocumentAs(<name>, <saveTo>)
Python Example	<code>oModule.SavePdfDocumentAs("Design Summary 1", "DS1.pdf")</code>

VB Syntax	SavePdfDocumentAs <name>, <saveTo>
VB Example	<code>oModule.SavePdfDocumentAs "Design Summary 1", "DS1.pdf"</code>

UpdateAllDocuments

Refreshes the contents of all created documents. This action is made on the folder rather than the individual document.

UI Access	Right click on Results > Documents in the Project Manager and click Update All Documents .
Parameters	None.
Return Value	None.

Python Syntax	UpdateAllDocuments()
Python Example	<code>oModule.UpdateAllDocuments()</code>

VB Syntax	UpdateAllDocuments
VB Example	<code>oModule.UpdateAllDocuments</code>

UpdateDocument

Refreshes the contents of the selected document.

UI Access	Right click on the created document in the Project Manager under Results > Documents and click Update Document .		
Parameters	Name	Type	Description
	<name>	String	Name of the document to be updated
Return Value	None.		

Python Syntax	UpdateDocument(<name>)
Python Example	oModule.UpdateDocument("Test UDD Report")

VB Syntax	UpdateDocument <name>
VB Example	oModule.UpdateDocument "Test UDD Report"

ViewHtmlDocument

Displays a pre-existing document as HTML.

UI Access	Right click on the created document in the Project Manager under Results > Documents and click View Xml Document .		
Parameters	Name	Type	Description
	<name>	String	Name of the document to be viewed as a HTML
Return Value	None		

Python Syntax	ViewHtmlDocument(<name>)
Python Example	oModule.ViewHtmlDocument("Design Summary 1")

VB Syntax	ViewHtmlDocument <name>
VB Example	oModule.ViewHtmlDocument "Design Summary 1"

ViewPdfDocument

Displays a pre-existing document as a PDF file.

UI Access	Right click on the created document in the Project Manager under Results > Documents and click View PDF Document .		
Parameters	Name	Type	Description
	<name>	String	Name of the document to be viewed as a PDF
Return Value	None.		

Python Syntax	ViewPdfDocument(<name>)
Python Example	oModule.ViewPdfDocument("Design Summary 1")

VB Syntax	ViewPdfDocument <name>
VB Example	oModule.ViewPdfDocument "Design Summary 1"

Explication of a Sample UDD Script

This VB script defines a document. It is a portion of one of the UDD example VB scripts:

```
Array("NAME:Test Report",           ' Name of the document
      "Test Report",                ' Description of the document
      "SysLib",                     ' Location of the python script: Syslib, Userlib, PeronalLib, etc.
      "TestUDDReport",              ' Relative path of the script in the UserDefinedDocuments folder
```

' This array is the start of the input definition:

Array("NAME:Inputs", ' Document Inputs keyword

' This array contains the Solution input:

Array("NAME:DLMetrics",	' Input name
"Solution",	' Solution Input Type
"Data Line Metrics",	' Input Description
-1,	' Solution ID
-1),	' Report ID

' This array contains the trace input:

Array("NAME:DQ0",	' Input name
"Trace",	' Trace Input Type
"DQ0",	' Input Description
-1,	' Solution ID
-1),	' Report ID

' This array contains the text input:

Array("NAME:Name",	' Input name
"Text",	' Text Input Type
"User Name",	' Input Description
Array("Sita Ramesh")),	' Default Value

' This array contains the Bool input:

Array("NAME:Summary",	' Input name
"Bool",	' Boolean Input Type
"Display Summary",	' Input Description
Array(true)),	' Default Value

' This array contains the number input:

```

Array("NAME:Version",           ' Input name
"Number",                      ' Number Input Type
"Script Version",              ' Input Description
Array(1021))),                 ' Default Value

' This array contains trace selection for the solution and trace inputs:
Array("NAME:DocTraces",        ' Document traces keyword

' This array has input for "DLMetrics":
Array("NAME:DLMetrics",        ' Input name

' This array defines a trace similar to the UDO. This trace definition is a user-defined solution:
Array("User Defined", "", "DDR3 AC-Timing 4-DQ1", Array("Context:=", ""), Array("Index:=", Array
("All"), "Trise:=", Array("Nominal"), "Tfall:=", Array("Nominal"), "Pulse_Width:=", Array("Nom-
inal"), "Data_Rate:=", Array("Nominal"), "Length:=", Array("Nominal")), Array("Probe Com-
ponent:=", Array(""))), Array()),

' This array is for input "DQ0":
Array("NAME:DQ0",

' This array defines a trace similar to the UDO. This trace definition is a Standard solution:
Array("Standard", "DQ0", "NexximTransient", Array("NAME:Context", "SimValueContext:=", Array(1,
0, 2, 0, false, false, -1, 1, 0, 1, 1, "", 0, 0, "DE", false, "0", "DP", _
false, "20000000", "DT", false, "0.001", "WE", false, "100ns", "WM", false, _
"100ns", "WN", false, "0ps", "WS", false, "0ps")), Array("Time:=", Array("All"), "Trise:=",
Array( _
"Nominal"), "Tfall:=", Array("Nominal"), "Pulse_Width:=", Array("Nominal"), "Data_Rate:=", Array
("Nominal"), "Length:=", Array("Nominal")), Array("Probe Component:=", Array( _
"DQ0")), Array()))

```

Example Python Script: Defining a Document

This script creates a user-defined solution and a document.

```
import ScriptEnv
ScriptEnv.Initialize("Ansoft.ElectronicsDesktop")
oDesktop.RestoreWindow()
oProject = oDesktop.SetActiveProject("BJT_Inverter")
oDesign = oProject.SetActiveDesign("Nexxim1")
oModule = oDesign.GetModule("UserDefinedSolutionModule")

oModule.CreateUserDefinedSolution("UDS Distance Trace Arithmetic Result1", "SysLib", "TraceArithmetic/Distance Sweep Trace Arithmetic",
[
    "Offset 1:=", "0",
    "Scale 1:=", "1",
    "Offset 2:=", "0",
    "Scale 2:=", "1",
    "Operation:=", "Add"
],
[
    [
        "Standard",
        "probel",
        "Transient",
        [
            style="font-family: monospace;">"NAME:Context",
            style="font-family: monospace;">"SimValueContext:=", [1,0,2,0,False,False,-
1,1,0,1,1,"",0,0,"DE",False,"0","DP",False,"500000000","DT",False,"0.001","NUMLEVELS",False,"0","-
WE",False,"10us","WM",False,"10us","WN",False,"0ns","WS",False,"0ns"]
        ],
        [
            "Time:=", ["All"]
        ],
        [
            "Probe Component:=", ["V(Port1)"]
        ]
    ]
]
```

```

        ],
        []
    ],
    [
        "Standard",
        "probe2",
        "Transient",
        [
            "NAME:Context",
            "SimValueContext:=", [1,0,2,0,False,False,-
1,1,0,1,1,"",0,0,"DE",False,"0","DP",False,"500000000","DT",False,"0.001","NUMLEVELS",False,"0",-
"WE",False,"10us","WM",False,"10us","WN",False,"0ns","WS",False,"0ns"]
        ],
        [
            "Time:=", ["All"]
        ],
        [
            "Probe Component:=", ["V(Port1)"]
        ],
        []
    ]
],
[])
oModule = oDesign.GetModule("UserDefinedDocuments")
oModule.AddDocument(
[
    "NAME:Test Report",
    "Test Report",
    "SysLib",
    "TestUDDInputs",
    [
        "NAME:Inputs",
        [
            "NAME:UDS1",

```

```
        "Solution",
        "UDS Distance Trace Arithmetic Result1",
        1000000,
        0
    ],
    [
        "NAME:UDS2",
        "Solution",
        "UDS Distance Trace Arithmetic Result2",
        1000000,
        2
    ]
],
[
    "NAME:DocTraces",
    [
        "NAME:UDS1",
        [
            "User Defined",
            "",
            "UDS Distance Trace Arithmetic Result1",
            [
                "Context:=", ""
            ],
            [
                "Distance:=", ["All"]
            ],
            [
                "Probe Component:=", [""]
            ],
            []
        ]
    ]
]
```

```
    ],
    [
        "NAME:UDS2",
        [
            "User Defined",
            "",
            "UDS Distance Trace Arithmetic Result1",
            [
                "Context:=", ""
            ],
            [
                "Distance:=", ["All"]
            ],
            [
                "Probe Component:=", [""]
            ],
            []
        ]
    ]
])
```


This page intentionally
left blank.

18 - User Defined Solutions Commands

User Defined Solution commands should be executed by the "UserDefinedSolutionModule" module.

```
Set oDesign = oProject.SetActiveDesign("TestDesign1")
Set oModule =
oDesign.GetModule("UserDefinedSolutionModule")
```

The list of commands is as follows:

[CreateUserDefinedSolution](#)

[DeleteUserDefinedSolutions](#)

[EditUserDefinedSolution](#)

CreateUserDefinedSolution

Creates a new user defined solution.

UI Access	Right-click on Results > Create User Defined Solution .		
Parameters	Name	Type	Description
	<SoluName>	String	Name of user defined solution.
	<LibType>	String	Indicates the library where the UDS plugin file is located. This parameter must be one of the following values: "SysLib", "UserLib", "PersonalLib".
	<RelativePath>	String	The path of the UDS plugin file relative to the "UserDefinedOutputs" sub-directory of the library specified by <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.
Return Value	String name of created user defined solution.		

Python Syntax	CreateUserDefinedSolution(<SoluName>, <LibType>, <RelativePath>, <PropList>, <ProbeSelections>, <DynamicProbes>)		
Python Example	<pre>oModule.CreateUserDefinedSolution("ConstantTimestep1", "SysLib", "ConstantTimestep", [], [], [])</pre>		

VB Syntax	CreateUserDefinedSolution <SoluName>, <LibType>, <RelativePath>, <PropList>, <ProbeSelections>, <DynamicProbes>		
VB Example	<pre>oModule.CreateUserDefinedSolution "User Defined Solution 1", _ "SysLib", "Example", _ Array("multiplication_factor:=", "1.2"), _ Array(Array("Modal Solution Data", "Probe 1", "Setup1 : LastAdaptive", _</pre>		

	<pre>Array(), Array("Freq:=", Array("All")), _ Array("Probe Component:=", Array("dB(S(1,1))")), Array()), _ Array("Modal Solution Data", "Probe 2", "Setup1 : LastAdaptive", Array(), _ Array("Freq:=", Array("All")), _ Array("Probe Component:=", Array("mag(S(1,1))")), Array()), _ Array(Array("Modal Solution Data", _ "Dynamic Probe 1", "Setup1 : LastAdaptive", Array(), _ Array("Freq:=", Array("All")), _ Array("Probe Component:=", Array("Freq")), Array()))</pre>

DeleteUserDefinedSolutions

Deletes one or more user defined solutions.

UI Access	Delete button from the User Defined Solutions dialog.		
Parameters	Name	Type	Description
	<SoluNames>	Array	Array of strings containing names of User Defined Solutions to be deleted.
Return Value	None.		

Python Syntax	DeleteUserDefinedSolutions(<SoluNames>)
Python Example	oModule.DeleteUserDefinedSolutions(["Solution1", "Solution2"])

VB Syntax	DeleteUserDefinedSolutions <SoluNames>
VB Example	<pre>oModule.DeleteUserDefinedSolutions _ Array("Solution1", "Solution2")</pre>

EditUserDefinedSolution

Modifies an existing user defined solution.

UI Access	Edit button from the User Defined Solutions dialog box.		
Parameters	Name	Type	Description
	<SoluName>	String	Name of user defined solution to be edited.
	<NewSoluName>	String	New name for the specified user defined solution.
	<LibType>	String	Indicates the library where the UDS plugin file is located. This parameter must be one of the following values: "SysLib", "UserLib", "PersonalLib".
	<RelativePath>	String	The path of the UDS plugin file relative to the "UserDefinedOutputs" sub-directory of the library specified by <LibType>.
	<PropList>	Array	Strings specify name-value pairs corresponding to the UDS properties specified in the plugin file. For example: 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.
Return Value	String name of update user defined solution.		

Python Syntax	<code>EditUserDefinedSolution(<SoluName>, <NewSoluName>, <LibType>, <RelativePath>, <PropList>, <ProbeSelections>, <DynamicProbes>)</code>
Python Example	<pre>oModule.EditUserDefinedSolution("ConstantTimestep1" "ConstantTimestep1After", "SysLib", "ConstantTimestep", [], [], [])</pre>

VB Syntax	<code>EditUserDefinedSolution <SoluName>, <NewSoluName>, <LibType>, <RelativePath>, <PropList>, <ProbeSelections>, <DynamicProbes></code>
VB Example	<pre>oModule.CreateUserDefinedSolution "User Defined Solution 1", _ "User Defined Solution 1", "SysLib", "Example", _ Array("multiplication_factor:=", "1.2"), _ Array(Array("Modal Solution Data", "Probe 1", "Setup1 : LastAdaptive", _ Array(), Array("Freq:=", Array("All"))), _ Array("Probe Component:=", Array("dB(S(1,1))")), Array()), _ Array("Modal Solution Data", "Probe 2", "Setup1 : LastAdaptive", Array(), _ Array("Freq:=", Array("All"))), _ Array("Probe Component:=", Array("mag(S(1,1))")), Array()), _ Array(Array("Modal Solution Data", _ "Dynamic Probe 1", "Setup1 : LastAdaptive", Array(), _</pre>

	<pre>Array("Freq:=", Array("All")), _ Array("Probe Component:=", Array("Freq")), Array()))</pre>

GetUserDefinedSolutionNames

Retrieves user defined solution names.

UI Access	N/A
Parameters	None.
Return Value	Array of strings containing solution names.

Python Syntax	<code>GetUserDefinedSolutionNames()</code>
Python Example	<code>oModule.GetUserDefinedSolutionNames()</code>

VB Syntax	<code>GetUserDefinedSolutionNames</code>
VB Example	<code>oModule.GetUserDefinedSolutionNames</code>

GetUserDefinedSolutionProperties

Obtains properties for a specified user defined solution.

UI Access	N/A

Parameters	Name	Type	Description
	<SoluName>	String	Name of a specified user defined solution.
Return Value	Array of strings containing properties values.		

Python Syntax	GetUserDefinedSolutionProperties(<SoluName>)
Python Example	<code>oModule.GetUserDefinedSolutionProperties("ConstantTimestep1")</code>

VB Syntax	GetUserDefinedSolutionProperties <SoluName>
VB Example	<code>oModule.GetUserDefinedSolutionProperties "ConstantTimestep1"</code>

This page intentionally
left blank.

19 - Network Data Explorer Script Commands

Network Data Explorer (NDE) scripting uses three objects, each with unique script commands:

- **Network Data Explorer** – top-level object obtained by calling `oNDE=oDesktop.GetTool("ndExplorer")`. Network Data Explorer commands are called using `oNDE`.
- **Network Data** – single set of S-parameters, corresponding to a single entry in the UI tree. Network Data commands are called using `oData`.
- **Post Process Settings** – settings that can be applied and removed from network data without making permanent changes to the underlying data. Post Process Settings commands are called using `oPostProc`.

Examples using the above objects:

```
oNDE = oDesktop.GetTool("ndExplorer")  
  
oData = oNDE.Open("D:\folder\test.s2p")  
  
success = oPostProc.AddDiffPair(2, 1, "Diff1", "Comm1", 100, 25)
```

The following commands are described in this section:

Warning: The following commands are preliminary. Their syntax may change. Be prepared to make changes to legacy scripts.

[AddDiffPair](#)

[Cascade \(SPISim\)](#)

[ClearDiffPairs](#)

[Clone](#)

[Close](#)

[Combine \(SPISim\)](#)

[Deembed \(SPISim\)](#)

[DeembedBack \(SPISim\)](#)

[DeembedFront \(SPISim\)](#)

[DisableDiffPairs](#)

[EnableDiffPairs](#)

[ExportCitiFile](#)

[ExportFullWaveSpice](#)

[ExportMatlab](#)

[ExportNMFData](#)

[ExportNetworkData](#)

[ExportSpreadsheet](#)

[ExportTouchstone](#)

[ExportTouchstone2](#)

[Extract \(SPISim\)](#)

[GetFrequencies](#)

[GetFrequencyCount](#)

[GetName](#)

[GetPortCount](#)

[GetPortNumber](#)

[GetPostProcSettings](#)

[GetSolutionVariation](#)

[GetVariation](#)

[HasSameData](#)

[LoadSolution](#)

[Open](#)

[Rename \(SPISim\)](#)

[Renormalize \(SPISim\)](#)

[Reorder](#)

[Reorder \(SPISim\)](#)

[Reset](#)

[SetAllPortImpedances](#)

[SetAllPortImpedances](#)

[SetPortDeembedDistance](#)

[SetPortImpedance](#)

[SetPostProcSettings](#)

[Smooth](#)

[Stretch \(SPISim\)](#)

[Terminate](#)

Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.

AddDiffPair

Specifies a differential pair from two terminal ports.

UI Access	From the Network Data Explorer tab, select Differential Pairs in the NDE ribbon to open the Differential Pairs window. Then select differential pairs from the Pairs group box and click Add Pairs >> .		
Parameters	Name	Type	Description
	<positiveTerminal>	Integer	The portNumber of the positive terminal.
	<negativeTerminal>	Integer	The portNumber of the negative terminal.
	<diffName>	String	The new name of the differential pin (e.g., Diff1).
	<commName>	String	The new name of the common pin (e.g., Comm1).
	<diffImpedance>	Double	The differential pin characteristic impedance.
	<commImpedance>	Double	The common pin characteristic impedance.
	<matched>	Boolean	Specify whether to use matched pair.
Note: The default value is False .			
Return Value	Boolean.		

Python Syntax	AddDiffPair()
Python Example	<code>success = oPostProc.AddDiffPair(2, 1, "Diff1", "Comm1", 100, 25)</code>

VB Syntax	AddDiffPair

VB Example	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.

UI Access	From the Network Data Explorer tab, select Transforms in the NDE ribbon. Then select Cascade from the drop-down menu.		
Parameters	Name	Type	Description
	<dataArray>	Array	These network data objects are cascaded together to create the new network data.
Return Value	Network IDispatch. Note: Cascade returns None if the specified network data does not have compatible terminal ports defined.		

Python Syntax	Cascade()

Python Example	<pre>oNDE = oDesktop.GetTool("ndExplorer") oData2 = oNDE.Cascade([oData1, oData2, oData3])</pre>

VB Syntax	Cascade
VB Example	<pre>Set oNDE = oDesktop.GetTool("ndExplorer") Set oData2 = oNDE.Cascade Array(oData1, oData2, oData3)</pre>

Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.

ClearDiffPairs

Clears all differential pair definitions. All other post-processing settings remain the same.

UI Access	From the Network Data Explorer tab, select Differential Pairs in the NDE ribbon to open the Differential Pairs window. Then select differential pairs from the rightmost box and click << Remove Pairs .
Parameters	None.
Return Value	None.

Python Syntax	ClearDiffPairs()
Python Example	<pre>oNDE = oDesktop.GetTool("ndExplorer")</pre>

	<code>oPostProc.ClearDiffPairs()</code>

VB Syntax	<code>ClearDiffPairs</code>
VB Example	<code>Set oNDE = oDesktop.GetTool("ndExplorer") oPostProc.ClearDiffPairs</code>

Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.

Clone

Creates a copy of the network data object.

UI Access	From the Network Data Explorer tab, select the network data object to clone. Then select Clone from the NDE ribbon.
Parameters	None.
Return Value	Network IDispatch.

Python Syntax	<code>Clone()</code>
Python Example	<code>oNDE = oDesktop.GetTool("ndExplorer") oData1 = oData.Clone()</code>

VB Syntax	Clone
VB Example	<pre>Set oNDE = oDesktop.GetTool("ndExplorer") Set oData1 = oData.Clone</pre>

Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.

Close

Closes the network data object. The object will no longer be accessible.

UI Access	From the Network Data Explorer tab, click Close in the NDE ribbon, or select File > Close .
Parameters	None.
Return Value	None.

Python Syntax	Close()
Python Example	<pre>oNDE = oDesktop.GetTool("ndExplorer") oData.Close()</pre>

VB Syntax	Close
VB Example	<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 Network Data Explorer tab, select Transforms in the NDE ribbon. Then select Combine from the drop-down menu.		
Parameters	Name	Type	Description
	<code><dataArray></code>	Array	These network data objects are combined to create the new network data.
	<code><freqTol></code>	Double	If $\text{abs}(f1 - f2) < \text{freqTol}$, $f1$ and $f2$ are considered the same frequency. Note: The default value is 100 .
Return Value	Network IDispatch. Note: Combine returns None if the network data does not have the same number of ports.		

Python Syntax	Combine()
Python Example	<pre>oNDE = oDesktop.GetTool("ndExplorer") oData4 = oNDE.Combine([oData1, oData2, oData3], 1000)</pre>

VB Syntax	Combine
VB Example	<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, dembedding the frontData and the backData from the front and back of the originalData.

Note: Deembed opens and interacts with **SPISim** to complete.

UI Access	From the Network Data Explorer tab, select Transforms in the NDE ribbon. Then select Deembed from the drop-down menu to open the Deembed... window, and choose Given 3 S-params in order of: Total, Front, Back from the Settings group box.		
Parameters	Name	Type	Description
	<originalData>	Object	Original network data that is deembeded.
	<frontData>	Object	Network data that is deembeded from the front of the original.

	<backData> Object Network data that is deembedded from the back of the original.
Return Value	Network IDispatch. Note: Deembed returns None if the specified network data does not have compatible terminal ports defined.

Python Syntax	Deembed()
Python Example	<pre>oNDE = oDesktop.GetTool("ndExplorer") oData2 = oNDE.Deembed(oData1, oDataFront, oDataBack)</pre>

VB Syntax	Deembed
VB Example	<pre>Set oNDE = oDesktop.GetTool("ndExplorer") Set oData2 = oNDE.Deembed oData1, oDataFront, oDataBack</pre>

Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.

DeembedBack (SPISim)

Creates a new network data, dembedding the backData from the back of the originalData.

Note: DeembedBack opens and interacts with **SPISim** to complete.

UI Access	From the Network Data Explorer tab, select Transforms in the NDE ribbon. Then select Deembed from the drop-down menu to open the Deembed... window, and choose Given 3 S-params in order of: Total, Back from the Settings group box.		
Parameters	Name	Type	Description
	<originalData>	Object	Original network data that is deembeded.
	<backData>	Object	Network data that is deembeded from the back of the original.
Return Value	Network IDispatch. Note: DeembedBack returns None if the specified network data does not have compatible terminal ports defined.		

Python Syntax	DeembedBack()
Python Example	<pre>oNDE = oDesktop.GetTool("ndExplorer") oData2 = oNDE.DeembedBack(oData1, oDataBack)</pre>

VB Syntax	DeembedBack
VB Example	<pre>Set oNDE = oDesktop.GetTool("ndExplorer") Set oData2 = oNDE.DeembedBack oData1, oDataBack</pre>

Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.

DeembedFront (SPISim)

Creates a new network data, dembedding the frontData from the front of the originalData.

Note: DeembedFront opens and interacts with **SPISim** to complete.

UI Access	From the Network Data Explorer tab, select Transforms in the NDE ribbon. Then select Deembed from the drop-down menu to open the Deembed... window, and choose Given 3 S-params in order of: Total, Front from the Settings group box.		
Parameters	Name	Type	Description
	<originalData>	Object	Original network data that is deembedded.
	<frontData>	Object	Network Data that is deembedded from the front of the original.
Return Value	Network IDispatch. Note: DeembedFront returns None if the specified network data does not have compatible terminal ports defined.		

Python Syntax	DeembedFront()
Python Example	<pre>oNDE = oDesktop.GetTool("ndExplorer") oData2 = oNDE.DeembedFront(oData1, oDataFront)</pre>

VB Syntax	DeembedFront
VB Example	<pre>Set oNDE = oDesktop.GetTool("ndExplorer") 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](#).

UI Access	From the Network Data Explorer tab, select Differential Pairs in the NDE ribbon to open the Differential Pairs window. Once differential pairs have been added to the rightmost box (i.e., select differential pairs from the Pairs group box and click Add Pairs >>), remove check marks from the Enabled column to deactivate the corresponding differential pairs.
Parameters	None.
Return Value	None.

Python Syntax	DisableDiffPairs()
Python Example	<pre>oNDE = oDesktop.GetTool("ndExplorer") oPostProc.DisableDiffPairs()</pre>

VB Syntax	DisableDiffPairs
VB Example	<pre>Set oNDE = oDesktop.GetTool("ndExplorer") 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.

UI Access	From the Network Data Explorer tab, select Differential Pairs in the NDE ribbon to open the Differential Pairs window. Once differential pairs have been added to the rightmost box (i.e., select differential pairs from the Pairs group box and click Add Pairs >>), add check marks to the Enabled column to activate the corresponding differential pairs.
Parameters	None.
Return Value	None.

Python Syntax	EnableDiffPairs()
Python Example	<pre>oNDE = oDesktop.GetTool("ndExplorer") success = oPostProc.EnableDiffPairs()</pre>

VB Syntax	EnableDiffPairs

VB Example	<pre>Set oNDE = oDesktop.GetTool("ndExplorer") Set success = oPostProc.EnableDiffPairs</pre>

Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.

ExportCitiFile

Writes the S-parameters to disk in Citifile format (*.cit text file).

UI Access	From the Network Data Explorer tab, select File > Save As to open a Save As explorer window. Then select Citifile (*.cit) from the Save as type: drop-down menu.		
Parameters	Name	Type	Description
	<filePath>	String	The full path to the file.
	<matrixType>	Integer	One of the following: • 0 – S-Parameters • 1 – Y-Parameters • 2 – Z-Parameters • 3 – Gamma • 4 – Impedance Note: The default value is 0.
	<formalType>	Integer	One of the following: • 0 – Magnitude/Angle (degrees)

			• 1 – Real/Imaginary • 2 – dB/Angle (degrees) Note: The default value is 0.
	<precision>	Integer	The number of significant figures. Note: The default value is 10.
	<frequencies>	Array of Doubles	A subset of the calculated frequencies. Any array values that don't have calculated data will be ignored. An empty array will cause all frequencies to be written. Note: The default value is an empty array.
Return Value	Boolean True if file was successfully written; else False.		

Python Syntax	ExportCitiFile()
Python Example	<pre>oNDE = oDesktop.GetTool("ndExplorer") success = oData.ExportCitiFile("D:\folder\nel133.cit", 0, 1, 12, [])</pre>

VB Syntax	ExportCitiFile
VB Example	<pre>Set oNDE = oDesktop.GetTool("ndExplorer") Set success = oData.ExportCitiFile "D:\folder\nel133.cit", 0, 1, 12, Array()</pre>

ExportFullWaveSpice

Use: Export FullWaveSpice data in a format of your choice.

Command: File > Export MacroModel > Broadband (SYZ, FWS....)

Syntax: ExportFullWaveSpice

```
"DesignName", // Design name. Can be left blank, if loading solution from a file.  
true/false, // true - solution loaded from file, false- loaded from design  
"Name", // If loading from design this is the solution name, else this is the  
           // full path of the file from which the solution is loaded  
"variation", // Pick a particular variation. Leave blank if no variation.  
Array("NAME:Frequencies"), // Optional; if none defined all frequencies are used  
Array("NAME:SpiceData", // Spice export options object  
"SpiceType:=", "SSS", // SpiceType can be "PSpice", "HSpice", "Spectre", "SSS",  
           // "Simplorer", "TouchStone1.0", "TouchStone2.0"  
"EnforcePassivity:=", false, // Enforce Passivity true/false  
"EnforceCausality:=", false, // Enforce Causality true/false  
"UseCommonGround:=", false, // Use common ground true/false  
"FittingError:=", 0.5, // Fitting error  
"MaxPoles:=", 400, // Maximum Order  
"PassivityType:=", "ConvexOptimization", // Passivity Type can be "ConvexOptimization",  
           // "PassivityByPerturbation", or "IteratedFittingOfPV"
```

```
"ColumnFittingType:=", "Column", // Column FittingType can be "Column", "Entry", "Matrix"
"SSFittingType:=", "TWA", // SS Fitting Type can be "TWA", "IterativeRational"
"RelativeErrorToleranc:=", false, // Relative error tolerance true/false
"TouchstoneFormat:=", "MA", // Touchstone Format "MA", "RI", "DB"
"TouchstoneUnits:=", "Hz", // Touchstone Units "Hz", "KHz", "MHz", "MHz"
"TouchStonePrecision:=", 8, // Touchstone precision
"ExportDirectory:=", "C:/Examples/LNA/", // Directory to export to
"ExportSpiceFileName:=", "Linckt_HBTest_2.sss", // Spice export file
"FullwaveSpiceFileName:=", "Linckt_HBTest.sss", // FWS file
"CreateNPortModel:=", true // Create a model based on the exported file true/false
)
```

Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.

ExportMatlab

Writes the S-parameters to disk in Matlab format (*.m text file).

UI Access	From the Network Data Explorer tab, select File > Save As to open a Save As explorer window. Then select MATLAB (*.m) from the Save as type: drop-down menu.		
Parameters	Name	Type	Description
	<filePath>	String	The full path to the file.
	<matrixType>	Integer	One of the following: • 0 – S-Parameters

			• 1 – Y-Parameters • 2 – Z-Parameters • 3 – Gamma • 4 – Impedance Note: The default value is 0.
	<i><formalType></i>	Integer	One of the following: • 0 – Magnitude/Angle (degrees) • 1 – Real/Imaginary • 2 – dB/Angle (degrees) Note: The default value is 0.
	<i><precision></i>	Integer	The number of significant figures. Note: The default value is 10.
	<i><frequencies></i>	Array of Doubles	A subset of the calculated frequencies. Any array values that don't have calculated data will be ignored. An empty array will cause all frequencies to be written. Note: The default value is an empty array.
Return Value	Boolean True if file was successfully written; else False.		

Python Syntax	ExportMatlab()
Python Example	<pre>oNDE = oDesktop.GetTool("ndExplorer") success = oData.ExportMatlab("D:\folder\nel33.m", 0, 1, 12, [])</pre>

VB Syntax	ExportMatlab
VB Example	<pre>Set oNDE = oDesktop.GetTool "ndExplorer" Set success = oData.ExportMatlab "D:\folder\nel33.m", 0, 1, 12, Array()</pre>

ExportNMFData

ExportNetworkData

Exports matrix solution data to a file.

UI Access	N/A		
Parameters	Name	Type	Description
	<DesignVariationKey>	String	Design variation key. Pass empty string for the current nominal variation.
	<SolnSelectionArray>	Array	Array of selected solutions. <pre>Array(<SolnSelector>, <SolnSelector>, ...)</pre> If more than one array entry, this indicates a combined Interpolating sweep.
	<SolnSelector>	String	Solution setup name and solution name, separated by a colon.
	<FileFormat>	Integer	File format value. 2 : Tab delimited spreadsheet format (.tab)

		3 : Touchstone (.snp) 4 : CitiFile (.cit) 7 : Matlab (.m) 8 : Terminal Z0 spreadsheet
<OutFile>	String	Full path to the file to write out.
<FreqsArray>	Array	The frequencies to export. The <FreqsArray> argument contains a vector (e.g. "1GHz", "2GHz", ...) to use, or "all". To export all frequencies, use Array("all"). If no frequencies are specified, all frequencies are used.
<DoRenorm>	Boolean	Specifies whether to renormalize the data before export.
<RenormImped>	Double	Real impedance value in ohms, for renormalization. Required in syntax, but ignored if DoRenorm is false.
<DataType>	String	Optional. Type: "S", "Y", or "Z". The matrix to export.
<pass>	Integer	Optional. The pass to export. This is ignored if the sourceName is a frequency sweep. Leaving out this value or specifying -1 gets all passes.
<ComplexFormat>	Integer	Optional. Type: "0", "1", or "2" The format to use for the exported data. 0 = Magnitude/Phase. 1 = Real/Imaginary. 2 = db/Phase.
<Precision>	Integer	Optional. Touchstone number of digits precision. Default if not specified is 15.
<UseExportFreqs>	Boolean	Specifies whether to use export frequencies.
<IncludeGammaComments>	Boolean	Specifies whether to include Gamma and Impedance comments.
<SupportNonStdExport>	Boolean	Specifies whether to support non-standard Touchstone extensions for mixed reference impedance.

Return Value	None.

Python Syntax	<code>ExportNetworkData(<DesignVariationKey>, <SolnSelectionArray>, <SolnSelector>, <FileFormat>, <OutFile>, <FreqsArray>, <DoRenorm>, <RenormImped>, [Optional <DataType>], [Optional <pass>], [Optional <ComplexFormat>], [Optional <Precision>], [Optional <UseExportFreqs>], [Optional <IncludeGammaComments>], [Optional <SupportNonStdExport>])</code>
Python Example	

VB Syntax	<code>ExportNetworkData <DesignVariationKey>, <SolnSelectionArray>, <SolnSelector>, <FileFormat>, <OutFile>, <FreqsArray>, <DoRenorm>, <RenormImped>, [Optional <DataType>], [Optional <pass>], [Optional <ComplexFormat>], [Optional <Precision>], [Optional <UseExportFreqs>], [Optional <IncludeGammaComments>], [Optional <SupportNonStdExport>]</code>
VB Example	<pre>oModule.ExportNetworkData "width='2in'", _ Array("Setup1:Sweep1"), 2, "c:\mydir\out.tab", _ Array("all"), false, 0 oModule.ExportNetworkData "width='2in'", _ Array("Setup1:Sweep1", "Setup1:Sweep2"), 3, _ "c:\mydir\out.s2p", Array(1.0e9, 1.5e9, 2.0e9), _ true, 50.0</pre>

Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.

ExportSpreadsheet

Writes the S-parameters to disk in spreadsheet format (*.tab text file).

UI Access	From the Network Data Explorer tab, select File > Save As to open a Save As explorer window. Then select Data Table (spreadsheet) (*.tab) from the Save as type: drop-down menu.		
Parameters	Name	Type	Description
	<filePath>	String	The full path to the file.
	<matrixType>	Integer	One of the following: • 0 – S-Parameters • 1 – Y-Parameters • 2 – Z-Parameters • 3 – Gamma • 4 – Impedance Note: The default value is 0.
	<formalType>	Integer	One of the following: • 0 – Magnitude/Angle (degrees) • 1 – Real/Imaginary • 2 – dB/Angle (degrees) Note: The default value is 0.
	<precision>	Integer	The number of significant figures.

			Note: The default value is 10 .
	<frequencies>	Array of Doubles	A subset of the calculated frequencies. Any array values that don't have calculated data will be ignored. An empty array will cause all frequencies to be written. Note: The default value is an empty array.
	<renormalize>	Boolean	If True , will renormalize the data to the value of renormImpedance before writing. Note: The default value is False .
	<renormImpedance>	Double	Data will be renormalized to this impedance before writing only if renormalize is True . Note: The default value is 50 ohms .
Return Value	Boolean True if file was successfully written; else False.		

Python Syntax	ExportSpreadsheet()
Python Example	<pre> oNDE = oDesktop.GetTool("ndExplorer") oData = oNDE.Open(<model path>) success = oData.ExportSpreadsheet("D:\folder\ne133.tab", 0, 1, 12, [], True, 23) </pre>

VB Syntax	ExportSpreadsheet
VB Example	<pre>Set oData=oDesktop.GetTool("ndExplorer") success = oData.ExportSpreadsheet "D:\folder\ne133.tab", 0, 1, 12, Array(), True, 23</pre>

Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.

ExportTouchstone

Writes the S-parameters to disk in Touchstone 1.0 format (*.snp text file).

UI Access	From the Network Data Explorer tab, select File > Save As to open a Save As explorer window. Then select Touchstone Format 1.0 (*.snp) from the Save as type: drop-down menu.		
Parameters	Name	Type	Description
	<filePath>	String	The full path to the file.
	<matrixType>	Integer	One of the following: • 0 – S-Parameters • 1 – Y-Parameters • 2 – Z-Parameters • 3 – Gamma • 4 – Impedance Note: The default value is 0.
	<formalType>	Integer	One of the following:

			• 0 – Magnitude/Angle (degrees) • 1 – Real/Imaginary • 2 – dB/Angle (degrees) Note: The default value is 0.
	<precision>	Integer	The number of significant figures. Note: The default value is 10.
	<frequencies>	Array of Doubles	A subset of the calculated frequencies. Any array values that don't have calculated data will be ignored. An empty array will cause all frequencies to be written. Note: The default value is an empty array.
	<renormalize>	Boolean	If True, will renormalize the data to the value of renormImpedance before writing. Note: The default value is False.
	<renormImpedance>	Double	Data will be renormalized to this impedance before writing only if renormalize is True. Note: The default value is 50 ohms.
	<freqUnits>	String	One of the following: "Hz", "KHz", "MHz", "GHz". Note: The default value is "Hz".
Return Value	Boolean True if file was successfully written; else False.		

Python Syntax	ExportTouchstone()
Python Example	<pre>oNDE = oDesktop.GetTool("ndExplorer") success = oData.ExportTouchstone("D:\folder\nel133.s2p", 0, 1, 12, [], True, 23, "GHz")</pre>

VB Syntax	ExportTouchstone
VB Example	<pre>Set oNDE = oDesktop.GetTool("ndExplorer") success = oData.ExportTouchstone "D:\folder\nel133.s2p", 0, 1, 12, Array(), True, 23, "GHz"</pre>

Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.

ExportTouchstone2

Writes the S-parameters to disk in Touchstone 2 format (*.ts text file).

UI Access	From the Network Data Explorer tab, select File > Save As to open a Save As explorer window. Then select Touchstone 2 Format (*.ts) from the Save as type: drop-down menu.		
Parameters	Name	Type	Description
	<filePath>	String	The full path to the file.

	<matrixType>	Integer	One of the following: • 0 – S-Parameters • 1 – Y-Parameters • 2 – Z-Parameters • 3 – Gamma • 4 – Impedance Note: The default value is 0.
	<formalType>	Integer	One of the following: • 0 – Magnitude/Angle (degrees) • 1 – Real/Imaginary • 2 – dB/Angle (degrees) Note: The default value is 0.
	<precision>	Integer	The number of significant figures. Note: The default value is 10.
	<frequencies>	Array of Doubles	A subset of the calculated frequencies. Any array values that don't have calculated data will be ignored. An empty array will cause all frequencies to be written. Note: The default value is an empty array.
	<renormalize>	Boolean	If True, will renormalize the data to the value of renormImpedance before writing.

			Note: The default value is False .
	<renormImpedance>	Double	Data will be renormalized to this impedance before writing only if renormalize is True . Note: The default value is 50 ohms .
	<freqUnits>	String	One of the following: "Hz", "KHz", "MHz", "GHz". Note: The default value is "Hz" .
Return Value	Boolean True if file was successfully written; else False.		

Python Syntax	ExportTouchstone2()
Python Example	<pre>oNDE = oDesktop.GetTool("ndExplorer") success = oData.ExportTouchstone2("D:\folder\nel133.ts", 0, 1, 12, [], True, 23, "GHz")</pre>

VB Syntax	ExportTouchstone2
VB Example	<pre>Set oNDE = oDesktop.GetTool("ndExplorer") Set success = oData.ExportTouchstone2 "D:\folder\nel133.ts", 0, 1, 12, Array(), True, 23, "GHz"</pre>

Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.

Extract (SPISim)

Creates a new smaller network data after removing data for the specified terminal port numbers.

Note: Extract opens and interacts with **SPISim** to complete.

UI Access	From the Network Data Explorer tab, select Transforms in the NDE ribbon. Then select Extract from the drop-down menu.		
Parameters	Name	Type	Description
	<oData>	Object	Data from this object is copied to a new network data and the copy is transformed.
	<portNumbers>	Array of Integers	The port numbers that will be retained (integers between 1 and <i>n</i>).
Return Value	Network IDispatch.		

Python Syntax	Extract()
Python Example	<pre>oNDE = oDesktop.GetTool("ndExplorer") oData2 = oNDE.Extract(oData1, [3, 2])</pre>

VB Syntax	Extract

VB Example	<pre>Set oNDE = oDesktop.GetTool "ndExplorer" Set oData2 = oNDE.Extract oData1, Array(3, 2)</pre>

Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.

GetFrequencies

Returns the frequency values with calculated data in the network data.

UI Access	None.
Parameters	None.
Return Value	Array of doubles.

Python Syntax	GetFrequencies()
Python Example	<pre>oNDE = oDesktop.GetTool("ndExplorer") Freqs = oData.GetFrequencies()</pre>

VB Syntax	GetFrequencies
VB Example	<pre>Set oNDE = oDesktop.GetTool("ndExplorer") Set Freqs = oData.GetFrequencies</pre>

Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.

GetFrequencyCount

Returns the number of frequencies with calculated data in the network data.

UI Access	None.
Parameters	None.
Return Value	Integer number of frequencies.

Python Syntax	GetFrequencyCount()
Python Example	<pre>oData = oDesktop.GetTool("ndExplorer") numFreqs = oData.GetFrequencyCount()</pre>

VB Syntax	GetFrequencyCount
VB Example	<pre>Set oData = oDesktop.GetTool("ndExplorer") Set numFreqs = oData.GetFrequencyCount</pre>

Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.

GetName

Returns the name of the network data.

UI Access	None.
Parameters	None.
Return Value	String name.

Python Syntax	GetName()
Python Example	<pre>oNDE = oDesktop.GetTool("ndExplorer") strName = oData.GetName()</pre>

VB Syntax	GetName
VB Example	<pre>Set oNDE = oDesktop.GetTool("ndExplorer") strName = oNDE.GetName</pre>

Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.

GetPortCount

Returns the total number of ports.

UI Access	From the Network Data Explorer tab, select Edit Ports in the NDE ribbon to open the Port properties window.
Parameters	None.
Return Value	Integer number of ports.

Python Syntax	GetPortCount()
Python Example	<pre>oNDE = oDesktop.GetTool("ndExplorer") num = oData.GetPortCount()</pre>

VB Syntax	GetPortCount
VB Example	<pre>Set oNDE = oDesktop.GetTool("ndExplorer") num = oData.GetPortCount</pre>

Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.

GetPortNumber

Returns the terminal port number for a given terminal port name.

Note: GetPortNumber can be used in other commands that require a port number, if a port name is preferable.

The PortNumber is the index of the name port in the original data and is not changed by any other settings applied to the oPostProc, including the Reorder settings. It is permanently linked to the port name and can be used interchangeably with the port name in many of the PostProcSettings script functions.

UI Access	From the Network Data Explorer tab, ensure the Full Port Names check box is activated. Port numbers will be represented in the object portNames.		
Parameters	Name	Type	Description
	<portName>	String	The designation of the port number.
Return Value	Integer.		

Python Syntax	GetPortNumber()
Python Example	<pre>oNDE = oDesktop.GetTool("ndExplorer") index = oPostProc.GetPortNumber("Input35")</pre>

VB Syntax	GetPortNumber
VB Example	<pre>Set oNDE = oDesktop.GetTool("ndExplorer") index = oPostProc.GetPortNumber "Input35"</pre>

Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.

GetPostProcSettings

Returns a copy of the IDispatch to the postprocessing settings for the network data (oPostProc).

Note: Making changes to the PostProcSettings will not change the network data. To make changes, call oData.SetPostProcSettings to change the network data.

UI Access	None.
Parameters	None.
Return Value	PostProcSettings IDispatch

Python Syntax	GetPostProcSettings()
Python Example	<pre>oNDE = oDesktop.GetTool("ndExplorer") oPostProc = oData.GetPostProcSettings()</pre>

VB Syntax	GetPostProcSettings
VB Example	<pre>Set oNDE = oDesktop.GetTool("ndExplorer") Set oPostProc = oData.GetPostProcSettings</pre>

Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.

GetSolutionVariation

Returns the network data object corresponding to a variation of the solution.

UI Access	None.		
Parameters	Name	Type	Description
	<solutionID>	Integer	ID of the solution (obtained with LoadSolution).
	<variation>	String	The variation key.
Return Value	Network IDispatch.		

Python Syntax	GetSolutionVariation()
Python Example	<pre>oNDE = oDesktop.GetTool("ndExplorer") oData = oNDE.GetSolutionVariation("O, "offset='0.1'")</pre>

VB Syntax	GetSolutionVariation
VB Example	<pre>Set oNDE = oDesktop.GetTool("ndExplorer") oData = oNDE.GetSolutionVariation "O, "offset='0.1'"</pre>

Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.

GetVariation

Returns the variation key for the network data.

UI Access	None.
Parameters	None.
Return Value	String variation key.

Python Syntax	GetVariation()
Python Example	<pre>oNDE = oDesktop.GetTool("ndExplorer") variation = oData.GetVariation()</pre>

VB Syntax	GetVariation
VB Example	<pre>Set oNDE = oDesktop.GetTool("ndExplorer") Set variation = oData.GetVariation</pre>

Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.

HasSameData

Compares one network data object with another network data. Actual calculated values will be compared and no interpolation will be done.

UI Access	None.		
Parameters	Name	Type	Description
	<NetworkData>	Object	The second network data to compare against.
	<matrixType>	Integer	One of the following: • 0 – S-Parameters • 1 – Y-Parameters • 2 – Z-Parameters • 3 – Gamma • 4 – Impedance Note: The default value is 0.
	<comparePortNames>	Boolean	If True , port names must be identical. Note: The default value is True.
	<compareNoise>	Boolean	If True , and both network data have noise data, the comparison will fail unless the noise values also match. Note: The default value is 10.
	<relativeTolerance>	Double	If the absoluteTolerance test fails, then $\text{abs}(\text{value1} - \text{value2}) / \max(\text{abs}(\text{value1}), \text{abs}(\text{value2}))$ must be less than this to be considered

			equal.
			Note: The default value is 1e-14 .
	<absoluteTolerance>	Double	If True , then $\text{abs}(\text{value1} - \text{value2}) < \text{absoluteTolerance}$.
			Note: The default value is 0 .
Return Value	Boolean True if the compared network data have the same frequency and matrix values; otherwise, False.		

Python Syntax	HasSameData()
Python Example	<pre>oNDE = oDesktop.GetTool("ndExplorer") success = oData.HasSameData(oData2, 0, True, False, 1e-10, 0)</pre>

VB Syntax	HasSameData
VB Example	<pre>Set oNDE = oDesktop.GetTool("ndExplorer") Set success = oData.HasSameData oData2, 0, True, False, 1e-10, 0</pre>

Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.

LoadSolution

Loads the specified design (all variations) and returns an integer ID.

UI Access	After analyzing a design, from the Project Manager window, expand the Project Tree and Analysis folders. Then right-click on the completed analysis icon and select Network Data Explorer to open the solution in the Network Data Explorer tab.		
Parameters	Name	Type	Description
	<projectName>	String	Full path to the Electronics Desktop file.
	<designName>	String	Name of the design.
	<solutionName>	String	Name of the solution.
Return Value	Integer ID (identifying the solution); < 0 if the solution is not loaded.		

Python Syntax	LoadSolution()
Python Example	<pre>oNDE = oDesktop.GetTool("ndExplorer") id = oNDE.LoadSolution("D:\folder\Tee.aedt", "HFSS_Test", "Setup1:Sweep1")</pre>

VB Syntax	LoadSolution
VB Example	<pre>Set oNDE = oDesktop.GetTool("ndExplorer") id = oNDE.LoadSolution "D:\folder\Tee.aedt", "HFSS_Test", "Setup1:Sweep1"</pre>

Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.

Open

Reads contents of file and returns the IDispatch for the new network data.

UI Access	From the Network Data Explorer tab, select Open in the NDE ribbon, or select File > Open to open an explorer window. Then navigate to the required S-parameter file.		
Parameters	Name	Type	Description
	<fileName>	String	Full path to the file containing S-parameters.
Return Value	Network IDispatch.		

Python Syntax	Open()
Python Example	<pre>oNDE = oDesktop.GetTool("ndExplorer") oData = oNDE.Open("D:\folder\test.s2p")</pre>

VB Syntax	Open
VB Example	<pre>Set oNDE = oDesktop.GetTool("ndExplorer") Set oData = oNDE.Open "D:\folder\test.s2p"</pre>

Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.

Rename (SPISim)

Creates a new network data of the same size with the terminal ports renamed.

Note: Rename opens and interacts with **SPISim** to complete.

UI Access	From the Network Data Explorer tab, select Transforms in the NDE ribbon. Then select Rename from the drop-down menu.		
Parameters	Name	Type	Description
	<oData>	Object	Data from this object is copied to a new network data and the copy is transformed.
	<portNames>	Array of Strings	The new names (one for each port).
Return Value	Network IDispatch.		

Python Syntax	Rename()
Python Example	<pre>oNDE = oDesktop.GetTool("ndExplorer") oData2 = oNDE.Rename(oData1, ["input", "control", "output"])</pre>

VB Syntax	Rename
VB Example	<pre>Set oNDE = oDesktop.GetTool "ndExplorer" Set oData2 = oNDE.Rename oData1, Array("input", "control", "output")</pre>

Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.

Renormalize (SPISim)

Creates a new network data of the same size after renormalizing the terminal ports using the specified impedance values.

Note: Rename opens and interacts with **SPISim** to complete.

UI Access	From the Network Data Explorer tab, select Transforms in the NDE ribbon. Then select Renormalize from the drop-down menu.		
Parameters	Name	Type	Description
	<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.
Return Value	Network IDispatch. Note: Renormalize returns None if there is not an impedance value for each port.		

Python Syntax	Renormalize()
Python Example	<pre>oNDE = oDesktop.GetTool("ndExplorer") oData2 = oNDE.Renormalize(oData1, [3, 2])</pre>

VB Syntax	Renormalize
VB Example	<pre>Set oNDE = oDesktop.GetTool("ndExplorer") oData2 = oNDE.Renormalize oData1, Array(3, 2)</pre>

Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.

Reorder

Rearranges the terminal port entries but does not change the port names (i.e., switching first and third terminal port positions will give switched values for S(1,1) and S(3,3) but S(Port1,Port1) and S(Port3,Port3) do not change).

UI Access	From the Network Data Explorer tab, select Edit Ports in the NDE ribbon to open the Port properties window. Click+drag the terminal port rows to reorder them. Make changes, as necessary, then click OK .		
Parameters	Name	Type	Description
	<newOrder>	Array of Integers	Must have one entry for each port; all numbers 1 to n must be present.
Return Value	None.		

Python Syntax	Reorder()
Python Example	<pre>oNDE = oDesktop.GetTool("ndExplorer") oPostProc.Reorder([3, 2, 1])</pre>

VB Syntax	Reorder
VB Example	<pre>Set oNDE = oDesktop.GetTool("ndExplorer") oPostProc.Reorder Array(3, 2, 1)</pre>

Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.

Reorder (SPISim)

Creates a new network data with the same number of terminal ports and with the port names in the same order. The data is also reordered so it corresponds to the new order (e.g., the data for S(Port1, Port1) may correspond to S(Port3, Port3).

Note: Reorder opens and interacts with **SPISim** to complete.

UI Access	From the Network Data Explorer tab, select Transforms in the NDE ribbon. Then select Reorder from the drop-down menu.		
Parameters	Name	Type	Description
	<oData>	Object	Data from this object is copied to a new network data and the copy is transformed.
	<portNumbers>	Array of Integers	The port numbers in new order (integers between 1 and <i>n</i>).
Return Value	Network IDispatch. Note: Reorder returns None if the array argument does not contain all the terminal port numbers, in any order.		

Python Syntax	Reorder()
Python Example	<pre>oNDE = oDesktop.GetTool("ndExplorer") oData2 = oNDE.Reorder([3, 2, 1])</pre>

VB Syntax	Reorder
VB Example	<pre>Set oNDE = oDesktop.GetTool("ndExplorer") Set oData2 = oNDE.Reorder Array(3, 2, 1)</pre>

Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.

Reset

Resets the post-processing to what it was when the network data was created.

Note: This is not equivalent to setting post-processing to an empty state. The network data may have been read from a file or created from solution data that already had post-processing settings.

Resetting the oPostProcSetting object does not change any data; it only changes the settings object. If you want to change a data object to apply the changed settings, you must follow the Reset function call with oData.SetPostProcSettings (oPostProc)

UI Access	From the Network Data Explorer tab, select Reset postprocessing in the NDE ribbon.

Parameters	None.
Return Value	None.

Python Syntax	Reset()
Python Example	<pre>oNDE = oDesktop.GetTool("ndExplorer") oPostProc.Reset()</pre>

VB Syntax	Reset
VB Example	<pre>Set oNDE = oDesktop.GetTool("ndExplorer") oPostProc.Reset</pre>

Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.

SetAllPortImpedances

Sets all terminal port impedances in a single call. The impedances are in an array of real or complex values.

UI Access	Through the UI, the terminal ports can only be edited individually. From the Network Data Explorer tab, select Edit Ports in the NDE ribbon to open the Port properties window. Make changes, then click OK .		
Parameters	Name	Type	Description
	<impedances>	Array of Doubles or String	Must have one entry for each port or a single complex (or real) value which will be applied to all ports.

Return Value	None.

Python Syntax	SetAllPortImpedances()
Python Example	<pre>oNDE = oDesktop.GetTool("ndExplorer") oPostProc.SetAllPortImpedances([23, "2+3i", 50]) -or- oData.SetAllPortImpedances(50)</pre>

VB Syntax	SetAllPortImpedances
VB Example	<pre>Set oData=oDesktop.GetTool("ndExplorer") oPostProc.SetAllPortImpedances Array(23, "2+3i", 50) -or- oData.SetAllPortImpedances 50</pre>

Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.

SetPortDeembedDistance

Sets terminal port impedance for a single port.

UI Access	From the Network Data Explorer tab, select Edit Ports in the NDE ribbon to open the Port properties window. Make changes, then click OK .

Parameters	Name	Type	Description
	<portNumber>	Integer	The port number of the changed terminal port (integers between 1 and n).
	<distance>	String	Impedance with units (e.g., "20mm"); if no units, meters are assumed
Return Value	None.		

Python Syntax	SetPortDeembedDistance()
Python Example	<pre>oNDE = oDesktop.GetTool("ndExplorer") oPostProc.SetPortDeembedDistance(2, "20mm")</pre>

VB Syntax	SetPortDeembedDistance
VB Example	<pre>Set oNDE = oDesktop.GetTool("ndExplorer") oPostProc.SetPortDeembedDistance 2, "20mm"</pre>

Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.

SetPortImpedance

Sets port impedance for a single port.

UI Access	From the Network Data Explorer tab, select Edit Ports in the NDE ribbon to open the Port properties window. Make changes, then click OK .

Parameters	Name	Type	Description
	<portNumber>	Integer	The port number of the changed terminal port (integers between 1 and <i>n</i>).
	<impedance>	Double or String	Double if impedance value is real; string format (e.g., 24+10i) for complex impedance.
Return Value	None.		

Python Syntax	SetPortImpedance()
Python Example	<pre>oNDE = oDesktop.GetTool("ndExplorer") oPostProc.SetPortImpedance(2, "25+3i")</pre>

VB Syntax	SetPortImpedance
VB Example	<pre>Set oNDE = oDesktop.GetTool("ndExplorer") oPostProc.SetPortImpedance 2, "25+3i"</pre>

Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.

SetPostProcSettings

Applies postprocessing settings (oPostProc) to the specified network data.

Note: Making changes to the PostProcSettings will not change the network data. To make changes, call `oData.SetPostProcSettings` to change the network data.

UI Access	None.		
Parameters	Name	Type	Description
	<PostProcSettings>	Object	The settings that will be applied to the data.
Return Value	None.		

Python Syntax	<code>SetPostProcSettings()</code>		
Python Example	<pre>oNDE = oDesktop.GetTool("ndExplorer") oData.SetPostProcSettings(oPostProc)</pre>		

VB Syntax	<code>SetPostProcSettings</code>		
VB Example	<pre>Set oNDE = oDesktop.GetTool("ndExplorer") oData.SetPostProcSettings oPostProc</pre>		

Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.

Smooth

Creates a new network data with values smoothed. The number of adjacent points smoothed is user-specified.

UI Access	From the Network Data Explorer tab, select Transforms in the NDE ribbon. Then select Smooth from the drop-down menu.		
Parameters	Name	Type	Description
	<oData>	Object	Data from this object is copied to a new network data and the copy is transformed.
	<order>	Integer	The number of adjacent points that are smoothed. Note: The default value is 10 .
	<enforceCausality>	Boolean	Casual data is calculated before smoothing. Note: The default value is False .
Return Value	Network IDispatch Note: Network IDispatch is returned only if the Smooth function was successfully able to perform the smoothing operation on the supplied data and meet the maximum 0.025 error tolerance. Otherwise it will return a < null object >. The user may want to try changing other input parameters, for example, <code>enforceCausality</code> .		

Python Syntax	Smooth()
Python Example	<pre>oNDE = oDesktop.GetTool("ndExplorer") oData2 = oNDE.Smooth(oData1, 3, True)</pre>

VB Syntax	Smooth
VB Example	<pre>Set oNDE = oDesktop.GetTool("ndExplorer") Set oData2 = oNDE.Smooth oData1, 3, True</pre>

Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.

Stretch (SPISim)

Creates a new network data representing a matrix of transmission lines with the length of those transmission lines multiplied by a factor of *n*.

Note: Stretch opens and interacts with **SPISim** to complete.

UI Access	From the Network Data Explorer tab, select Transforms in the NDE ribbon. Then select Stretch from the drop-down menu.		
Parameters	Name	Type	Description
	<oData>	Object	Data from this object is copied to a new network data and the copy is transformed.
	<factor>	Double	The transmission line lengths are multiplied by this number.
Return Value	Network IDispatch. Note: Stretch returns None if the network data argument does not represent a matrix of transmission lines.		

Python Syntax	Stretch()
Python Example	<pre>oNDE = oDesktop.GetTool("ndExplorer") oData2 = oNDE.Stretch(oData1, 3.1)</pre>

VB Syntax	Stretch
VB Example	<pre>Set oNDE = oDesktop.GetTool("ndExplorer") Set oData2 = oNDE.Stretch oData1, 3.1</pre>

Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.

Terminate

Creates a new network data with specified ports terminated.

UI Access	From the Network Data Explorer tab, select Transforms in the NDE ribbon. Then select Terminate from the drop-down menu.		
Parameters	Name	Type	Description
	<oData>	Object	Data from this object is copied to a new network data and the copy is transformed.
	<portNumbers>	Array of Integers	The port numbers that are terminated (integers between 1 and <i>n</i>).
	<termImpedances>	Array of Complex	The impedance values used to terminate the specified ports.

	Numbers
Return Value	Network IDispatch. Note: Terminate returns None if the port numbers are not valid or there are no impedance value for the port numbers.

Python Syntax	Terminate()
Python Example	<pre>oNDE = oDesktop.GetTool("ndExplorer") oData2 = oNDE.Terminate(oData1, [2, 3], [23, "10+2i"])</pre>

VB Syntax	Terminate
VB Example	<pre>Set oNDE = oDesktop.GetTool("ndExplorer") Set oData2 = oNDE.Terminate oData1, Array(2, 3), Array(23, "10+2i")</pre>

This page intentionally
left blank.

20 - CompInstance Script Commands

CompInstance commands should be executed by the **oDesign2** or **oPropHost2** object.

```
Set oDesign2 = CompInstance.GetParentDesign
```

```
Set oPropHost2 = CompInstance.GetPropHost
```

Callback Scripting Using CompInstance Object

Callback scripts are scripts that can be set in the Property Dialog for individual properties by clicking the button in the Callback column and choosing a script that is saved with the project. Callback scripts can contain any legal script commands including general Ansys script function calls (e.g., `GetApplicationName()`). In addition, Callback scripts can also call functions on a special object named `CompInstance`.

You can obtain an interface to a `CompInstance` in a schematic or layout by calling `oEditor.GetCompInstanceFromRefDes(refDes)`. For more information see Layout Scripting and Schematic Scripting. This interface is also available as a `CompInstance` object in [CompInstance event callbacks](#), such as placing a component in a layout or schematic.

Definitions

<propName> = text string

<value> = double

<valueText> = text string

<fileName> = full path file name

<choices> = string containing menu choices separated by commas

<initialChoice> = string containing initial choice for menu; must be one of the **<choices>**

<scriptName> = string containing name of script stored in project

<bool> is 1 for true or 0 for false

<editorName> is either "Layout" or "SchematicEditor"

The topics for this section include:

[ComplInstance Functions](#)

ComplInstance Functions

Following are commands that can be used to manipulate properties from a ComplInstance script.

The topics for this section include:

[GetComponentName](#)

GetEditor

[GetInstanceID](#)

[GetInstanceName](#)

[GetParentDesign](#)

GetPropHost

[GetPropServerName](#)

GetComponentName

Returns the name of the component corresponding to this ComplInstance.

UI Access	NA		
Parameters	Name	Type	Description

	None
Return Value	String name of component (e.g. MS_TRL) and stores it in "name"

Python Syntax	GetComponentName()
Python Example	<code>name = CompInstance.GetComponentName()</code>

VB Syntax	GetComponentName()
VB Example	<code>name = CompInstance.GetComponentName()</code>

GetInstanceID [Component Instance]

Use: Returns the instanceID of the CompInstance.

Command: None

Syntax: GetInstanceID()

Return Value: String

VB Example: `id = CompInstance.GetInstanceID();`

Returns id of compInstance (e.g. 7) and stores it in "id".

Note that this is not the same number as the one used in the RefDes.

Python Syntax	GetInstanceID()

Python Example	<code>id = CompInstance.GetInstanceID()</code>

GetInstanceName [Component Instance]

Use: Returns the instance name of the component corresponding to this CompInstance.

Command: None

Syntax: GetInstanceName()

Return Value: String

VB Example: `name = CompInstance.GetInstanceName()`;

Returns instanceName (e.g. A7) of compInstance and stores it in "name".

Note that the Instance Name is not the same as the RefDes.

GetParentDesign

Use: Returns an interface to the compInstance's parent design.

Command: None

Syntax: GetParentDesign()

Return Value: Returns interface to design.

Example: `Set oDesign2 = CompInstance.GetParentDesign()`;

Returns the interface to the design containing the compInstance.

This interface can be used to call Design functions. See: [Design Object Script Commands](#).

Python Syntax	GetParentDesign()
Python Example	<code>oDesign2 = CompInstance.GetParentDesign()</code>

GetPropServerName

Use: Returns the PropServerName of the Component corresponding to this CompInstance.

Command: None

Syntax: GetPropServerName()

Return Value: String

VB Example:

```
name = CompInstance.GetPropServerName();
```

Returns propserver name of compInstance (e.g., CompInst@MS_TRL;7) and stores it in "name".

Python Syntax	GetPropServerName()
Python Example	<code>name = CompInstance.GetPropServerName()</code>

This page intentionally
left blank.

21 - 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:

[AddMaterial](#)

[CloneMaterial](#)

[DoesMaterialExist](#)

[EditMaterial](#)

[ExportMaterial](#)

[RemoveMaterial](#)

Related Topics:

[Component Manager Script Commands](#)

[Material Manager Script Commands](#)

[Model Manager Script Commands](#)

[Network Data Explorer Manager Script Commands](#)

[Script and Library Scripts](#)

[Symbol Manager Script Commands](#)

Add [component manager]

*Use:*Add a component

Command: Tools > Edit Configured Libraries > Components > Add Component

*Syntax:*Add Array("NAME:<ComponentName>",
 "Info:=", <ComponentInfo>,
 "RefBase:=", <string>, // reference designator
 "NumParts:=", <int>, // parts per component
 "OriginalComponent:=", <string>
 "Terminal:=", <TerminalInfo>,
 "Terminal:=", <TerminalInfo>, ...
 // The remaining parameters are optional
 Array("NAME:Parameters", // any combo of the following
 "VariableProp:=", <VariableInfo>,
 "CheckboxProp:=", <CheckBoxInfo>,
 "ButtonProp:=", <ButtonInfo>,
 "TextProp:=", <TextInfo>,
 "NumberProp:=", <NumberInfo>,
 "SeparatorProp:=", <SeparatorInfo>,
 "ValueProp:=", <ValueInfo>,
 "MenuProp:=", <MenuInfo>),

```
Array("NAME:Properties", // any combo of the following
"CheckboxProp:=", <CheckBoxInfo>,
"TextProp:=", <TextInfo>,
"NumberProp:=", <NumberInfo>,
"SeparatorProp:=", <SeparatorInfo>,
"ValueProp:=", <ValueInfo>,
"MenuProp:=", <MenuInfo>),
"VPointProp:=", <VPointInfo>,
"PointProp:=", <PointInfo>),
Array("Quantities",
"QuantityProp:=", <QuantityPropInfo>...),
Array("NAME:CosimDefinitions",
<CosimDefInfo>,
<CosimDefInfo>...)
```

Return Value:<string>

```
// composite name of the component.
// If the name requested conflicts with the name of an existing
// component, the requested name is altered to be unique.
// The name returned reflects any change made to be unique.
```

Parameters:<ComponentName>:

<string> // [simple name](#) of the component

<ComponentInfo>:

Array("Type:=", <TypeInfo>,

"NumTerminals:=", <int>,

"DataSource:=", <string>,

"ModifiedOn:=", <ModifiedOnInfo>,

"Manufacturer:=", "<string>,"

"Symbol:=", <string>,"

"Footprint:=", <string>,"

"Description:=", <string>,"

"InfoTopic:=", <string>,"

"InfoHelpFile:=", <string>,"

"IconFile:=", <string>,"

"LibraryName:=", "",

"OriginalLocation:=", "Project", // Project Location

"Author:=", <string>,"

"OriginalAuthor:=", <string>,"

"CreationDate:=", <int>)

<TypeInfo>:

An integer that is the or-ing of bits for each product listed below. The default setting is 0xffffffff (4294967295) which indicates valid for all products. In the component editing dialog, checking different boxes in the "Specify products for which this component is valid" grid control sets specific flags that correspond to the following hex/decimal settings:

Nexxim -- 100 binary, 4 decimal, 0x4

SIwaveDeNovo -- 1000 binary, 8 decimal, 0x8

Simplorer -- 10000 binary, 16 decimal, 0x10

MaxwellCircuit -- 100000 binary, 32 decimal, 0x20

<ModifiedOnInfo>:

An integer that corresponds to the number of seconds that have elapsed since 00:00 hours, Jan 1, 1970 UTC from the system clock.

<TerminalInfo>:

Array(<string>, // symbol pin

<string> // footprint pin

<string>, // gate name

<bool>, // shared

<int>, // equivalence number

<int>, // what to do if unconnected: flag as error:0, ignore:1

<string> // description

<Nature>)

<Nature>:

<string> // content varies as follows

Nexxim/Circuit:

"Electrical" // the only choice

Simplorer:

// several choices

"Electrical", "Magnetic", "Fluidic", "Translational",

"Translational_V", "Rotational", "Rotational_V",

""Radiant", "Thermal", or <VHDLPackageName>

<VHDLPackageName>:

<string> // in the form <Library>.<Package>

<Library>:

<string> // name of the VHDL library

<Package>:

<string> // name of the VHDL package

<VariableInfo>:

```
Array(<string>, // name
<FlagLetters>,
<string>, // description
"CB:=", <string>, // optional - script for call back
<string>) // value: number, variable, or expression
```

<FlagLetters>:

```
<string> // "D" - has description parameter,
// "RD" - readonly & has description parameter,
// or "RHD" - readonly, hidden, & has description parameter
```

<CheckBoxInfo>:

```
Array(<string>, // name
<FlagLetters>,
<string>, // description
"CB:=", <string>, // optional - script for call back
<bool>) // value: true or false
```

<ButtonInfo>:


```
Array(<string>, // name
<FlagLetters>,
<string>, // description
<string>, // button title
<string>, // extra text
<ClientID>,
"ButtonPropClientData:= ", <ClientDataArray>)
```

```
<ClientID>:
<int> // specifies Button Prop Client
// 0 - unknown, ButtonPropClientData
// array will be empty
// 1 - Netlist Prop Client
// 2 - not used
// 3 - File Name Prop Client
```

```
<ClientDataArray>:
varies with <ClientID>
```

```
<ClientID> is 0 or 1: empty array
```

Array()

<ClientID> is 3:

Array("InternalFormatText:=", "<prefix><RelativePath>")

<prefix>:

<string> // "<Project>", "<PersonalLib>", "<UserLib>", or "<SysLib>"

<RelativePath>:

<string> // relative path to file from <prefix>

<TextInfo>:

Array(<string>, // name

<FlagLetters>,

<string>, // description

"CB:=", <string>, // optional - script for call back

<string>) // value: a text string

<NumberInfo>:

Array(<string>, // name

<FlagLetters>,

<string>, // description

"CB:=", <string>, // optional - script for call back

<real>, // value: a number

<string>) // units

<SeparatorInfo>:

Array(<string>, // name

<FlagLetters>,

<string>, // description

"CB:=", <string>, // optional - script for call back

<string>) // value: a text string

<ValueInfo>:

Array(<string>, // name

<FlagLetters>,

<string>, // description

"CB:=", <string>, // optional - script for call back

<string>) // value: a number, variable or expression

<MenuPropInfo>:

Array(<string>, // name

```
<FlagLetters>,  
<string>, // description  
<string>, // menu choices - separated by commas  
<int>) // 0 based index of current menu choice
```

```
<VPointInfo>:  
Array(<string>, // name  
<FlagLetters>,  
<string>, // description  
"CB:=", <string>, // optional - script for call back  
<string>, // x value: number with length units  
<string>) // y value: number with length units
```

```
<PointInfo>:  
Array(<string>, // name  
<FlagLetters>,  
<string>, // description  
"CB:=", <string>, // optional - script for call back  
<real>, // x value  
<real>) // y value
```

```
<QuantityPropInfo>:
```

```
Array(<string>, // name  
<FlagLetters>,  
<string>, // description  
<string>, // value  
<TypeString>,  
<TypeStringDependentInfo>)
```

```
<TypeString>:  
<string> // "Across", "Through", or "Free"
```

```
<TypeStringDependentInfo>:
```

```
<TypeString> is "Free" :  
<string>, // direction: "In", "Out", "InOut", or "DontCare"  
// Following <string> is not present if direction is "DontCare"  
<string> // when to calculate: "BeforeAnalogSolver",  
// "BeforeStateGraph", "AfterStateGraph", or "DontCareWhen"
```

```
<TypeString> is "Across" or "Through":  
<int>, // terminal 1  
<int> // terminal 2
```

```
<CosimDefInfo>:  
Array("NAME:CosimDefinition",  
"CosimulatorType:=", <int>,  
"CosimDefName:=", <string> // "HFSS 3D Layout", "Circuit",  
// "Custom", or "Netlist"  
"IsDefinition:=", <bool>,  
final array member(s) vary with CosimDefName)
```

final array members for HFSS 3D Layout:

```
"CosimStackup:=", <string>,  
"CosimDmbedRatio:=", <int>
```

final array members for Circuit:

```
"ExportAsNport:=", <int>,  
"UsePjt:=", <int>
```

final array member for Custom:

```
"DefinitionCompName:=", <string>
```

final array member for Netlist:

"NetlistString:=", <string>

VB Example:

Dim name

```
oComponentManager.Add (Array("NAME:MyComponent", _  
"Info:=", Array("Type:=", 4294901767, _  
"NumTerminals:=", 2, _  
"DataSource:=", "", _  
"ModifiedOn:=", 1071096503, _  
"Manufacturer:=", "Ansys", _  
"Symbol:=", "bendo", _  
"Footprint:=", "BENDO", _  
"Description:=", "", _  
"InfoTopic:=", "", _  
"InfoHelpFile:=", "", _  
"IconFile:=", "", _  
"LibraryName:=", "", _  
"OriginalLocation:=", "Project", _  
"Author:=", "", _  
"OriginalAuthor:=", "", _
```

```
"CreationDate:= ", 1147460679), _  
"Refbase:=", "U", _  
"NumParts:=", 1, _  
"OriginalComponent:=", "", _  
"Terminal:=", Array("n1", _  
"n1", _  
"A", _  
false, _  
0, _  
1, _  
"", _  
"Electrical"), _  
"Terminal:=", Array("n2", _  
"n2", _  
"A", _  
false, _  
1, _  
0, _  
"", _  
"Electrical"), _  
Array("NAME:Parameters", _
```



```
"MenuProp:=", Array("CoSimulator", _  
"D", _  
"", _  
"Default,HFSS 3D Layout,Circuit,Custom,Netlist", _  
0), _  
"ButtonProp:=", Array("CosimDefinition", _  
"D", _  
"", _  
"", _  
"Edit", _  
0, _  
"ButtonPropClientData:=", Array()), _  
Array("NAME:CosimDefinitions", _  
Array("NAME:CosimDefinition", _  
"CosimulatorType:=", 0, _  
"CosimDefName:=", "HFSS 3D Layout", _  
"IsDefinition:=", true, _  
"CosimStackup:=", "Layout stackup", _  
"CosimDmbedRatio:=", 3), _  
Array("NAME:CosimDefinition", _
```

```
"CosimulatorType:=", 1, _
"CosimDefName:=", "Circuit", _
"IsDefinition:=", true, _
"ExportAsNport:=", 0, _
"UsePjt:=", 0), _
Array("NAME:CosimDefinition", _
"CosimulatorType:=", 2, _
"CosimDefName:=", "Custom", _
"IsDefinition:=", true, _
"DefinitionCompName:=", ""), _
Array("NAME:CosimDefinition", _
"CosimulatorType:=", 3, _
"CosimDefName:=", "Netlist", _
"IsDefinition:=", true, _
"NetlistString:=", "")))
```

Python Syntax	Add ["NAME:<ComponentName>", "Info:=", <ComponentInfo>, "RefBase:=", <string>, // reference designator "NumParts:=", <int>, // parts per component "OriginalComponent:=", <string>

```

"Terminal:=", <TerminalInfo>,
"Terminal:=", <TerminalInfo>, ...

The remaining parameters are optional.

["NAME:Parameters", // any combo of the following
"VariableProp:=", <VariableInfo>,
"CheckboxProp:=", <CheckBoxInfo>,
"ButtonProp:=", <ButtonInfo>,
"TextProp:=", <TextInfo>,
"NumberProp:=", <NumberInfo>,
"SeparatorProp:=", <SeparatorInfo>,
"ValueProp:=", <ValueInfo>,
"MenuProp:=", <MenuInfo>],
["NAME:Properties", Any combination of the following:
"CheckboxProp:=", <CheckBoxInfo>,
"TextProp:=", <TextInfo>,
"NumberProp:=", <NumberInfo>,
"SeparatorProp:=", <SeparatorInfo>,
"ValueProp:=", <ValueInfo>,
"MenuProp:=", <MenuInfo>,
"VPointProp:=", <VPointInfo>,

```

	<pre>"PointProp:=", <PointInfo>], ["Quantities", "QuantityProp:=", <QuantityPropInfo>...], ["NAME:CosimDefinitions", <CosimDefInfo>, <CosimDefInfo>...]]</pre>
Python Example	<pre>oComponentManager.Add(["NAME:Component", "Info:=", ["Type:=", 0, "NumTerminals:=", 0, "DataSource:=", "", "ModifiedOn:=", 1467910752, "Manufacturer:=", "", "Symbol:=", "Component", "ModelNames:=", "", "Footprint:=", "", "Description:=", "", "InfoTopic:=", "", "InfoHelpFile:=", "",</pre>

	<pre>"IconFile:=", "", "Library:=", "", "OriginalLocation:=", "Project", "IEEE:=", "", "Author:=", "", "OriginalAuthor:=", "", "CreationDate:=", 1467910746, "ExampleFile:=", ""], "Refbase:=", "U", "NumParts:=", 1, "ModSinceLib:=", True, "CompExtID:=", 2])</pre>

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", "X:=", 2, "Y:=", 4], ["NAME:Coordinate", "X:=", 6, "Y:=", 8]])</pre>

```
]
]
)
oDesign.AddDataset (
[
"NAME:$ds1",
[
"NAME:Coordinates",
[
"NAME:Coordinate",
"X:=", 2,
"Y:=", 4
],
[
"NAME:Coordinate",
"X:=", 6,
"Y:=", 8
]
]
]
]
```

	)

VB Syntax	AddDataset <DatasetDataArray>
VB Example	<pre>oProject.AddDatasetArray("NAME:ds1", Array("NAME:Coordinates", Array("NAME:Coordinate", "X:=", 1, "Y:=", 2, Array("NAME:Coordinate", "X:=", 3, "Y:=", 4), Array("NAME:Coordinate", "X:=", 5, "Y:=", 7), Array("NAME:Coordinate", "X:=", 6, "Y:=", 20))) oDesign.AddDatasetArray("NAME:ds1", Array("NAME:Coordinates", Array("NAME:Coordinate", "X:=", 1, "Y:=", 2, Array("NAME:Coordinate", "X:=", 3, "Y:=", 4), Array("NAME:Coordinate", "X:=", 5, "Y:=", 7), Array("NAME:Coordinate", "X:=", 6, "Y:=", 20)))</pre>

AddDefinitionFromBlock

Adds a material definition from block text (same definition format as would be contained in the material library file) by library type (using definition folder name). This scripting command directly supports the .AMAT (or .ASURF) definition formats.

UI Access	N/A		
Parameters	Name	Type	Description

	<defBlock>	String	Text of the new material definition in block form.
	<defFolderName>	String	Library type (by definition folder name)
	<newTimeStamp>	String	New timestamp (time_t as integer number of seconds since 1/1/1970 12:00am, as string), default is current time
	<replaceExisting>	Boolean	True to replace existing, False to choose a new unique name if an existing definition is found
Return Value	A property scripting object for the definition.		

P- yt- h- o- n S- y- nt- ax	AddDefinitionFromBlock(<defBlock>, <defFolderName>, <newTimeStamp>, <replaceExisting>)		
P- yt- h- o- n E- x- a- m- pl- e	<pre>oProject = oDesktop.NewProject() oProject.InsertDesign("HFSS", "HFSSDesign1", "DrivenModal", "") oDesign = oProject.SetActiveDesign("HFSSDesign1") oEditor = oDesign.SetActiveEditor("3D Modeler") oEditor.CreateBox([</pre>		

```
"NAME:BoxParameters",

"XPosition:="          , "-0.4mm",
    "YPosition:="      , "-1mm",
    "ZPosition:="      , "0mm",
    "XSize:="          , "1.4mm",
    "YSize:="          , "1.6mm",
    "ZSize:="          , "0.6mm"

],

[

"NAME:Attributes",
    "Name:="            , "Box1",

"Flags:="              , "",

"Color:="              , "(143 175 143)",

"Transparency:="       , 0,

"PartCoordinateSystem:=", "Global",

"UDMId:="              , "",
```

```

    "MaterialValue:="          , "\"vacuum\"",

    "SurfaceMaterialValue:=", "\"\"",

    "SolveInside:="          , True,

    "ShellElement:="          , False,

    "ShellElementThickness:=", "0mm",

    "IsMaterialEditable:="    , True,

    "UseMaterialAppearance:=", False,

    "IsLightweight:="          , False

  ])

oDefinitionManager = oProject.GetDefinitionManager()

defBlock = "$begin 'vacuum2' $begin 'AttachedData' $begin 'MatAppearanceData' property_data-
='appearance_data' Red=230 Green=230 Blue=230 Transparency=0.95 $end 'MatAppearanceData'
$end 'AttachedData' simple('permittivity', 1) ModTime=1499970477 $end 'vacuum2'"

added = oDefinitionManager.AddDefinitionFromBlock(defBlock, "Materials", "10101010", True)
addedName = ''

  if isinstance(added, basestring):

```

```
        addedName = added
    elif isinstance(added, list):
        addedName = added[0]
    else:

        addedName = added.GetName().replace("Materials:", "")
    AddInfoMessage(os.path.basename(__file__) + " result: " + addedName)
    materialNameInQuotes = "\"" + addedName + "\""
    oEditor.ChangeProperty(
        [
            "NAME:AllTabs",

            [
                "NAME:Geometry3DAttributeTab",

                [
                    "NAME:PropServers",

                    "Box1"
                ],
            ],
        ],
```

```
[
    "NAME:ChangedProps",
    [
        "NAME:Material",
        "Value:=", materialNameInQuotes
    ]
    ]
]
```

AddMaterial

Adds a local material.

UI Access	Add Material in the material editor.		
Parameters	Name	Type	Description
	<MaterialParams>	Array	["NAME: <name of the material to be added>",

			<MatProperty>, <MatProperty>, ...]
<MatProperty>	Array		For simple material: "<PropertyName>:=", <value> For anisotropic material: ["NAME:<PropertyName>", "property_type:=", "AnisoProperty", "unit:=", <Unit>," "component1:=", <value>," "component2:=", <value>," "component3:=", <value>))]
<PropertyName>	String		Should be one of the following (depending on the material, design, and solution types): Electromagnetic (Maxwell-exclusive material properties omitted, see Maxwell Scripting help): "permittivity", "permeability", "conductivity", "dielectric_loss_tangent", "magnetic_loss_tangent", "electric_coercivity", "magnetic_coercivity", "saturation_mag", "lande_g_factor", "delta_H", "delta_h_freq", "mass_

		density" Thermal (including solids, Icepak fluid flow, and Mechanical rotating fluid modeling): "thermal_conductivity", "mass_density", "specific_heat", "thermal_expansion_coefficient", "thermal_material_type", "viscosity", "diffusivity", "molecular_mass", "clarity_type" Structural: "mass_density", "youngs_modulus", "poissons_ratio", "thermal_expansion_coefficient"
<Unit>	String	Possible values (Maxwell-exclusive properties omitted, see Maxwell Scripting Help; other missing entries are unitless): conductivity: "siemens/m" saturation_mag: "uTesla", "mTesla", "tesla", "kTesla", "uGauss", "mGauss", "gauss", "kGauss" delta_H: "A_per_meter", "kA_per_meter", "Oe", "kOe" delta_h_frequency: "Hz", "kHz", "MHz", "GHz", "THz", "rps", "per_sec" mass_density: "kg/m^3" thermal_conductivity: "W/m-C" specific_heat: "J/kg-C" youngs_modulus: "N/m^2"

	<table><tr><td></td><td></td><td>thermal_expansion_coefficient: "1/C"</td></tr></table>			thermal_expansion_coefficient: "1/C"
		thermal_expansion_coefficient: "1/C"		
Return Value	None			

Python Syntax	AddMaterial (["NAME:<MaterialName>", <MatProperty>, <MatProperty>, ...])
Python Example	<pre>oDefinitionManager.AddMaterial(["permittivity:=", "2.2", "0.002"]) oDefinitionManager.AddMaterial [("NAME:Material2",_ "dielectric_loss_tangent:=", "44", Array("NAME:saturation_mag",_ "property_type:=", "AnisoProperty",_ "unit:=", "Gauss",_ "component1:=", "11", _ "component2:=", "22", _ "component3:=", "33"), _ "delta_H:=", "440e")]</pre>

VB Syntax	AddMaterial Array("NAME:<MaterialName>", <MatProperty>, <MatProperty>, ...)
VB Example	<pre>oDefinitionManager.AddMaterial Array("permittivity:=", "2.2", "0.002") oDefinitionManager.AddMaterial Array("NAME:Material2",_ "dielectric_loss_tangent:=", "44", Array("NAME:saturation_mag",_ "property_type:=", "AnisoProperty",_ "unit:=", "Gauss",_ "component1:=", "11", _ "component2:=", "22", _ "component3:=", "33"), _ "delta_H:=", "440e")</pre>

Add [padstack manager]

Use: Add a padstack

Command: Tools > Edit Configured Libraries > Padstacks > Add Padstack

Syntax: Add Array("NAME:<PadstackName>",
 "ModTime:=", <ModifiedOnInfo>,
 "Library:=", "", // name of the library

```
"LibLocation:=", "Project", // location of the named library
```

```
Array("NAME:psd",
```

```
"nam:=", <PadstackName>,
```

```
"lib:=", "", // name of the library
```

```
"mat:=", "", // hole plating material
```

```
"plt:=", "0", // percent of hole's radius filled by plating
```

```
Array("NAME:pds",
```

```
<LayerGeometryArray>,
```

```
<LayerGeometryArray....>),
```

```
"hle:=", <PadInfo>
```

```
"hRg:=", <HoleRange>,
```

```
"sbsh:=", <SolderballShape>,
```

```
"sbpl:=", <SolderballPlacement>,
```

```
"sbr:=", <string>, // solderball diameter, real with units
```

```
"sb2:=", <string>, // solderball mid diameter, real with units
```

```
"sbn:=", <string>), // name of solderball material
```

```
"ppl:=", <PadPortLayerArray>)
```

Return Value: [simple name](#) of the added padstack

// If the name requested conflicts with the name of an existing

// padstack, the requested name is altered to be unique.

// The name returned reflects any change made to be unique.

Parameters: <PadstackName>:

<string> // [simple name](#) of padstack to create

<ModifiedOnInfo>:

An integer that corresponds to the number of seconds that have elapsed since 00:00 hours, Jan 1, 1970 UTC from the system clock.

<LayerGeometryArray>:

Array("Name:lgm",

"lay:=", <string>, // definition layer name

"id:=", <int>, // definition layer id

"pad:=", <PadInfo>, // pad

"ant:=", <PadInfo>, // antipad

"thm:=", <PadInfo>, // thermal pad

"X:=", <string>, // pad x connection, real with units

"Y:=", <string>, // pad y connection, real with units

"dir:=", <DirectionString>) // pad connection direction

<PadInfo>:

```
Array("shp:=", <PadShape>,  
"Szs:=", <DimensionArray>,  
"X:=", <string>, // x offset, real with units  
"Y:=", <string>, // y offset, real with units  
"R:=", <string>) // rotation, real with units
```

<PadShape>:

<string> one of these choices

"No" // no pad

"Cir" // Circle

"Sq" // Square

"Rct" // Rectangle

"Ov" // Oval

"Blt" // Bullet

"Ply" // Polygons

"R45" // Round 45 thermal

"R90" // Round 90 thermal

"S45" // Square 45 thermal

"S90" // Square 90 thermal

<DimensionArray>:

Array(<string>, ...) // each string is a real with units for one of the dimensions of the shape

<DirectionString>:

<string> one of these choices

"No" // no direction

"Any" // any direction

"0" // 0 degrees

"45" // 45 degrees

"90" // 90 degrees

"135" // 135 degrees

"180" // 180 degrees

"225" // 225 degrees

"270" // 270 degrees

"315" // 315 degrees

<HoleRange>:

<string> one of these choices

"Thr" // through all layout layers

"Beg" // from upper pad layer to lowest layout layer

"End" // from upper layout layer to lowest pad layer

"UTL" // from upper pad layer to lowest pad layer

<SolderballShape>:

<string> one of these choices

"None" // no solderball

"Cyl" // cylinder solderball

"Sph" // spheroid solderball

<SolderballPlacement>:

<string> one of these choices

"abv" // above padstack

"blw" // below padstack

<PadPortLayerArray>:

Array(<int>, <int>,.....) where each int is a layer id

VB Example:

```
oPadstackManager.Add Array("NAME:Circle - through3", "ModTime:=", 1235765635, "Library:=", _  
"", "LibLocation:=", "Project", Array("NAME:psd", "nam:=", "Circle - through3", "lib:=", _  
"", "mat:=", "", "plt:=", "0", Array("NAME:pds", Array("NAME:lgm", "lay:=", "Top Signal",  
"id:=", _  
0, "pad:=", Array("shp:=", "Cir", "Szs:=", Array("2.5mm"), "X:=", "0mm", "Y:=", _  
"0mm", "R:=", "0deg"), "ant:=", Array("shp:=", "Cir", "Szs:=", Array("3.5mm"), "X:=", _  
"0mm", "Y:=", "0mm", "R:=", "0"), "thm:=", Array("shp:=", "No", "Szs:=", Array(), "X:=", _
```

```
"0mm", "Y:=", "0mm", "R:=", "0"), "X:=", "0mm", "Y:=", "0mm", "dir:=", "Any"), Array("NAME:lgm",  
"lay:=", _  
"SignalA", "id:=", 1, "pad:=", Array("shp:=", "Cir", "Szs:=", Array("2mm"), "X:=", _  
"0mm", "Y:=", "0mm", "R:=", "0deg"), "ant:=", Array("shp:=", "Cir", "Szs:=", Array( _  
"3mm"), "X:=", "0mm", "Y:=", "0mm", "R:=", "0"), "thm:=", Array("shp:=", "No", "Szs:=", Array(),  
"X:=", _  
"0mm", "Y:=", "0mm", "R:=", "0"), "X:=", "0mm", "Y:=", "0mm", "dir:=", "Any"), Array("NAME:lgm",  
"lay:=", _  
"SignalB", "id:=", 2, "pad:=", Array("shp:=", "Cir", "Szs:=", Array("2mm"), "X:=", _  
"0mm", "Y:=", "0mm", "R:=", "0deg"), "ant:=", Array("shp:=", "Cir", "Szs:=", Array( _  
"3mm"), "X:=", "0mm", "Y:=", "0mm", "R:=", "0deg"), "thm:=", Array("shp:=", "No", "Szs:=", Array  
(, "X:=", _  
"0mm", "Y:=", "0mm", "R:=", "0"), "X:=", "0mm", "Y:=", "0mm", "dir:=", "Any"), Array("NAME:lgm",  
"lay:=", _  
"Ground", "id:=", 3, "pad:=", Array("shp:=", "No", "Szs:=", Array(), "X:=", _  
"0mm", "Y:=", "0mm", "R:=", "0deg"), "ant:=", Array("shp:=", "No", "Szs:=", Array(), "X:=", _  
"0mm", "Y:=", "0mm", "R:=", "0"), "thm:=", Array("shp:=", "R90", "Szs:=", Array( _  
"3mm", "0.75mm", "1mm"), "X:=", "0mm", "Y:=", "0mm", "R:=", "0"), "X:=", "0mm", "Y:=", _  
"0mm", "dir:=", "Any"), Array("NAME:lgm", "lay:=", "Bottom signal", "id:=", 5, "pad:=", Array  
("shp:=", _  
"Cir", "Szs:=", Array("1mm"), "X:=", "0mm", "Y:=", "0mm", "R:=", "0deg"), "ant:=", Array("shp:=",  
_
```

```
"Cir", "Szs:=", Array("2mm"), "X:=", "0mm", "Y:=", "0mm", "R:=", "0deg"), "thm:=", Array
("shp:=", _
"No", "Szs:=", Array(), "X:=", "0mm", "Y:=", "0mm", "R:=", "0"), "X:=", "0mm", "Y:=", _
"0mm", "dir:=", "Any")), "hle:=", Array("shp:=", "Cir", "Szs:=", Array("1.5mm"), "X:=", _
"0mm", "Y:=", "0mm", "R:=", "0deg"), "hRg:=", "End", "sbsh:=", "Sph", "sbpl:=", _
"abv", "sbr:=", "750um", "sb2:=", "1200um", "1200um", "sbn:=", "solder"), "ppl:=", Array( _
0, 1, 2, 3, 5))
```

Add [symbol manager]

Use: Add a symbol

Command: Tools > Edit Configured Libraries > Symbols > Add Symbol

Syntax: Add Array("NAME:<SymbolName>",
 "ModTime:=", <ModifiedTimeInfo>,
 "Library:=", "", // Library name
 "LibLocation:=", "Project", // Project Location
 <PinDefInfo>,
 <PinDefInfo>,... // optional, to define pins
 <GraphicsDataInfo>, // optional, to define graphics
 <PropDisplayMapInfo>)) // optional, to define property displays

Return Value: <string>

 // [composite name](#) of the symbol.

// If the name requested conflicts with the name of an existing
// symbol, the requested name is altered to be unique.
// The name returned reflects any change made to be unique.

Parameters: <SymbolName>:

<string> // [simple name](#) of the symbol being added

<ModifiedOnInfo>:

An integer that corresponds to the number of seconds that have elapsed
since 00:00 hours, Jan 1, 1970 UTC from the system clock.

<PinDefInfo>:

Array("NAME:PinDef",
"Pin:=", Array (<string>, // pin name
<real>, // x location
<real>, // y location
<real>, // angle in radians
<PinType>,
<real>, // line width
<real>, // line length

```
<bool>, // mirrored  
<int>, // color  
<bool>, // true if visible, false if not  
<string>, // hidden net name  
<OptionalPinInfo>, // optional info  
<PropDisplayMapInfo>)) // optional
```

```
<PinType>:  
<string> // "N" : normal pin  
// "I" : input pin  
// "O" : output pin
```

```
<OptionalPinInfo>:  
// Specify both or neither  
<bool>, // true if name is to be shown  
<bool>, // true if number is to be shown
```

```
<PropDisplayMapInfo>:  
Array("NAME:PropDisplayMap",  
<PropDisplayInfo>,
```

<PropDisplayInfo>,...)

<PropDisplayInfo>:

<NameString>, Array(<DisplayTypeInfo>,

<DisplayLocationInfo>,

<int>, // optional, level number

<TextInfo>)

<NameString>:

<string> // PropertyName:=, where PropertyName is the name of

// the property to be displayed

<DisplayTypeInfo>:

<int> // 0 : No display

// 1 : Display name only

// 2 : Display value only

// 3 : Display both name and value

// 4: Display evaluated value only

// 5: Display both name and evaluated value

<DisplayLocationInfo>:

```
<int> // 0 : Left  
// 1 : Top  
// 2 : Right  
// 3 : Bottom  
// 4 : Center  
// 5 : Custom placement
```

```
<GraphicsDataInfo>:  
Array("NAME:Graphics",  
// one or more of the following  
<RectInfo>,  
<CircleInfo>,  
<ArcInfo>,  
<LineInfo>,  
<PolygonInfo>,  
<TextInfo>,  
<ImageInfo>)
```

```
<RectInfo>:  
"Rect:=", Array(<real>, // line width  
<int>, // fill pattern
```

<int>, // color

<real>, // angle, in radians

<real>, // x position of center

<real>, // y position of center

<real>, // width

< real>) // height

<CircleInfo>:

"Circle:=", Array(<real>, // line width

<int>, // fill pattern

<int>, // color

<real>, // x position of center

<real>, // y position of center

< real>) // radius

<ArcInfo>:

"Arc:=", Array(<real>, // line width

<int>, // line pattern

<int>, // color

<real>, // x position of center

<real>, // y position of center
< real>, // radius
<real>, // start angle, in radians
<end>) // end angle, in radians

<LineInfo>:

"Line:=", Array(<real>, // line width
<int>, // line pattern
<int>, // color
<PointInfo>, // must specify at least 2 points
<PointInfo>...)
<PointInfo>:
<real>, // x position
<real> // y position

<PolygonInfo>:

"Polygon:=", Array(<real>, // line width
<int>, // fill pattern
<int>, // color
<PointInfo>, // must specify at least 3 points
<PointInfo>...)

<TextInfo>:

"Text:=", Array(<real>, // x position

<real>, // y position

<real>, // angle, in radians

<Justification>,

<bool>, // is plotter font

<string>, // font name

<int>, // color

<string>) // text string

<Justification>:

<int> // 0 : left top

// 1 : left base

// 2 : left bottom

// 3 : center top

// 4 : center base

// 5 : center bottom

// 6 : right top

// 7 : right base

// 8 : right bottom

<ImageInfo>:

"Image:=", Array(<RectInfo>,

<ImageData>,

<bool>) // is mirrored

<ImageData>:

<string>, // file path

<int>, // 0 : use the file path and link to it

// 1 : ignore file path and use next parameter

<string> // text data, only present if preceding int is 1

VB Example:

```
oSymbolManager.Add Array("NAME:MySymbol", _  
"ModTime:=", 1070989137, _  
"Library:=", "", _  
"LibLocation:=", "Project", _  
Array("NAME:PinDef", _  
"Pin:=", Array ("newpin0", _  
0.00254, _ 0, _  
0, _
```



```
"N", _  
0, _  
0.00254, _  
false, _  
0, _  
true, _  
"", _  
false, _  
false)), _  
Array("NAME:PinDef", _  
"Pin:=", Array ("newpin1", _  
-0.00254, _  
0, _  
3.14159265358979, _  
"N", _  
0, _  
0.00254, _  
false, _  
0, _  
true, _
```

```
"",_  
false, _  
false)),  
Array("NAME:Graphics", _  
"Rect:=", Array(0, _  
0, _  
12566272, _  
0, _  
4.33680868994202e-019, _  
-0.000635, _  
0.00508, _  
0.002794), _  
"Circle:=", Array(0, _  
0, _  
12566272, _  
0.000127, _  
0.000635, _  
0.000635)))
```

AreMaterialPropertiesEqual

Checks whether named material compared to added material data have equivalent major properties.

UI Access	None.		
Parameters	Name	Type	Description
	material_data	<string>	This will be the same format as the AddMaterial() input parameter, material_data. For example, material_data can be something like this.["NAME:Mold_Material", "CoordinateSystemType:=", "Cartesian", "BulkOrSurfaceType:=", 1, ["NAME:PhysicsTypes", "set:=", ["Thermal"]], "thermal_conductivity:=", "0.8", "mass_density:=", "1980", "specific_heat:=", "960"].
	<MaterialName>	<String>	Name of the material to compare with material database
Return Value	Boolean: • True – named materials have equivalent major properties. • False – named materials do not have equivalent major properties.		

Python Syntax	AreMaterialPropetiesEqual(<material_data>, '<MaterialName>')
Python Example	oDefinitionManager.AreMaterialPropertiesEqual(material_data, 'Mold_Material')

VB Syntax	AreMaterialPropertiesEqual <material_data>, <MaterialName>
VB Example	oDefinitionManager.AreMaterialPropetiesEqual material_data, "modified_epoxy"

AreSurfaceMaterialPropertiesEqual

Checks whether named surface material compared to added surface material data have equivalent major properties.

UI Access	None.		
Parameters	Name	Type	Description
	material_data	<string>	This will be the same format as the AddMaterial() input parameter, material_data. For example, material_data can be something like this.["NAME:Mold_Material", "CoordinateSystemType:=", "Cartesian", "BulkOrSurfaceType:=", 1, ["NAME:PhysicsTypes", "set:=", ["Thermal"]], "thermal_conductivity:=", "0.8", "mass_density:=", "1980", "specific_heat:=", "960"]
	<MaterialName>	<String>	Name of the surface material to compare with material database
Return Value	Boolean: • True – named surface materials have equivalent major properties. • False – named surface materials do not have equivalent major properties.		

Python Syntax	AreSurfaceMaterialPropetiesEqual(<material_data>, '<MaterialName>')
Python Example	oDefinitionManager.AreSurfaceMaterialPropertiesEqual(material_data, 'Mold_Material')

VB Syntax	AreMaterialPropertiesEqual <material_data>, <MaterialName>
VB Example	oDefinitionManager.AreMaterialPropetiesEqual material_data, "modified_epoxy"

CloneMaterial

Clones a local material.

UI Access	N/A		
Parameters	Name	Type	Description
	<matName>	String	Name of existing material.
	<newName>	String	Name for newly cloned material.
Return Value	Boolean: • 1 - Material is cloned. • 0 - Existing material not found or a conflict with the new material name.		

Python Syntax	CloneMaterial (<matName>, <newName>)
Python Example	oDefinitionManager.CloneMaterial ("copper1", "copper3")

VB Syntax	CloneMaterial <matName>, <newName>
VB Example	oDefinitionManager.CloneMaterial "copper1", "copper3"

DeleteDataset

Deletes a specified dataset. This can be executed by the oProject, or oDesign variables.

UI Access	Project > Datasets > Remove.		
Parameters	Name	Type	Description
	<DatasetName>	String	Name of the dataset found in the project.

Return Value	None.

Python Syntax	DeleteDataset (<DatasetName>)
Python Example	<pre>oProject.DeleteDataset('\$ds1') oDesign.DeleteDataset('\$ds1')</pre>

VB Syntax	DeleteDataset <DatasetName>
VB Example	<pre>oProject.DeleteDataset "\$ds1" oDesign.DeleteDataset "\$ds1"</pre>

DoesMaterialExist

Checks for the presence of a material in the library by name

UI Access	None.		
Parameters	Name	Type	Description
	<MaterialName>	<String>	Name of the material to search for in the material database
Return Value	Boolean: • True – specified material exists. • False – specified material does not exist.		

Python Syntax	DoesMaterialExist(<MaterialName>)
Python Example	oDefinitionManager.DoesMaterialExist("modified_epoxy")

VB Syntax	DoesMaterialExist <MaterialName>
VB Example	oDefinitionManager.DoesMaterialExist "modified_epoxy"

Edit [component manager]

Modifies an existing component

Command: Tools > Edit Configured Libraries > Components > Edit Component

Syntax: Edit <ComponentName>,

```
    Array("NAME:<NewComponentName>",  
    "Info:=", <ComponentInfo>,  
    "RefBase:=", <string>, // reference designator  
    "NumParts:=", <int>, // parts per component  
    "OriginalComponent:=", <string>  
    "Terminal:=", <TerminalInfo>,  
    "Terminal:=", <TerminalInfo>, ...  
    // The remaining parameters are optional  
    Array("NAME:Parameters", // any combo of the following  
    "VariableProp:=", <VariableInfo>,
```

```
"CheckboxProp:=", <CheckBoxInfo>,
"ButtonProp:=", <ButtonInfo>,
"TextProp:=", <TextInfo>,
"NumberProp:=", <NumberInfo>,
"SeparatorProp:=", <SeparatorInfo>,
"ValueProp:=", <ValueInfo>,
"MenuProp:=", <MenuInfo>),
Array("NAME:Properties", // any combo of the following
"CheckboxProp:=", <CheckBoxInfo>,
"TextProp:=", <TextInfo>,
"NumberProp:=", <NumberInfo>,
"SeparatorProp:=", <SeparatorInfo>,
"ValueProp:=", <ValueInfo>,
"MenuProp:=", <MenuInfo>),
"VPointProp:=", <VPointInfo>,
"PointProp:=", <PointInfo>),
Array("Quantities",
"QuantityProp:=", <QuantityPropInfo>...),
Array("NAME:CosimDefinitions",
<CosimDefInfo>,
<CosimDefInfo>...)
```


Return Value: <string>

// [composite name](#) of the component.
// If the name requested conflicts with the name of an existing
// component, the requested name is altered to be unique.
// The name returned reflects any change made to be unique.

Parameters: <ComponentName>:

<string> // [composite name](#) of the component to edit

<NewComponentName>:

<string> // new [simple name](#) for the component

<ComponentInfo>:

Array("Type:=", <TypeInfo>,
"NumTerminals:=", <int>,
"DataSource:=", <string>,
"ModifiedOn:=", <ModifiedOnInfo>,
"Manufacturer:=", "<string>,"
"Symbol:=", <string>,
"Footprint:=", <string>,

```
"Description:=", <string>,  
"InfoTopic:=", <string>,  
"InfoHelpFile:=", <string>,  
"IconFile:=", <string>,  
"LibraryName:=", <string>,  
"OriginalLocation:=", <string>, // Project Location  
"Author:=", <string>,  
"OriginalAuthor:=", <string>,  
"CreationDate:= ", <int>)
```

<TypeInfo>:

An integer that is the or-ing of bits for each product listed below. The default setting is 0xffffffff (4294967295) which indicates valid for all products. In the component editing dialog, checking different boxes in the "Specify products for which this component is valid" grid control sets specific flags that correspond to the following hex/decimal settings:

Nexxim -- 100 binary, 4 decimal, 0x4

SIwaveDeNovo -- 1000 binary, 8 decimal, 0x8

Simplorer -- 10000 binary, 16 decimal, 0x10

MaxwellCircuit -- 100000 binary, 32 decimal, 0x20

<ModifiedOnInfo>:

An integer that corresponds to the number of seconds that have elapsed since 00:00 hours, Jan 1, 1970 UTC from the system clock.

<TerminalInfo>:

Array(<string>, // symbol pin

<string> // footprint pin

<string>, // gate name

<bool>, // shared

<int>, // equivalence number

<int>, // what to do if unconnected: flag as error:0, ignore:1

<string>, // description

<Nature>)

<Nature>:

<string> // content varies as follows

Nexxim/Circuit:

"Electrical" // the only choice

Simplorer:

// several choices

"Electrical", "Magnetic", "Fluidic", "Translational",

"Translational_V", "Rotational", "Rotational_V",

""Radiant", "Thermal", or <VHDLPackageName>

<VHDLPackageName>:

<string> // in the form <Library>.<Package>

<Library>:

<string> // name of the VHDL library

<Package>:

<string> // name of the VHDL package

<VariableInfo>:

Array(<string>, // name

<FlagLetters>,

<string>, // description

"CB:=", <string>, // optional - script for call back

<string>) // value: number, variable, or expression

<FlagLetters>:

<string> // "D" - has description parameter,

// "RD" - readonly & has description parameter,
// or "RHD" - readonly, hidden, & has description parameter

<CheckBoxInfo>:

Array(<string>, // name

<FlagLetters>,

<string>, // description

"CB:=", <string>, // optional - script for call back

<bool>) // value: true or false

<ButtonInfo>:

Array(<string>, // name

<FlagLetters>,

<string>, // description

<string>, // button title

<string>, // extra text

<ClientID>,

"ButtonPropClientData:= ", <ClientDataArray>)

<ClientID>:

<int> // specifies Button Prop Client
// 0 - unknown, "ButtonPropClientData"
// array will be empty
// 1 - Netlist Prop Client
// 2 - not used
// 3 - File Name Prop Client

<ClientDataArray>:
varies with <ClientID>

<ClientID> is 0 or 1: empty array
Array()

<ClientID> is 3:
Array("InternalFormatText:=", "<prefix><RelativePath>")

<prefix>:
<string> // "<Project>", "<PersonalLib>", "<UserLib>", or "<SysLib>"

<RelativePath>:
<string> // relative path to file from <prefix>

<TextInfo>:

Array(<string>, // name

<FlagLetters>,

<string>, // description

"CB:=", <string>, // optional - script for call back

<string>) // value: a text string

<NumberInfo>:

Array(<string>, // name

<FlagLetters>,

<string>, // description

"CB:=", <string>, // optional - script for call back

<real>, // value: a number

<string>) // units

<SeparatorInfo>:

Array(<string>, // name

<FlagLetters>,

<string>, // description

"CB:=", <string>, // optional - script for call back

<string>) // value: a text string

<ValueInfo>:

Array(<string>, // name

<FlagLetters>,

<string>, // description

"CB:=", <string>, // optional - script for call back

<string>) // value: a number, variable or expression

<MenuPropInfo>:

Array(<string>, // name

<FlagLetters>,

<string>, // description

<string>, // menu choices - separated by commas

<int>) // 0 based index of current menu choice

<VPointInfo>:

Array(<string>, // name

<FlagLetters>,

<string>, // description

"CB:=", <string>, // optional - script for call back

<string>, // x value: number with length units

<string>) // y value: number with length units

<PointInfo>:

Array(<string>, // name

<FlagLetters>,

<string>, // description

"CB:=", <string>, // optional - script for call back

<real>, // x value

<real>) // y value

<QuantityPropInfo>:

Array(<string>, // name

<FlagLetters>,

<string>, // description

<string>, // value

<TypeString>,

<TypeStringDependentInfo>)

<TypeString>:

<string> // "Across", "Through", or "Free"

<TypeStringDependentInfo>:

"Free" :

<string>, // direction: "In", "Out", "InOut", or "DontCare"

// Following <string> is not present if direction is "DontCare"

<string> // when to calculate: "BeforeAnalogSolver",

// "BeforeStateGraph", "AfterStateGraph", or "DontCareWhen"

"Across" or "Through"

<int>, // terminal 1

<int> // terminal 2

<CosimDefInfo>:

Array("NAME:CosimDefinition",

"CosimulatorType:=", <int> ,

"CosimDefName:=", <string> // "HFSS3D", "Circuit",

// "Custom", or "Netlist"

"IsDefinition:=", <bool> ,

final array member(s) vary with CosimDefName)

final array members for HFSS 3D Layout:

```
"CosimStackup:=", <string>,  
"CosimDmbedRatio:=", <int>
```

final array members for Circuit:

```
"ExportAsNport:=", <int>,  
"UsePjt:=", <int>
```

final array member for Custom:

```
"DefinitionCompName:=", <string>
```

final array member for Netlist:

```
"NetlistString:=", <string>
```

VB Example:

```
Dim name  
name = oComponentManager.Edit ("Nexxim Circuit Elements\BJTs:Level01_NPN", _  
Array("NAME:Level01_NPN", _  
"Info:=", Array("Type:=", 4294901764, _  
"NumTerminals:=", 3, _  
"DataSource:=", "Ansys built-in component", _  
"ModifiedOn:=", 1152722112, _
```

```
"Manufacturer:=", "", _
"Symbol:=", "nexx_bjt_npn", _
"Footprint:=", "", _
"Description:=", "BJT, GP, NPN", _
"InfoTopic:=", "NXBJT1.htm", _
"InfoHelpFile:=", "nexximcomponents.chm", _
"IconFile:=", "bjtsn.bmp", _
"Library:=", "Nexxim Circuit Elements\BJTs", _
"OriginalLocation:=", "SysLibrary ", _
"Author:=", "", _
"OriginalAuthor:=", "", _
"CreationDate:=", 1152722102), _
"Refbase:=", "Q", _
"NumParts:=", 1, _
"Terminal:=", Array("collector", _
"collector", _
"A", _
false, _
6, _
0, _
"", _
```

```
"Electrical"), _  
"Terminal:=", Array("base", _  
"base", _  
"A", _  
false, _  
7, _  
0, _  
"", _  
"Electrical"), _  
"Terminal:=", Array("emitter", _  
"emitter", _  
"A", _  
false, _  
8, _  
0, _  
"", _  
"Electrical"), _  
Array("NAME:Parameters", _  
"TextProp:=", Array("LabelID", _  
"HD", _
```

```
"Property string for netlist ID", _
"Q@ID"), _
"TextProp:=", Array("MOD", _
"D", _
"Name of model data reference", _
"required"), _
"VariableProp:=", Array("AREA", _
"D", _
"Emitter area multiplying factor, which affects curents, resistances, and capacitances", _ "1"),
_
"VariableProp=", Array("AREAB", _
"D", _
"Base AREA", _
"1"), _
"VariableProp:=", Array( "AREAC", _
"D", _
Collector AREA", _
"1"), _
"VariableProp:=", Array("DTEMP", _
"D", _
"The difference between element and circuit temperature deg Cel)", _
"0"), _
```

```
"VariableProp:=", Array("M", _  
"D", _  
"Multiplier factor to simulate multiple BJTs in paralel", _ "1"), _  
"ButtonProp:=", Array("NexximNetlist", _  
"HD", _  
"", _  
"Q@ID %0 %1 %2 *MOD(@MOD) *AREA (AREA=@AREA) " & _  
" *AREAB (AREAB=@AREAB) *AREAC (AREAC=@ " & _  
"AREAC) *DTEMP (DTEMP=@DTEMP) *M (M=@M) ", _  
"Q@ID %0 %1 %2 *MOD(@MOD) " & _ "*AREA (AREA=@AREA) AREAB (AREAB=@AREAB) *AREAC (AREAC=@ " & _  
"AREAC) *DTEMP (DTEMP=@DTEMP) *M (M=@M) ", _  
1, _  
"ButtonPropClientData:=", Array()), _  
"TextProp:=", Array( "ModelName", _  
"HD", _  
"", _  
"Q"))))  
  
VB Example:  
  
Dim name2  
name2 = oComponentManager.Edit "MyComponent", _
```

```
(Array("NAME:MyOtherComponent", _  
"Info:=", Array("Type:=", 4294901767, _  
"NumTerminals:=", 2, _  
"DataSource:=", "", _  
"ModifiedOn:=", 1071096503, _  
"Manufacturer:=", "Ansys", _  
"Symbol:=", "bendo", _  
"Footprint:=", "BENDO", _  
"Description:=", "", _  
"InfoTopic:=", "", _  
"InfoHelpFile:=", "", _  
"IconFile:=", "", _  
"LibraryName:=", "", _  
"OriginalLocation:=", "Project", _  
"Author:=", "", _  
"OriginalAuthor:=", "", _  
"CreationDate:= ", 1147460679), _  
"Refbase:=", "U", _  
"NumParts:=", 1, _  
"OriginalComponent:=", "", _  
"Terminal:=", Array("n1", _
```



```
"n1", _  
"A", _  
false, _  
0, _  
0, _  
"", _  
"Electrical"), _  
"Terminal:=", Array("n2", _  
"n2", _  
"A", _  
false, _  
1, _  
0, _  
"", _  
Electrical"), _  
Array("NAME:Parameters", _  
"MenuProp:=", Array("CoSimulator", _  
"D", _  
"", _  
"Default,HFSS3D,Circuit,Custom,Netlist", _
```

```
0), _  
"ButtonProp:=", Array("CosimDefinition", _  
"D", _  
"", _  
"", _  
"Edit", _  
0, _  
"ButtonPropClientData:=", Array()), _  
Array("NAME:CosimDefinitions", _  
Array("NAME:CosimDefinition", _  
"CosimulatorType:=", 0, _  
"CosimDefName:=", "HFSS3D", _  
"IsDefinition:=", true, _  
"CosimStackup:=", "Layout stackup", _  
"CosimDmbedRatio:=", 3), _  
Array("NAME:CosimDefinition", _  
"CosimulatorType:=", 1, _  
"CosimDefName:=", "Circuit", _  
"IsDefinition:=", true, _  
"ExportAsNport:=", 0, _  
"UsePjt:=", 0), _
```

```
Array("NAME:CosimDefinition", _  
      "CosimulatorType:=", 2, _  
      "CosimDefName:=", "Custom", _  
      "IsDefinition:=", true, _  
      "DefinitionCompName:=", ""), _  
Array("NAME:CosimDefinition", _  
      "CosimulatorType:=", 3, _  
      "CosimDefName:=", "Netlist", _  
      "IsDefinition:=", true, _  
      "NetlistString:=", "")))
```

Python Syntax	Edit <ComponentName>, ["NAME:<NewComponentName>", "Info:=", <ComponentInfo>, "RefBase:=", <string>, // reference designator "NumParts:=", <int>, // parts per component "OriginalComponent:=", <string> "Terminal:=", <TerminalInfo>, "Terminal:=", <TerminalInfo>, ... #The remaining parameters are optional

```
["NAME:Parameters", // any combo of the following
```

```
"VariableProp:=", <VariableInfo>,
```

```
"CheckboxProp:=", <CheckBoxInfo>,
```

```
"ButtonProp:=", <ButtonInfo>,
```

```
"TextProp:=", <TextInfo>,
```

```
"NumberProp:=", <NumberInfo>,
```

```
"SeparatorProp:=", <SeparatorInfo>,
```

```
"ValueProp:=", <ValueInfo>,
```

```
"MenuProp:=", <MenuInfo>],
```

```
["NAME:Properties", # any combo of the following
```

```
"CheckboxProp:=", <CheckBoxInfo>,
```

```
"TextProp:=", <TextInfo>,
```

```
"NumberProp:=", <NumberInfo>,
```

```
"SeparatorProp:=", <SeparatorInfo>,
```

```
"ValueProp:=", <ValueInfo>,
```

```
"MenuProp:=", <MenuInfo>),
```

```
"VPointProp:=", <VPointInfo>,
```

```
"PointProp:=", <PointInfo>),
```

```
["Quantities",
```

```
"QuantityProp:=", <QuantityPropInfo>...],
```

```
["NAME:CosimDefinitions",
```

	<pre> <CosimDefInfo>, <CosimDefInfo>...]) </pre>
Python Example	<pre> name = oComponentManager.Edit ("Simplorer Circuit Elements\BJTs:Level01_NPN", _ ["NAME:Level01_NPN", "Info:=", ["Type:=", 4294901764,_ "NumTerminals:=", 3, "DataSource:=", "Ansoft built-in component",_ "ModifiedOn:=", 1152722112, "Manufacturer:=", "", _ "Symbol:=", "nexx_bjt_npn", "Footprint:=", "", _ "Description:=", "BJT, GP, NPN", "InfoTopic:=", "NXBJT1.htm", _ "InfoHelpFile:=", "nexximcomponents.chm", "IconFile:=", "bjtsn.bmp", _ "Library:=", "Nexxim Circuit Elements\BJTs",_ "OriginalLocation:=", "SysLibrary ", "Author:=", "", _ "OriginalAuthor:=", "", "CreationDate:=", 1152722102], _ "Refbase:=", "Q", "NumParts:=", 1, "Terminal:=", ["collector", _ "collector", "A", false, 6, 0, "", "Electrical"], _ "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", _ </pre>

```

"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"]]])

```

**Python
Example
2**

```

name2 = oComponentManager.Edit ("MyComponent",_

(["NAME:MyOtherComponent","Info:=", ["Type:=", 4294901767, _

"NumTerminals:=", 2, "DataSource:=", "", _

"ModifiedOn:=", 1071096503, "Manufacturer:=", "Ansoft", _

"Symbol:=", "bendo", "Footprint:=", "BENDO", _

"Description:=", "", "InfoTopic:=", "", _

"InfoHelpFile:=", "", "IconFile:=", "", _

"LibraryName:=", "", "OriginalLocation:=", "Project", _

"Author:=", "", "OriginalAuthor:=", "", _

```

```
"CreationDate:= ", 1147460679], "Refbase:=", "U", _  
  
"NumParts:=", 1, "OriginalComponent:=", "", _  
  
"Terminal:=", ["n1", "n1", "A", false, 0, 0, "", _  
  
"Electrical"], "Terminal:=", ["n2", "n2", "A", _  
  
false, 1, 0, "", Electrical"], ["NAME:Parameters", _  
  
"MenuProp:=", ["CoSimulator", "D", "", _  
  
"Default,Custom,Netlist", 0], "ButtonProp:=", ["CosimDefinition", _  
  
"D", "", "", "Edit", 0, "ButtonPropClientData:=", []], _  
  
["NAME:CosimDefinitions", ["NAME:CosimDefinition", _  
  
"CosimulatorType:=", 0, "CosimDefName:=", "HFSS3D", _
```



```

"IsDefinition:=", true, "CosimStackup:=", "Layout stackup", _

"CosimDmbedRatio:=", 3], ["NAME:CosimDefinition", _

"CosimulatorType:=", 1, "CosimDefName:=", "", _

"IsDefinition:=", true, "ExportAsNport:=", 0, _

"UsePjt:=", 0], ["NAME:CosimDefinition", _

"CosimulatorType:=", 2, "CosimDefName:=", "Custom", _

"IsDefinition:=", true, "DefinitionCompName:=", ""], _

["NAME:CosimDefinition", "CosimulatorType:=", 3, _

"CosimDefName:=", "Netlist", "IsDefinition:=", true, _

```

	<code>"NetlistString:=", ""]]])</code>

EditDataset

Modifies a dataset. This can be executed by the oProject, or oDesign variables.

UI Access	Project > Datasets > Edit.		
Parameters	Name	Type	Description
	<OriginalName>	String	Name of the original dataset.
	<DatasetDataArray>	Array	Data for the modified dataset.
Return Value	None.		

Python Syntax	EditDataset (<OriginalName> <DatasetDataArray>)
Python Example	<pre>oProject.EditDataset ("ds1" ["NAME:ds2", ["NAME:Coordinates", ["NAME:Coordinate", "X:=", 1, "Y:=", 2],]</pre>

```
        "NAME:Coordinate",
        "X:=", 3, "Y:=", 4
    ]
]
]
)
oDesign.EditDataset ("ds1"
["NAME:ds2",
    ["NAME:Coordinates",
        [
            "NAME:Coordinate",
            "X:=", 1, "Y:=", 2
        ],
        [
            "NAME:Coordinate",
            "X:=", 3, "Y:=", 4
        ]
    ]
]
]
)
```

VB Syntax	EditDataset <OriginalName> <DatasetDataArray>
VB Example	<pre>oProject.EditDataset "ds1" Array("NAME:ds2",_ Array("NAME:Coordinates",Array("NAME:Coordinate", "X:=", 1, "Y:=", 2), Array("NAME:Coordinate", "X:=", 3, "Y:=", 4))) oDesign.EditDataset "ds1" Array("NAME:ds2",_ Array("NAME:Coordinates",Array("NAME:Coordinate", "X:=", 1, "Y:=", 2), Array("NAME:Coordinate", "X:=", 3, "Y:=", 4)))</pre>

EditMaterial

Modifies an existing material.

UI Access	View/Edit Materials command in the material editor		
Parameters	Name	Type	Description
	<OriginalName>	String	Name of the material before editing.
	<MatProperties>	Array	Structured array containing material properties:
			["NAME:<New material name>", "CoordinateSystemType:=", <string>, "BulkOrSurfaceType:=" , <integer>,]

			<pre>["NAME:PhysicsTypes", "set:=" , <array containing string physics types> , <Optional ModifierDataArray>, "permeability:=" , <string containing value>, "conductivity:=" , <string containing value>, "thermal_conductivity:=" , <string containing value>, "mass_density:=" , <string containing value>, "specific_heat:=" , <string containing value>, "youngs_modulus:=" , <string containing value>, "poissons_ratio:=" , <string containing value>, "thermal_expansion_coefficient:=" , <string containing value>]</pre>
	<ModifierDataArray>	Array	Optional structured array containing thermal or spatial modifiers: <pre>["NAME:ModifierData", ["NAME:<ThermalModifierData or SpatialModifierData>", "modifier_data:=" , <"thermal_modifier_data" or "spa-</pre>

			<pre>tial_modifier_data">, ["NAME:<all_thermal_modifiers or all_spatial_mod- ifiers>", ["NAME:<modifierName>", "Property::=" , <string property being mod- ified>, "Index::=" , <integer>, "prop_modifier::=" , <"thermal_modifier" or "spa- tial_modifier">, "use_free_form::=" , <Boolean>, "free_form_value::=" , <string modifier value>,]]]]</pre>
Return Value	None.		

Python Syntax	EditMaterial (<OriginalName>, <MatProperties>)
Python	Without Modifiers:

Example

```
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_coefficient:=" , "7.2e-006"
]
)
```

With Thermal Modifier:

```
oDefinitionManager.EditMaterial("copper",  
    [  
        "NAME:copper",  
        "CoordinateSystemType:=", "Cartesian",  
        "BulkOrSurfaceType:=" , 1,  
        [  
            "NAME:PhysicsTypes",  
            "set:=" , ["Electromagnetic","Thermal","Structural"]  
        ],  
        [  
            "NAME:ModifierData",  
            [  
                "NAME:ThermalModifierData",  
                "modifier_data:=" , "thermal_modifier_data",  
                [  
                    "NAME:all_thermal_modifiers",  
                    [  
                        "NAME:one_thermal_modifier",  
                        "Property:=" , "permittivity",  
                        "Index:=" , 0,  
                        "prop_modifier:=" , "thermal_modifier",
```



```
        "use_free_form:=" , True,  
        "free_form_value:=" , "if(Temp > 1000cel, 1, if(Temp < -273.15cel, 1, 1))"  
    ]  
]  
]  
],  
"permeability:=" , "0.999991",  
"conductivity:=" , "58000000",  
"thermal_conductivity:=" , "400",  
"mass_density:=" , "8933",  
"specific_heat:=" , "385",  
"youngs_modulus:=" , "120000000000",  
"poissons_ratio:=" , "0.38",  
"thermal_expansion_coefficient:=" , "1.77e-05"  
])
```

Transient Solve, Non-linear Drude Data Plasma

```
import ScriptEnv  
  
ScriptEnv.Initialize("Ansoft.ElectronicsDesktop")  
oDesktop.RestoreWindow()  
  
oProject = oDesktop.SetActiveProject("Drude_plasma_parameters_r231")
```

```

oDefinitionManager = oProject.GetDefinitionManager()
oDefinitionManager.EditMaterial("Drude",
    [
        "NAME:Drude",
        "CoordinateSystemType:=", "Cartesian",
        "BulkOrSurfaceType:="    , 1,
        [
            "NAME:PhysicsTypes",
            "set:="                , ["Electromagnetic"]
        ],
        [
            "NAME:AttachedData",
            [
                "NAME:MatNonLinearDrudeFreqDepData",
                "property_data:="    , "nonlinear_drude_data",
                "EpsilonInfinity:="  , "1",
                "PlasmaFrequency:="  , "4.62348462366278GHz",
                "CollisionFrequency:=" , "0.00054491190162662GHz",
                "FieldBreakdown:="    , "10000V_per_meter",
                "PlasmaMaintainFrequency:=", "2.31174231183139GHz",
                "NeutralDensity:="    , 2.65164580488373E+20,
            ]
        ]
    ]
)

```

	<pre>"ElectronDensity:=" , 2.65164580488373E+17, "CollisionRateConstant:=", 2.05499505485618E-15]]])</pre>

VB Syntax	EditMaterial <OriginalName>, <MatProperties>
VB Example	Without Modifiers: <pre>oDefinitionManager.EditMaterial "alumina_92pct", Array("NAME:alumina_92pct", "CoordinateSystemType:=", "Cartesian", "BulkOrSurfaceType:=", 1, Array("NAME:PhysicsTypes", "set:=", Array("Electromagnetic","Thermal","Structural")), "permittivity:=", "9.3", "dielectric_loss_tangent:=", "0.008", "thermal_conductivity:=", "26", "mass_density:=", "3720",</pre>

```
"specific_heat:=", "790",
"youngs_modulus:=", "267000000000",
"poissons_ratio:=", "0.26",
"thermal_expansion_coefficient:=", "7.2e-006"
)
```

With Thermal Modifier:

```
oDefinitionManager.EditMaterial "copper",
  Array("NAME:copper",
    "CoordinateSystemType:=", "Cartesian",
    "BulkOrSurfaceType:=" , 1,
    Array("NAME:PhysicsTypes",
      "set:=" , Array("Electromagnetic","Thermal","Structural")),
    Array("NAME:ModifierData",
      Array("NAME:ThermalModifierData",
        "modifier_data:=" , "thermal_modifier_data",
        Array("NAME:all_thermal_modifiers",
          Array("NAME:one_thermal_modifier",
            "Property:=" , "permittivity",
            "Index:=" , 0,
            "prop_modifier:=" , "thermal_modifier",
            "use_free_form:=" , True,
```

	<pre> "free_form_value:=" , "if(Temp > 1000cel, 1, if(Temp < -273.15cel, 1, 1))")))), "permeability:=" , "0.999991", "conductivity:=" , "58000000", "thermal_conductivity:=" , "400", "mass_density:=" , "8933", "specific_heat:=" , "385", "youngs_modulus:=" , "120000000000", "poissons_ratio:=" , "0.38", "thermal_expansion_coefficient:=" , "1.77e-05") </pre>

Edit [padstack manager]

Use: Edit an existing padstack.

Command: Tools > Edit Configured Libraries > Padstacks > Edit Padstack

Syntax: Edit <PadstackName>,

```

  Array("NAME:<NewPadstackName>",
  "ModTime:=", <ModifiedOnInfo>,

```

```
"Library:=", "", // name of the library
"LibLocation:=", "Project", // location of the named library
Array("NAME:psd",
"nam:=", "<PadstackName>",
"lib:=", "", // name of the library
"mat:=", "", // hole plating material
"plt:=", "0", // percent of hole's radius filled by plating
Array("NAME:pds",
<LayerGeometryArray>,
<LayerGeometryArray....>,
"hle:=", <PadInfo>
"hRg:=", <HoleRange>,
"sbsh:=", <SolderballShape>,
"sbpl:=", <SolderballPlacement>,
"sbr:=", <string>, // solderball diameter, real with units
"sb2:=", <string>, // solderball mid diameter, real with units
"sbn:=", <string>), // name of solderball material
"ppl:=", <PadPortLayerArray>)
```

Return Value: <string> // [composite name](#) of the padstack

// If the name requested conflicts with the name of an existing

// padstack, the requested name is altered to be unique.

// The name returned reflects any change made to be unique.

Parameters: <PadstackName>:

<string> // [composite name](#) of padstack to edit

<NewPadstackName>:

<string> // new [simple name](#) for padstack

<ModifiedOnInfo>:

An integer that corresponds to the number of seconds that have elapsed since 00:00 hours, Jan 1, 1970 UTC from the system clock.

<LayerGeometryArray>:

Array("Name:lgm",

"lay:=", <string>, // definition layer name

"id:=", <int>, // definition layer id

"pad:=", <PadInfo>, // pad

"ant:=", <PadInfo>, // antipad

"thm:=", <PadInfo>, // thermal pad

"X:=", <string>, // pad x connection, real with units

"Y:=", <string>, // pad y connection, real with units
"dir:=", <DirectionString>) // pad connection direction

<PadInfo>:

Array("shp:=", <PadShape>,
"Szs:=", <DimensionArray>,
"X:=", <string>, // x offset, real with units
"Y:=", <string>, // y offset, real with units
"R:=", <string>) // rotation, real with units

<PadShape>:

<string> one of these choices

"No" // no pad

"Cir" // Circle

"Sq" // Square

"Rct" // Rectangle

"Ov" // Oval

"Blt" // Bullet

"Ply" // Polygons

"R45" // Round 45 thermal

"R90" // Round 90 thermal

"S45" // Square 45 thermal

"S90" // Square 90 thermal

<DimensionArray>:

Array(<string>, ...) // each string is a real with units for one of the
// dimensions of the shape

<DirectionString>:

<string> one of these choices

"No" // no direction

"Any" // any direction

"0" // 0 degrees

"45" // 45 degrees

"90" // 90 degrees

"135" // 135 degrees

"180" // 180 degrees

"225" // 225 degrees

"270" // 270 degrees

"315" // 315 degrees

<HoleRange>:

<string> one of these choices

"Thr" // through all layout layers

"Beg" // from upper pad layer to lowest layout layer

"End" // from upper layout layer to lowest pad layer

"UTL" // from upper pad layer to lowest pad layer

<SolderballShape>:

<string> one of these choices

"None" // no solderball

"Cyl" // cylinder solderball

"Sph" // spheroid solderball

<SolderballPlacement>:

<string> one of these choices

"abv" // above padstack

"blw" // below padstack

<PadPortLayerArray>:

Array(<int>, <int>,....) where each int is a layer id

VB Example:

```
oPadstackManager.Edit "Circle - through1", Array("NAME:Circle - through1", "ModTime:=", _
```

```
1235765635, "Library:=", "", "LibLocation:=", "Project", Array("NAME:psd", "nam:=", _
"Circle - through1", "lib:=", "", "mat:=", "", "plt:=", "0", Array("NAME:pds", Array("NAME:lgm",
"lay:=", _
"Top Signal", "id:=", 0, "pad:=", Array("shp:=", "Cir", "Szs:=", Array("2.5mm"), "X:=", _
"0mm", "Y:=", "0mm", "R:=", "0deg"), "ant:=", Array("shp:=", "Cir", "Szs:=", Array( _
"3.5mm"), "X:=", "0mm", "Y:=", "0mm", "R:=", "0"), "thm:=", Array("shp:=", "No", "Szs:=", Array
(), "X:=", _
"0mm", "Y:=", "0mm", "R:=", "0"), "X:=", "0mm", "Y:=", "0mm", "dir:=", "Any"), Array("NAME:lgm",
"lay:=", _
"SignalA", "id:=", 1, "pad:=", Array("shp:=", "Cir", "Szs:=", Array("2mm"), "X:=", _
"0mm", "Y:=", "0mm", "R:=", "0deg"), "ant:=", Array("shp:=", "Cir", "Szs:=", Array( _
"3mm"), "X:=", "0mm", "Y:=", "0mm", "R:=", "0"), "thm:=", Array("shp:=", "No", "Szs:=", Array(),
"X:=", _
"0mm", "Y:=", "0mm", "R:=", "0"), "X:=", "0mm", "Y:=", "0mm", "dir:=", "Any"), Array("NAME:lgm",
"lay:=", _
"SignalB", "id:=", 2, "pad:=", Array("shp:=", "Cir", "Szs:=", Array("2mm"), "X:=", _
"0mm", "Y:=", "0mm", "R:=", "0deg"), "ant:=", Array("shp:=", "Cir", "Szs:=", Array( _
"3mm"), "X:=", "0mm", "Y:=", "0mm", "R:=", "0deg"), "thm:=", Array("shp:=", "No", "Szs:=", Array
(), "X:=", _
"0mm", "Y:=", "0mm", "R:=", "0"), "X:=", "0mm", "Y:=", "0mm", "dir:=", "Any"), Array("NAME:lgm",
"lay:=", _
"Ground", "id:=", 3, "pad:=", Array("shp:=", "No", "Szs:=", Array(), "X:=", _
"0mm", "Y:=", "0mm", "R:=", "0deg"), "ant:=", Array("shp:=", "No", "Szs:=", Array(), "X:=", _
```

```
"0mm", "Y:=", "0mm", "R:=", "0"), "thm:=", Array("shp:=", "R90", "Szs:=", Array( _
"3mm", "0.75mm", "1mm"), "X:=", "0mm", "Y:=", "0mm", "R:=", "0"), "X:=", "0mm", "Y:=", _
"0mm", "dir:=", "Any"), Array("NAME:lgm", "lay:=", "Bottom signal", "id:=", 5, "pad:=", Array
("shp:=", _
"Cir", "Szs:=", Array("1mm"), "X:=", "0mm", "Y:=", "0mm", "R:=", "0deg"), "ant:=", Array
("shp:=", _
"Cir", "Szs:=", Array("2mm"), "X:=", "0mm", "Y:=", "0mm", "R:=", "0deg"), "thm:=", Array
("shp:=", _
"No", "Szs:=", Array(), "X:=", "0mm", "Y:=", "0mm", "R:=", "0"), "X:=", "0mm", "Y:=", _
"0mm", "dir:=", "Any")), "hle:=", Array("shp:=", "Cir", "Szs:=", Array("1.5mm"), "X:=", _
"0mm", "Y:=", "0mm", "R:=", "0deg"), "hRg:=", "End", "sbsh:=", "Sph", "sbpl:=", _
"abv", "sbr:=", "750um", "sb2:=", "1200um", "1200um", "sbn:=", "solder"), "ppl:=", Array( _
0, 1, 2, 3, 5))
```

ExportDataset

Exports a dataset to a named file. This can be executed by the oProject, or oDesign variables.

UI Access	Project > Datasets > Export.		
Parameters	Name	Type	Description
	<datasetFilePath>	String	The full path to the file.
Return Value	None.		

Python Syntax	ExportDataset (<datasetFilePath>)

Python Example	<pre>oProject.ExportDataset('e:/tmp/dsdata.txt') oDesign.ExportDataset('e:/tmp/dsdata.txt')</pre>

VB Syntax	ExportDataset <datasetFilePath>
VB Example	<pre>oProject.ExportDataset "e:/tmp/dsdata.txt" oDesign.ExportDataset "e:/tmp/dsdata.txt"</pre>

Export [footprint manager]

Use: Export a footprint to a library

Command: Tools > Edit Configured Libraries > Footprints > Export to Library

Syntax: Export Array("NAME:<LibraryName>",

 <FootprintName>,

 <FootprintName>...),

 <LibraryLocation>

Return Value: None

Parameters: <LibraryName>:

 <string> // name of the library

 <FootprintName>:

 <string> // [composite name](#) of footprint to export

```
<LibraryLocation>:  
<string> // location of the library in <LibraryName>  
// One of "Project", "PersonalLib", or "UserLib"
```

```
VB Example:  
oFootprintManager.Export Array("NAME:mylib", "Distributed Footprints:BPAD"), "PersonalLib"
```

ExportMaterial

Exports a local material to a library.

UI Access	Export to Library command in the material editor.		
Parameters	Name	Type	Description
	<ExportData>	Array	["NAME:<LibraryName>", <MaterialName>, <MaterialName>, ...]
	<LibraryName>	String	Name of the exported library.
	<MaterialName>	String	Name of the material to be exported.
	<LibraryLocation>	String	Location to save the library. Only "PersonalLib" and "UserLib" are allowed.
Return Value	None.		

Python Syntax	ExportMaterial (<ExportData>, <LibraryLocation>)
Python Example	oDefinitionManager.ExportMaterial (["NAME:mylib", _

	"Material1", "Material2", "Material3"], "PersonalLib")

VB Syntax	ExportMaterial <ExportData>, <LibraryLocation>
VB Example	oDefinitionManager.ExportMaterial Array("NAME:mylib", _ "Material1", "Material2", "Material3"), "PersonalLib"

Export [padstack manager]

Use: Export a padstack to a library

Command: Tools > Edit Configured Libraries > Padstacks > Export to Library

Syntax: Export Array("NAME:<LibraryName>",

<PadstackName>,

<PadstackName>...),

<LibraryLocation>

Return Value: None

Parameters: <LibraryName>:

<string> // name of the library

<PadstackName>:

<string> // [simple name](#) of padstack to export

```
<LibraryLocation>:  
<string> // location of the library in <LibraryName>  
// One of "Project", "PersonalLib", or "UserLib"
```

VB Example:

```
oPadstackManager.Export Array("NAME:mylib", "myPadstack"), "PersonalLib"
```

ExportScript

Use: Export to Library in the script definition manager

Command: None

Syntax: ExportScript <ExportData>,<Library location>

Return Value: None

Parameters: <ExportData>

```
Array("NAME:<LibraryName>",<ScriptName>,<ScriptName>,...)
```

VB Example:

```
oProject.ExportComponent Array("NAME:mylib", "myscript"), "PersonalLib"
```

Python Syntax	ExportScript(<ExportData>,<Library location>)
Python Example	<pre>oProject.ExportComponent (["NAME:mylib", "myscript"], "PersonalLib")</pre>

Export [symbol manager]

Use: Exports symbol(s) to a library

Command: Tools > Edit Configured Libraries > Symbols > Export to Library

Syntax: Export Array("NAME:<LibraryName>",

 <SymbolName>,

 <SymbolName>...),

 <LibraryLocation>

Return Value: None

Parameters: <LibraryName>:

 <string> // name of the library

 <SymbolName>:

 <string> // [composite name](#) of symbol to export

 <LibraryLocation>:

 <string> // location of the library in <LibraryName>

 // One of "Project", "PersonalLib", or "UserLib"

VB Example:

```
oSymbolManager.Export _  
Array("NAME Nexxim Circuit Elements\Distributed\Distributed:bendo:mylib", _ "mySymbol"), "Per-  
sonalLib"
```

GetProjextMaterialNames

Returns the material names belonging to an active Project.

UI Access	N/A		
Parameters	Name	Type	Description
	None		
Return Value	String names of the materials in the active project.		

Python Syntax	GetProjectMaterialNames()
Python Example	<pre>oProject = oDesktop.GetActiveProject() oDefinitionManager = oProject.GetDefinitionManager() materials = oDefinitionManager.GetProjectMaterialNames() AddWarningMessage(str(materials))</pre>

GetPropertyValue

Returns the value of a single property belonging to a specific *<PropServer>* and *<PropTab>*. This function is available with the Project, Design or Editor objects, including definition editors.

Tip: Use the script recording feature and edit a property, and then view the resulting script to see the format for that property.

UI Access	N/A		
Parameters	Name	Type	Description
	<code><PropTab></code>	String	One of the following, where tab titles are shown in parentheses: • PassedParameterTab ("Parameter Values") • DefinitionParameterTab (Parameter Defaults") • LocalVariableTab ("Variables" or "Local Variables") • ProjectVariableTab ("Project variables") • ConstantsTab ("Constants") • BaseElementTab ("Symbol" or "Footprint") • ComponentTab ("General") • Component("Component") • CustomTab ("Intrinsic Variables") • Quantities ("Quantities") • Signals ("Signals")
	<code><PropServer></code>	String	An object identifier, generally returned from another script method, such as <code>CompInst@R;2;3</code>
	<code><PropName></code>	String	Name of the property.
Return Value	String value of the property.		

Python Syntax	<code>GetPropertyValue (<PropTab>, <PropServer>, <PropName>)</code>
Python Example	<pre>selectionArray = oEditor.GetSelections() for k in selectionArray:</pre>

	<pre>val = oEditor.GetPropertyValue("PassedParameterTab", k, "R") ...</pre>

VB Syntax	<code>GetPropertyValue (<PropTab>, <PropServer>, <PropName>)</code>
VB Example	<pre>selectionArray = oEditor.GetSelections for k in selectionArray: val = oEditor.GetPropertyValue("PassedParameterTab", k, "R") ...</pre>

ImportDataset

Imports a dataset from a named file. This can be executed by the oProject, or oDesign variables. The name of the dataset is file-name+index number (e.g., dsdata1) unless the filename ends with a trailing number. When there is a trailing number at the end, we will remove the number and use first unused index. Alternatively, the name of the dataset can be explicitly defined by providing a string as an optional second argument.

UI Access	Project > Datasets > Import.		
Parameters	Name	Type	Description
	<code><datasetFileFullPath></code>	String	The full path to the file containing the dataset values. *.tab files recommended (see note below).
	<code><optionalDatasetName></code>	String	<i>Optional.</i> User-defined dataset name.
Return Value	None.		

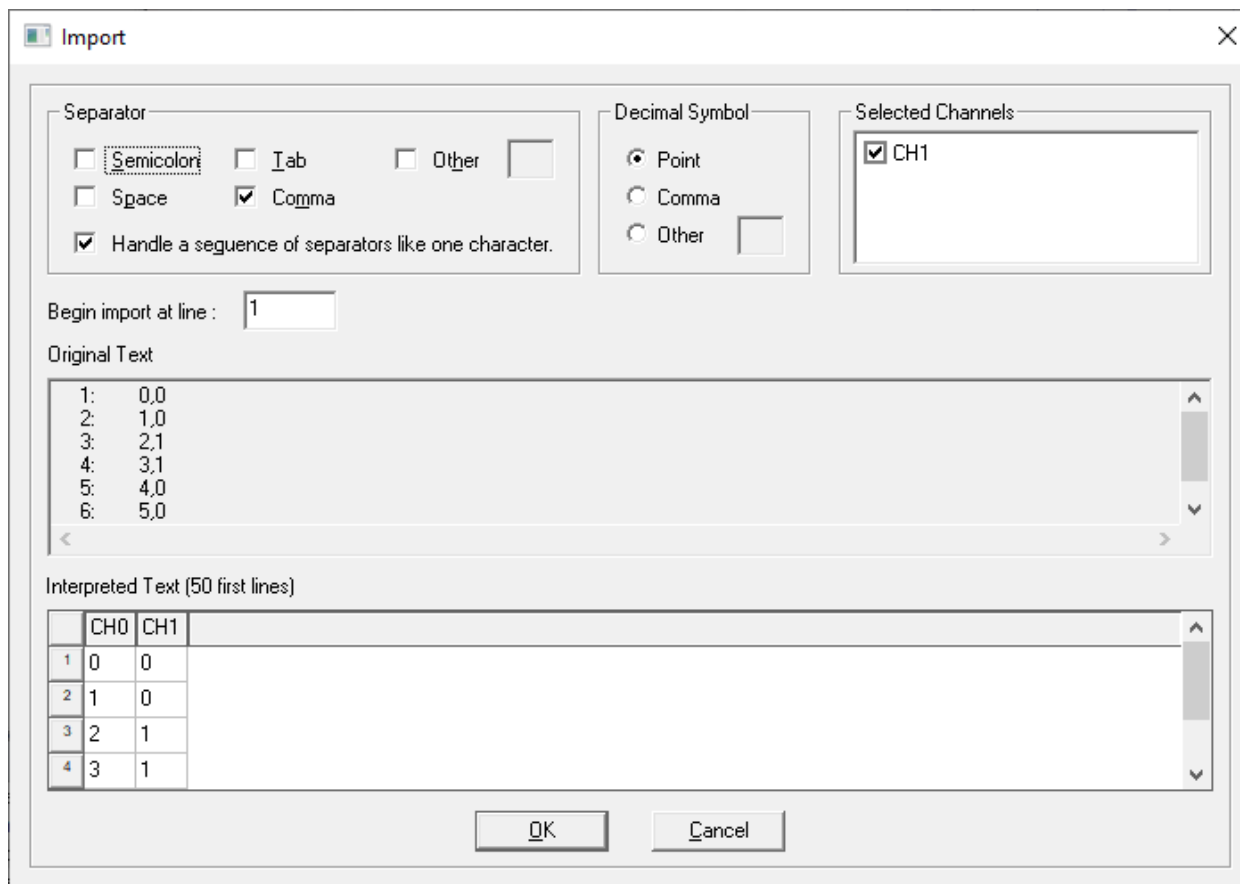
Python Syntax	ImportDataset (<datasetFilePath>, <optionalDatasetName>)
Python Example	<pre>oProject.ImportDataset('e:\tmp\dsdata.tab') oDesign.ImportDataset('e:\tmp\dsdata.tab') oProject.ImportDataset('e:\tmp\dsdata.tab', 'MyDatasetName') oDesign.ImportDataset('e:\tmp\dsdata.tab', 'MyDatasetName')</pre>

VB Syntax	ImportDataset <datasetFilePath>, <optionalDatasetName>
VB Example	<pre>oProject.ImportDataset "e:\tmp\dsdata.tab" oDesign.ImportDataset "e:\tmp\dsdata.tab" oProject.ImportDataset "e:\tmp\dsdata.tab", "MyDatasetName" oDesign.ImportDataset "e:\tmp\dsdata.tab", "MyDatasetName"</pre>

Note About File Types:

Tab-delimited or space-delimited files with the extension *.tab are the recommended file type. When using ImportDataset at the Design level, *.tab is the only file type supported.

At the Project level, other file types are supported (for example, *.csv). However, after calling the command, you must configure the file import format manually through the Electronics Desktop GUI by selecting **Project > Datasets** and clicking **Import**.



Remove [component manager]

Remove a component from a library

Command: Tools > Edit Configured Libraries > Components > Remove Component

Syntax: Remove <ComponentName>,

 <IsProjectComponent>,

 <LibraryName>,

 <LibraryLocation>

Return Value: None

Parameters: <ComponentName>:

 <string> // [composite name](#) of the component to remove

 <IsProjectComponent>:

 <bool>

 <LibraryName>:

 <string> // name of the library

 <LibraryLocation>:

 <string> // location of the library in <LibraryName>

 // One of "Project", "PersonalLib", or "UserLib"

VB Example:

```
oComponentManager.Remove "Nexxim Circuit Elements\BJTs:Level01_NPN", _ true, "Project"
```

Python Syntax	Remove (<ComponentName>, <IsProjectComponent>, <LibraryName>, <LibraryLocation>)
Python Example	<pre>oComponentManager.Remove ("Simplorer Circuit Elements\BJTs:Level01_NPN", _ true, "Project")</pre>

Remove [footprint manager]

Use: Removes a footprint from a library

Command: Tools > Edit Configured Libraries > Footprints > Remove Footprint

Syntax: Remove <FootprintName>,

 <IsProjectFootprint>,

 <LibraryName>,

 <LibraryLocation>

Return Value: None

Parameters: <FootprintName>:

 <string> // [composite name](#) of the footprint to remove

 <IsProjectFootprint>:

 <bool>

 <LibraryName>:

 <string> // name of the library

 <LibraryLocation>:


```
<string> // location of the library in <LibraryName>  
// One of "Project", "PersonalLib", or "UserLib"
```

VB Example:

```
oFootprintManager.Remove "BPAD", true, "Distributed Footprints", "Project"  
oFootprintManager.Remove "BPAD", false, "MyLib", "PersonalLib"
```

RemoveMaterial

Removes a material from a library.

UI Access	Remove Material(s) command in the material editor		
Parameters	Name	Type	Description
	<MaterialName>	String	Name of the material to be removed.
	<IsProjectMaterial>	Boolean	If True, assumes the material is a project material. The last two parameters will be ignored. If False, the material is not a project material.
	<LibraryName>	String	Name of the user or personal library where the material resides.
	<LibraryLocation>	String	Location of library. Valid options:"UserLib" or "PersonalLib".
Return Value	None.		

Python Syntax	RemoveMaterial (<MaterialName>, <IsProjectMaterial>, <LibraryName>, <LibraryLocation>)

Python Example	<pre>oDefinitionManager.RemoveMaterial (["Material1", false, "mo0907", "UserLib"])</pre>

VB Syntax	<pre>RemoveMaterial <MaterialName>, <IsProjectMaterial>, <LibraryName>, <LibraryLocation></pre>
VB Example	<pre>oDefinitionManager.RemoveMaterial "Material1", false, "mo0907", "UserLib"</pre>

Remove [padstack manager]

Use: Removes a padstack from a library

Command: Tools > Edit Configured Libraries > Padstacks > Remove Padstacks

Syntax: Remove <PadstackName>,

 <IsProjectPadstack>,

 <LibraryName>,

 <LibraryLocation>

Return Value: None

Parameters: <PadstackName>:

 <string> // [simple name](#) of the padstack to remove

 <IsProjectPadstack>:

 <bool>

<LibraryName>:

<string> // name of the library

<LibraryLocation>:

<string> // location of the library in <LibraryName>

// One of "Project", "PersonalLib", or "UserLib"

VB Example:

```
oPadstackManager.Remove "Polygon SMT", true, "Padstacks", "Project"
```

```
oPadstackManager.Remove "Polygon SMT", false, "MyLib", "PersonalLib"
```

RemoveScript

Use: Remove Script in the script definition manager

Command: None

Syntax: RemoveScript <ScriptName>,<IsProjectScript>, <LibraryName>,<LibraryLocation>

Return Value: None

Parameters: <ScriptName>

Type: <string>

<IsProjectScript>

Type: <bool>

<LibraryName>

Type: <string>

<LibraryLocation>

Type: <string>

VB Example:

```
oDefinitionManager.RemoveScript "myscript", true, "Local", "Project"
```

Python Syntax	RemoveScript (<ScriptName>,<IsProjectScript>, <LibraryName>,<LibraryLocation>)
Python Example	<pre>oDefinitionManager.RemoveScript("myscript", true, "Local", "Project")</pre>

Remove [symbol manager]

Use: Removes a symbol from a library

Command: Tools > Edit Configured Libraries > Symbols > Remove Symbol

Syntax: Remove <SymbolName>,

 <IsProjectSymbol>,

 <LibraryName>,

 <LibraryLocation>

Return Value: None

Parameters: <SymbolName>:

 <string> // [composite name](#) of the symbol to remove

<IsProjectSymbol>:

<bool>

<LibraryName>:

<string> // name of the library

<LibraryLocation>:

<string> // location of the library in <LibraryName>

// One of "Project", "PersonalLib", or "UserLib"

VB Example:

```
oSymbolManager.Remove "Nexxim Circuit Elements\Distributed\Distributed:bendo", true, "Project"
```

RemoveUnusedDefinitions

Removes any unused project definitions.

UI Access	Tools > Project Tools > Remove Unused Definitions.		
Parameters	Name	Type	Description
	<Definitions>	Array	Definitions to be removed, such as materials and surface materials.
Return Value	None.		

Python Syntax	RemoveUnusedDefinitions(<Definitions>)
Python Example	<pre>oProject.RemoveUnusedDefinitions([["NAME:Materials", "Al-Extruded"], ["NAME:SurfaceMaterials", "Steel-oxidised-surface"]])</pre>

VB Syntax	RemoveUnusedDefinitions <Definitions>
VB Example	<pre>oProject.RemoveUnusedDefinitions Array(Array("NAME:Materials", "Al-Extruded"), Array ("NAME:SurfaceMaterials", "Steel-oxidised-surface"))</pre>

SetPropertyValue

Sets the value of a single property belonging to a specific PropServer and PropTab. This function is available with the Project, Design or Editor objects, including definition editors. This is not supported for properties of the following types: ButtonProp, PointProp, V3DPointProp, and VPointProp. Only the ChangeProperty command can be used to modify these properties.

Use the script recording feature and edit a property, and then view the resulting script entry or use GetPropertyValue for the desired property to see the expected format.

UI Access	N/A		
Parameters	Name	Type	Description
	<propTab>	String	One of the following, where tab titles are shown in parentheses: - PassedParameterTab ("Parameter Values") - DefinitionParameterTab (Parameter Defaults") - LocalVariableTab ("Variables" or "Local Variables") - ProjectVariableTab ("Project variables") - ConstantsTab ("Constants") - BaseElementTab ("Symbol" or "Footprint") - ComponentTab ("General") - Component("Component") - CustomTab ("Intrinsic Variables") - Quantities ("Quantities") - Signals ("Signals")
	<propServer>	String	An object identifier, generally returned from another script method, such as CompInst@R;2;3
	<propName>	String	Name of the property.
	<propValue>	String	The value for the property

Return Value	None.

Python Syntax	SetPropertyValue(<propTab>, <propServer>, <propName>, <propValue>)
Python Example	oEditor.SetPropertyValue("PassedParameterTab", "k", "R", "2200")

VB Syntax	SetPropertyValue <propTab>, <propServer>, <propName>, <propValue>
VB Example	oEditor.SetPropertyValue "PassedParameterTab", "k", "R", "2200"

UpdateDefFromBlock

Updates a material definition from block text (same definition format as would be contained in the material library file) by library type (using definition folder name). This scripting command directly supports the .AMAT (or .ASURF) definition formats.

UI Access	N/A		
Parameters	Name	Type	Description
	<targetDefName>	String	Name of the target definition, i.e. the name of the material to update.
	<defBlock>	String	Text of the new material definition in block format (string); this block could use a new definition name, which will cause a rename as part of the update.
	<defFolderName>	String	Library type (by definition, folder name).
	<newTimeStamp>	String	New timestamp string (time_t as integer, number of seconds since 1/1/1970 12:00am), default is current time.
Return Value	A property scripting object for the definition.		

P- yt- h- o- n S- y- nt- ax	UpdateDefFromBlock(<targetDefName>, <defBlock>, <defFolderName>, <newTimeStamp>)
P- yt- h- o- n E- x- a- m- pl- e	<pre> oProject = oDesktop.NewProject() oProject.InsertDesign("HFSS", "HFSSDesign1", "DrivenModal", "") oDesign = oProject.SetActiveDesign("HFSSDesign1") oEditor = oDesign.SetActiveEditor("3D Modeler") oDefinitionManager = oProject.GetDefinitionManager() defBlock = "\$begin 'vacuum2' \$begin 'AttachedData' \$begin 'MatAppearanceData' property_data- ='appearance_data' Red=230 Green=230 Blue=230 Transparency=0.95 \$end 'MatAppearanceData' \$end 'AttachedData' simple('permittivity', 1) ModTime=1499970477 \$end 'vacuum2'" added = oDefinitionManager.AddDefinitionFromBlock(defBlock, "Materials", "10101010", True) addedName = '' if isinstance(added, basestring): addedName = added elif isinstance(added, list): addedName = added[0] </pre>

```
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

UI Access	N/A		
Parameters	Name	Type	Description
	<targetDefName>	String	Name of the target definition, i.e. the name of the material to update. Name of the target definition, i.e. the

		name of the material to update (string). Name of the target definition, i.e. the name of the material to update (string).		
	<desiredDefName>	String	New name to use for a rename. If this is the same as <i>targetDefName</i> , it means the definition is not being renamed. If this is blank, the block name will be used for this purpose instead.	New name to use for a rename. If this is the same as <i>targetDefName</i> , it means the definition is not being renamed. If this is blank, the block name will be used for this purpose instead.
	<defBlock>	String	Text of the updated material definition in block form. (Same form as the defBlock input to the AddDefinitionFromBlock and UpdateDefFromBlock commands, for example.) Note that, unlike in the UpdateDefFromBlock command, the name of the definition in the block can be ignored, provided that desiredDefName is set.	
	<defFolderName>	String	Library type (by definition, folder name).	
	<newTimeStamp>	String	New timestamp string (time_t as integer, number of seconds since 1/1/1970 12:00am), default is current	

			time.
	<code><continuelIfConflict></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.
Return Value	A property scripting object (extended definition object) for the definition. (In the event of a failure, the command throws an exception and there will be no result.)		

Python Syntax	<code>UpdateDefFromBlockEx(<targetDefName>, <desiredDefName>, <defBlock>, <defFolderName>, <newTimeStamp>, <continuelIfConflict>)</code>
Python Example	<pre># Note that the material block name will be ignored in this case defBlock = "\$begin 'unused_name' \$begin 'AttachedData' \$begin \ 'MatAppearanceData' property_data='appearance_data' \ Red=255 Green=0 Blue=0 Transparency=0.19 \$end \ 'MatAppearanceData' \$end 'AttachedData' \ simple('permittivity', 1.9) ModTime=1509090909 \ \$end 'unused_name'"</pre>

	<pre>try: updated = oDefinitionManager.UpdateDefFromBlockEx("red_material", "red_material", defBlock, "Materials", "1609090909") updatedName = updated.GetName().replace("Materials:", "") AddInfoMessage("Result: " + updatedName) except: AddErrorMessage("Unexpected error in UpdateDefFromBlockEx")</pre>

UpdateDefinitions

Updates all definitions. The **Messages** window reports when definitions are updated, or warns when definitions cannot be found.

UI Access	Tools > Project Tools > Update Definitions. Click Select All , then Update .
Parameters	None.
Return Value	None.

Python Syntax	UpdateDefinitions()
Python Example	oProject.UpdateDefinitions()

VB Syntax	UpdateDefinitions
VB Example	<code>oProject.UpdateDefinitions</code>

Component Manager Script Commands

The component manager provides access to components in a project. The manager object is accessed via the definition manager.

```
Set oDefinitionManager = oProject.GetDefinitionManager()
```

```
Set oComponentManager = oDefinitionManager.GetManager("Component")
```

The topics for this section include:

[Add](#)

[AddNPortData](#)

[AddSolverOnDemandModel](#)

[Edit](#)

[EditSolverOnDemandModel](#)

[EditWithComps](#)

[Export](#)

[GetNPortData](#)

[GetSolverOnDemandData](#)

[GetSolverOnDemandModelList](#)

[Remove](#)

[RemoveSolverOnDemandMode](#)

[UpdateDynamicLink](#)

Add [component manager]

*Use:*Add a component

Command: Tools > Edit Configured Libraries > Components > Add Component

*Syntax:*Add Array("NAME:<ComponentName>",
 "Info:=", <ComponentInfo>,
 "RefBase:=", <string>, // reference designator
 "NumParts:=", <int>, // parts per component
 "OriginalComponent:=", <string>
 "Terminal:=", <TerminalInfo>,
 "Terminal:=", <TerminalInfo>, ...
 // The remaining parameters are optional
 Array("NAME:Parameters", // any combo of the following
 "VariableProp:=", <VariableInfo>,
 "CheckboxProp:=", <CheckBoxInfo>,
 "ButtonProp:=", <ButtonInfo>,
 "TextProp:=", <TextInfo>,
 "NumberProp:=", <NumberInfo>,
 "SeparatorProp:=", <SeparatorInfo>,
 "ValueProp:=", <ValueInfo>,

```
"MenuProp:=", <MenuInfo>),  
Array("NAME:Properties", // any combo of the following  
"CheckboxProp:=", <CheckBoxInfo>,  
"TextProp:=", <TextInfo>,  
"NumberProp:=", <NumberInfo>,  
"SeparatorProp:=", <SeparatorInfo>,  
"ValueProp:=", <ValueInfo>,  
"MenuProp:=", <MenuInfo>),  
"VPointProp:=", <VPointInfo>,  
"PointProp:=", <PointInfo>),  
Array("Quantities",  
"QuantityProp:=", <QuantityPropInfo>...),  
Array("NAME:CosimDefinitions",  
  <CosimDefInfo>,  
  <CosimDefInfo>...)
```

Return Value:<string>

```
// composite name of the component.  
// If the name requested conflicts with the name of an existing  
// component, the requested name is altered to be unique.  
// The name returned reflects any change made to be unique.
```


Parameters:<ComponentName>:

<string> // [simple name](#) of the component

<ComponentInfo>:

Array("Type:=", <TypeInfo>,

"NumTerminals:=", <int>,

"DataSource:=", <string>,

"ModifiedOn:=", <ModifiedOnInfo>,

"Manufacturer:=", "<string>,"

"Symbol:=", <string>,"

"Footprint:=", <string>,"

"Description:=", <string>,"

"InfoTopic:=", <string>,"

"InfoHelpFile:=", <string>,"

"IconFile:=", <string>,"

"LibraryName:=", "",

"OriginalLocation:=", "Project", // Project Location

"Author:=", <string>,"

"OriginalAuthor:=", <string>,"

"CreationDate:=", <int>)

<TypeInfo>:

An integer that is the or-ing of bits for each product listed below. The default setting is 0xffffffff (4294967295) which indicates valid for all products. In the component editing dialog, checking different boxes in the "Specify products for which this component is valid" grid control sets specific flags that correspond to the following hex/decimal settings:

Nexxim -- 100 binary, 4 decimal, 0x4

SlwaveDeNovo -- 1000 binary, 8 decimal, 0x8

Simplorer -- 10000 binary, 16 decimal, 0x10

MaxwellCircuit -- 100000 binary, 32 decimal, 0x20

<ModifiedOnInfo>:

An integer that corresponds to the number of seconds that have elapsed since 00:00 hours, Jan 1, 1970 UTC from the system clock.

<TerminalInfo>:

Array(<string>, // symbol pin

<string> // footprint pin

<string >, // gate name

<bool>, // shared

<int>, // equivalence number

<int>, // what to do if unconnected: flag as error:0, ignore:1

<string> // description

<Nature>)

<Nature>:

<string> // content varies as follows

Nexxim/Circuit:

"Electrical" // the only choice

Simplorer:

// several choices

"Electrical", "Magnetic", "Fluidic", "Translational",

"Translational_V", "Rotational", "Rotational_V",

""Radiant", "Thermal", or <VHDLPackageName>

<VHDLPackageName>:

<string> // in the form <Library>.<Package>

<Library>:

<string> // name of the VHDL library

<Package>:

<string> // name of the VHDL package

<VariableInfo>:

Array(<string>, // name

<FlagLetters>,

<string>, // description

"CB:=", <string>, // optional - script for call back

<string>) // value: number, variable, or expression

<FlagLetters>:

<string> // "D" - has description parameter,

// "RD" - readonly & has description parameter,

// or "RHD" - readonly, hidden, & has description parameter

<CheckBoxInfo>:

Array(<string>, // name

<FlagLetters>,

<string>, // description

"CB:=", <string>, // optional - script for call back

<bool>) // value: true or false

<ButtonInfo>:

Array(<string>, // name

<FlagLetters>,

<string>, // description

<string>, // button title

<string>, // extra text

<ClientID>,

"ButtonPropClientData:= ", <ClientDataArray>)

<ClientID>:

<int> // specifies Button Prop Client

// 0 - unknown, ButtonPropClientData

// array will be empty

// 1 - Netlist Prop Client

// 2 - not used

// 3 - File Name Prop Client

<ClientDataArray>:

varies with <ClientID>

<ClientId> is 0 or 1: empty array

Array()

<ClientID> is 3:

Array("InternalFormatText:=", "<prefix><RelativePath>")

<prefix>:

<string> // "<Project>", "<PersonalLib>", "<UserLib>", or "<SysLib>"

<RelativePath>:

<string> // relative path to file from <prefix>

<TextInfo>:

Array(<string>, // name

<FlagLetters>,

<string>, // description

"CB:=", <string>, // optional - script for call back

<string>) // value: a text string

<NumberInfo>:

Array(<string>, // name

```
<FlagLetters>,  
<string>, // description  
"CB:=", <string>, // optional - script for call back  
<real>, // value: a number  
<string>) // units
```

```
<SeparatorInfo>:  
Array(<string>, // name  
<FlagLetters>,  
<string>, // description  
"CB:=", <string>, // optional - script for call back  
<string>) // value: a text string
```

```
<ValueInfo>:  
Array(<string>, // name  
<FlagLetters>,  
<string>, // description  
"CB:=", <string>, // optional - script for call back  
<string>) // value: a number, variable or expression
```

```
<MenuPropInfo>:
```

```
Array(<string>, // name
<FlagLetters>,
<string>, // description
<string>, // menu choices - separated by commas
<int>) // 0 based index of current menu choice
```

```
<VPointInfo>:
```

```
Array(<string>, // name
<FlagLetters>,
<string>, // description
"CB:=", <string>, // optional - script for call back
<string>, // x value: number with length units
<string>) // y value: number with length units
```

```
<PointInfo>:
```

```
Array(<string>, // name
<FlagLetters>,
<string>, // description
"CB:=", <string>, // optional - script for call back
<real>, // x value
<real>) // y value
```


<QuantityPropInfo>:

```
Array(<string>, // name
<FlagLetters>,
<string>, // description
<string>, // value
<TypeString>,
<TypeStringDependentInfo>)
```

<TypeString>:

<string> // "Across", "Through", or "Free"

<TypeStringDependentInfo>:

<TypeString> is "Free" :

<string>, // direction: "In", "Out", "InOut", or "DontCare"

// Following <string> is not present if direction is "DontCare"

<string> // when to calculate: "BeforeAnalogSolver",

// "BeforeStateGraph", "AfterStateGraph", or "DontCareWhen"

<TypeString> is "Across" or "Through":

<int>, // terminal 1

<int> // terminal 2

<CosimDefInfo>:

Array("NAME:CosimDefinition",

"CosimulatorType:=", <int>,

"CosimDefName:=", <string> // "HFSS 3D Layout", "Circuit",

// "Custom", or "Netlist"

"IsDefinition:=", <bool>,

final array member(s) vary with CosimDefName)

final array members for HFSS 3D Layout:

"CosimStackup:=", <string>,

"CosimDmbedRatio:=", <int>

final array members for Circuit:

"ExportAsNport:=", <int>,

"UsePjt:=", <int>

final array member for Custom:

"DefinitionCompName:=", <string>

final array member for Netlist:

"NetlistString:=", <string>

VB Example:

```
Dim name
oComponentManager.Add (Array("NAME:MyComponent", _
"Info:=", Array("Type:=", 4294901767, _
"NumTerminals:=", 2, _
"DataSource:=", "", _
"ModifiedOn:=", 1071096503, _
"Manufacturer:=", "Ansys", _
"Symbol:=", "bendo", _
"Footprint:=", "BENDO", _
"Description:=", "", _
"InfoTopic:=", "", _
"InfoHelpFile:=", "", _
"IconFile:=", "", _
"LibraryName:=", "", _
"OriginalLocation:=", "Project", _
"Author:=", "", _
```

```
"OriginalAuthor:=", "", _
"CreationDate:= ", 1147460679), _
"Refbase:=", "U", _
"NumParts:=", 1, _
"OriginalComponent:=", "", _
"Terminal:=", Array("n1", _
"n1", _
"A", _
false, _
0, _
1, _
"", _
"Electrical"), _
"Terminal:=", Array("n2", _
"n2", _
"A", _
false, _
1, _
0, _
"", _
"Electrical"), _
```

```
Array("NAME:Parameters", _  
  "MenuProp:=", Array("CoSimulator", _  
    "D", _  
    "", _  
    "Default,HFSS 3D Layout,Circuit,Custom,Netlist", _  
  ), _  
  "ButtonProp:=", Array("CosimDefinition", _  
    "D", _  
    "", _  
    "", _  
    "Edit", _  
  ), _  
  "ButtonPropClientData:=", Array()), _  
  Array("NAME:CosimDefinitions", _  
    Array("NAME:CosimDefinition", _  
      "CosimulatorType:=", 0, _  
      "CosimDefName:=", "HFSS 3D Layout", _  
      "IsDefinition:=", true, _  
      "CosimStackup:=", "Layout stackup", _  
      "CosimDmbedRatio:=", 3), _
```

```
Array("NAME:CosimDefinition", _  
  "CosimulatorType:=", 1, _  
  "CosimDefName:=", "Circuit", _  
  "IsDefinition:=", true, _  
  "ExportAsNport:=", 0, _  
  "UsePjt:=", 0), _  
Array("NAME:CosimDefinition", _  
  "CosimulatorType:=", 2, _  
  "CosimDefName:=", "Custom", _  
  "IsDefinition:=", true, _  
  "DefinitionCompName:=", ""), _  
Array("NAME:CosimDefinition", _  
  "CosimulatorType:=", 3, _  
  "CosimDefName:=", "Netlist", _  
  "IsDefinition:=", true, _  
  "NetlistString:=", "")))
```

Python Syntax	Add [("NAME:<ComponentName>", "Info:=", <ComponentInfo>, "RefBase:=", <string>, // reference designator "NumParts:=", <int>, // parts per component

```

"OriginalComponent:=", <string>
"Terminal:=", <TerminalInfo>,
"Terminal:=", <TerminalInfo>, ...
The remaining parameters are optional.
["NAME:Parameters", // any combo of the following
"VariableProp:=", <VariableInfo>,
"CheckboxProp:=", <CheckBoxInfo>,
"ButtonProp:=", <ButtonInfo>,
"TextProp:=", <TextInfo>,
"NumberProp:=", <NumberInfo>,
"SeparatorProp:=", <SeparatorInfo>,
"ValueProp:=", <ValueInfo>,
"MenuProp:=", <MenuInfo>],
["NAME:Properties", Any combination of the following:
"CheckboxProp:=", <CheckBoxInfo>,
"TextProp:=", <TextInfo>,
"NumberProp:=", <NumberInfo>,
"SeparatorProp:=", <SeparatorInfo>,
"ValueProp:=", <ValueInfo>,
"MenuProp:=", <MenuInfo>,

```

	<pre>"VPointProp:=", <VPointInfo>, "PointProp:=", <PointInfo>], ["Quantities", "QuantityProp:=", <QuantityPropInfo>...], ["NAME:CosimDefinitions", <CosimDefInfo>, <CosimDefInfo>...]]</pre>
Python Example	<pre>oComponentManager.Add(["NAME:Component", "Info:=", ["Type:=", 0, "NumTerminals:=", 0, "DataSource:=", "", "ModifiedOn:=", 1467910752, "Manufacturer:=", "", "Symbol:=", "Component", "ModelNames:=", "", "Footprint:=", "", "Description:=", "", "InfoTopic:=", "",</pre>

	<pre>"InfoHelpFile:=", "", "IconFile:=", "", "Library:=", "", "OriginalLocation:=", "Project", "IEEE:=", "", "Author:=", "", "OriginalAuthor:=", "", "CreationDate:=", 1467910746, "ExampleFile:=", ""], "Refbase:=", "U", "NumParts:=", 1, "ModSinceLib:=", True, "CompExtID:=", 2])</pre>

AddNPortData [component manager]

Use: Adds a component using the specified data

Command: Project Menu > Add Model > Add Nport Model

Syntax: AddNPortData Array("NAME:<ComponentDataName>",

 "ComponentDataType:=", "NportData",

 "name:=", <string>, // Name of the item

```
"filename:=", <string>, // Path to the file to find the data
"numberOfports:=", <int>,
"filelocation:=", <LocationType>,
"domain:=", <string>, // "time" or "frequency"
"datamode:=", <string> // "EnterData", "Import", or "Link"
"devicename:=", <string>,
"ImpedanceTab:=", <bool>,
"NoiseDataTab:=", <bool>,
"DCBehaviorTab:=", <bool>,
"SolutionName:=", <string>,
"displayformat:=", <DisplayInfo>,
"datatype:=", <string>, // "SMatrix", "YMatrix", or "ZMatrix"
"ShowRefPin:=", <bool>,
"RefNodeCheckbox:=", <bool>, ...
<ProductOptionsInfo>)
```

Return Value: <string>

```
// composite name of the component.
// If the name requested conflicts with the name of an existing
// component, the requested name is altered to be unique.
// The name returned reflects any change made to be unique.
```

Parameters: <ComponentDataName>:

<string> // [simple name](#) of the component

<LocationType>:

<string> // one of "UsePath", "PersonalLib", "UserLib", "SysLib",
// or "Project".

<dcInfo>:

<string> // one of "DCOpen", "DCShort", "DCShShort",
// "DCNone", or "DCEmpty".

<DisplayInfo>:

<string> // one of "MagnitudePhase", "ReallImaginary",
// or "DbPhase".

<ProductOptionsInfo>:

// The remaining parameters differ by product

// HFSS 3D Layout - doesn't support interpolation/DC behavior

"DCOption:=", -1,

```
"InterpOption:=", -1,
```

```
"ExtrapOption:=", -1,
```

```
"DataType:=", 0
```

```
// Nexxim
```

```
"DCOption:=", <NexximDCOption>,
```

```
"InterpOption:=", <NexximInterpOption>,
```

```
"ExtrapOption:=", <NexximExtrapOption>,
```

```
"DataType:=", 2
```

```
<NexximDCOption>:
```

```
<int> // 0 : Zero Padding
```

```
// 1 : Same as last point
```

```
// 2 : Linear extrapolation from last 2 points
```

```
// 3 : Constant magnitude, linear phase extrapolation
```

```
// 4 : Leave all signal lines open circuited
```

```
// 5 : Short all signal lines together
```

```
// 6 : Short all signal lines to ground
```

```
<NexximInterpOption>:
```

```
<int> // 0 : Step
```

// 1 : Linear

<NexximExtrapOption>:

<int> // 0 : Zero padding

// 1 : Same as last point

// 2 : Linear extrapolation from last 2 points

// 3 : Constant magnitude, linear phase extrapolation

<CircuitDCOption>:

<int> // 0 : Leave all signal lines open circuited

// 1 : Short all signal lines together

// 2 : Short all signal lines to ground

// 3 : Extrapolate from data provided (not recommended)

<CircuitInterpOption>:

<int> // 0 : Linear

// 1 : Cubic spline

// 2 : Rational polynomial

<CircuitExtrapOption>:

<int> // 0 : Same as interpolation

// 1 : Zero padding

// 2 : Same as last point

VB Example:

```
oComponentManager.AddNPortData Array("NAME:ComponentData", _  
"ComponentDataType=", "NPortData", _  
"name=", "NportData", _  
"filename=", "", _  
"numberofports=", 2, _  
"filelocation=", "UsePath", _  
"domain=", "frequency", _  
"datamode=", "Import", _  
"devicename=", "", _  
"ImpedenceTab=", true, _  
"NoiseDataTab=", true, _  
"DCBehaviorTab=", true, _  
"SolutionName=", "", _  
"displayformat=", "MagnitudePhase", _  
"datatype=", "SMatrix", _  
"ShowRefPin=", true, _  
"RefNodeCheckbox=", false, _
```

```
"DCOption:=", 3, _  
"InterpOption:=", 1, _  
"ExtrapOption:=", 3, _  
"DataType:=", 2, _  
"DCOption:=", 3, _  
"InterpOption:=", 1, _  
"ExtrapOption:=", 3, _  
"DataType:=", 2)
```

AddSolverOnDemandModel

Use: This method looks for a local component of the name passed in, and to this component it adds an SOD model definition using the information passed in the VARIANT. It returns the name of the SOD model added.

Parameters: BSTR component name.

Parameters: VARIANT which is the SOD model data.

Return Value: Returns the name of the model added.

ClearSolutionCache [component manager]

Use: Clear the solution cache for dynamic link component.

Command: Each of the following commands will clear the solution cache:

- **Dynamic Link Item RCM > Clear Solution Cache**
- **Dynamic Link Component in schematic RCM > Clear Solution Cache**

Syntax: ClearSolutionCache <Component Name >

Return Value: None

Parameters: <component name> is the name of the dynamic link component

VB Example:

```
oComponentManager.ClearSolutionCache "TeeModel1"
```

Edit [component manager]

Modifies an existing component

Command: Tools > Edit Configured Libraries > Components > Edit Component

Syntax: Edit <ComponentName>,

```
    Array("NAME:<NewComponentName>",  
    "Info:=", <ComponentInfo>,  
    "RefBase:=", <string>, // reference designator  
    "NumParts:=", <int>, // parts per component  
    "OriginalComponent:=", <string>  
    "Terminal:=", <TerminalInfo>,  
    "Terminal:=", <TerminalInfo>, ...  
    // The remaining parameters are optional  
    Array("NAME:Parameters", // any combo of the following  
    "VariableProp:=", <VariableInfo>,  
    "CheckboxProp:=", <CheckBoxInfo>,  
    "ButtonProp:=", <ButtonInfo>,  
    "TextProp:=", <TextInfo>,
```



```
"NumberProp:=", <NumberInfo>,  
"SeparatorProp:=", <SeparatorInfo>,  
"ValueProp:=", <ValueInfo>,  
"MenuProp:=", <MenuInfo>),  
Array("NAME:Properties", // any combo of the following  
"CheckboxProp:=", <CheckBoxInfo>,  
"TextProp:=", <TextInfo>,  
"NumberProp:=", <NumberInfo>,  
"SeparatorProp:=", <SeparatorInfo>,  
"ValueProp:=", <ValueInfo>,  
"MenuProp:=", <MenuInfo>),  
"VPointProp:=", <VPointInfo>,  
"PointProp:=", <PointInfo>),  
Array("Quantities",  
"QuantityProp:=", <QuantityPropInfo>...),  
Array("NAME:CosimDefinitions",  
  <CosimDefInfo>,  
  <CosimDefInfo>...)
```

Return Value: <string>

// [composite name](#) of the component.

// If the name requested conflicts with the name of an existing
// component, the requested name is altered to be unique.
// The name returned reflects any change made to be unique.

Parameters: <ComponentName>:

<string> // [composite name](#) of the component to edit

<NewComponentName>:

<string> // new [simple name](#) for the component

<ComponentInfo>:

Array("Type:=", <TypeInfo>,

"NumTerminals:=", <int>,

"DataSource:=", <string>,

"ModifiedOn:=", <ModifiedOnInfo>,

"Manufacturer:=", "<string>,"

"Symbol:=", <string>,"

"Footprint:=", <string>,"

"Description:=", <string>,"

"InfoTopic:=", <string>,"

"InfoHelpFile:=", <string>,"

```
"IconFile:=", <string>,  
"LibraryName:=", <string>,  
"OriginalLocation:=", <string>, // Project Location  
"Author:=", <string>,  
"OriginalAuthor:=", <string>,  
"CreationDate:= ", <int>)
```

<TypeInfo>:

An integer that is the or-ing of bits for each product listed below. The default setting is 0xffffffff (4294967295) which indicates valid for all products. In the component editing dialog, checking different boxes in the "Specify products for which this component is valid" grid control sets specific flags that correspond to the following hex/decimal settings:

Nexxim -- 100 binary, 4 decimal, 0x4

SlwaveDeNovo -- 1000 binary, 8 decimal, 0x8

Simplorer -- 10000 binary, 16 decimal, 0x10

MaxwellCircuit -- 100000 binary, 32 decimal, 0x20

<ModifiedOnInfo>:

An integer that corresponds to the number of seconds that have elapsed since 00:00 hours, Jan 1, 1970 UTC from the system clock.

<TerminalInfo>:

```
Array(<string>, // symbol pin
<string> // footprint pin
<string>, // gate name
<bool>, // shared
<int>, // equivalence number
<int>, // what to do if unconnected: flag as error:0, ignore:1
<string>, // description
<Nature>)
```

<Nature>:

<string> // content varies as follows

Nexxim/Circuit:

"Electrical" // the only choice

Simplorer:

// several choices

"Electrical", "Magnetic", "Fluidic", "Translational",

"Translational_V", "Rotational", "Rotational_V",

""Radiant", "Thermal", or <VHDLPackageName>

<VHDLPackageName>:

<string> // in the form <Library>.<Package>

<Library>:

<string> // name of the VHDL library

<Package>:

<string> // name of the VHDL package

<VariableInfo>:

Array(<string>, // name

<FlagLetters>,

<string>, // description

"CB:=", <string>, // optional - script for call back

<string>) // value: number, variable, or expression

<FlagLetters>:

<string> // "D" - has description parameter,

// "RD" - readonly & has description parameter,

// or "RHD" - readonly, hidden, & has description parameter

<CheckBoxInfo>:

Array(<string>, // name

<FlagLetters>,

<string>, // description

"CB:=", <string>, // optional - script for call back

<bool>) // value: true or false

<ButtonInfo>:

Array(<string>, // name

<FlagLetters>,

<string>, // description

<string>, // button title

<string>, // extra text

<ClientID>,

"ButtonPropClientData:= ", <ClientDataArray>)

<ClientID>:

<int> // specifies Button Prop Client

// 0 - unknown, "ButtonPropClientData

// array will be empty

// 1 - Netlist Prop Client

// 2 - not used

// 3 - File Name Prop Client

<ClientDataArray>:

varies with <ClientID>

<ClientID> is 0 or 1: empty array

Array()

<ClientID> is 3:

Array("InternalFormatText:=", "<prefix><RelativePath>")

<prefix>:

<string> // "<Project>", "<PersonalLib>", "<UserLib>", or "<SysLib>"

<RelativePath>:

<string> // relative path to file from <prefix>

<TextInfo>:

```
Array(<string>, // name
<FlagLetters>,
<string>, // description
"CB:=", <string>, // optional - script for call back
<string>) // value: a text string
```

<NumberInfo>:

```
Array(<string>, // name
<FlagLetters>,
<string>, // description
"CB:=", <string>, // optional - script for call back
<real>, // value: a number
<string>) // units
```

<SeparatorInfo>:

```
Array(<string>, // name
<FlagLetters>,
<string>, // description
"CB:=", <string>, // optional - script for call back
<string>) // value: a text string
```


<ValueInfo>:

```
Array(<string>, // name  
<FlagLetters>,  
<string>, // description  
"CB:=", <string>, // optional - script for call back  
<string>) // value: a number, variable or expression
```

<MenuPropInfo>:

```
Array(<string>, // name  
<FlagLetters>,  
<string>, // description  
<string>, // menu choices - separated by commas  
<int>) // 0 based index of current menu choice
```

<VPointInfo>:

```
Array(<string>, // name  
<FlagLetters>,  
<string>, // description  
"CB:=", <string>, // optional - script for call back  
<string>, // x value: number with length units  
<string>) // y value: number with length units
```

<PointInfo>:

Array(<string>, // name

<FlagLetters>,

<string>, // description

"CB:=", <string>, // optional - script for call back

<real>, // x value

<real>) // y value

<QuantityPropInfo>:

Array(<string>, // name

<FlagLetters>,

<string>, // description

<string>, // value

<TypeString>,

<TypeStringDependentInfo>)

<TypeString>:

<string> // "Across", "Through", or "Free"

<TypeStringDependentInfo>:

"Free" :

<string>, // direction: "In", "Out", "InOut", or "DontCare"

// Following <string> is not present if direction is "DontCare"

<string> // when to calculate: "BeforeAnalogSolver",

// "BeforeStateGraph", "AfterStateGraph", or "DontCareWhen"

"Across" or "Through"

<int>, // terminal 1

<int> // terminal 2

<CosimDefInfo>:

Array("NAME:CosimDefinition",

"CosimulatorType:=", <int>,

"CosimDefName:=", <string> // "HFSS3D", "Circuit",

// "Custom", or "Netlist"

"IsDefinition:=", <bool>,

final array member(s) vary with CosimDefName)

final array members for HFSS 3D Layout:

"CosimStackup:=", <string>,

"CosimDmbedRatio:=", <int>

final array members for Circuit:

"ExportAsNport:=", <int>,

"UsePjt:=", <int>

final array member for Custom:

"DefinitionCompName:=", <string>

final array member for Netlist:

"NetlistString:=", <string>

VB Example:

```
Dim name
```

```
name = oComponentManager.Edit ("Nexxim Circuit Elements\BJTs:Level01_NPN", _  
Array("NAME:Level01_NPN", _  
"Info:=", Array("Type:=", 4294901764,_  
"NumTerminals:=", 3, _  
"DataSource:=", "Ansys built-in component", _  
"ModifiedOn:=", 1152722112, _  
"Manufacturer:=", "", _  
"Symbol:=", "nexx_bjt_npn", _
```

```
"Footprint:=", "", _
"Description:=", "BJT, GP, NPN", _
"InfoTopic:=", "NXBJT1.htm", _
"InfoHelpFile:=", "nexximcomponents.chm", _
"IconFile:=", "bjtsn.bmp", _
"Library:=", "Nexxim Circuit Elements\BJTs", _
"OriginalLocation:=", "SysLibrary ", _
"Author:=", "", _
"OriginalAuthor:=", "", _
"CreationDate:=", 1152722102), _
"Refbase:=", "Q", _
"NumParts:=", 1, _
"Terminal:=", Array("collector", _
"collector", _
"A", _
false, _
6, _
0, _
"", _
"Electrical"), _
```

```
"Terminal:=", Array("base", _  
"base", _  
"A", _  
false, _  
7, _  
0, _  
"", _  
"Electrical"), _  
"Terminal:=", Array("emitter", _  
"emitter", _  
"A", _  
false, _  
8, _  
0, _  
"", _  
"Electrical"), _  
Array("NAME:Parameters", _  
"TextProp:=", Array("LabelID", _  
"HD", _  
"Property string for netlist ID", _  
"Q@ID"), _
```

```
"TextProp:=", Array("MOD", _  
"D", _  
"Name of model data reference", _  
"required"), _  
"VariableProp:=", Array("AREA", _  
"D", _  
"Emitter area multiplying factor, which affects curents, resistances, and capacitances", _ "1"),  
_  
"VariableProp=", Array("AREAB", _  
"D", _  
"Base AREA", _  
"1"), _  
"VariableProp:=", Array( "AREAC", _  
"D", _  
Collector AREA", _  
"1"), _  
"VariableProp:=", Array("DTEMP", _  
"D", _  
"The difference between element and circuit temperature deg Cel)", _  
"0"), _  
"VariableProp:=", Array("M", _
```

```

"D", _
"Multiplier factor to simulate multiple BJTs in paralel", _ "1"), _
"ButtonProp:=", Array("NexximNetlist", _
"HD", _
"", _
"Q@ID %0 %1 %2 *MOD(@MOD) *AREA (AREA=@AREA) " & _
" *AREAB (AREAB=@AREAB) *AREAC (AREAC=@ " & _
"AREAC) *DTEMP (DTEMP=@DTEMP) *M (M=@M) ", _
"Q@ID %0 %1 %2 *MOD(@MOD) " & _ "*AREA (AREA=@AREA) AREAB (AREAB=@AREAB) *AREAC (AREAC=@ " & _
"AREAC) *DTEMP (DTEMP=@DTEMP) *M (M=@M) ", _
1, _
"ButtonPropClientData:=", Array()), _
"TextProp:=", Array( "ModelName", _
"HD", _
"", _
"Q"))))

```

VB Example:

```

Dim name2
name2 = oComponentManager.Edit "MyComponent", _
(Array("NAME:MyOtherComponent", _
"Info:=", Array("Type:=", 4294901767, _

```



```
"NumTerminals:=", 2, _  
"DataSource:=", "", _  
"ModifiedOn:=", 1071096503, _  
"Manufacturer:=", "Ansys", _  
"Symbol:=", "bendo", _  
"Footprint:=", "BENDO", _  
"Description:=", "", _  
"InfoTopic:=", "", _  
"InfoHelpFile:=", "", _  
"IconFile:=", "", _  
"LibraryName:=", "", _  
"OriginalLocation:=", "Project", _  
"Author:=", "", _  
"OriginalAuthor:=", "", _  
"CreationDate:= ", 1147460679), _  
"Refbase:=", "U", _  
"NumParts:=", 1, _  
"OriginalComponent:=", "", _  
"Terminal:=", Array("n1", _  
"n1", _
```

```
"A", _  
false, _  
0, _  
0, _  
"", _  
"Electrical"), _  
"Terminal:=", Array("n2", _  
"n2", _  
"A", _  
false, _  
1, _  
0, _  
"", _  
Electrical"), _  
Array("NAME:Parameters", _  
"MenuProp:=", Array("CoSimulator", _  
"D", _  
"", _  
"Default,HFSS3D,Circuit,Custom,Netlist", _  
0), _  
"ButtonProp:=", Array("CosimDefinition", _
```

```
"D", _  
"  
"  
"  
"Edit", _  
0, _  
"ButtonPropClientData:=", Array()), _  
Array("NAME:CosimDefinitions", _  
Array("NAME:CosimDefinition", _  
"CosimulatorType:=", 0, _  
"CosimDefName:=", "HFSS3D", _  
"IsDefinition:=", true, _  
"CosimStackup:=", "Layout stackup", _  
"CosimDmbedRatio:=", 3), _  
Array("NAME:CosimDefinition", _  
"CosimulatorType:=", 1, _  
"CosimDefName:=", "Circuit", _  
"IsDefinition:=", true, _  
"ExportAsNport:=", 0, _  
"UsePjt:=", 0), _  
Array("NAME:CosimDefinition", _
```

```
"CosimulatorType:=", 2, _
"CosimDefName:=", "Custom", _
"IsDefinition:=", true, _
"DefinitionCompName:=", ""), _
Array("NAME:CosimDefinition", _
"CosimulatorType:=", 3, _
"CosimDefName:=", "Netlist", _
"IsDefinition:=", true, _
"NetlistString:=", "")))
```

Python Syntax	<pre>Edit <ComponentName>, ["NAME:<NewComponentName>", "Info:=", <ComponentInfo>, "RefBase:=", <string>, // reference designator "NumParts:=", <int>, // parts per component "OriginalComponent:=", <string> "Terminal:=", <TerminalInfo>, "Terminal:=", <TerminalInfo>, ... #The remaining parameters are optional ["NAME:Parameters", // any combo of the following "VariableProp:=", <VariableInfo>,</pre>

```

"CheckboxProp:=", <CheckBoxInfo>,
"ButtonProp:=", <ButtonInfo>,
"TextProp:=", <TextInfo>,
"NumberProp:=", <NumberInfo>,
"SeparatorProp:=", <SeparatorInfo>,
"ValueProp:=", <ValueInfo>,
"MenuProp:=", <MenuInfo>],
["NAME:Properties", # any combo of the following
"CheckboxProp:=", <CheckBoxInfo>,
"TextProp:=", <TextInfo>,
"NumberProp:=", <NumberInfo>,
"SeparatorProp:=", <SeparatorInfo>,
"ValueProp:=", <ValueInfo>,
"MenuProp:=", <MenuInfo>),
"VPointProp:=", <VPointInfo>,
"PointProp:=", <PointInfo>),
["Quantities",
"QuantityProp:=", <QuantityPropInfo>...],
["NAME:CosimDefinitions",
<CosimDefInfo>,

```

	<CosimDefInfo>...])
Python Example	<pre> name = oComponentManager.Edit ("Simplorer Circuit Elements\BJTs:Level01_NPN", _ ["NAME:Level01_NPN", "Info:=", ["Type:=", 4294901764,_ "NumTerminals:=", 3, "DataSource:=", "Ansoft built-in component",_ "ModifiedOn:=", 1152722112, "Manufacturer:=", "", _ "Symbol:=", "nexx_bjt_npn", "Footprint:=", "", _ "Description:=", "BJT, GP, NPN", "InfoTopic:=", "NXBJT1.htm", _ "InfoHelpFile:=", "nexximcomponents.chm", "IconFile:=", "bjtsn.bmp", _ "Library:=", "Nexxim Circuit Elements\BJTs",_ "OriginalLocation:=", "SysLibrary ", "Author:=", "", _ "OriginalAuthor:=", "", "CreationDate:=", 1152722102], _ "Refbase:=", "Q", "NumParts:=", 1, "Terminal:=", ["collector", _ "collector", "A", false, 6, 0, "", "Electrical"], _ "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"], _ </pre>

	<pre> "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"]]]) </pre>
Python	<pre> name2 = oComponentManager.Edit ("MyComponent",_ </pre>

**Example
2**

```
(["NAME:MyOtherComponent","Info:=", ["Type:=", 4294901767, _  
  
"NumTerminals:=", 2, "DataSource:=", "", _  
  
"ModifiedOn:=", 1071096503, "Manufacturer:=", "Ansoft", _  
  
"Symbol:=", "bendo", "Footprint:=", "BENDO", _  
  
"Description:=", "", "InfoTopic:=", "", _  
  
"InfoHelpFile:=", "", "IconFile:=", "", _  
  
"LibraryName:=", "", "OriginalLocation:=", "Project", _  
  
"Author:=", "", "OriginalAuthor:=", "", _  
  
"CreationDate:= ", 1147460679], "Refbase:=", "U", _
```



```

"NumParts:=", 1, "OriginalComponent:=", "", _

"Terminal:=", ["n1", "n1", "A", false, 0, 0, "", _

"Electrical"], "Terminal:=", ["n2", "n2", "A", _

false, 1, 0, "", Electrical"], ["NAME:Parameters", _

"MenuProp:=", ["CoSimulator", "D", "", _

"Default,Custom,Netlist", 0], "ButtonProp:=", ["CosimDefinition", _

"D", "", "", "Edit", 0, "ButtonPropClientData:=", []], _

["NAME:CosimDefinitions", ["NAME:CosimDefinition", _

"CosimulatorType:=", 0, "CosimDefName:=", "HFSS3D", _

```

```
"IsDefinition:=", true, "CosimStackup:=", "Layout stackup", _  
  
"CosimDmbedRatio:=", 3], ["NAME:CosimDefinition", _  
  
"CosimulatorType:=", 1, "CosimDefName:=", "", _  
  
"IsDefinition:=", true, "ExportAsNport:=", 0, _  
  
"UsePjt:=", 0], ["NAME:CosimDefinition", _  
  
"CosimulatorType:=", 2, "CosimDefName:=", "Custom", _  
  
"IsDefinition:=", true, "DefinitionCompName:=", ""], _  
  
["NAME:CosimDefinition", "CosimulatorType:=", 3, _  
  
"CosimDefName:=", "Netlist", "IsDefinition:=", true, _  
  
"NetlistString:=", ""]]])
```


EditSolverOnDemandModel

Use: This method looks for a local component of the name passed in, and in this component it looks for an SOD model using the name passed in the second BSTR. It modifies the SOD model using the data in the VARIANT. It returns the name of the SOD model edited.

Return Value: Returns the name of the model edited.

Parameters: BSTR component name.

Parameters: BSTR SOD model name.

Parameters: VARIANT which is the new SOD model data (can include changed name).

EditWithComps [component manager]

Edit an existing component.

Command: None

Syntax: EditWithComps <ComponentName>,

```
    Array("NAME:<NewComponentName>",  
    "ModTime:=", <ModifiedTimeInfo>,  
    "Library:=", <string>, // Library name  
    "LibLocation:=", <string>, // Project Location  
    <PinDefInfo>,  
    <PinDefInfo>,... // optional, to define pins  
    <GraphicsDataInfo>, // optional, to define graphics  
    <PropDisplayMapInfo>), // optional, to define property displays
```

Array(<ListOfComponentNames>) // Component names

Return Value: <string>

// [composite name](#) of the component.

// If the name requested conflicts with the name of an existing

// component, the requested name is altered to be unique.

// The name returned reflects any change made to be unique.

Parameters: <ComponentName>:

<string> // [composite name](#) of the component being edited

<NewComponentName>:

<string> // new [simple name](#) for the component

<ModifiedOnInfo>:

An integer that corresponds to the number of seconds that have elapsed

since 00:00 hours, Jan 1, 1970 UTC from the system clock.

<PinDefInfo>:

Array("NAME:PinDef",

"Pin:=", Array (<string>, // pin name

<real>, // x location
<real>, // y location
<real>, // angle in radians
<PinType>,
<real>, // line width
<real>, // line length
<bool>, // mirrored
<int>, // color
<bool>, // true if visible, false if not
<string>, // hidden net name
<OptionalPinInfo>, // optional info
<PropDisplayMapInfo>)) // optional

<PinType>:
<string> // "N" : normal pin
// "I" : input pin
// "O" : output pin

<OptionalPinInfo>:
// Specify both or neither

<bool>, // true if name is to be shown

<bool>, // true if number is to be shown

<PropDisplayMapInfo>:

Array("NAME:PropDisplayMap",

<PropDisplayInfo>,

<PropDisplayInfo>,...)

<PropDisplayInfo>:

<NameString>, Array(<DisplayTypeInfo>,

<DisplayLocationInfo>,

<int>, // optional, level number

<TextInfo>)

<NameString>:

<string> // PropertyName:=, where PropertyName is the name of

// the property to be displayed

<DisplayTypeInfo>:

<int> // 0 : No display

// 1 : Display name only

// 2 : Display value only
// 3 : Display both name and value
// 4: Display evaluated value only
// 5: Display both name and evaluated value

<DisplayLocationInfo>:

<int> // 0 : Left

// 1 : Top

// 2 : Right

// 3 : Bottom

// 4 : Center

// 5 : Custom placement

<GraphicsDataInfo>:

Array("NAME:Graphics",

// one or more of the following

<RectInfo>,

<CircleInfo>,

<ArcInfo>,

<LineInfo>,

```
<PolygonInfo>,  
<TextInfo>,  
<ImageInfo>)
```

```
<RectInfo>:  
"Rect:=", Array(<real>, // line width  
<int>, // fill pattern  
<int>, // color  
<real>, // angle, in radians  
<real>, // x position of center  
<real>, // y position of center  
<real>, // width  
< real>) // height
```

```
<CircleInfo>:  
"Circle:=", Array(<real>, // line width  
<int>, // fill pattern  
<int>, // color  
<real>, // x position of center  
<real>, // y position of center  
< real>) // radius
```


<ArcInfo>:

"Arc:=", Array(<real>, // line width

<int>, // line pattern

<int>, // color

<real>, // x position of center

<real>, // y position of center

< real>, // radius

<real>, // start angle, in radians

<end>) // end angle, in radians

<LineInfo>:

"Line:=", Array(<real>, // line width

<int>, // line pattern

<int>, // color

<PointInfo>, // must specify at least 2 points

<PointInfo>...)

<PointInfo>:

<real>, // x position

<real> // y position

<PolygonInfo>:

"Polygon:=", Array(<real>, // line width

<int>, // fill pattern

<int>, // color

<PointInfo>, // must specify at least 3 points

<PointInfo>...)

<TextInfo>:

"Text:=", Array(<real>, // x position

<real>, // y position

<real>, // angle, in radians

<Justification>,

<bool>, // is plotter font

<string>, // font name

<int>, // color

<string>) // text string

<Justification>:

<int> // 0 : left top

// 1 : left base

// 2 : left bottom

// 3 : center top

// 4 : center base

// 5 : center bottom

// 6 : right top

// 7 : right base

// 8 : right bottom

<ImageInfo>:

"Image:=", Array(<RectInfo>,

<ImageData>,

<bool>) // is mirrored

<ImageData>:

<string>, // file path

<int>, // 0 : use the file path and link to it

// 1 : ignore file path and use next parameter

<string> // text data, only present if preceding int is 1

<ListOfComponentNames>:

<string>,<string> ...

// The list may be empty. When not empty, each string that is listed is a component

// that references the component to be edited. Prior to editing, a clone of the component is

// made, and the components that are listed are modified so that they now refer to

// the clone.

VB Example:

```
Dim nam
```

```
oModelManager.EditWithComps_
```

```
("Nexxim Circuit Elements\Distributed\Distributed:bendo", _
```

```
Array("NAME:bendo new name", _
```

```
"ModTime:=", 1152722165, _
```

```
"Library:=", "Nexxim Circuit Elements\Distributed\Distributed", _
```

```
"LibLocation:=", "SysLibrary", _
```

```
Array("NAME:PinDef", _
```

```
"Pin:=", Array( "n1", _
```

```
-0.00254, _
```

```
0.00254, _
```

```
0, _
```

```
"N", _
```

```
0, _
```

```
0, _  
false, _  
0, _  
true, _  
"", _  
false, _  
false)), _  
Array("NAME:PinDef", _  
"Pin:=", Array( "n2", _  
0.00254, _  
-0.00254, _  
1.5708, _  
"N", _  
0, _  
0, _  
false, _  
0, _  
true, _  
"", _  
false, _
```

```
false)), _
Array("NAME:Graphics", _
"Polygon=", Array( 0, 0, 12566272, 0, 0.00381, 0, .00127, 0.00127, _
0.00127, 0.00127, 0, 0.00381, 0, 0.00381, _
0.002032, 0.00127, 0.00381), _
"Line=", Array(0, 1, 12566272, -0.00254, 0.00254, 0, .00254), _
"Line=", Array(0, 1, 12566272, 0.00254, -0.00254, .00254, 0)), _
Array("NAME:PropDisplayMap", _
"W=", Array(3, _
5, _
0, _
"Text=", Array( 2.1684E-019, _
0.00504119, _
0, _
1, _
5, _
false, _
"Arial", _
0, _
"W=***")))), _
Array("MY_COMP")
```

Export [component manager]

Export component(s) to a library

Command: Tools > Edit Configured Libraries > Components > Export to Library

Syntax: Export Array("NAME:<LibraryName>",

 <ComponentName>,

 <ComponentName>...),

 <LibraryLocation>

Return Value: None

Parameters: <LibraryName>:

 <string> // name of the library

 <ComponentName>:

 <string> // [composite name](#) of the component to export

 <LibraryLocation>:

 <string> // location of the library in <LibraryName>

 // One of "Project", "PersonalLib", or "UserLib"

VB Example:

```
oComponentManager.Export Array("NAME:mylib", "Nexxim Circuit Elements\BJTs:Level01_NPN"),"PersonalLib"
```

Python Syntax	Export(["NAME:<LibraryName>", <ComponentName>, <ComponentName>...], <LibraryLocation>)
Python Example	<pre>oComponentManager.Export (["NAME:mylib", "Simplorer Circuit Elements\BJTs:Level01_NPN"], "PersonalLib")</pre>

GetData [component manager]

Gets data that describes the definition.

Command: None

Syntax: GetData(<DefinitionName>)

Return Value: <DefinitionData> This is an array of data for the definition.

Parameters: <DefinitionName>:

<string> // [composite name](#) of the definition to edit

VB Example:

```
Dim compData  
compData = oComponentManager.GetData ("Level01_NPN")
```


Note:

GetData allows the user to access definition information, make modifications, and then use the Edit or EditWithComps script commands to save the modified definition. Accordingly, for each type of definition, the array data returned to GetData should be the same array information that is supplied to the [Edit](#) or [EditWithComps](#) commands.

Python Syntax	GetData(<DefinitionName>)
Python Example	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()
```

Python Syntax	GetNames()
Python Example	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")
```

Python Syntax	IsUsed(<ComponentName>)
Python Example	IsUsed = oComponentManager.IsUsed("MyComponent")

Remove [component manager]

Remove a component from a library

Command: Tools > Edit Configured Libraries > Components > Remove Component

Syntax: Remove <ComponentName>,

 <IsProjectComponent>,

 <LibraryName>,

 <LibraryLocation>

Return Value: None

Parameters: <ComponentName>:

 <string> // [composite name](#) of the component to remove

 <IsProjectComponent>:

 <bool>

 <LibraryName>:

<string> // name of the library

<LibraryLocation>:
<string> // location of the library in <LibraryName>
// One of "Project", "PersonalLib", or "UserLib"

VB Example:

```
oComponentManager.Remove "Nexxim Circuit Elements\BJTs:Level01_NPN", _ true, "Project"
```

Python Syntax	Remove (<ComponentName>, <IsProjectComponent>, <LibraryName>, <LibraryLocation>)
Python Example	oComponentManager.Remove ("Simplorer Circuit Elements\BJTs:Level01_NPN", _ true, "Project")

RemoveSolverOnDemandModel

Use: This method looks for a local component of the name passed in, and in this component it looks for an SOD model of the name passed in and deletes the SOD model definition from the component.

Parameters: BSTR component name.

Parameters: BSTR SOD model name

Return Value: None.

RemoveUnused [component manager]

Removes components that are not used in the design.

Command: **Project->Remove Unused Definitions** is similar but operates slightly different and does not record script commands.

Syntax: RemoveUnused()

Return Value: <bool> True if one or more components are removed.

Parameters: None

VB Example:

```
Dim removedDefs  
removedDefs = oComponentManager.RemoveUnused()
```

Note:

The order of calls to RemoveUnused is significant. As a result, removing definitions in an unordered fashion may cause other components in dependent definitions to be rendered unusable.

Also, the symbol and footprint of an unused component are not unusable until after the component itself is removed using the Component Manager Remove script.

Python Syntax	RemoveUnused()
Python Example	removedDefs = oComponentManager.RemoveUnused()

Update Dynamic Link [component manager]

Use: Reads data from the linked design and updates the dynamic link component. This will update the following: properties, solutions, ports, geometry.

Command: Each of the following commands will record a non-undoable script command:

- Dynamic Link Item RCM > Refresh Dynamic Link**
- Dynamic Link Component in schematic RCM > Refresh Dynamic Link**
- Click the **Refresh Dynamic Link** button in the Footprint tab of the **Properties Window** for selected Dynamic Link components in the Layout Editor.

Syntax: UpdateDynamicLink(<component name>)

Return Value: None

Parameters: <component name> is the name of the dynamic link component

VB Example:

```
oComponentManager.UpdateDynamicLink("TeeModel_L1")
```

Add [footprint manager]

Use: Add a footprint

Command: Tools > Edit Configured Libraries > Footprints > Add Footprint

Syntax: Add Array("NAME:<FootprintName>,"

"ModTime:=", <ModifiedOnInfo>,"

"Library:=", "",

"LibLocation:=", "Project",

"OkayToMirror:=", <bool>,"

"DefUnits:=", <UnitType>,"

Array(NAME:Lyrs",

"Layer:=", <LayerArray>,"

"Layer:=", <LayerArray>...",

```
"SLayer:=", <StackupLayerArray>,  
"SLayer:=", <StackupLayerArray>...),  
"ActLyr:=", <string>, // name of active layer  
"Tol:=", <ToleranceArray> // optional  
<PrimitivesInfo>, // optional  
<PinsInfo>, // optional  
<ViasInfo>, // optional  
<EdgeportsInfo>, // optional  
<ComponentPropertyInfo>,  
<ScriptInfo>) // optional, specified for scripted footprints
```

Return Value: <string>

```
// composite name of the footprint.  
// If the name requested conflicts with the name of an existing  
// footprint, the requested name is altered to be unique.  
// The name returned reflects any change made to be unique.
```

Parameters: <FootprintName>:

```
<string> // simple name of footprint to create
```

```
<ModifiedOnInfo>:
```

An integer that corresponds to the number of seconds that have elapsed

since 00:00 hours, Jan 1, 1970 UTC from the system clock.

<UnitType>:

<string> // default length units to use if units are not specified in other

// parameters

<LayerArray>:

Array("N:=", <string>, // layer name

"ID:=", <int> ,

"T:=", <LayerTypeInfo>, // layer type

"TB:=", <TopBottomInfo> ,

"Col:=", <int>, // optional - color

"Pat:=", <int>, // optional - fill pattern

"Vis:=", <bool>, // optional - are objects on layer visible

"Sel:=", <bool>, // optional - are objects on layer selectable

"L:=", <bool>) // optional

// are objects on layer locked (can't be edited)

<LayerTypeInfo>:

<string> // one of: signal, dielectric, metalized signal, assembly, silkscreen, soldermask, solderpaste, glue, or user

<TopBottomInfo>:

<string> // one of: top, neither, bottom, or template

<StackupLayerArray>:

Array(<LayerArray>,

"Elev:=", <ElevationInfo>,

"SubL:=", Array("Th:=", <Dimension>,

"LElev:=", <Dimension>,

"R:=", <Dimension>,

"M:=", <MaterialInfo>))

<ElevationInfo>:

<string> // "top" - snap to top

// "mid" - snap to middle

// "bot" - snap to bottom

// "edit" - manual edit

// "none"

<Dimension>:

<string> // real number, may include units

<MaterialInfo>:

<string> // name of the layer material

<ToleranceArray>:

Array(<real>, // distance tolerance

<real>, // angle tolerance (radians)

<real>) // dimensionless tolerance

<PrimitivesInfo>:

Array("NAME:Prims",

// one or more of the following

<RectInfo>,

<CircleInfo>,

<ArcInfo>,

<LineInfo>,

<PolygonInfo>,

<TextInfo>,

<ImageInfo>)

<RectInfo>:

```
"Rect:=", Array(<real>, // line width
<int>, // fill pattern
<int>, // color
<real>, // angle, in radians
<real>, // x position of center
<real>, // y position of center
<real>, // width
< real>) // height
```

```
<CircleInfo>:
```

```
"Circle:=", Array(<real>, // line width
<int>, // fill pattern
<int>, // color
<real>, // x position of center
<real>, // y position of center
< real>) // radius
```

```
<ArcInfo>:
```

```
"Arc:=", Array(<real>, // line width
<int>, // line pattern
<int>, // color
```

<real>, // x position of center
<real>, // y position of center
< real>, // radius
<real>, // start angle, in radians
<end>) // end angle, in radians

<LineInfo>:

"Line:=", Array(<real>, // line width
<int>, // line pattern
<int>, // color
<PointInfo>, // must specify at least 2 points
<PointInfo>...)
<PointInfo>:
<real>, // x position
<real> // y position

<PolygonInfo>:

"Polygon:=", Array(<real>, // line width
<int>, // fill pattern
<int>, // color

<PointInfo>, // must specify at least 3 points

<PointInfo>...)

<TextInfo>:

"Text:=", Array(<real>, // x position

<real>, // y position

<real>, // angle, in radians

<Justification>,

<bool>, // is plotter font

<string>, // font name

<int>, // color

<string>) // text string

<Justification>: <int>

// 0 : left top

// 1 : left base

// 2 : left bottom

// 3 : center top

// 4 : center base

// 5 : center bottom

// 6 : right top

// 7 : right base

// 8 : right bottom

<ImageInfo>:

"Image:=", Array(<RectInfo>,

<ImageData>,

<bool>) // is mirrored

<ImageData>:

<string>, // file path

<int>, // 0 : use the file path and link to it

// 1 : ignore file path and use next parameter

<string> // text data, only present if preceding int is 1

<PinsInfo>:

Array("NAME:Pins",

"P:=", <PinArray>,

"P:=", <PinArray>,...)

<PinArray>:

Array("Port:=", Array("Id:=", <int>,

"Clr:=", <real>, // optional - clearance
"N:=", <string>), // pin name
"Pos:=", Array("x:=", <Location>, // padstack (x,y) position
"y:=", <Location>),
"VRt:=", <Angle>, // optional - rotation
"HD:=", <Size>, // optional - hole diameter
<PadstackImplementationInfo>)

<Location>:

<string> specifying real number and units (may use variables)

<Angle>:

<string> specifying angle with a real number and units

<Size>:

<string> specifying size with a real number and units

<PadstackImplementationInfo>:

If another with the same implementation has already been specified:

"Ref:=", <int> // id of the other

If not:

```
"Ref:=", <string>, // name
"Frm:=", <int>, // id of highest layer
"To:=", <int>, // id of lowest layer
<LayerPlacementInfo>,
"Man:=", <int>, // optional, 1 if manually placed, 0 if not (default)
"Use:=", Array(<PadUseInfo>, <PadUseInfo>...) // array may be empty
```

```
<LayerPlacementInfo>:
"Lyr:=", Array("Mrg:=", <int>, // optional,
// 1 if all layers have been merged (default)
// 0 if layer are not merged
"Flp:=", <int>, // optional,
// 1 if placed bottom up
// 0 if not (default)
"Map:=", <ParentToLocalLayerInfo>)
```

```
<ParentToLocalLayerInfo>:
Array("U:=", <DirectionOfUniqueness>,
"F:=", Array(<int>, <int>, ...), // forward mapping
// -1 is not mapped
```

```
"B:=", Array(<int>, <int>, ...) // backward mapping  
// -1 is not mapped
```

```
<PadUseInfo>:
```

```
"Pad:=", Array("Lid:=", <int>, // layer id  
"T:=", <string>, // type : "connected", "thermal",  
// "no_pad", "not_connected",  
// or "not_connected_thermal"  
"Man:=", <int>) // optional, 1 if manually placed  
// 0 if not (default)
```

```
<DirectionOfUniqueness>:
```

```
<string> // one of "forward", "backward", or "two ways"
```

```
<ViasInfo>:
```

```
Array("NAME:Vias", <VialInfo>, <VialInfo>...)
```

```
<VialInfo>:
```

```
"V:=", Array("Id:=", <int>,  
"N:=", <string>, // name  
"Pos:=", Array("x:=", <Location>, // via (x,y) position
```

"y:=", <Location>),

"VRt:=", <Angle>, // optional - rotation

<ImplementationInfo>

<EdgeportsInfo>:

Array("NAME:EPorts", <EdgePortArray>, <EdgePortArray>...)

<EdgePortArray>:

Array("NAME:EP",

"LP:=", Array("Id:=", <int>, // port id

"N:=", <string>), // port name

"Eo:=", Array(<edge description>, <edge description>,...))

<edge description> for primitive edges

"et:=", "pe", "pr:=", <id>, "ei:=", <edge#>

<id>: integer that is the primitive id

<edge#>: integer that is the edge number on the primitive

<edge description> for via edges

```
"et:=", "pse", "layer:=", <layer id>, "se:=", <via id>,  
"sx:=", <start X location>, "sy:=", <start Y location>, "ex:=", <end X location>,  
"ey:=", <end Y location>, "h:=", <arc height>, "rad:=", <radians>
```

<via id>: an integer that is the id of the via to use

<layer id>: an integer that is the id of the layer of the pad of the via to use

<start X location>

<start Y Location>:

doubles that are the X, Y location of the start point of the edge arc <end X location>

<end Y Location>:

doubles that are the X, Y location of the end point of the edge arc

<arc height>: double giving the height of the edge arc (0 for a straight edge)

<radians>: double giving the arc size in radians (0 for a straight edge)

<ComponentPropertyInfo>:

```
Array("NAME:CProps",
```

```
"VariableProp:=", <VariableInfo>,  
"VariableProp:=", <VariableInfo>,  
...)
```

```
<VariableInfo>:  
Array(<string>, // name  
<FlagLetters>,  
<string>, // description  
"CB:=", <string>, // optional - script for call back  
<string>) // value: number, variable, or expression
```

```
<ScriptInfo>:  
Array("NAME:script",  
"language:=", <string>, // one of "javascript" or "vbscript"  
"UsesScript:=", true,  
"script:=", <string>) // contents of script
```

VB Example:

```
oFootprintManager.Add (Array("NAME:BCL", _  
"ModTime:=", 1023388445, _
```

```
"Library:=", "", _  
"LibLocation:=", "Project", _  
"OkayToMirror:=", false, _  
"DefUnits:=", "mm", _  
Array("NAME:Lyrs", _  
"Layer:=", Array("N:=", "Measures", _  
"ID:=", 8, _  
"T:=", "measures", _  
"Col:=", 4144959, _  
"Pat:=", 1), _  
"Layer:=", Array("N:=", "Rats", _  
"ID:=", 1, _  
"T:=", "rat", _  
"Col:=", 16711680, _  
"Pat:=", 1), _  
"Layer:=", Array("N:=", "Errors", _  
"ID:=", 2, _  
"T:=", "error", _  
"Col:=", 255, _  
"Pat:=", 1, _  
"L:=", true), _
```



```
"Layer:=", Array("N:=", "Symbols", _  
"ID:=", 3, _  
"T:=", "symbol", _  
"Col:=", 8323199, _  
"Pat:=", 4), _  
"Layer:=", Array("N:=", "Assembly", _  
"ID:=", 4, _  
"T:=", "assembly", _  
"TB:=", "top", _  
"Col:=", 16711680, _  
Pat:=", 3), _  
"Layer:=", Array("N:=", "Silkscreen", _  
"ID:=", 5, _  
"T:=", "silkscreen", _  
TB:=", "top", _  
"Col:=", 8454143, _  
"Pat:=", 6), _  
SLayer:=", Array("Layer:=", Array("N:=", "Cover", _  
"ID:=", 12, _  
"T:=", "metalizedsignal", _
```

```
"Col:=", 32639, _  
Pat:=", 7), _  
"Elev:=", "mid", _  
"SubL:=", Array("Th:=", "0mm", _  
"LElev:=", "118.110236220472mil", _  
"R:=", "0mm", _  
"Mat:=", "")), _  
"SLayer:=", Array("Layer:=", Array("N:=", "Dielec3", _  
"ID:=", 11, _  
"T:=", "dielectric", _  
"Col:=", 8323199, _  
"Pat:=", 6), _  
"Elev:=", "none", _  
"SubL:=", Array("Th:=", "1mm", _  
"LElev:=", "78.740157480315mil", _  
"R:=", "0mm", _  
"Mat:=", "")), _  
"SLayer:=", Array("Layer:=", Array("N:=", "Trace2", _  
"ID:=", 10, _  
"T:=", "signal", _  
"Col:=", 8355584, _
```

```
"Pat:=", 5), _  
"Elev:=", "mid", _  
"SubL:=", Array("Th:=", "0mm", _  
"LElev:=", "78.740157480315mil", _  
"R:=", "0mm", _  
"Mat:=", "")), _  
"SLayer:=", Array("Layer:=", Array("N:=", "Dielec2", _  
"ID:=", 6, _  
"T:=", "dielectric", _  
"Col:=", 8323072, _  
"Pat:=", 4), _  
"Elev:=", "none", _  
"SubL:=", Array("Th:=", "1mm", _  
"LElev:=", "39.3700787401575mil", _  
"R:=", "0mm", _  
"Mat:=", "")), _  
"SLayer:=", Array("Layer:=", Array("N:=", "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:Lyrs",
 "Layer:=", <LayerArray>,"
 "Layer:=", <LayerArray>...,


```
"SLayer:=", <StackupLayerArray>,  
"SLayer:=", <StackupLayerArray>...),  
"ActLyr:=", <string>, // name of active layer  
"Tol:=", <ToleranceArray> // optional  
<PrimitivesInfo>, // optional  
<PinsInfo>, // optional  
<ViasInfo>, // optional  
<EdgeportsInfo>, // optional  
<ComponentPropertyInfo>,  
<ScriptInfo>, // optional, specified for scripted footprints  
Array(<ListOfComponentNames>) // Component names
```

Return Value: <string>

```
// composite name of the footprint.  
// If the name requested conflicts with the name of an existing  
// footprint, the requested name is altered to be unique.  
// The name returned reflects any change made to be unique.
```

Parameters: <FootprintName>:

```
<string> // composite name of the footprint being edited
```

<NewFootprintName>:

<string> // new [simple name](#) for the footprint

<ModifiedOnInfo>:

An integer that corresponds to the number of seconds that have elapsed since 00:00 hours, Jan 1, 1970 UTC from the system clock.

<**UnitType**>:

<string> // default length units to use if units are not specified in other
// parameters

<**LayerArray**>:

Array("N:", <string>, // layer name

"ID:", <int> ,

"T:", <LayerTypeInfo>, // layer type

"TB:", <TopBottomInfo> ,

"Col:", <int>, // optional - color

"Pat:", <int>, // optional - fill pattern

"Vis:", <bool>, // optional - are objects on layer visible

"Sel:", <bool>, // optional - are objects on layer selectable

"L:", <bool>) // optional

// are objects on layer locked (can't be edited)

<LayerTypeInfo>:

<string> // one of: signal, dielectric, metalized signal, assembly, silkscreen, soldermask, solderpaste, glue, or user

<TopBottomInfo>:

<string> // one of: top, neither, bottom, or template

<StackupLayerArray>:

```
Array(<LayerArray>,  
"Elev:=", <ElevationInfo>,  
"SubL:=", Array("Th:=", <Dimension>,  
"LElev:=", <Dimension>,  
"R:=", <Dimension>,  
"M:=", <MaterialInfo>))
```

<ElevationInfo>:

<string> // "top" - snap to top

// "mid" - snap to middle

// "bot" - snap to bottom

```
// "edit" - manual edit
```

```
// "none"
```

<Dimension>:

```
<string> // real number, may include units
```

<MaterialInfo>:

```
<string> // name of the layer material
```

<ToleranceArray>:

```
Array(<real>, // distance tolerance
```

```
<real>, // angle tolerance (radians)
```

```
<real>) // dimensionless tolerance
```

<PrimitivesInfo>:

```
Array("NAME:Prims",
```

```
// one or more of the following
```

```
<RectInfo>,
```

```
<CircleInfo>,
```

```
<ArcInfo>,
```

```
<LineInfo>,
```

```
<PolygonInfo>,  
<TextInfo>,  
<ImageInfo>)
```

<RectInfo>:

```
"Rect:=", Array(<real>, // line width  
<int>, // fill pattern  
<int>, // color  
<real>, // angle, in radians  
<real>, // x position of center  
<real>, // y position of center  
<real>, // width  
< real>) // height
```

<CircleInfo>:

```
"Circle:=", Array(<real>, // line width  
<int>, // fill pattern  
<int>, // color  
<real>, // x position of center  
<real>, // y position of center
```

< real>) // radius

<ArcInfo>:

"Arc:=", Array(<real>, // line width

<int>, // line pattern

<int>, // color

<real>, // x position of center

<real>, // y position of center

< real>, // radius

<real>, // start angle, in radians

<end>) // end angle, in radians

<LineInfo>:

"Line:=", Array(<real>, // line width

<int>, // line pattern

<int>, // color

<PointInfo>, // must specify at least 2 points

<PointInfo>...)

<PointInfo>:

<real>, // x position

<real> // y position

<PolygonInfo>:

```
"Polygon:=", Array(<real>, // line width
<int>, // fill pattern
<int>, // color
<PointInfo>, // must specify at least 3 points
<PointInfo>...)
```

<TextInfo>:

```
"Text:=", Array(<real>, // x position
<real>, // y position
<real>, // angle, in radians
<Justification>,
<bool>, // is plotter font
<string>, // font name
<int>, // color
<string>) // text string
<Justification>: <int>
// 0 : left top
// 1 : left base
```

// 2 : left bottom
// 3 : center top
// 4 : center base
// 5 : center bottom
// 6 : right top
// 7 : right base
// 8 : right bottom

<ImageInfo>:

"Image:=", Array(<RectInfo>,
<ImageData>,
<bool>) // is mirrored

<ImageData>:

<string>, // file path
<int>, // 0 : use the file path and link to it
// 1 : ignore file path and use next parameter
<string> // text data, only present if preceding int is 1

<PinsInfo>:

Array("NAME:Pins",


```
"P:=", <PinArray>,  
"P:=", <PinArray>,...)
```

<PinArray>:

```
Array("Port:=", Array("Id:=", <int>,  
"Clr:=", <real>, // optional - clearance  
"N:=", <string>), // pin name  
"Pos:=", Array("x:=", <Location>, // padstack (x,y) position  
"y:=", <Location>),  
"VRt:=", <Angle>, // optional - rotation  
"HD:=", <Size>, // optional - hole diameter  
<PadstackImplementationInfo>)
```

<Location>:

<string> specifying real number and units (may use variables)

<Angle>:

<string> specifying angle with a real number and units

<Size>:

<string> specifying size with a real number and units

<PadstackImplementationInfo>:

If another with the same implementation has already been specified:

"Ref:=", <int> // id of the other

If not:

"Ref:=", <string>, // name

"Frm:=", <int>, // id of highest layer

"To:=", <int>, // id of lowest layer

<LayerPlacementInfo>,

"Man:=", <int>, // optional, 1 if manually placed, 0 if not (default)

"Use:=", Array(<PadUseInfo>, <PadUseInfo>...) // array may be empty

<LayerPlacementInfo>:

"Lyr:=", Array("Mrg:=", <int>, // optional,

// 1 if all layers have been merged (default)

// 0 if layer are not merged

"Flp:=", <int>, // optional,

// 1 if placed bottom up

// 0 if not (default)

"Map:=", <ParentToLocalLayerInfo>)

<ParentToLocalLayerInfo>:

```
Array("U:=", <DirectionOfUniqueness>,  
"F:=", Array(<int>, <int>, ...), // forward mapping  
// -1 is not mapped  
"B:=", Array(<int>, <int>, ...)) // backward mapping  
// -1 is not mapped
```

<PadUseInfo>:

```
"Pad:=", Array("Lid:=", <int>, // layer id  
"T:=", <string>, // type : "connected", "thermal",  
// "no_pad", "not_connected",  
// or "not_connected_thermal"  
"Man:=", <int>) // optional, 1 if manually placed  
// 0 if not (default)
```

<DirectionOfUniqueness>:

```
<string> // one of "forward", "backward", or "two ways"
```

<ViasInfo>:

```
Array("NAME:Vias", <VialInfo>, <VialInfo>...)
```

<VialInfo>:

```
"V:=", Array("Id:=", <int>,  
"N:=", <string>, // name  
"Pos:=", Array("x:=", <Location>, // via (x,y) position  
"y:=", <Location>),  
"VRt:=", <Angle>, // optional - rotation  
<ImplementationInfo>
```

<EdgeportsInfo>:

```
Array("NAME:EPorts", <EdgePortArray>, <EdgePortArray>...)
```

<EdgePortArray>

```
Array("NAME:EP",  
"LP:=", Array("Id:=", <int>, // port id  
"N:=", <string>), // port name  
"Eo:=", Array(<edge description>, <edge description>,...))
```

<edgedescription> for primitive edges

"et:=", "pe", "pr:=", <id>, "ei:=", <edge#>

<id>: integer that is the primitive id

<edge#>: integer that is the edge number on the primitive

<edge description> for via edges

"et:=", "pse", "sel:=", <"via">, "layer:=", <layer id>,

"sx:=", <start X location>, "sy:=", <start Y location>, "ex:=", <end X location>, "ey:=", <end Y location>, "h:=", <arc height>,
"rad:=", <radians>

<"via">: text that is the name of the via to use

<layer id>: an integer that is the id of the layer of the pad of the via to use

<start X location>, <start Y Location>:

doubles that are the X, Y location of the start point of the edge arc

<end X location>, <end Y Location>:

doubles that are the X, Y location of the end point of the edge arc

<arc height>: double giving the height of the edge arc (0 for a straight edge)

<radians>: double giving the arc size in radians (0 for a straight edge)

<ComponentPropertyInfo>:

```
Array("NAME:CProps",  
"VariableProp:=", <VariableInfo>,  
"VariableProp:=", <VariableInfo>,  
...)
```

<VariableInfo>:

```
Array(<string>, // name  
<FlagLetters>,  
<string>, // description  
"CB:=", <string>, // optional - script for call back  
<string>) // value: number, variable, or expression
```

<ScriptInfo>:

```
Array("NAME:script",  
"language:=", <string>, // one of "javascript" or "vbscript"  
"UsesScript:=", true,  
"script:=", <string>) // contents of script
```

<ListOfComponentNames>:

<string>,<string> ...

// The list may be empty. When not empty, each string that is listed is a component

// that references the footprint to be edited. Prior to editing, a clone of the footprint is

// made, and the components that are listed are modified so that they now refer to

// the clone.

VB Example:

```
Dim nam
```

```
oFootprintManager.EditWithComps _
```

```
("Nexxim Circuit Elements\Distributed\Distributed:bendo", _
```

```
Array("NAME:bendo new name", _
```

```
"ModTime:=", 1152722165, _
```

```
"Library:=", "Nexxim Circuit Elements\Distributed\Distributed", _
```

```
"LibLocation:=", "SysLibrary", _
```

```
Array("NAME:PinDef", _
```

```
"Pin:=", Array( "n1", _
```

```
-0.00254, _
```

```
0.00254, _
```

```
0, _
```

```
"N", _  
0, _  
0, _  
false, _  
0, _  
true, _  
"", _  
false, _  
false)), _  
Array("NAME:PinDef", _  
"Pin:=", Array( "n2", _  
0.00254, _  
-0.00254, _  
1.5708, _  
"N", _  
0, _  
0, _  
false, _  
0, _  
true, _  
"", _
```



```
false, _
false)), _
Array("NAME:Graphics", _
"Polygon:=", Array( 0, 0, 12566272, 0, 0.00381, 0, .00127, 0.00127, _
0.00127, 0.00127, 0, 0.00381, 0, 0.00381, _
0.002032, 0.00127, 0.00381), _
"Line:=", Array(0, 1, 12566272, -0.00254, 0.00254, 0, .00254), _
"Line:=", Array(0, 1, 12566272, 0.00254, -0.00254, .00254, 0)), _
Array("NAME:PropDisplayMap", _
"W:=", Array(3, _
5, _
0, _
"Text:=", Array( 2.1684E-019, _
0.00504119, _
0, _
1, _
5, _
false, _
"Arial", _
0, _
```

```
"W=***") ) ) ) , _  
Array("MY_COMP")
```

Export [footprint manager]

Use: Export a footprint to a library

Command: Tools > Edit Configured Libraries > Footprints > Export to Library

Syntax: Export Array("NAME:<LibraryName>",

<FootprintName>,

<FootprintName>...),

<LibraryLocation>

Return Value: None

Parameters: <LibraryName>:

<string> // name of the library

<FootprintName>:

<string> // [composite name](#) of footprint to export

<LibraryLocation>:

<string> // location of the library in <LibraryName>

// One of "Project", "PersonalLib", or "UserLib"

VB Example:

```
oFootprintManager.Export Array("NAME:mylib", "Distributed Footprints:BPAD"), "PersonalLib"
```

GetData [footprint manager]

FootprintGetData

Use: Gets data that describes the definition.

Command: None

Syntax: GetData(<DefinitionName>)

Return Value: <DefinitionData> This is an array of data for the definition.

Parameters: <DefinitionName>:

<string> // [composite name](#) of the definition to edit

VB Example:

```
Dim footprintData
```

```
footprintData = oFootprintManager.GetData("Nexxim Circuit Elements\Distributed\Nexxim_Footprints:MCPL13_Nexx")
```

Note:

GetData allows the user to access definition information, make modifications, and then use the Edit or EditWithComps script commands to save the modified definition. Accordingly, for each type of definition, the array data returned to GetData should be the same array information that is supplied to the [Edit](#) or [EditWithComps](#) commands.

GetNames [footprint manager]

Use: Returns the names of the footprints (used and unused) in a design. The following script command, **IsUsed**, can then be used to separate used and unused footprints.

Command: None

Syntax: GetNames()

Return Value: An array of strings

Parameters: None

VB Example:

```
Dim footprintNames  
footprintNames = oFootprintManager.GetNames()
```

IsUsed [footprint manager]

Use: Used to determine if a footprint is used in the design.

Command: None

Syntax: IsUsed(<FootprintName>)

Return Value: <Boolean> // true if the specified footprint is used in the design

Parameters: <FootprintName>:

 <string>

VB Example:

```
Dim isUsed  
isUsed = oFootprintManager.IsUsed("MyFootprint")
```

Remove [footprint manager]

Use: Removes a footprint from a library

Command: Tools > Edit Configured Libraries > Footprints > Remove Footprint

Syntax: Remove <FootprintName>,

 <IsProjectFootprint>,

 <LibraryName>,

 <LibraryLocation>

Return Value: None

Parameters: <FootprintName>:

 <string> // [composite name](#) of the footprint to remove

 <IsProjectFootprint>:

 <bool>

 <LibraryName>:

 <string> // name of the library

 <LibraryLocation>:

 <string> // location of the library in <LibraryName>

 // One of "Project", "PersonalLib", or "UserLib"

VB Example:

```
oFootprintManager.Remove "BPAD", true, "Distributed Footprints", "Project"  
oFootprintManager.Remove "BPAD", false, "MyLib", "PersonalLib"
```

RemoveUnused [footprint manager]

Use: Removes footprints that are not used in the design.

Command: **Project->Remove Unused Definitions** is similar but operates slightly different and does not record script commands.

Syntax: RemoveUnused()

Return Value: <bool> True if one or more footprints are removed.

Parameters: None

VB Example:

```
Dim removedDefs  
removedDefs = oFootprintManager.RemoveUnused()
```

Note:

The order of calls to RemoveUnused is significant. As a result, removing definitions in an unordered fashion may cause other footprints in dependent definitions to be rendered unusable.

Also, the symbol and footprint of an unused component are not unusable until after the component itself is removed using the Component ManagerRemove script.

Material Manager Script Commands

The material manager provides access to materials in a project. The manager object is accessed via the definition manager.

```
Set oDefinitionManager = oProject.GetDefinitionManager()
```

```
Set oMaterialManager = oDefinitionManager.GetManager("Material")
```

The topics for this section include:

[GetData](#)

[GetNames](#)

[GetProperties](#)

[IsUsed](#)

[RemoveUnused](#)

GetData [material manager]

Use: Get material data

Command: None

Syntax: GetData <material_name>

Return Value: Array of material data

Parameters: <material_name>

Type: string

Value: Name of the project material

VB Example:

```
materialData = oMaterialMgr.GetData("MyMaterial2")
```

Python Syntax	GetData(<SimpleName>)
Python Example	<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", "per-
mittivity=", "0.123") )

oMaterialMgr.Add( Array("NAME:MyMaterial2", "CoordinateSystemType=", "Cartesian", "per-
mittivity=", "0.456") )

Dim materialNames

materialNames = oMaterialMgr.GetNames()
```

Python Syntax	GetNames()
Python Example	materialnames = oMaterialManager.GetNames()

GetProperties [material manager]

Get material properties. Differs from GetData in that only material properties available to the user are returned by GetProperties.

UI Access	NA		
Parameters	Name	Type	Description
	<material_name>	string Boolean	Name of the project material True or False. If you use False or only a single argument, then GetProperties returns only the properties that are different from the defaults.

Return Value	Array of material data

Python Syntax	GetProperties (<materialname>)
Python Example	<pre>oMaterialManager.GetProperties("Gold", True) oMaterialManager.GetProperties("vacuum")</pre>

VB Syntax	GetProperties <material_name>
VB Example	<pre>materialProps = oMaterialMgr.GetProperties("MyMaterial2")</pre>

IsUsed [material manager]

Use: Checks if a project material is in use

Command: None

Syntax: IsUsed <material_name>

Return Value: Returns 'True' if the material is in use.

Parameters: <material_name>

Type: string

Value: Name of the project material to check.

VB Example:

```
used = oMaterialMgr.IsUsed( "MyMaterial2" )
```

Python Syntax	IsUsed(<ComboName>)
Python Example	<code>isused = oMaterialManager.IsUsed("mylib:mymaterial")</code>

RemoveUnused [material manager]

Use: Remove all unused materials from the project.

Command: None

Syntax: RemoveUnused

Return Value: None

Parameters: None

VB Example:

```
oMaterialMgr.RemoveUnused()
```

Python Syntax	RemoveUnused()
Python Example	<code>thereWereExtras = oMaterialManager.RemoveUnused()</code>

Model Manager Script Commands

The model manager provides access to models in a project. The manager object is accessed via the definition manager.

```
Set oDefinitionManager = oProject.GetDefinitionManager()
```

```
Set oModelManager = oDefinitionManager.GetManager("Model")
```

The model manager script commands are listed below:

[Add](#)

[ConvertToDynamic](#)

[ConvertToParametric](#)

[Edit](#)

[EditWithComps](#)

[Export](#)

[GetData](#)

[GetNames](#)

[IsUsed](#)

[Remove](#)

[RemoveUnused](#)

Add [model manager]

Use: Add a model

Command: Tools > Edit Configured Libraries > Models > Add Model

Syntax: Add Array("NAME:<modelName>",
 "ModTime:=", <ModifiedTimeInfo>,
 "Library:=", "", // Library name
 "LibLocation:=", "Project", // Project Location
 <PinDefInfo>,
 <PinDefInfo>,... // optional, to define pins
 <GraphicsDataInfo>, // optional, to define graphics
 <PropDisplayMapInfo>)) // optional, to define property displays

Return Value: <string>

// [composite name](#) of the model.
// If the name requested conflicts with the name of an existing
// model, the requested name is altered to be unique.
// The name returned reflects any change made to be unique.

Parameters: <modelName>:

<string> // [simple name](#) of the model being added

<ModifiedOnInfo>:

An integer that corresponds to the number of seconds that have elapsed
since 00:00 hours, Jan 1, 1970 UTC from the system clock.

<PinDefInfo>:

Array("NAME:PinDef",
"Pin:=", Array (<string>, // pin name
<real>, // x location
<real>, // y location
<real>, // angle in radians

```
<PinType>,  
<real>, // line width  
<real>, // line length  
<bool>, // mirrored  
<int>, // color  
<bool>, // true if visible, false if not  
<string>, // hidden net name  
<OptionalPinInfo>, // optional info  
<PropDisplayMapInfo>)) // optional
```

```
<PinType>:  
<string> // "N" : normal pin  
// "I" : input pin  
// "O" : output pin
```

```
<OptionalPinInfo>:  
// Specify both or neither  
<bool>, // true if name is to be shown  
<bool>, // true if number is to be shown
```

<PropDisplayMapInfo>:

```
Array("NAME:PropDisplayMap",  
<PropDisplayInfo>,  
<PropDisplayInfo>,...)
```

<PropDisplayInfo>:

```
<NameString>, Array(<DisplayTypeInfo>,  
<DisplayLocationInfo>,  
<int>, // optional, level number  
<TextInfo>)  
<NameString>:  
<string> // PropertyName:=, where PropertyName is the name of  
// the property to be displayed
```

<DisplayTypeInfo>:

```
<int> // 0 : No display  
// 1 : Display name only  
// 2 : Display value only  
// 3 : Display both name and value  
// 4: Display evaluated value only
```

// 5: Display both name and evaluated value

<DisplayLocationInfo>:

<int> // 0 : Left

// 1 : Top

// 2 : Right

// 3 : Bottom

// 4 : Center

// 5 : Custom placement

<GraphicsDataInfo>:

Array("NAME:Graphics",

// one or more of the following

<RectInfo>,

<CircleInfo>,

<ArcInfo>,

<LineInfo>,

<PolygonInfo>,

<TextInfo>,

<ImageInfo>)

<RectInfo>:

```
"Rect:=", Array(<real>, // line width  
<int>, // fill pattern  
<int>, // color  
<real>, // angle, in radians  
<real>, // x position of center  
<real>, // y position of center  
<real>, // width  
< real>) // height
```

<CircleInfo>:

```
"Circle:=", Array(<real>, // line width  
<int>, // fill pattern  
<int>, // color  
<real>, // x position of center  
<real>, // y position of center  
< real>) // radius
```

<ArcInfo>:

```
"Arc:=", Array(<real>, // line width
```

<int>, // line pattern

<int>, // color

<real>, // x position of center

<real>, // y position of center

< real>, // radius

<real>, // start angle, in radians

<end>) // end angle, in radians

<LineInfo>:

"Line:=", Array(<real>, // line width

<int>, // line pattern

<int>, // color

<PointInfo>, // must specify at least 2 points

<PointInfo>...)

<PointInfo>:

<real>, // x position

<real> // y position

<PolygonInfo>:

"Polygon:=", Array(<real>, // line width

<int>, // fill pattern

<int>, // color

<PointInfo>, // must specify at least 3 points

<PointInfo>...)

<TextInfo>:

"Text:=", Array(<real>, // x position

<real>, // y position

<real>, // angle, in radians

<Justification>,

<bool>, // is plotter font

<string>, // font name

<int>, // color

<string>) // text string

<Justification>:

<int> // 0 : left top

// 1 : left base

// 2 : left bottom

// 3 : center top

// 4 : center base

// 5 : center bottom

// 6 : right top

// 7 : right base

// 8 : right bottom

<ImageInfo>:

"Image:=", Array(<RectInfo>,

<ImageData>,

<bool>) // is mirrored

<ImageData>:

<string>, // file path

<int>, // 0 : use the file path and link to it

// 1 : ignore file path and use next parameter

<string> // text data, only present if preceding int is 1

VB Example:

```
oModelManager.Add Array("NAME:MyModel", _
```

```
"ModTime:=", 1070989137, _
```

```
"Library:=", "", _
```

```
"LibLocation:=", "Project", _
```

```
Array("NAME:PinDef", _
```

```
"Pin:=", Array ("newpin0", _  
0.00254, _  
0, _  
0, _  
"N", _  
0, _  
0.00254, _  
false, _  
0, _  
true, _  
"", _  
false, _  
false)), _  
Array("NAME:PinDef", _  
"Pin:=", Array ("newpin1", _  
-0.00254, _  
0, _  
3.14159265358979, _  
"N", _  
0, _
```

```
0.00254, _  
false, _  
0, _  
true, _  
"", _  
false, _  
false)),  
Array("NAME:Graphics", _  
"Rect:=", Array(0, _  
0, _  
12566272, _  
0, _  
4.33680868994202e-019, _  
-0.000635, _  
0.00508, _  
0.002794), _  
"Circle:=", Array(0, _  
0, _  
12566272, _  
0.000127, _  
-0.000635, _
```

```
0.000635)))
```

ConvertToDynamic

Use: Build a new dynamic model based on an existing parametric model.

Command: Right-click on a model under Definitions/Models in the Project Tree and choose ConvertToDynamic.

Syntax: ConvertToDynamic(defName, newname)

Return Value: <newname> // Name of the new model added

Parameters: <defName> // Model that is the base for the new conversion

VB Example:

```
Dim newname
```

```
newname = oModelManager.ConvertToDynamic([in] BSTR defName, [out, retval] BSTR* newName);
```

ConvertToParametric

Use: Build a new parametric model based on an existing dynamic model.

Command: Right-click on a model under Definitions/Models in the Project Tree and choose ConvertToParametric.

Syntax: ConvertToParametric(defName, newname)

Return Value: <newname> // Name of the new model added

Parameters: <defName> // Model that is the base for the new conversion

VB Example: Dim newname

```
newname = oModelManager.ConvertToParametric([in] BSTR defName, [out, retval] BSTR* newName);
```

Edit [deprecated]

Deprecated command — please use [EditWithComps](#).

EditWithComps [model manager]

Use: Edit an existing model.

Command: None

Syntax: EditWithComps <ModelName>,
 Array("NAME:<NewModelName>",
 "ModTime:=", <ModifiedTimeInfo>,
 "Library:=", <string>, // Library name
 "LibLocation:=", <string>, // Project Location
 <PinDefInfo>,
 <PinDefInfo>,... // optional, to define pins
 <GraphicsDataInfo>, // optional, to define graphics
 <PropDisplayMapInfo>), // optional, to define property displays
 Array(<ListOfComponentNames>) // Component names

Return Value: <string>

 // [composite name](#) of the model.
 // If the name requested conflicts with the name of an existing
 // model, the requested name is altered to be unique.
 // The name returned reflects any change made to be unique.

Parameters: <ModelName>:

<string> // [composite name](#) of the model being edited

<NewModelName>:

<string> // new [simple name](#) for the model

<ModifiedOnInfo>:

An integer that corresponds to the number of seconds that have elapsed since 00:00 hours, Jan 1, 1970 UTC from the system clock.

<PinDefInfo>:

Array("NAME:PinDef",

"Pin:=", Array (<string>, // pin name

<real>, // x location

<real>, // y location

<real>, // angle in radians

<PinType>,

<real>, // line width

<real>, // line length

<bool>, // mirrored

<int>, // color

```
<bool>, // true if visible, false if not  
<string>, // hidden net name  
<OptionalPinInfo>, // optional info  
<PropDisplayMapInfo>)) // optional
```

```
<PinType>:  
<string> // "N" : normal pin  
// "I" : input pin  
// "O" : output pin
```

```
<OptionalPinInfo>:  
// Specify both or neither  
<bool>, // true if name is to be shown  
<bool>, // true if number is to be shown
```

```
<PropDisplayMapInfo>:  
Array("NAME:PropDisplayMap",  
<PropDisplayInfo>,  
<PropDisplayInfo>,...)
```

```
<PropDisplayInfo>:
```

```
<NameString>, Array(<DisplayTypeInfo>,  
<DisplayLocationInfo>,  
<int>, // optional, level number  
<TextInfo>)
```

```
<NameString>:  
<string> // PropertyName:=, where PropertyName is the name of  
// the property to be displayed
```

```
<DisplayTypeInfo>:  
<int> // 0 : No display  
// 1 : Display name only  
// 2 : Display value only  
// 3 : Display both name and value  
// 4: Display evaluated value only  
// 5: Display both name and evaluated value
```

```
<DisplayLocationInfo>:  
<int> // 0 : Left  
// 1 : Top
```

```
// 2 : Right  
// 3 : Bottom  
// 4 : Center  
// 5 : Custom placement
```

```
<GraphicsDataInfo>:  
Array("NAME:Graphics",  
// one or more of the following  
<RectInfo>,  
<CircleInfo>,  
<ArcInfo>,  
<LineInfo>,  
<PolygonInfo>,  
<TextInfo>,  
<ImageInfo>)
```

```
<RectInfo>:  
"Rect:=", Array(<real>, // line width  
<int>, // fill pattern  
<int>, // color  
<real>, // angle, in radians
```

<real>, // x position of center

<real>, // y position of center

<real>, // width

< real>) // height

<CircleInfo>:

"Circle:=", Array(<real>, // line width

<int>, // fill pattern

<int>, // color

<real>, // x position of center

<real>, // y position of center

< real>) // radius

<ArcInfo>:

"Arc:=", Array(<real>, // line width

<int>, // line pattern

<int>, // color

<real>, // x position of center

<real>, // y position of center

< real>, // radius

<real>, // start angle, in radians

<end>) // end angle, in radians

<LineInfo>:

"Line:=", Array(<real>, // line width

<int>, // line pattern

<int>, // color

<PointInfo>, // must specify at least 2 points

<PointInfo>...)

<PointInfo>:

<real>, // x position

<real> // y position

<PolygonInfo>:

"Polygon:=", Array(<real>, // line width

<int>, // fill pattern

<int>, // color

<PointInfo>, // must specify at least 3 points

<PointInfo>...)

<TextInfo>:

```
"Text:=", Array(<real>, // x position  
<real>, // y position  
<real>, // angle, in radians  
<Justification>,  
<bool>, // is plotter font  
<string>, // font name  
<int>, // color  
<string>) // text string
```

```
<Justification>:  
<int> // 0 : left top  
// 1 : left base  
// 2 : left bottom  
// 3 : center top  
// 4 : center base  
// 5 : center bottom  
// 6 : right top  
// 7 : right base  
// 8 : right bottom
```

<ImageInfo>:

"Image:=", Array(<RectInfo>,

<ImageData>,

<bool>) // is mirrored

<ImageData>:

<string>, // file path

<int>, // 0 : use the file path and link to it

// 1 : ignore file path and use next parameter

<string> // text data, only present if preceding int is 1

<ListOfComponentNames>:

<string>,<string> ...

// The list may be empty. When not empty, each string that is listed is a component

// that references the model to be edited. Prior to editing, a clone of the model is

// made, and the components that are listed are modified so that they now refer to

// the clone.

VB Example:

Dim nam

oModelManager.EditWithComps_


```
("Nexxim Circuit Elements\Distributed\Distributed:bendo", _  
Array("NAME:bendo new name", _  
"ModTime:=", 1152722165, _  
"Library:=", "Nexxim Circuit Elements\Distributed\Distributed", _  
"LibLocation:=", "SysLibrary", _  
Array("NAME:PinDef", _  
"Pin:=", Array( "n1", _  
-0.00254, _  
0.00254, _  
0, _  
"N", _  
0, _  
0, _  
false, _  
0, _  
true, _  
"", _  
false, _  
false)), _  
Array("NAME:PinDef", _
```

```
"Pin:=", Array( "n2", _  
0.00254, _  
-0.00254, _  
1.5708, _  
"N", _  
0, _  
0, _  
false, _  
0, _  
true, _  
"", _  
false, _  
false)), _  
Array("NAME:Graphics", _  
"Polygon:=", Array( 0, 0, 12566272, 0, 0.00381, 0, .00127, 0.00127, _  
0.00127, 0.00127, 0, 0.00381, 0, 0.00381, _  
0.002032, 0.00127, 0.00381), _  
"Line:=", Array(0, 1, 12566272, -0.00254, 0.00254, 0, .00254), _  
"Line:=", Array(0, 1, 12566272, 0.00254, -0.00254, .00254, 0)), _  
Array("NAME:PropDisplayMap", _  
"W:=", Array(3, _
```

```
5, _  
0, _  
"Text:=", Array( 2.1684E-019, _  
0.00504119, _  
0, _  
1, _  
5, _  
false, _  
"Arial", _  
0, _  
"W=***")))), _  
rray("MY_COMP")
```

Export [model manager]

Use: Exports model(s) to a library

Command: Tools > Edit Configured Libraries > Models > Export to Library

Syntax: Export Array("NAME:<LibraryName>",

<ModelName>,

<ModelName>...),

<LibraryLocation>

Return Value: None

```
Parameters: <LibraryName>:  
    <string> // name of the library  
  
    <ModelName>:  
    <string> // composite name of model to export  
  
    <LibraryLocation>:  
    <string> // location of the library in <LibraryName>  
    // One of "Project", "PersonalLib", or "UserLib"
```

```
VB Example:  
  
oModelManager.Export _  
Array("NAME Nexxim Circuit Elements\Distributed\Distributed:bendo:mylib", _ "myModel"), "Per-  
sonalLib"
```

Python Syntax	Export (["NAME:<LibraryName>", <ComboName>, <ComboName>...], <LibraryLocation>)
Python Example	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.

Python Syntax	GetData(<ComboName>)
Python Example	<pre>dataArray = oModelManager.GetData ("mylib:mymodel")</pre>

GetNames [model manager]

Use: Returns the names of the models (used and unused) in a design. The following script command, **IsUsed**, can then be used to separate used and unused models.

Command: None

Syntax: GetNames()

Return Value: An array of strings

Parameters: None

VB Example:

```
Dim modelNames
```

```
modelNames = oModelManager.GetNames()
```

Python Syntax	GetNames()
Python Example	modelnames = oModelManager.GetNames()

IsUsed [model manager]

Use: Used to determine if a model is used in the design.

Command: None

Syntax: IsUsed(<ModelName>)

Return Value: <Boolean> // true if the specified model is used in the design

Parameters: <ModelName>:

<string>

VB Example:

```
Dim isUsed
```

```
isUsed = oModelManager.IsUsed("MyModel")
```

Python Syntax	IsUsed(<ComboName>)

Python Example	<pre>isused = oModelManager.IsUsed ("mylib:mymodel")</pre>

Remove [model manager]

Use: Removes a model from a library

Command: Tools > Edit Configured Libraries > Models > Remove Model

Syntax: Remove <ModelName>,

 <IsProjectModel>,

 <LibraryName>,

 <LibraryLocation>

Return Value: None

Parameters: <ModelName>:

 <string> // [composite name](#) of the model to remove

<IsProjectModel>:

 <bool>

<LibraryName>:

 <string> // name of the library

<LibraryLocation>:

 <string> // location of the library in <LibraryName>

// One of "Project", "PersonalLib", or "UserLib"

VB Example:

```
oModelManager.Remove "Nexxim Circuit Elements\Distributed\Distributed:bendo", true, "Project"
```

Python Syntax	Remove (<ModelName>, <IsLocal>, <LibraryName>, <LibraryLocation>)
Python Example	<pre>oModelManager.Export(["NAME:mylib", "model1", "model2"])</pre>

RemoveUnused [model manager]

Use: Removes models that are not used in the design.

Command: **Project->Remove Unused Definitions** is similar but operates slightly different and does not record script commands.

Syntax: RemoveUnused()

Return Value: <bool> True if one or more models are removed.

Parameters: None

VB Example:

```
Dim removedDefs  
removedDefs = oModelManager.RemoveUnused()
```


Note:

The order of calls to RemoveUnused is significant. As a result, removing definitions in an unordered fashion may cause other models in dependent definitions to be rendered unusable.

Also, the model and footprint of an unused component are not unusable until after the component itself is removed using the Component Manager Remove script.

Python Syntax	RemoveUnused()
Python Example	<code>thereWereExtras = oModelManager.RemoveUnused()</code>

Network Data Explorer Manager Script Commands

The network data Explorer (NDE) Manager provides access to certain NDE data.

```
Set oDefinitionManager = oProject.GetDefinitionManager()
```

```
Set oNdExplorerManager = oDefinitionManager.GetManager("NdExplorer")
```

For NDE scripts accessed via the ndExplorer tool, see: [Network Data Explorer Script Commands](#).

The topics for this section include:

[ExportFullWaveSpice](#)

[ExportNetworkData](#)

[ExportNMFData](#)

ExportFullWaveSpice

Use: Export FullWaveSpice data in a format of your choice.

Command: File > Export MacroModel > Broadband (SYZ, FWS....)

Syntax: ExportFullWaveSpice

```
"DesignName", // Design name. Can be left blank, if loading solution from a file.  
true/false, // true - solution loaded from file, false- loaded from design  
"Name", // If loading from design this is the solution name, else this is the  
           // full path of the file from which the solution is loaded  
"variation", // Pick a particular variation. Leave blank if no variation.  
Array("NAME:Frequencies"), // Optional; if none defined all frequencies are used  
Array("NAME:SpiceData", // Spice export options object  
"SpiceType:=", "SSS", // SpiceType can be "PSpice", "HSpice", "Spectre", "SSS",  
           // "Simplorer", "TouchStone1.0", "TouchStone2.0"  
"EnforcePassivity:=", false, // Enforce Passivity true/false  
"EnforceCausality:=", false, // Enforce Causality true/false  
"UseCommonGround:=", false, // Use common ground true/false  
"FittingError:=", 0.5, // Fitting error  
"MaxPoles:=", 400, // Maximum Order  
"PassivityType:=", "ConvexOptimization", // Passivity Type can be "ConvexOptimization",  
           // "PassivityByPerturbation", or "IteratedFittingOfPV"  
"ColumnFittingType:=", "Column", // Column FittingType can be "Column", "Entry", "Matrix"
```

```
"SSFittingType:=", "TWA", // SS Fitting Type can be "TWA", "IterativeRational"
"RelativeErrorToleranc:=", false, // Relative error tolerance true/false
"TouchstoneFormat:=", "MA", // Touchstone Format "MA", "RI", "DB"
"TouchstoneUnits:=", "Hz", // Touchstone Units "Hz", "KHz", "MHz", "MHz"
"TouchStonePrecision:=", 8, // Touchstone precision
"ExportDirectory:=", "C:/Examples/LNA/", // Directory to export to
"ExportSpiceFileName:=", "Linckt_HBTest_2.sss", // Spice export file
"FullwaveSpiceFileName:=", "Linckt_HBTest.sss", // FWS file
"CreateNPortModel:=", true // Create a model based on the exported file true/false
)
```

ExportNetworkData

Exports matrix solution data to a file.

UI Access	N/A		
Parameters	Name	Type	Description
	<DesignVariationKey>	String	Design variation key. Pass empty string for the current nominal variation.
	<SolnSelectionArray>	Array	Array of selected solutions. Array(<SolnSelector>, <SolnSelector>, ...) If more than one array entry, this indicates a combined Interpolating sweep.
	<SolnSelector>	String	Solution setup name and solution name, separated by a colon.

<FileFormat>	Integer	File format value. 2 : Tab delimited spreadsheet format (.tab) 3 : Touchstone (.sNp) 4 : CitiFile (.cit) 7 : Matlab (.m) 8 : Terminal Z0 spreadsheet
<OutFile>	String	Full path to the file to write out.
<FreqsArray>	Array	The frequencies to export. The <FreqsArray> argument contains a vector (e.g. "1GHz", "2GHz", ...) to use, or "all". To export all frequencies, use Array("all"). If no frequencies are specified, all frequencies are used.
<DoRenorm>	Boolean	Specifies whether to renormalize the data before export.
<RenormImped>	Double	Real impedance value in ohms, for renormalization. Required in syntax, but ignored if DoRenorm is false.
<DataType>	String	Optional. Type: "S", "Y", or "Z". The matrix to export.
<pass>	Integer	Optional. The pass to export. This is ignored if the sourceName is a frequency sweep. Leaving out this value or specifying -1 gets all passes.
<ComplexFormat>	Integer	Optional. Type: "0", "1", or "2" The format to use for the exported data. 0 = Magnitude/Phase. 1= Real/Imaginary. 2= db/Phase.
<Precision>	Integer	Optional. Touchstone number of digits precision. Default if not specified is 15.
<UseExportFreqs>	Boolean	Specifies whether to use export frequencies.
<IncludeGammaComments>	Boolean	Specifies whether to include Gamma and Impedance comments.

	<i><SupportNonStdExport></i>	Boolean	Specifies whether to support non-standard Touchstone extensions for mixed reference impedance.
Return Value	None.		

Python Syntax	<code>ExportNetworkData(<DesignVariationKey>, <SolnSelectionArray>, <SolnSelector>, <FileFormat>, <OutFile>, <FreqsArray>, <DoRenorm>, <RenormImped>, [Optional <DataType>], [Optional <pass>], [Optional <ComplexFormat>], [Optional <Precision>], [Optional <UseExportFreqs>], [Optional <IncludeGammaComments>], [Optional <SupportNonStdExport>])</code>		
Python Example			

VB Syntax	<code>ExportNetworkData <DesignVariationKey>, <SolnSelectionArray>, <SolnSelector>, <FileFormat>, <OutFile>, <FreqsArray>, <DoRenorm>, <RenormImped>, [Optional <DataType>], [Optional <pass>], [Optional <ComplexFormat>], [Optional <Precision>], [Optional <UseExportFreqs>], [Optional <IncludeGammaComments>], [Optional <SupportNonStdExport>]</code>		
VB Example	<pre>oModule.ExportNetworkData "width='2in'", _ Array("Setup1:Sweep1"), 2, "c:\mydir\out.tab", _ Array("all"), false, 0 oModule.ExportNetworkData "width='2in'", _ Array("Setup1:Sweep1", "Setup1:Sweep2"), 3, _ "c:\mydir\out.s2p", Array(1.0e9, 1.5e9, 2.0e9), _ true, 50.0</pre>		

ExportNMFData

Add [padstack manager]

Use: Add a padstack

Command: Tools > Edit Configured Libraries > Padstacks > Add Padstack

Syntax: Add Array("NAME:<PadstackName>",
 "ModTime:=", <ModifiedOnInfo>,
 "Library:=", "", // name of the library
 "LibLocation:=", "Project", // location of the named library
 Array("NAME:psd",
 "nam:= ", <PadstackName>,
 "lib:=", "", // name of the library
 "mat:=", "", // hole plating material
 "plt:=", "0", // percent of hole's radius filled by plating
 Array("NAME:pds",
 <LayerGeometryArray>,
 <LayerGeometryArray....>,
 "hle:=", <PadInfo>
 "hRg:=", <HoleRange>,
 "sbsh:=", <SolderballShape>,
 "sbpl:=", <SolderballPlacement>,
 "sbr:=", <string>, // solderball diameter, real with units

"sb2:=", <string>, // solderball mid diameter, real with units

"sbn:=", <string>), // name of solderball material

"ppl:=", <PadPortLayerArray>)

Return Value: [simple name](#) of the added padstack

// If the name requested conflicts with the name of an existing

// padstack, the requested name is altered to be unique.

// The name returned reflects any change made to be unique.

Parameters: <PadstackName>:

<string> // [simple name](#) of padstack to create

<ModifiedOnInfo>:

An integer that corresponds to the number of seconds that have elapsed
since 00:00 hours, Jan 1, 1970 UTC from the system clock.

<LayerGeometryArray>:

Array("Name:lgm",

"lay:=", <string>, // definition layer name

"id:=", <int>, // definition layer id

"pad:=", <PadInfo>, // pad

"ant:=", <PadInfo>, // antipad
"thm:=", <PadInfo>, // thermal pad
"X:=", <string>, // pad x connection, real with units
"Y:=", <string>, // pad y connection, real with units
"dir:=", <DirectionString> // pad connection direction

<PadInfo>:

Array("shp:=", <PadShape>,
"Szs:=", <DimensionArray>,
"X:=", <string>, // x offset, real with units
"Y:=", <string>, // y offset, real with units
"R:=", <string>) // rotation, real with units

<PadShape>:

<string> one of these choices

"No" // no pad

"Cir" // Circle

"Sq" // Square

"Rct" // Rectangle

"Ov" // Oval

"Blit" // 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....>),  
"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

"Blit" // Bullet

"Ply" // Polygons

"R45" // Round 45 thermal

"R90" // Round 90 thermal

"S45" // Square 45 thermal

"S90" // Square 90 thermal

<DimensionArray>:

Array(<string>, ...) // each string is a real with units for one of the
// dimensions of the shape

<DirectionString>:

<string> one of these choices

"No" // no direction

"Any" // any direction

"0" // 0 degrees

"45" // 45 degrees

"90" // 90 degrees

"135" // 135 degrees

"180" // 180 degrees

"225" // 225 degrees

"270" // 270 degrees

"315" // 315 degrees

<HoleRange>:

<string> one of these choices

"Thr" // through all layout layers

"Beg" // from upper pad layer to lowest layout layer

"End" // from upper layout layer to lowest pad layer

"UTL" // from upper pad layer to lowest pad layer

<SolderballShape>:

<string> one of these choices

"None" // no solderball

"Cyl" // cylinder solderball

"Sph" // spheroid solderball

<SolderballPlacement>:

<string> one of these choices

"abv" // above padstack

"blw" // below padstack

<PadPortLayerArray>:

Array(<int>, <int>,....) where each int is a layer id

VB Example:

```
oPadstackManager.Edit "Circle - through1", Array("NAME:Circle - through1", "ModTime:=", _  
1235765635, "Library:=", "", "LibLocation:=", "Project", Array("NAME:psd", "nam:=", _
```

```

"Circle - through1", "lib:=", "", "mat:=", "", "plt:=", "0", Array("NAME:pds", Array("NAME:lgm",
"lay:=", _
"Top Signal", "id:=", 0, "pad:=", Array("shp:=", "Cir", "Szs:=", Array("2.5mm"), "X:=", _
"0mm", "Y:=", "0mm", "R:=", "0deg"), "ant:=", Array("shp:=", "Cir", "Szs:=", Array( _
"3.5mm"), "X:=", "0mm", "Y:=", "0mm", "R:=", "0"), "thm:=", Array("shp:=", "No", "Szs:=", Array
(), "X:=", _
"0mm", "Y:=", "0mm", "R:=", "0"), "X:=", "0mm", "Y:=", "0mm", "dir:=", "Any"), Array("NAME:lgm",
"lay:=", _
"SignalA", "id:=", 1, "pad:=", Array("shp:=", "Cir", "Szs:=", Array("2mm"), "X:=", _
"0mm", "Y:=", "0mm", "R:=", "0deg"), "ant:=", Array("shp:=", "Cir", "Szs:=", Array( _
"3mm"), "X:=", "0mm", "Y:=", "0mm", "R:=", "0"), "thm:=", Array("shp:=", "No", "Szs:=", Array(),
"X:=", _
"0mm", "Y:=", "0mm", "R:=", "0"), "X:=", "0mm", "Y:=", "0mm", "dir:=", "Any"), Array("NAME:lgm",
"lay:=", _
"SignalB", "id:=", 2, "pad:=", Array("shp:=", "Cir", "Szs:=", Array("2mm"), "X:=", _
"0mm", "Y:=", "0mm", "R:=", "0deg"), "ant:=", Array("shp:=", "Cir", "Szs:=", Array( _
"3mm"), "X:=", "0mm", "Y:=", "0mm", "R:=", "0deg"), "thm:=", Array("shp:=", "No", "Szs:=", Array
(), "X:=", _
"0mm", "Y:=", "0mm", "R:=", "0"), "X:=", "0mm", "Y:=", "0mm", "dir:=", "Any"), Array("NAME:lgm",
"lay:=", _
"Ground", "id:=", 3, "pad:=", Array("shp:=", "No", "Szs:=", Array(), "X:=", _
"0mm", "Y:=", "0mm", "R:=", "0deg"), "ant:=", Array("shp:=", "No", "Szs:=", Array(), "X:=", _
"0mm", "Y:=", "0mm", "R:=", "0"), "thm:=", Array("shp:=", "R90", "Szs:=", Array( _
"3mm", "0.75mm", "1mm"), "X:=", "0mm", "Y:=", "0mm", "R:=", "0"), "X:=", "0mm", "Y:=", _

```

```
"0mm", "dir:=", "Any"), Array("NAME:lgm", "lay:=", "Bottom signal", "id:=", 5, "pad:=", Array
("shp:=", _
"Cir", "Szs:=", Array("1mm"), "X:=", "0mm", "Y:=", "0mm", "R:=", "0deg"), "ant:=", Array("shp:=",
_
"Cir", "Szs:=", Array("2mm"), "X:=", "0mm", "Y:=", "0mm", "R:=", "0deg"), "thm:=", Array("shp:=",
_
"No", "Szs:=", Array(), "X:=", "0mm", "Y:=", "0mm", "R:=", "0"), "X:=", "0mm", "Y:=", _
"0mm", "dir:=", "Any")), "hle:=", Array("shp:=", "Cir", "Szs:=", Array("1.5mm"), "X:=", _
"0mm", "Y:=", "0mm", "R:=", "0deg"), "hRg:=", "End", "sbsh:=", "Sph", "sbpl:=", _
"abv", "sbr:=", "750um", "sb2:=", "1200um", "1200um", "sbn:=", "solder"), "ppl:=", Array( _
0, 1, 2, 3, 5))
```

EditWithComps [padstack manager]

Use: Edit an existing padstack.

Command: None

Syntax: EditWithComps <PadstackName>,

```
    Array("NAME:<NewPadstackName>",
    "ModTime:=", <ModifiedOnInfo>,
    "Library:=", "", // name of the library
    "LibLocation:=", "Project", // location of the named library
    Array("NAME:psd",
    "nam:=", <PadstackName>,
```

```
"lib:=", "", // name of the library
"mat:=", "", // hole plating material
"plt:=", "0", // percent of hole's radius filled by plating
Array("NAME:pds",
<LayerGeometryArray>,
<LayerGeometryArray....>,
"hle:=", <PadInfo>
"hRg:=", <HoleRange>,
"sbsh:=", <SolderballShape>,
"sbpl:=", <SolderballPlacement>,
"sbr:=", <string>, // solderball diameter, real with units
"sb2:=", <string>, // solderball mid diameter, real with units
"sbn:=", <string>), // name of solderball material
"ppl:=", <PadPortLayerArray>,
Array(<ListOfComponentNames>) // Component names
```

Return Value: <string>

```
// composite name of the padstack.
// If the name requested conflicts with the name of an existing
// padstack, the requested name is altered to be unique.
// The name returned reflects any change made to be unique.
```

Parameters: <PadstackName>:

<string> // [composite name](#) of the padstack being edited

<NewPadstackName>:

<string> // new [simple name](#) for the padstack

<ModifiedOnInfo>:

An integer that corresponds to the number of seconds that have elapsed since 00:00 hours, Jan 1, 1970 UTC from the system clock.

<LayerGeometryArray>:

Array("Name:lgm",

"lay:=", <string>, // definition layer name

"id:=", <int>, // definition layer id

"pad:=", <PadInfo>, // pad

"ant:=", <PadInfo>, // antipad

"thm:=", <PadInfo>, // thermal pad

"X:=", <string>, // pad x connection, real with units

"Y:=", <string>, // pad y connection, real with units

"dir:=", <DirectionString>) // pad connection direction

<PadInfo>:

Array("shp:=", <PadShape>,

"Szs:=", <DimensionArray>,

"X:=", <string>, // x offset, real with units

"Y:=", <string>, // y offset, real with units

"R:=", <string>) // rotation, real with units

<PadShape>:

<string> one of these choices

"No" // no pad

"Cir" // Circle

"Sq" // Square

"Rct" // Rectangle

"Ov" // Oval

"Blt" // Bullet

"Ply" // Polygons

"R45" // Round 45 thermal

"R90" // Round 90 thermal

"S45" // Square 45 thermal

"S90" // Square 90 thermal

<DimensionArray>:

Array(<string>, ...) // each string is a real with units for one of the

// dimensions of the shape

<DirectionString>:

<string> one of these choices

"No" // no direction

"Any" // any direction

"0" // 0 degrees

"45" // 45 degrees

"90" // 90 degrees

"135" // 135 degrees

"180" // 180 degrees

"225" // 225 degrees

"270" // 270 degrees

"315" // 315 degrees

<HoleRange>:

<string> one of these choices

"Thr" // through all layout layers

"Beg" // from upper pad layer to lowest layout layer

"End" // from upper layout layer to lowest pad layer

"UTL" // from upper pad layer to lowest pad layer

<SolderballShape>:

<string> one of these choices

"None" // no solderball

"Cyl" // cylinder solderball

"Sph" // spheroid solderball

<SolderballPlacement>:

<string> one of these choices

"abv" // above padstack

"blw" // below padstack

<PadPortLayerArray>:

Array(<int>, <int>,....) where each int is a layer id

<ListOfComponentNames>:

<string>,<string> ...

// The list may be empty. When not empty, each string that is listed is a component


```
// that references the padstack to be edited. Prior to editing, a clone of the padstack is  
// made, and the components that are listed are modified so that they now refer to  
// the clone.
```

VB Example:

```
oPadstackManager.EditWithComps "Circle - through1", Array("NAME:Circle - through1", "ModTime:=",  
_  
1235765635, "Library:=", "", "LibLocation:=", "Project", Array("NAME:psd", "nam:=", _  
"Circle - through1", "lib:=", "", "mat:=", "", "plt:=", "0", Array("NAME:pds", Array("NAME:lgm",  
"lay:=", _  
"Top Signal", "id:=", 0, "pad:=", Array("shp:=", "Cir", "Szs:=", Array("2.5mm"), "X:=", _  
"0mm", "Y:=", "0mm", "R:=", "0deg"), "ant:=", Array("shp:=", "Cir", "Szs:=", Array(_  
"3.5mm"), "X:=", "0mm", "Y:=", "0mm", "R:=", "0"), "thm:=", Array("shp:=", "No", "Szs:=", Array  
( ), "X:=", _  
"0mm", "Y:=", "0mm", "R:=", "0"), "X:=", "0mm", "Y:=", "0mm", "dir:=", "Any"), Array("NAME:lgm",  
"lay:=", _  
"SignalA", "id:=", 1, "pad:=", Array("shp:=", "Cir", "Szs:=", Array("2mm"), "X:=", _  
"0mm", "Y:=", "0mm", "R:=", "0deg"), "ant:=", Array("shp:=", "Cir", "Szs:=", Array(_  
"3mm"), "X:=", "0mm", "Y:=", "0mm", "R:=", "0"), "thm:=", Array("shp:=", "No", "Szs:=", Array(),  
"X:=", _  
"0mm", "Y:=", "0mm", "R:=", "0"), "X:=", "0mm", "Y:=", "0mm", "dir:=", "Any"), Array("NAME:lgm",  
"lay:=", _
```

```

"SignalB", "id:=", 2, "pad:=", Array("shp:=", "Cir", "Szs:=", Array("2mm"), "X:=", _
"0mm", "Y:=", "0mm", "R:=", "0deg"), "ant:=", Array("shp:=", "Cir", "Szs:=", Array( _
"3mm"), "X:=", "0mm", "Y:=", "0mm", "R:=", "0deg"), "thm:=", Array("shp:=", "No", "Szs:=", Array
(), "X:=", _
"0mm", "Y:=", "0mm", "R:=", "0"), "X:=", "0mm", "Y:=", "0mm", "dir:=", "Any"), Array("NAME:lgm",
"lay:=", _
"Ground", "id:=", 3, "pad:=", Array("shp:=", "No", "Szs:=", Array(), "X:=", _
"0mm", "Y:=", "0mm", "R:=", "0deg"), "ant:=", Array("shp:=", "No", "Szs:=", Array(), "X:=", _
"0mm", "Y:=", "0mm", "R:=", "0"), "thm:=", Array("shp:=", "R90", "Szs:=", Array( _
"3mm", "0.75mm", "1mm"), "X:=", "0mm", "Y:=", "0mm", "R:=", "0"), "X:=", "0mm", "Y:=", _
"0mm", "dir:=", "Any"), Array("NAME:lgm", "lay:=", "Bottom signal", "id:=", 5, "pad:=", Array
("shp:=", _
"Cir", "Szs:=", Array("1mm"), "X:=", "0mm", "Y:=", "0mm", "R:=", "0deg"), "ant:=", Array
("shp:=", _
"Cir", "Szs:=", Array("2mm"), "X:=", "0mm", "Y:=", "0mm", "R:=", "0deg"), "thm:=", Array
("shp:=", _
"No", "Szs:=", Array(), "X:=", "0mm", "Y:=", "0mm", "R:=", "0"), "X:=", "0mm", "Y:=", _
"0mm", "dir:=", "Any")), "hle:=", Array("shp:=", "Cir", "Szs:=", Array("1.5mm"), "X:=", _
"0mm", "Y:=", "0mm", "R:=", "0deg"), "hRg:=", "End", "sbsh:=", "Sph", "sbpl:=", _
"abv", "sbr:=", "750um", "sb2:=", "1200um", "1200um", "sbn:=", "solder"), "ppl:=", Array( _
0, 1, 2, 3, 5), Array(""))

```

Export [padstack manager]

Use: Export a padstack to a library

Command: Tools > Edit Configured Libraries > Padstacks > Export to Library

Syntax: Export Array("NAME:<LibraryName>",
 <PadstackName>,
 <PadstackName>...),
 <LibraryLocation>

Return Value: None

Parameters: <LibraryName>:

<string> // name of the library

<PadstackName>:

<string> // [simple name](#) of padstack to export

<LibraryLocation>:

<string> // location of the library in <LibraryName>

// One of "Project", "PersonalLib", or "UserLib"

VB Example:

```
oPadstackManager.Export Array("NAME:mylib", "myPadstack"), "PersonalLib"
```

GetData [padstack manager]

Use: Gets data that describes the definition.

Command: None

Syntax: GetData(<DefinitionName>)

Return Value: <DefinitionData> This is an array of data for the definition.

Parameters: <DefinitionName>:

<string> // [composite name](#) of the definition to edit

VB Example:

```
Dim padstackData  
padstackData = oPadstackManager.GetData("NoPad SMT East")
```

Note:

GetData allows the user to access definition information, make modifications, and then use the Edit or EditWithComps script commands to save the modified definition. Accordingly, for each type of definition, the array data returned to GetData should be the same array information that is supplied to the [Edit](#) or [EditWithComps](#) commands.

GetNames [padstack manager]

Use: Returns the names of the padstack (used and unused) in a design. The following script command, **IsUsed**, can then be used to separate used and unused padstacks.

Command: None

Syntax: GetNames()

Return Value: An array of strings

Parameters: None

VB Example:

```
Dim padstackNames
```

```
padstackNames = oPadstackManager.GetNames()
```

IsUsed [padstack manager]

Use: Used to determine if a component is used in the design.

Command: None

Syntax: IsUsed(<PadstackName>)

Return Value: <Boolean> // true if the specified padstack is used in the design

Parameters: <PadstackName>:

<string>

VB Example:

```
Dim isUsed
```

```
isUsed = oPadstackManager.IsUsed("MyPadstack")
```

Remove [padstack manager]

Use: Removes a padstack from a library

Command: Tools > Edit Configured Libraries > Padstacks > Remove Padstacks

Syntax: Remove <PadstackName>,

<IsProjectPadstack>,

<LibraryName>,

<LibraryLocation>

Return Value: None

Parameters: <PadstackName>:

<string> // [simple name](#) of the padstack to remove

<IsProjectPadstack>:

<bool>

<LibraryName>:

<string> // name of the library

<LibraryLocation>:

<string> // location of the library in <LibraryName>

// One of "Project", "PersonalLib", or "UserLib"

VB Example:

```
oPadstackManager.Remove "Polygon SMT", true, "Padstacks", "Project"
```

```
oPadstackManager.Remove "Polygon SMT", false, "MyLib", "PersonalLib"
```

RemoveUnused [padstack manager]

Use: Removes padstacks that are not used in the design.

Command: **Project->Remove Unused Definitions** is similar but operates slightly different and does not record script commands.

Syntax: RemoveUnused()

Return Value: <bool> True if one or more padstacks are removed.

Parameters: None

VB Example:

```
Dim removedDefs
removedDefs = oPadstackManager.RemoveUnused()
```

Note:

The order of calls to RemoveUnused is significant. As a result, removing definitions in an unordered fashion may cause other padstacks in dependent definitions to be rendered unusable.

Script and Library Scripts

The definition manager provides access to materials in a project. The manager object is accessed via the definition manager.

```
Set oDefinitionManager = oProject.GetDefinitionManager()
```

The script and library script commands are listed below.

[AddScript](#)

[EditScript](#)

[ExportScript](#)

[RemoveScript](#)

[ModifyLibraries](#)

AddScript

Adds a script to the definition manager.

UI Access	N/A

Parameters	Name	Type	Description
	<code><AddScriptArray></code>	Array	Structured array. <code>Array("NAME:<string script name>", "ScriptLang:=", <string "vbscript" or "javascript"> "ScriptText:=, <string text of script>)</code>
Return Value	None.		

Python Syntax	<code>AddScript(<AddScriptArray>)</code>
Python Example	<pre>oDefinitionManager.AddScript(["NAME:MyScript", "ScriptLang:=", "vbscript", "ScriptText:=, "MsgBox(\"HelloWorld\") "])</pre>

VB Syntax	<code>AddScript <AddScriptArray></code>
VB Example	<pre>oDefinitionManager.AddScript Array("NAME:MyScript", "ScriptLang:=", "vbscript", "ScriptText:=, "MsgBox(""HelloWorld"") ")</pre>

EditScript

Edits a script in the definition manager.

UI Access	N/A		
Parameters	Name	Type	Description
	<OriginalName>	String	Name of the script to be edited.
	<EditScriptArray>	Array	Structured array. Array("NAME:<string script name>", "ScriptLang:=", <string "vbscript" or "javascript"> "ScriptText:=", <string text of script>)
Return Value	None.		

Python Syntax	EditScript(<OriginalName>, <EditScriptArray>)
Python Example	<pre>oDefinitionManager.EditScript("MyScript", ["NAME:MyNewScript", "ScriptLang:=", "vbscript", "ScriptText:=", "MsgBox(\"HelloAgain\") "])</pre>

VB Syntax	EditScript <OriginalName> <EditScriptArray>
VB Example	<pre>oDefinitionManager.EditScript "MyScript" Array("NAME:MyNewScript",</pre>

	"ScriptLang:=", "vbscript", "ScriptText:=", "MsgBox("HelloAgain")")")

ExportScript

Use: Export to Library in the script definition manager

Command: None

Syntax: ExportScript <ExportData>,<Library location>

Return Value: None

Parameters: <ExportData>

Array("NAME:<LibraryName>",<ScriptName>,<ScriptName>,...)

VB Example:

oProject.ExportComponent Array("NAME:mylib", "myscript"), "PersonalLib"

Python Syntax	ExportScript(<ExportData>,<Library location>)
Python Example	oProject.ExportComponent (["NAME:mylib", "myscript"], "PersonalLib")

ModifyLibraries

Use: Configure Libraries on the Tools menu

Command: None

Syntax: ModifyLibraries <DesignName>,Array(<ConfigLibArray>)

Return Value: None

Parameters: <DesignName>

Type: <string>

```
<ConfigLibArray>
Array("NAME:<LibraryType>,<ConfiguredLib>,<ConfiguredLib>,...),...
<ConfiguredLib> // blank to leave unchanged
<DefinitionType>
Array("<libraryname >","<libraryname>",<libraryname>,...)
VB Example:
oDefinitionManager.ModifyLibraries "MyCircuit", _
Array("NAME:PersonalLib"), _
Array("NAME:UserLib"), _
Array("NAME:SystemLib", _
"Symbols:=", Array( "Circuit Elements", "Symbols", _
"ParamExtraElements\PE_Symbols", _
"Vendor Elements\Nonlinear"))
```

Python Syntax	ModifyLibraries (<DesignName>,<ConfigLibArray>)]
Python Example	oDefinitionManager.ModifyLibraries("MyCircuit", _ ["NAME:PersonalLib"], _ ["NAME:UserLib"], _

	<pre>["NAME: SystemLib", _ "Symbols:=", ["Circuit Elements", "Symbols", _ "ParamExtraElements\PE_Symbols", _ "Vendor Elements\Nonlinear"]])</pre>

RemoveScript

Use: Remove Script in the script definition manager

Command: None

Syntax: RemoveScript <ScriptName>, <IsProjectScript>, <LibraryName>, <LibraryLocation>

Return Value: None

Parameters: <ScriptName>

Type: <string>

<IsProjectScript>

Type: <bool>

<LibraryName>

Type: <string>

<LibraryLocation>

Type: <string>

VB Example:

```
oDefinitionManager.RemoveScript "myscript", true, "Local", "Project"
```

Python Syntax	<code>RemoveScript (<ScriptName>,<IsProjectScript>, <LibraryName>,<LibraryLocation>)</code>
Python Example	<pre>oDefinitionManager.RemoveScript("myscript", true, "Local", "Project")</pre>

Symbol Manager Script Commands

The symbol manager provides access to symbols in a project. The manager object is accessed via the definition manager.

```
Set oDefinitionManager = oProject.GetDefinitionManager()
```

```
Set oSymbolManager = oDefinitionManager.GetManager("Symbol")
```

The symbol manager script commands are listed below.

[Add](#)

[BringToFront](#)

[Edit](#)

[EditWithComps](#)

[Export](#)

[GetData](#)

[GetNames](#)

[IsUsed](#)

[Remove](#)

[RemoveUnused](#)

Add [symbol manager]

Use: Add a symbol

Command: Tools > Edit Configured Libraries > Symbols > Add Symbol

Syntax: Add Array("NAME:<SymbolName>",
 "ModTime:=", <ModifiedTimeInfo>,
 "Library:=", "", // Library name
 "LibLocation:=", "Project", // Project Location
 <PinDefInfo>,
 <PinDefInfo>,... // optional, to define pins
 <GraphicsDataInfo>, // optional, to define graphics
 <PropDisplayMapInfo>)) // optional, to define property displays

Return Value: <string>

 // [composite name](#) of the symbol.
 // If the name requested conflicts with the name of an existing
 // symbol, the requested name is altered to be unique.
 // The name returned reflects any change made to be unique.

Parameters: <SymbolName>:

 <string> // [simple name](#) of the symbol being added

<ModifiedOnInfo>:

An integer that corresponds to the number of seconds that have elapsed since 00:00 hours, Jan 1, 1970 UTC from the system clock.

<PinDefInfo>:

```
Array("NAME:PinDef",  
"Pin:=", Array (<string>, // pin name  
<real>, // x location  
<real>, // y location  
<real>, // angle in radians  
<PinType>,  
<real>, // line width  
<real>, // line length  
<bool>, // mirrored  
<int>, // color  
<bool>, // true if visible, false if not  
<string>, // hidden net name  
<OptionalPinInfo>, // optional info  
<PropDisplayMapInfo>)) // optional
```

<PinType>:

<string> // "N" : normal pin

// "I" : input pin

// "O" : output pin

<OptionalPinInfo>:

// Specify both or neither

<bool>, // true if name is to be shown

<bool>, // true if number is to be shown

<PropDisplayMapInfo>:

Array("NAME:PropDisplayMap",

<PropDisplayInfo>,

<PropDisplayInfo>,...)

<PropDisplayInfo>:

<NameString>, Array(<DisplayTypeInfo>,

<DisplayLocationInfo>,

<int>, // optional, level number

<TextInfo>)

<NameString>:

<string> // PropertyName:=, where PropertyName is the name of
// the property to be displayed

<DisplayTypeInfo>:

<int> // 0 : No display

// 1 : Display name only

// 2 : Display value only

// 3 : Display both name and value

// 4: Display evaluated value only

// 5: Display both name and evaluated value

<DisplayLocationInfo>:

<int> // 0 : Left

// 1 : Top

// 2 : Right

// 3 : Bottom

// 4 : Center

// 5 : Custom placement

```
<GraphicsDataInfo>:  
Array("NAME:Graphics",  
// one or more of the following  
<RectInfo>,  
<CircleInfo>,  
<ArcInfo>,  
<LineInfo>,  
<PolygonInfo>,  
<TextInfo>,  
<ImageInfo>)
```

```
<RectInfo>:  
"Rect:=", Array(<real>, // line width  
<int>, // fill pattern  
<int>, // color  
<real>, // angle, in radians  
<real>, // x position of center  
<real>, // y position of center  
<real>, // width  
< real>) // height
```

<CircleInfo>:

```
"Circle:=", Array(<real>, // line width  
<int>, // fill pattern  
<int>, // color  
<real>, // x position of center  
<real>, // y position of center  
< real>) // radius
```

<ArcInfo>:

```
"Arc:=", Array(<real>, // line width  
<int>, // line pattern  
<int>, // color  
<real>, // x position of center  
<real>, // y position of center  
< real>, // radius  
<real>, // start angle, in radians  
<end>) // end angle, in radians
```

<LineInfo>:

```
"Line:=", Array(<real>, // line width
```

<int>, // line pattern

<int>, // color

<PointInfo>, // must specify at least 2 points

<PointInfo>...)

<PointInfo>:

<real>, // x position

<real> // y position

<PolygonInfo>:

"Polygon:=", Array(<real>, // line width

<int>, // fill pattern

<int>, // color

<PointInfo>, // must specify at least 3 points

<PointInfo>...)

<TextInfo>:

"Text:=", Array(<real>, // x position

<real>, // y position

<real>, // angle, in radians

<Justification>,

<bool>, // is plotter font

<string>, // font name

<int>, // color

<string>) // text string

<Justification>:

<int> // 0 : left top

// 1 : left base

// 2 : left bottom

// 3 : center top

// 4 : center base

// 5 : center bottom

// 6 : right top

// 7 : right base

// 8 : right bottom

<ImageInfo>:

"Image:=", Array(<RectInfo>,

<ImageData>,

<bool>) // is mirrored

<ImageData>:

<string>, // file path

<int>, // 0 : use the file path and link to it

// 1 : ignore file path and use next parameter

<string> // text data, only present if preceding int is 1

VB Example:

```
oSymbolManager.Add Array("NAME:MySymbol", _  
"ModTime:=", 1070989137, _  
"Library:=", "", _  
"LibLocation:=", "Project", _  
Array("NAME:PinDef", _  
"Pin:=", Array ("newpin0", _  
0.00254, _ 0, _  
0, _  
"N", _  
0, _  
0.00254, _  
false, _  
0, _  
true, _  
"", _
```

```
false, _  
false)), _  
Array("NAME:PinDef", _  
"Pin:=", Array ("newpin1", _  
-0.00254, _  
0, _  
3.14159265358979, _  
"N", _  
0, _  
0.00254, _  
false, _  
0, _  
true, _  
"", _  
false, _  
false)),  
Array("NAME:Graphics", _  
"Rect:=", Array(0, _  
0, _  
12566272, _
```

```
0, _  
4.33680868994202e-019, _  
-0.000635, _  
0.00508, _  
0.002794), _  
"Circle:=", Array(0, _  
0, _  
12566272, _  
0.000127, _  
0.000635, _  
0.000635)))
```

BringToFront [symbol manager]

Use: Changes the drawing for the symbol so that the specified objects are drawn on top of other overlapping objects.

Command: Draw > Bring To Front

Syntax: BringToFront Array("NAME:Selections", "Selections:=", Array (<Object>, <Object>, ...))

Return Value: None

Parameters: **<Object>**

<string> // object to bring to the front

VB Example:

```
oDefinitionEditor.BringToFront Array("NAME:Selections", "Selections:=", Array( "SchObj@10"))
```


Edit [deprecated]

Deprecated command — please use [EditWithComps](#).

EditWithComps [symbol manager]

Use: Edit an existing symbol.

Command: None

Syntax: EditWithComps <SymbolName>,
 Array("NAME:<NewSymbolName>",
 "ModTime:=", <ModifiedTimeInfo>,
 "Library:=", <string>, // Library name
 "LibLocation:=", <string>, // Project Location
 <PinDefInfo>,
 <PinDefInfo>,... // optional, to define pins
 <GraphicsDataInfo>, // optional, to define graphics
 <PropDisplayMapInfo>), // optional, to define property displays
 Array(<ListOfComponentNames>) // Component names

Return Value: <string>

 // [composite name](#) of the symbol.
 // If the name requested conflicts with the name of an existing
 // symbol, the requested name is altered to be unique.

// The name returned reflects any change made to be unique.

Parameters: <SymbolName>:

<string> // [composite name](#) of the symbol being edited

<NewSymbolName>:

<string> // new [simple name](#) for the symbol

<ModifiedOnInfo>:

An integer that corresponds to the number of seconds that have elapsed since 00:00 hours, Jan 1, 1970 UTC from the system clock.

<PinDefInfo>:

Array("NAME:PinDef",
"Pin:=", Array (<string>, // pin name
<real>, // x location
<real>, // y location
<real>, // angle in radians
<PinType>,
<real>, // line width
<real>, // line length

```
<bool>, // mirrored  
<int>, // color  
<bool>, // true if visible, false if not  
<string>, // hidden net name  
<OptionalPinInfo>, // optional info  
<PropDisplayMapInfo>)) // optional
```

```
<PinType>:  
<string> // "N" : normal pin  
// "I" : input pin  
// "O" : output pin
```

```
<OptionalPinInfo>:  
// Specify both or neither  
<bool>, // true if name is to be shown  
<bool>, // true if number is to be shown
```

```
<PropDisplayMapInfo>:  
Array("NAME:PropDisplayMap",  
<PropDisplayInfo>,
```

<PropDisplayInfo>,...)

<PropDisplayInfo>:

<NameString>, Array(<DisplayTypeInfo>,

<DisplayLocationInfo>,

<int>, // optional, level number

<TextInfo>)

<NameString>:

<string> // PropertyName:=, where PropertyName is the name of

// the property to be displayed

<DisplayTypeInfo>:

<int> // 0 : No display

// 1 : Display name only

// 2 : Display value only

// 3 : Display both name and value

// 4: Display evaluated value only

// 5: Display both name and evaluated value

<DisplayLocationInfo>:

<int> // 0 : Left

// 1 : Top

// 2 : Right

// 3 : Bottom

// 4 : Center

// 5 : Custom placement

<GraphicsDataInfo>:

Array("NAME:Graphics",

// one or more of the following

<RectInfo>,

<CircleInfo>,

<ArcInfo>,

<LineInfo>,

<PolygonInfo>,

<TextInfo>,

<ImageInfo>)

<RectInfo>:

"Rect:=", Array(<real>, // line width

<int>, // fill pattern

<int>, // color

<real>, // angle, in radians

<real>, // x position of center

<real>, // y position of center

<real>, // width

< real>) // height

<CircleInfo>:

"Circle:=", Array(<real>, // line width

<int>, // fill pattern

<int>, // color

<real>, // x position of center

<real>, // y position of center

< real>) // radius

<ArcInfo>:

"Arc:=", Array(<real>, // line width

<int>, // line pattern

<int>, // color

<real>, // x position of center

<real>, // y position of center
< real>, // radius
<real>, // start angle, in radians
<end>) // end angle, in radians

<LineInfo>:

"Line:=", Array(<real>, // line width
<int>, // line pattern
<int>, // color
<PointInfo>, // must specify at least 2 points
<PointInfo>...)
<PointInfo>:
<real>, // x position
<real> // y position

<PolygonInfo>:

"Polygon:=", Array(<real>, // line width
<int>, // fill pattern
<int>, // color
<PointInfo>, // must specify at least 3 points

<PointInfo>...)

<TextInfo>:

"Text:=", Array(<real>, // x position

<real>, // y position

<real>, // angle, in radians

<Justification>,

<bool>, // is plotter font

<string>, // font name

<int>, // color

<string>) // text string

<Justification>:

<int> // 0 : left top

// 1 : left base

// 2 : left bottom

// 3 : center top

// 4 : center base

// 5 : center bottom

// 6 : right top

// 7 : right base

// 8 : right bottom

<ImageInfo>:

"Image:=", Array(<RectInfo>,

<ImageData>,

<bool>) // is mirrored

<ImageData>:

<string>, // file path

<int>, // 0 : use the file path and link to it

// 1 : ignore file path and use next parameter

<string> // text data, only present if preceding int is 1

<ListOfComponentNames>:

<string>,<string> ...

// The list may be empty. When not empty, each string that is listed is a component

// that references the symbol to be edited. Prior to editing, a clone of the symbol is

// made, and the components that are listed are modified so that they now refer to

// the clone.

VB Example:

```
Dim nam
oSymbolManager.EditWithComps _
("Nexxim Circuit Elements\Distributed\Distributed:bendo", _
Array("NAME:bendo new name", _
"ModTime:=", 1152722165, _
"Library:=", "Nexxim Circuit Elements\Distributed\Distributed", _
"LibLocation:=", "SysLibrary", _
Array("NAME:PinDef", _
"Pin:=", Array( "n1", _
-0.00254, _
0.00254, _
0, _
"N", _
0, _
0, _
false, _
0, _
true, _
"", _
false, _
```

```
false)), _  
Array("NAME:PinDef", _  
"Pin:=", Array( "n2", _  
0.00254, _  
-0.00254, _  
1.5708, _  
"N", _  
0, _  
0, _  
false, _  
0, _  
true, _  
"", _  
false, _  
false)), _  
Array("NAME:Graphics", _  
"Polygon:=", Array( 0, 0, 12566272, 0, 0.00381, 0, .00127, 0.00127, _  
0.00127, 0.00127, 0, 0.00381, 0, 0.00381, _  
0.002032, 0.00127, 0.00381), _  
"Line:=", Array(0, 1, 12566272, -0.00254, 0.00254, 0, .00254), _
```

```
"Line:=", Array(0, 1, 12566272, 0.00254, -0.00254, .00254, 0)), _  
Array("NAME:PropDisplayMap", _  
"W:=", Array(3, _  
5, _  
0, _  
"Text:=", Array( 2.1684E-019, _  
0.00504119, _  
0, _  
1, _  
5, _  
false, _  
"Arial", _  
0, _  
"W=***")))) , _  
Array("MY_COMP")
```

Export [symbol manager]

Use: Exports symbol(s) to a library

Command: Tools > Edit Configured Libraries > Symbols > Export to Library

Syntax: Export Array("NAME:<LibraryName>",
 <SymbolName>,

<SymbolName>...),

<LibraryLocation>

Return Value: None

Parameters: <LibraryName>:

<string> // name of the library

<SymbolName>:

<string> // [composite name](#) of symbol to export

<LibraryLocation>:

<string> // location of the library in <LibraryName>

// One of "Project", "PersonalLib", or "UserLib"

VB Example:

```
oSymbolManager.Export _  
Array("NAME Nexxim Circuit Elements\Distributed\Distributed:bendo:mylib", _ "mySymbol"), "Per-  
sonalLib"
```

GetData [symbol manager]

Use: Gets data that describes the definition.

Command: None

Syntax: GetData(<DefinitionName>)

Return Value: <DefinitionData> This is an array of data for the definition.

Parameters: <DefinitionName>:

<string> // [composite name](#) of the definition to edit

VB Example:

```
Dim symbolData
```

```
symbolData = oSymbolManager.GetData("Nexxim Circuit Elements\Hspice:nexx_bjt_npn")
```

Note:

GetData allows the user to access definition information, make modifications, and then use the Edit or EditWithComps script commands to save the modified definition. Accordingly, for each type of definition, the array data returned to GetData should be the same array information that is supplied to the [Edit](#) or [EditWithComps](#) commands.

GetNames [symbol manager]

Use: Returns the names of the symbols (used and unused) in a design. The following script command, **IsUsed**, can then be used to separate used and unused symbols.

Command: None

Syntax: GetNames()

Return Value: An array of strings

Parameters: None

VB Example:

```
Dim symbolNames
```

```
symbolNames = oSymbolManager.GetNames()
```

IsUsed [symbol manager]

Use: Used to determine if a symbol is used in the design.

Command: None

Syntax: IsUsed(<SymbolName>)

Return Value: <Boolean> // true if the specified symbol is used in the design

Parameters: <SymbolName>:

<string>

VB Example: Dim isUsed

isUsed = oSymbolManager.IsUsed("MySymbol")

Remove [symbol manager]

Use: Removes a symbol from a library

Command: Tools > Edit Configured Libraries > Symbols > Remove Symbol

Syntax: Remove <SymbolName>,

<IsProjectSymbol>,

<LibraryName>,

<LibraryLocation>

Return Value: None

Parameters: <SymbolName>:

<string> // [composite name](#) of the symbol to remove

<IsProjectSymbol>:

<bool>

<LibraryName>:

<string> // name of the library

<LibraryLocation>:

<string> // location of the library in <LibraryName>

// One of "Project", "PersonalLib", or "UserLib"

VB Example:

```
oSymbolManager.Remove "Nexxim Circuit Elements\Distributed\Distributed:bendo", true, "Project"
```

RemoveUnused [symbol manager]

Use: Removes symbols that are not used in the design.

Command: **Project->Remove Unused Definitions** is similar but operates slightly different and does not record script commands.

Syntax: RemoveUnused()

Return Value: <bool> True if one or more symbols are removed.

Parameters: None

VB Example:

```
Dim removedDefs  
removedDefs = oSymbolManager.RemoveUnused()
```


Note:

The order of calls to RemoveUnused is significant. As a result, removing definitions in an unordered fashion may cause other symbols in dependent definitions to be rendered unusable.

Also, the symbol and footprint of an unused component are not unusable until after the component itself is removed using the Component Manager Remove script.

22 - Core Global Script Context Commands

To run these commands:

```
import CoreGlobalScriptContextFunctions

CoreGlobalScriptContextFunctions.[CommandName]
```

The following are general script commands recognized by the **CoreGlobalScriptContextFunctions** object:

- [AddErrorMessage](#)
- [AddFatalMessage](#)
- [AddInfoMessage](#)
- [AddWarningMessage](#)
- [LogDebug](#)
- [LogError](#)

AddErrorMessage

Adds an error message to the **Message Manager** window. AddErrorMessage is a function of CoreGlobalScriptContextFunctions.

UI Access	N/A		
Parameters	Name	Type	Description
	<message>	String	Error message.
Return Value	None.		

Python Syntax	AddErrorMessage(<message>)
Python Example	<pre>import CoreGlobalScriptContextFunctions CoreGlobalScriptContextFunctions.AddErrorMessage('My error message.')</pre>

VB Syntax	N/A
VB Example	N/A. Script cannot be run in VBScript.

AddFatalMessage

Adds a fatal error message to the **Message Manager** window. AddFatalMessage is a function of CoreGlobalScriptContextFunctions.

UI Access	N/A		
Parameters	Name	Type	Description
	<message>	String	Error message.
Return Value	None.		

Python Syntax	AddFatalMessage(<message>)
Python Example	<pre>import CoreGlobalScriptContextFunctions CoreGlobalScriptContextFunctions.AddFatalMessage('My fatal error message.')</pre>

VB Syntax	N/A
VB Example	N/A. Script cannot be run in VBScript.

AddInfoMessage

Adds an informational message to the **Message Manager** window. AddInfoMessage is a function of CoreGlobalScriptContextFunctions.

UI Access	N/A		
Parameters	Name	Type	Description
	<message>	String	Informational message.
Return Value	None.		

Python Syntax	AddInfoMessage(<message>)
Python Example	<pre>import CoreGlobalScriptContextFunctions CoreGlobalScriptContextFunctions.AddInfoMessage('My info.')</pre>

VB Syntax	N/A
VB Example	N/A. Script cannot be run in VBScript.

AddWarningMessage

Adds a warning message to the **Message Manager** window. AddWarningMessage is a function of CoreGlobalScriptContextFunctions.

UI Access	N/A		
Parameters	Name	Type	Description
	<message>	String	Warning message.
Return Value	None.		

Python Syntax	AddWarningMessge(<message>)
Python Example	<pre>import CoreGlobalScriptContextFunctions CoreGlobalScriptContextFunctions.AddWarningMessage('My warn- ing.')</pre>

VB Syntax	N/A
VB Example	N/A. Script cannot be run in VBScript.

LogDebug

Adds a debug line to the log specified at **Tools > Debug Logging**. LogDebug is a function of CoreGlobalScriptContextFunctions.

UI Access	N/A		
Parameters	Name	Type	Description
	<message>	String	Debug message.
Return Value	None.		

Python Syntax	LogDebug(<message>)

Python Example	<pre>import CoreGlobalScriptContextFunctions CoreGlobalScriptContextFunctions.LogDebug('My debug message.')</pre>

VB Syntax	N/A
VB Example	N/A. Script cannot be run in VBScript.

LogError

Adds an error line to the log specified at **Tools > Debug Logging**. LogError is a function of CoreGlobalScriptContextFunctions.

UI Access	N/A		
Parameters	Name	Type	Description
	<error>	String	Error to log.
Return Value	None.		

Python Syntax	LogError(<error>)
Python Example	<pre>import CoreGlobalScriptContextFunctions CoreGlobalScriptContextFunctions.LogError('My error.')</pre>

VB Syntax	N/A
VB Example	N/A. Script cannot be run in VBScript.

23 - Example Scripts

This section contains [VBScript](#) and [IronPython](#) example scripts.

VBScript Example Scripts

VBScript Examples:

- [Variable Helix Script](#)
- [HFSS Data Export Script](#)
- [ExampleGetLayeredImpedanceBoundaryPropertyNamesandValues.htm](#)

Data Export Script

The following sample script demonstrates how to export data and save it to a file. The output data in the example script is in 3 columns. The first column is frequency in GHz, the second is the Real part of S11, and the third is the Img part of S11. It uses a tab-delimited format. The output is done using output variables.

The frequency sweep data must be entered correctly. If it is incorrect, the script will request a freq point that does not exist and execution will stop.

The script includes comment lines, which are preceded by an apostrophe (') and offer explanations for each subsequent line or lines.

```
Dim oAnsoftApp
Dim oDesktop
Dim oProject
Dim oDesign
Dim oEditor
Dim oModule

Set oAnsoftApp = CreateObject("Ansoft.ElectronicsDesktop")
Set oDesktop = oAnsoftApp.GetAppDesktop()
set oProject = oDesktop.GetActiveProject
set oDesign = oProject.GetActiveDesign()

Dim oFS,oFile,x,y,z,path,range,
Dim arr2,del_f,freq,cfreq,val,temp,stn,stw,i,line
'
' Input the desired file name.
'
```

```
path = inputbox("Input the file name" & chr(13) & _
"Note: If you do not specify a path the file will " & _
"be placed in the script directory", _
"File", "C:\hfss_export.txt", 50, 50)
,
    ' If the user clicks Cancel the path will be blank, in which
    case the script should just exit.
If path <>"" then
    ,
    ' Create the file, open it for data entry, and output the
    column labels.
    ,
Set oFS = CreateObject("Scripting.FileSystemObject")
Set ofile = oFS.CreateTextFile (path)
line = "Freq" & chr(9) & "RE(S11)" & chr(9) & "IMG(S11)"
ofile.WriteLine line
    ,
    ' Input the needed freq, solution, and sweep data and clean it
    up.
    ,
msgbox("For the following input make sure it matches " & _
"the frequencies defined in your sweep")
range = inputbox("Input the range of frequencies in GHz " & _
"and number of points", _
"Frequency", "8,12,10", 50, 50)
    ,
    ' The following 2 lines define the 2 output variables.
    ,
oDesign.CreateOutputVariable "re_S", "re(S(port1,port1))"
oDesign.CreateOutputVariable "im_S", "im(S(port1,port1))"
arr = split (range, ",")
```

```
arr(0) = Trim(arr(0))
arr(1) = Trim(arr(1))
arr(2) = Trim(arr(2))
if cint(arr(2)) <> 1 then
del_f = (arr(1)-arr(0))/(arr(2)-1)
else
del_f = 0
end if

temp = InputBox("Input the Setup and Sweep number to use:"_
& chr(13) & "(e.g. input 1,2 for Setup1 and Sweep2)", _
"Solution Data","1,1",50,50)
arr2 = split(temp,",")
stn = arr2(0)
swn = arr2(1)
stn = Trim(stn)
swn = Trim(swn)

'
' Loop through the freq points.
'

for i=1 to arr(2) step 1
freq = arr(0) + (cint(i)-1)*del_f
x=freq
cfreq="Freq='" & freq & "Ghz'"

'
' Get the values of the output variables for the desired freq.
'

val = oDesign.GetOutputVariableValue("re_S","Setup" & _
stn & " :Sweep" & swn,cfreq, "")
y = val
val = oDesign.GetOutputVariableValue("im_S","Setup" & _
```

```
stn & " : Sweep" & swncfreq, "")
z = val
    '
    ' Create the line of text to send to the file and write it to
    ' the file.
    '
line = x & chr(9) & y & chr(9) & z
ofile.WriteLine line
Next
    '
    ' Delete the 2 output variables before finishing.
    '
oDesign.DeleteOutputVariable "re_S"
oDesign.DeleteOutputVariable "im_S"
    '
    ' Close the file.
    '
ofile.close
End if
```

Variable Helix Script

This sample script creates a tapered helix. Tapering helices is not supported from the interface.

The script includes comment lines, which are preceded by an apostrophe (') and offer explanations for each subsequent line or lines.

```
Dim oAnsoftApp
Dim oDesktop
Dim oProject
Dim oDesign
Dim oEditor
Dim oModule
Set oAnsoftApp = CreateObject("Ansoft.ElectronicsDesktop")
```

```
Set oDesktop = oAnsoftApp.GetAppDesktop()
Set oProject = oDesktop.GetActiveProject()
Set oDesign = oProject.GetActiveDesign()
Set oEditor = oDesign.SetActiveEditor("3D Modeler")

' Declare the arrays and variables needed for building the polyline.
,

Dim points(), segments()
Dim NumPoints, R(2), P(2), PointsPerTurn, Turns, Units
,

' Establish the constant Pi.
Pi = 4*Atn(1)

' Retrieve the variable helix parameters from the user.
' Start with the input for unit selection.
,

Units = InputBox("Select the units:" & Chr(13) & _
    "(cm,mm,um,in,mil)", "Variable Helix", "mil", 50, 50)
,

' Check to make sure it is a valid unit.
,

Select Case Units
    Case "m"
Units = ""
    Case "cm"
    Case "mm"
    Case "um"
    Case "in"
    Case "mil"
    Case Else
MsgBox("Invalid Units - defaults to m")
```

```
Units = ""
End Select
    ,
    ' Obtain the other user-defined parameters.
    ,

Turns = InputBox("Select the number of turns (must be
    integer):", "Variable Helix", 2, 50, 50)
PointsPerTurn = InputBox("Select the points per turn:", _
    "Variable Helix", 16, 50, 50)
R(0) = InputBox("Select the initial Radius: ", _
    "Variable Helix", 10, 50, 50)
R(1) = InputBox("Select the final Radius: ", _
    "Variable Helix", 10, 50, 50)
P(0) = InputBox("Select the initial Pitch: ", _
    "Variable Helix", 4, 50, 50)
P(1) = InputBox("Select the final Pitch: ", _
    "Variable Helix", 4, 50, 50)
NumPoints = Turns*PointsPerTurn
,
    ' Initialize the points and segments arrays.
    ,

Redim points (NumPoints+1)
Redim segments (NumPoints)
points(0) = "NAME:PolylinePoints"
segments(0) = "NAME:PolylineSegments"
    ,
    ' Build the Point and Segment Arrays needed in the HFSS polyline call.
    ,

For n = 1 To (NumPoints+1)
    Angle = (n-1)*2*Pi/PointsPerTurn
```

```
Radius = R(0) + ((n-1)/NumPoints)*(R(1)-R(0))
Pitch = P(0) + ((n-1)/NumPoints)*(P(1)-P(0))
Rise = (n-1)*Pitch/PointsPerTurn

XValue = cstr(Radius*cos(Angle)) & Units
YValue = cstr(Radius*sin(Angle)) & Units
ZValue = cstr(Rise) & Units
points(n) = Array("NAME:PLPoint", "X:=", XValue, "Y:=", _
YValue, "Z:=", ZValue)
,
' Create the line segments between each of the pairs of points.
,

If n<=NumPoints Then
segments(n) = Array("NAME:PLSegment", "SegmentType:=", _
"Line", "StartIndex:=", (n-1), "NoOfPoints:=", 2)
End If
Next
,
' Create the polyline.
,

oEditor.CreatePolyline _
Array("NAME:PolylineParameters", "IsPolylineCovered:=", true, _
"IsPolylineClosed:=", false, points, segments), _
Array("NAME:Attributes", "Name:=", "Line_Helix", "Flags:=", _
"", "Color:=", "(132 132 193)", "Transparency:=", 0.4, _
"PartCoordinateSystem:=", "Global", "MaterialName:=", _
"vacuum", "SolveInside:=", true)
,
' Create the helix cross-section.
,
```

```
oEditor.CreateCircle _
    Array("NAME:CircleParameters", "IsCovered:=", true, "XCenter:=", _
        cstr(R(0))&Units, "YCenter:=", 0, "ZCenter:=", 0, "Radius:=", _
        "1"&Units, "WhichAxis:=", "Y"), _
    Array("NAME:Attributes", "Name:=", "Circle_Helix", "Flags:=", _
        "", "Color:=", "(132 132 193)", "Transparency:=", 0.4, _
        "PartCoordinateSystem:=", "Global", "MaterialName:=", "vacuum", _
        "SolveInside:=", true)
    ,
    ' Sweep the cross-section along the path.
    ,
```

```
oEditor.SweepAlongPath _
    Array("NAME:Selections", "Selections:=", _
        "Circle_Helix,Line_Helix"),
    Array("NAME:PathSweepParameters", "DraftAngle:=", "0deg", _
        "DraftType:=", "Round", "TwistAngle:=", "0deg")
```

GetPropNames and GetPropValues for Layered Impedance Boundary Script

The following example script used Object Oriented commands to GetPropName for a Layered Impedance Boundary, and then use the names to use GetPropValue for those properties.

```
' -----
' Script Recorded by Ansys Electronics Desktop Version 2022.1.0
' 14:55:18 Oct 25, 2022
' -----

Option Explicit

test_layered_impedance()

Sub test_layered_impedance()

    Dim oAnsoftApp

    Dim oDesktop

    Dim oProject

    Dim oDesign
```

```
Dim oEditor
Dim oModule
Dim oBoundaries
Dim Layered_Boundaries
Dim single_boundary
Dim oBoundary
Dim Get_Boundary_Properties
Dim Layer_Name
Dim Layer_Type
Dim Layer_Roughness
Dim Layer_Inf_Ground_Plane
Dim Layer_Two_Sided
Dim Layer_Shell_Element
Dim Layer_Number_of_Layers
Dim Layer_L1_Thickness
Dim Layer_L1_Material
Dim Layer_L2_Thickness
Dim Layer_L2_Material
Set oAnsoftApp = CreateObject("Ansoft.ElectronicsDesktop.2022.1")
Set oDesktop = oAnsoftApp.GetAppDesktop()
oDesktop.RestoreWindow
Set oProject = oDesktop.SetActiveProject("test1")
Set oDesign = oProject.SetActiveDesign("HFSSDesign1")
Set oModule = oDesign.GetModule("BoundarySetup")

Set oBoundaries = oDesign.GetChildObject("Boundaries")
Layered_Boundaries = oModule.GetBoundariesOfType("Layered Imped-
ance")
For Each single_boundary In Layered_Boundaries
    Set oBoundary = oBoundaries.GetChildObject(single_boundary)
```

```
'Get_Boundary_Properties = oBoundary.GetPropNames()
Layer_Name = oBoundary.GetPropValue("Name")
Layer_Type = oBoundary.GetPropValue("Type")
Layer_Roughness = oBoundary.GetPropValue("Roughness")
Layer_Inf_Ground_Plane = oBoundary.GetPropValue("Inf Ground
Plane")
Layer_Two_Sided = oBoundary.GetPropValue("Two sided")
Layer_Shell_Element = oBoundary.GetPropValue("Shell Element")
Layer_Number_of_Layers = oBoundary.GetPropValue("Number of Lay-
ers")
Layer_L1_Thickness = oBoundary.GetPropValue("Layer 1/Thickness")
Layer_L1_Material = oBoundary.GetPropValue("Layer 1/Material")
Layer_L2_Thickness = oBoundary.GetPropValue("Layer 2/Thickness")
Layer_L2_Material = oBoundary.GetPropValue("Layer 2/Material")
MsgBox ("Name = " & Layer_Name & vbNewLine _
& "Type = " & Layer_Type & vbNewLine _
& "Roughness = " & Layer_Roughness & vbNewLine _
& "Inf Ground = " & Layer_Inf_Ground_Plane & vbNewLine _
& "Two-Sided = " & Layer_Two_Sided & vbNewLine _
& "Shell Element = " & Layer_Shell_Element & vbNewLine _
& "Number of Layers = " & Layer_Number_of_Layers & vbNewLine _
& "Layer 1 Thickness = " & Layer_L1_Thickness & vbNewLine _
& "Layer 1 Material = " & Layer_L1_Material & vbNewLine _
& "Layer 2 Thickness = " & Layer_L2_Thickness & vbNewLine _
& "Layer 2 Material = " & Layer_L2_Material)
```

Next

End Sub

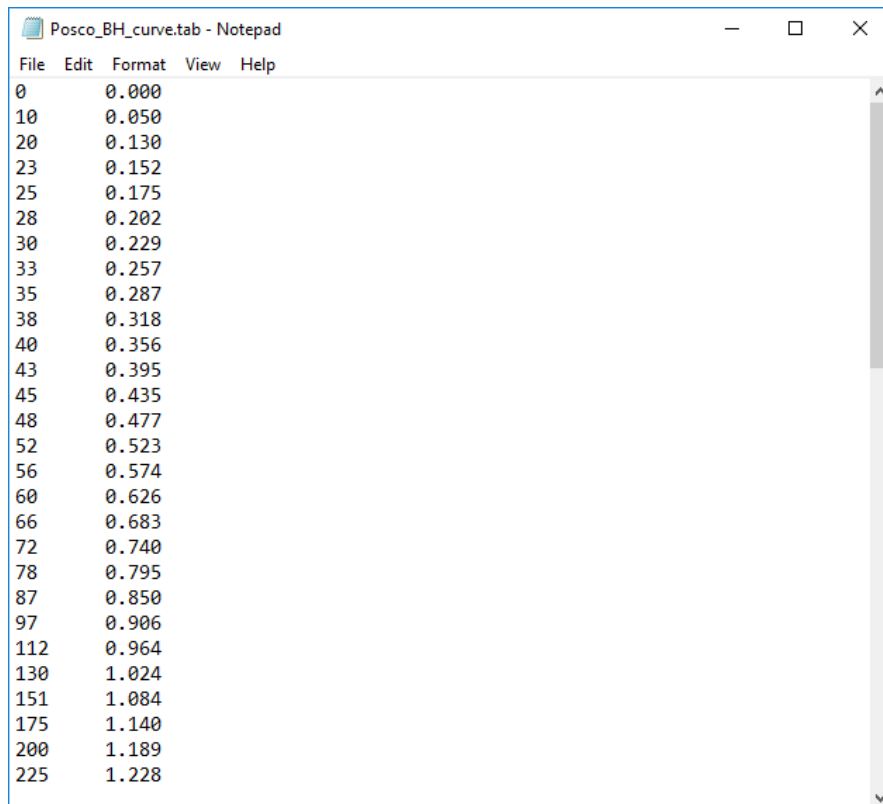
IronPython Example Scripts

IronPython Examples:

- [BH Coordinates Python Script](#)
- [Equation Based Curve Python Script](#)

BH Coordinates Python Script

This sample Python script adds a material ("Posco 35PN250") with BH coordinates from a specified file (Posco_BH_curve.tab):



0	0.000
10	0.050
20	0.130
23	0.152
25	0.175
28	0.202
30	0.229
33	0.257
35	0.287
38	0.318
40	0.356
43	0.395
45	0.435
48	0.477
52	0.523
56	0.574
60	0.626
66	0.683
72	0.740
78	0.795
87	0.850
97	0.906
112	0.964
130	1.024
151	1.084
175	1.140
200	1.189
225	1.228

To recreate this tab file, paste [the text below the script](#) into a text editor and save as Posco_BH_curve.tab.

The script itself includes comment lines, which are preceded by # and offer explanations for the subsequent line(s).

Script Contents

```
# specify path to the file with BH coordinates
path_to_file = r"D:\Posco_BH_curve.tab"

# specify name of the material
material_name = "Posco 35PN250"

oProject = oDesktop.GetActiveProject()
```



```
oDefinitionManager = oProject.GetDefinitionManager()

""" create list with B and H points to pass to AddMaterial command
bh_coordinates list is a three dimensional array: 1D: name, 2D:
Coordinate list, 3D: BH point"""

bh_coordinates = ["NAME:BHCoordinates", ["NAME:DimUnits", "", ""]]

with open(path_to_file) as input_file:
    for line in input_file:
        h, b = line.split()

        bh_coordinates.append(["NAME:Coordinate", [
            "NAME:CoordPoint",
            float(h),
            float(b)
        ]
        ])

# create a new magnetic material with BH curve
oDefinitionManager.AddMaterial(
[
    "NAME:" + material_name,
    "CoordinateSystemType:=", "Cartesian",
    "BulkOrSurfaceType:=" , 1,
    [
        "NAME:PhysicsTypes",
        "set:=" , ["Electromagnetic"]
    ],
    [
        "NAME:permeability",
        "property_type:=" , "nonlinear",
        "BTypeForSingleCurve:=" , "normal",
        "HUnit:=" , "A_per_meter", # unit can be specified as variable
```

```
"BUnit:=" , "tesla", # unit can be specified as variable
"IsTemperatureDependent:=", False,
bh_coordinates,
[
    "NAME:Temperatures"
]
],
"conductivity:=" , "1818181.82",
[
    "NAME:magnetic_coercivity",
    "property_type:=" , "VectorProperty",
    "Magnitude:=" , "0A_per_meter",
    "DirComp1:=" , "1",
    "DirComp2:=" , "0",
    "DirComp3:=" , "0"
]
])
```

Posco_BH_Curve.tab Contents

0	0.000
10	0.050
20	0.130
23	0.152
25	0.175
28	0.202
30	0.229
33	0.257
35	0.287
38	0.318
40	0.356
43	0.395

45	0.435
48	0.477
52	0.523
56	0.574
60	0.626
66	0.683
72	0.740
78	0.795
87	0.850
97	0.906
112	0.964
130	1.024
151	1.084
175	1.140
200	1.189
225	1.228
252	1.258
282	1.283
317	1.304
357	1.324
402	1.343
450	1.360
500	1.375
550	1.387
602	1.398
657	1.407
717	1.417
784	1.426
867	1.437
974	1.448

1117	1.461
1299	1.478
1515	1.495
1752	1.513
2000	1.529
2253	1.543
2521	1.557
2820	1.570
3167	1.584
3570	1.600
4021	1.617
4503	1.634
5000	1.652
5503	1.669
6021	1.685
6570	1.701
7167	1.717
7839	1.733
8667	1.749
9745	1.769
11167	1.793
12992	1.820
15146	1.851
17518	1.882
20000	1.909
22552	1.931
25417	1.948
28906	1.961
33333	1.971
38932	1.981

Equation Based Curve Python Script

This sample Python script creates an equation based curve that produces a helix.

```
from math import pi, sin, cos

oProject = oDesktop.GetActiveProject()
oDesign = oProject.GetActiveDesign()
oEditor = oDesign.SetActiveEditor("3D Modeler")

Start_t = 0
End_t = pi*2

Npoint = 128
Nsection = Npoint-1

d_t = (End_t-Start_t)/Nsection

for n in range(1,Nsection):

    P1 = Start_t+d_t*(n-1)
    P2 = P1+d_t
    X_t1 = cos(P1*6)
    Y_t1 = sin(P1*6)
    Z_t1 = P1
    X_t2 = cos(P2*6)
    Y_t2 = sin(P2*6)
    Z_t2 = P2

oEditor.CreatePolyline(
    [
```

```
"NAME:PolylineParameters",
"IsPolylineCovered:=" , True,
"IsPolylineClosed:=" , False,
[
"NAME:PolylinePoints",
[
"NAME:PLPoint",
"X:=" , '1mm*' + str(X_t1),
"Y:=" , '1mm*' + str(Y_t1),
"Z:=" , '1mm*' + str(Z_t1)
],
[
"NAME:PLPoint",
"X:=" , '1mm*' + str(X_t2),
"Y:=" , '1mm*' + str(Y_t2),
"Z:=" , '1mm*' + str(Z_t2)
]
],
[
"NAME:PolylineSegments",
[
"NAME:PLSegment",
"SegmentType:=" , "Line",
"StartIndex:=" , 0,
"NoOfPoints:=" , 2
]
],
[
"NAME:PolylineXSection",
"XSectionType:=" , "None",
```

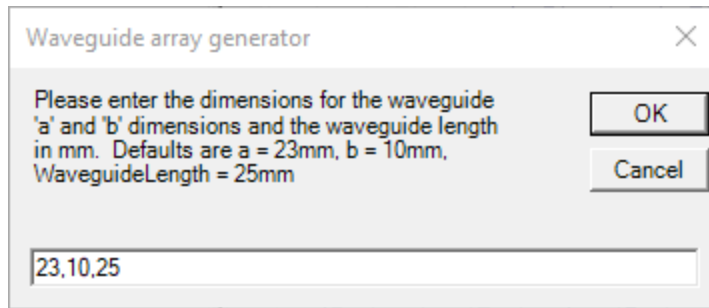
```
"XSectionOrient:=" , "Auto",
"XSectionWidth:=" , "0mm",
"XSectionTopWidth:=" , "0mm",
"XSectionHeight:=" , "0mm",
"XSectionNumSegments:=" , "0",
"XSectionBendType:=" , "Corner"
]
],
[
  "NAME:Attributes",
  "Name:=" , "Polyline"+str(n),
  "Flags:=" , "",
  "Color:=" , "(132 132 193)",
  "Transparency:=" , 0,
  "PartCoordinateSystem:=" , "Global",
  "UDMId:=" , "",
  "MaterialValue:=" , "\"vacuum\"",
  "SurfaceMaterialValue:=" , "\"\"",
  "SolveInside:=" , True,
  "IsMaterialEditable:=" , True,
  "UseMaterialAppearance:=" , False,
  "IsLightweight:=" , False
]
])
```

HFSS Waveguide Array Python Script

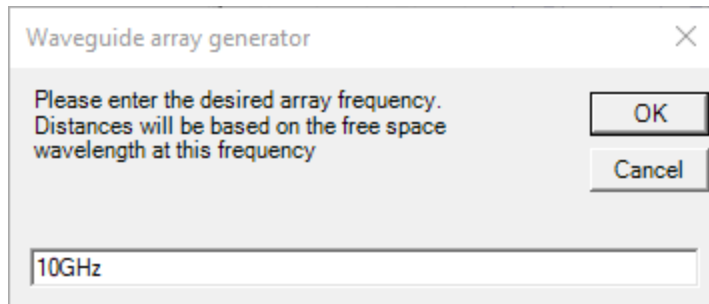
This sample Python script creates an HFSS Waveguide array. The script includes comment lines, which are preceded by an apostrophe ('), that offer explanations for each subsequent line or lines. The script includes examples of creating a 3D model, boundaries and excitation, solution setup and sweep, as well as reports.

You must insert an HFSS project before running the script. Running the script causes a series of input dialogs to appear that lets you set parameters.

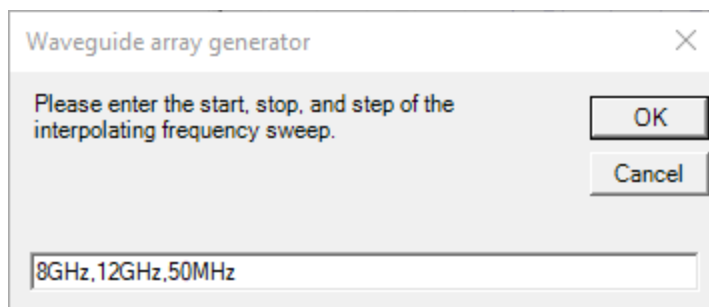
The first dialog asks for input for a and b dimensions and the waveguide length.



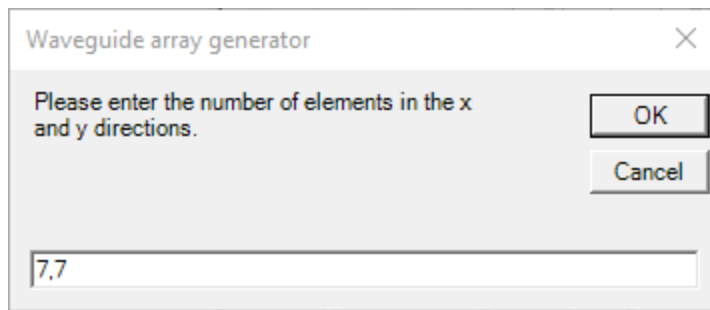
The next asks for the array frequency.



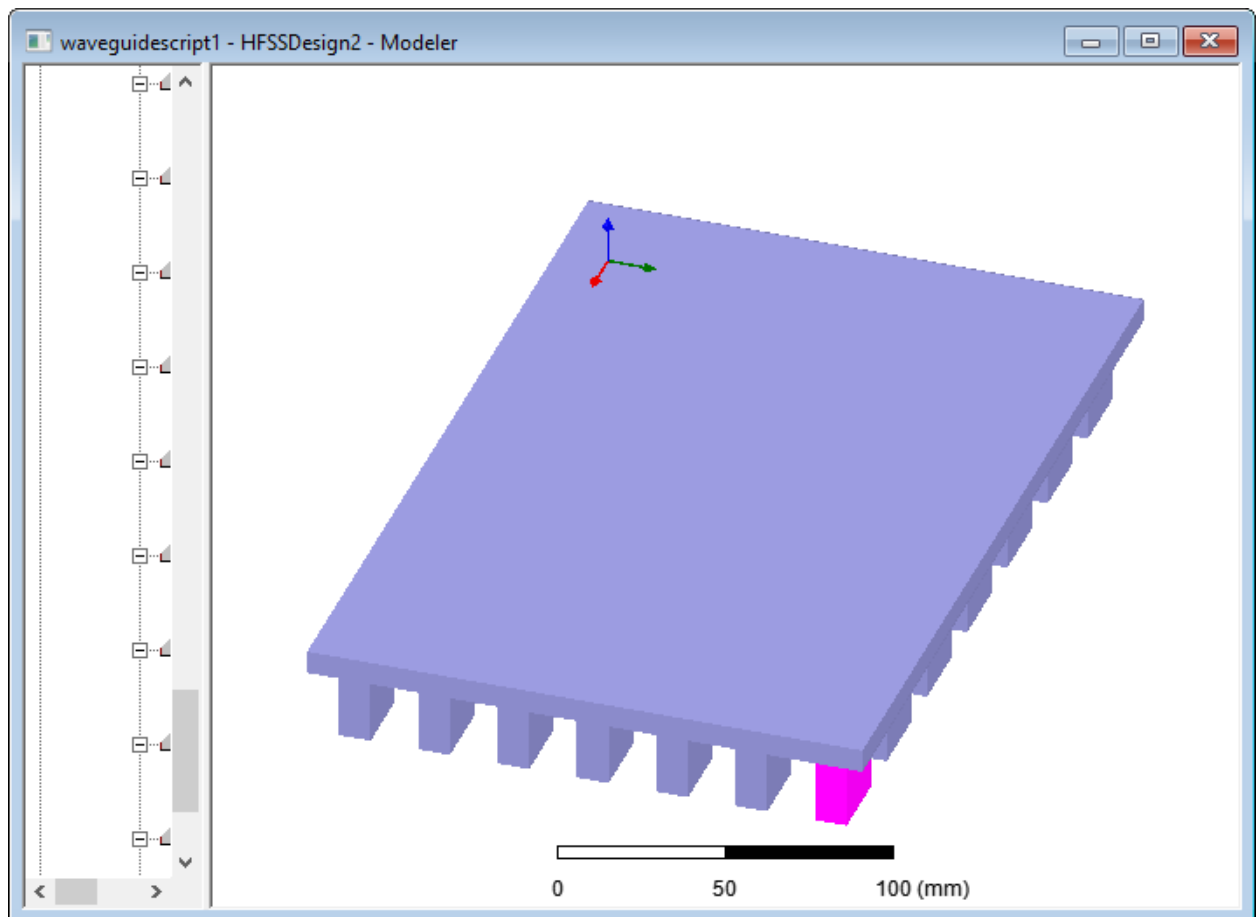
The next asks for start, stop and step values for in an interpolating frequency sweep.



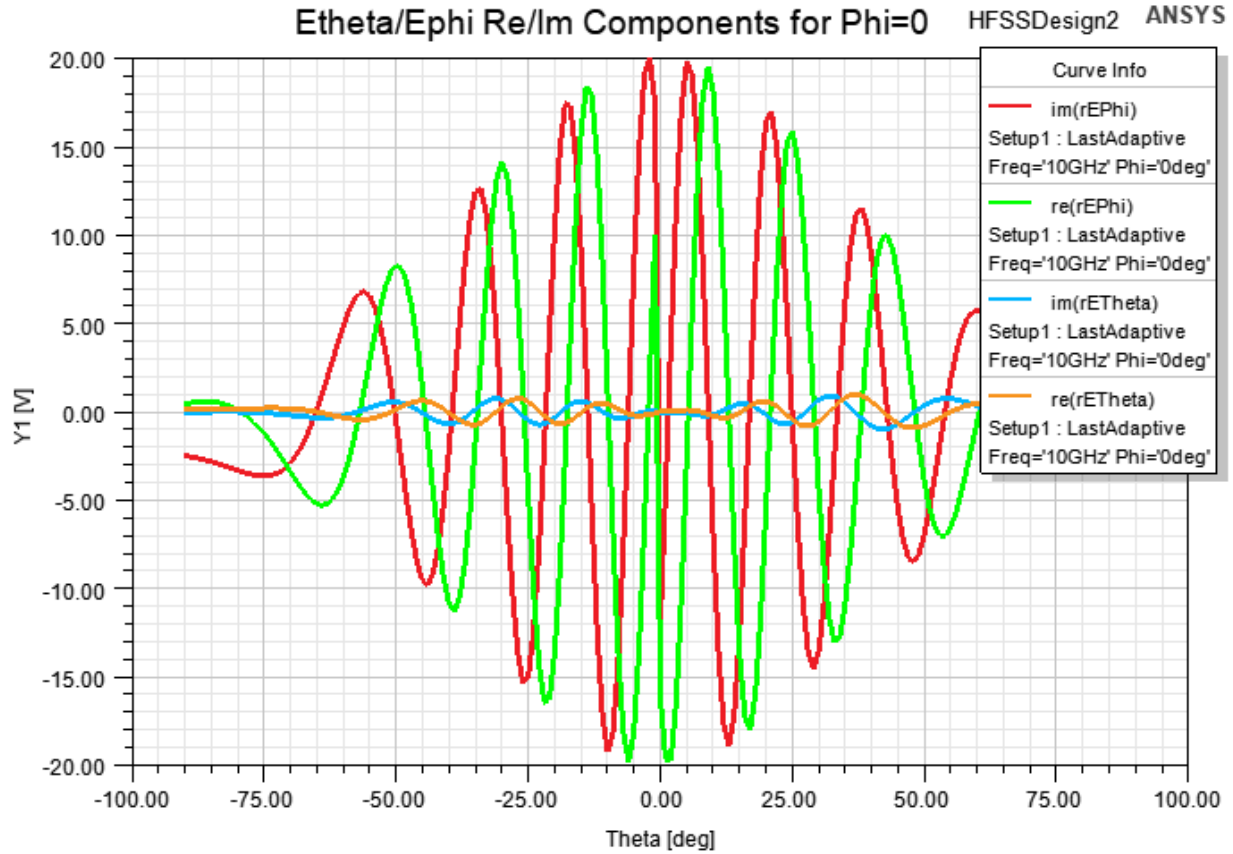
The last one asks for the array definition.

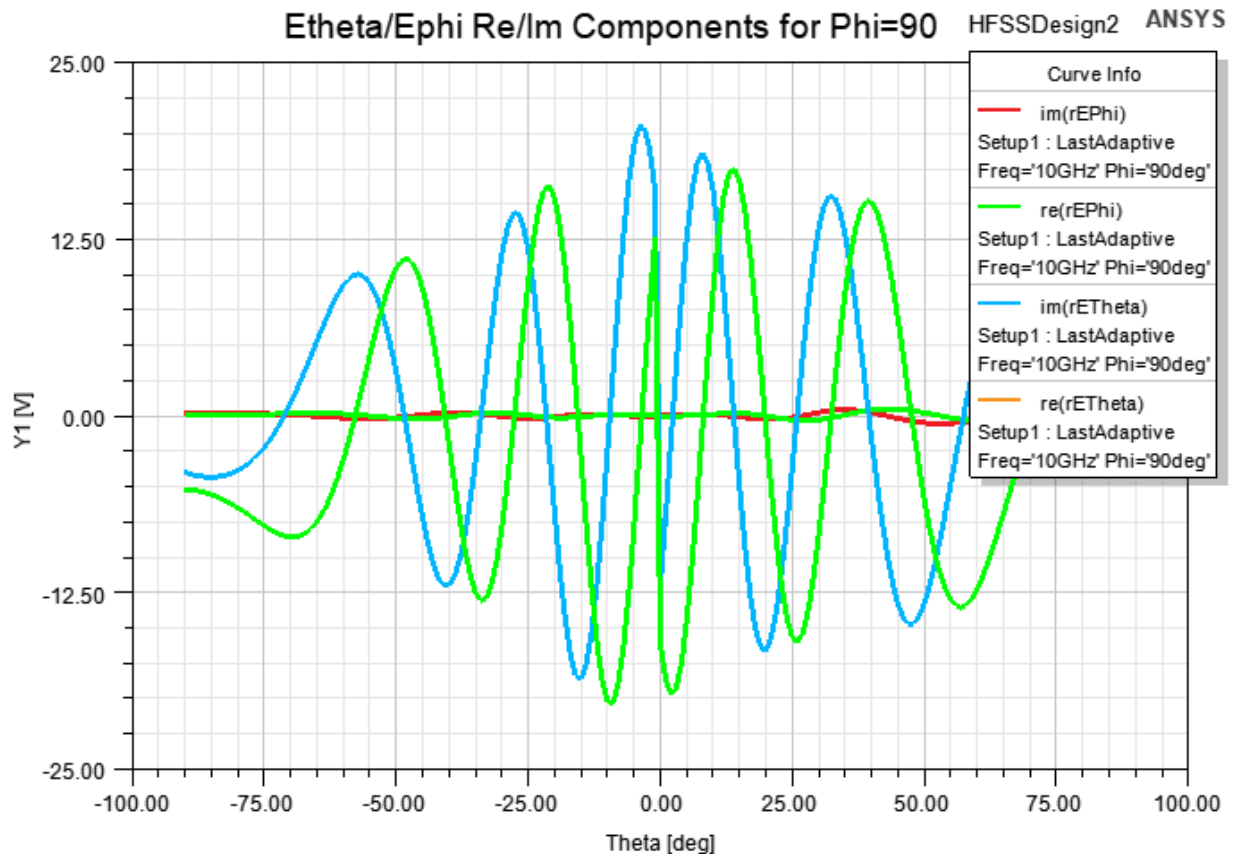


The following figure shows the waveguide array generated by the defaults.



After running the simulation, the plots defined by the script shows the following results.





The waveguide script in Python follows.

```
import clr

clr.AddReferenceByPartialName("Microsoft.VisualBasic")

from Microsoft.VisualBasic.Constants import
vbOKOnly, vbOKCancel, vbAbortRetryIgnore, vbYesNoCancel, vbYesNo, vbRetryC-
ancel

from Microsoft.VisualBasic.Constants import
vbOK, vbCancel, vbAbort, vbRetry, vbIgnore, vbYes, vbNo

from Microsoft.VisualBasic.Interaction import InputBox, MsgBox

oProject = oDesktop.GetActiveProject()

oDesign = oProject.GetActiveDesign()

oEditor = oDesign.SetActiveEditor("3D Modeler")
```

```
# Ask for dimensions for waveguide dimensions
# -----

dim = InputBox("Please enter the dimensions for the waveguide 'a' and
'b' dimensions "

    + "and the waveguide length in mm. Defaults are a = 23mm, b = 10mm,
    WaveguideLength = 25mm",

    "Waveguide array generator", "23,10,25")
Dimensions = dim.split(',')

a_str = Dimensions[0] + "mm"
b_str = Dimensions[1] + "mm"
b_over2 = float(Dimensions[1])/2

WaveguideLength_str = Dimensions[2] + "mm"

#Ask for the frequency of operation
#-----

Frequency = InputBox("Please enter the desired array frequency.
Distances will " +

    "be based on the free space wavelength at this frequency",

    "Waveguide array generator", "10GHz")

#Ask for the start, stop, and step of the interpolating frequency
sweep
#-----
--

inputText = InputBox("Please enter the start, stop, and step of the
interpolating " +

    "frequency sweep.", "Waveguide array generator",
    "8GHz,12GHz,50MHz")

StartFrequency, StopFrequency, StepFrequency = inputText.split(',')
```

```
#Ask for the number of elements in the x and y directions
#-----

inputText = InputBox("Please enter the number of elements in the x
and y directions.",

    "Waveguide array generator", "7,7")
numX, numY = [float(elem) for elem in inputText.split(',')]
TotalElements = numX*numY

# Make variables and set them equal to the values from the user input
# -----

oDesign.ChangeProperty(
    [
        "NAME:AllTabs",
        [
            "NAME:LocalVariableTab",
            [
                "NAME:PropServers",
                "LocalVariables"
            ],
            [
                "NAME:NewProps",
                [
                    "NAME:a", "PropType:=", "VariableProp", "UserDef:=", True,
                    "Value:=", a_str
                ],
            ],
            [
                "NAME:b", "PropType:=", "VariableProp", "UserDef:=", True,
                "Value:=", b_str
            ],
            [
```

```
"NAME:NumX", "PropType:=", "VariableProp", "UserDef:=", True,
"Value:=", numX
],
[
"NAME:NumY", "PropType:=", "VariableProp", "UserDef:=", True,
"Value:=", numY
],
[
"NAME:WaveguideLength", "PropType:=", "VariableProp", "User-
Def:=", True,
"Value:=", WaveguideLength_str
],
[
"NAME:Frequency", "PropType:=", "VariableProp", "UserDef:=",
True,
"Value:=", Frequency
],
[
"NAME:Lambda", "PropType:=", "VariableProp", "UserDef:=", True,
"Value:=", "c0/" + Frequency
],
[
"NAME:RadBoundDist", "PropType:=", "VariableProp", "UserDef:=",
True,
"Value:=", "Lambda/4"
]
]
]
])
#Create the radiation box
#-----
```

```
oEditor.CreateBox(  
  [  
    "NAME:BoxParameters",  
    "XPosition:=" , "-a/2-RadBoundDist",  
    "YPosition:=" , "-b/2-RadBoundDist",  
    "ZPosition:=" , "0mm",  
    "XSize:=" , "NumX*a+(NumX-1)*Lambda/2+2*RadBoundDist",  
    "YSize:=" , "NumY*b+(NumY-1)*Lambda/2+2*RadBoundDist",  
    "ZSize:=" , "RadBoundDist"  
  ],  
  [  
    "NAME:Attributes",  
    "Name:=" , "RadiationBox",  
    "Flags:=" , "",  
    "Color:=" , "(132 132 193)",  
    "Transparency:=" , 0.8,  
    "PartCoordinateSystem:=" , "Global",  
    "UDMId:=" , "",  
    "MaterialValue:=" , "\"vacuum\"",  
    "SurfaceMaterialValue:=" , "\"\"",  
    "SolveInside:=" , True,  
    "IsMaterialEditable:=" , True,  
    "UseMaterialAppearance:=" , False,  
    "IsLightweight:=" , False  
  ]  
)
```

```
oEditor.FitAll() # Zoom out  
#Create first element  
#-----  
oEditor.CreateBox(  

```

```
[
    "NAME:BoxParameters",
    "XPosition:=" , "-a/2",
    "YPosition:=" , "-b/2",
    "ZPosition:=" , "0mm",
    "XSize:=" , "a",
    "YSize:=" , "b",
    "ZSize:=" , "-WaveguideLength"
],
[
    "NAME:Attributes",
    "Name:=" , "Element1",
    "Flags:=" , "",
    "Color:=" , "(132 132 193)",
    "Transparency:=" , 0.8,
    "PartCoordinateSystem:=" , "Global",
    "UDMId:=" , "",
    "MaterialValue:=" , "\"vacuum\"",
    "SurfaceMaterialValue:=" , "\"\"",
    "SolveInside:=" , True,
    "IsMaterialEditable:=" , True,
    "UseMaterialAppearance:=" , False,
    "IsLightweight:=" , False
])

# Define the port
# -----
# Get the numeric value of half of the b dimension of the port.
# The values fed in to the "Start" and "End" arrays cannot have mathematical operators. For example, if HFSS
```


has a variable 'b', and a desired coordinate is 'b/2', that value cannot be entered here as "b/2". The number

has to be computed explicitly in script and entered as a string such as "-200mm".

```
oModule = oDesign.GetModule("BoundarySetup")
```

```
# get bottom face ID
```

```
element1FaceID = oEditor.GetFaceByPosition(
```

```
[
    "NAME:Parameters",
    "BodyName:=", "Element1",
    "XPosition:=", "-a/2",
    "YPosition:=", "-b/2",
    "ZPosition:=", "-WaveguideLength"
])
```

```
oModule.AssignWavePort(
```

```
[
    "NAME:WavePort1",
    "Faces:=", [element1FaceID],
    "NumModes:=", 1,
    "RenormalizeAllTerminals:=", True,
    "UseLineModeAlignment:=", False,
    "DoDeembed:=", False,
    [
        "NAME:Modes",
        [
            "NAME:Model",
            "ModeNum:=", 1,
            "UseIntLine:=", True,
            [
                "NAME:IntLine",
```

```
"Start:=" , ["0mm", "-" +str(b_over2) + "mm", "-" + Wave-
guideLength_str],
"End:=" , ["0mm", str(b_over2) + "mm", "-" + WaveguideLength_str]
],
"AlignmentGroup:=" , 0,
"CharImp:=" , "Zpi"
]
],
"ShowReporterFilter:=" , False,
"ReporterFilter:=" , [True],
"UseAnalyticAlignment:=", False
])

#Set the radiation boundary
#-----
# Get face IDs for further assignment
top_face_id = oEditor.GetFaceByPosition(["NAME:FaceParameters",
    "BodyName:=", "RadiationBox",
    "XPosition:=", "0mm",
    "YPosition:=", "NumY*b-b/2+(NumY-1)*Lambda/2+RadBoundDist",
    "ZPosition:=", "0mm"])

faceIDs = [int(elem) for elem in oEditor.GetFaceIDs("RadiationBox")
if elem != str(top_face_id)]
oModule.AssignRadiation(
[
    "NAME:Radiation",
    "Faces:=" , faceIDs,
    "IsFssReference:=" , False,
    "IsForPML:=" , False
])
```

```
# Copy and paste elements/ports into a rectangular array.
# Duplicate boundaries with geometry" must be turned on under Tools-
>Options->HFSS Options
# -----
-----

ElementNum = 1
for i in range(1, int(numX)+1):
    for j in range(1, int(numY)+1):
        if ElementNum == 1:
            pass

elif ElementNum <= numY: #If in the first column, only
    oEditor.Copy(["NAME:Selections", "Selections:=", "Element1"])
    oEditor.Paste()
    oEditor.Move(["NAME:Selections", "Selections:=", "Element" + str
(ElementNum)],
["NAME:TranslateParameters", "CoordinateSystemID:=", -1,
"TranslateVectorX:=", "0mm",
"TranslateVectorY:=", str(j-1) + "*" + (b+Lambda/2)",
"TranslateVectorZ:=", "0mm"]])

elif ElementNum > numY:
    oEditor.Copy(["NAME:Selections", "Selections:=", "Element1"])
    oEditor.Paste()
    oEditor.Move(["NAME:Selections", "Selections:=", "Element" + str
(ElementNum)],
["NAME:TranslateParameters", "CoordinateSystemID:=", -1,
"TranslateVectorX:=", str(i-1) + "*" + (a+Lambda/2)",
"TranslateVectorY:=", str(j-1) + "*" + (b+Lambda/2)",
"TranslateVectorZ:=", "0mm"]])
```

```
ElementNum += 1

#Create the setup and interpolating sweep
#-----
oModule = oDesign.GetModule("AnalysisSetup")
oModule.InsertSetup("HfssDriven",
[
    "NAME:Setup1",
    "AdaptMultipleFreqs:" , False,
    "Frequency:" , Frequency,
    "MaxDeltaS:" , 0.02,
    "PortsOnly:" , False,
    "UseMatrixConv:" , False,
    "MaximumPasses:" , 15,
    "MinimumPasses:" , 1,
    "MinimumConvergedPasses:" , 1,
    "PercentRefinement:" , 50,
    "IsEnabled:" , True,
    "BasisOrder:" , 1,
    "DoLambdaRefine:" , True,
    "DoMaterialLambda:" , True,
    "SetLambdaTarget:" , False,
    "Target:" , 0.3333,
    "UseMaxTetIncrease:" , False,
    "PortAccuracy:" , 2,
    "UseABConPort:" , False,
    "SetPortMinMaxTri:" , False,
    "UseDomains:" , False,
    "UseIterativeSolver:" , False,
    "SaveRadFieldsOnly:" , False,
```

```
"SaveAnyFields:=" , True,  
"IESolverType:=" , "Auto",  
"LambdaTargetForIESolver:=", 0.15,  
"UseDefaultLambdaTgtForIESolver:=", True  
])
```

```
oModule.InsertFrequencySweep("Setup1",  
[  
  "NAME:InterpolatingSweep",  
  "IsEnabled:=" , True,  
  "RangeType:=" , "LinearStep",  
  "RangeStart:=" , StartFrequency,  
  "RangeEnd:=" , StopFrequency,  
  "RangeStep:=" , StepFrequency,  
  "Type:=" , "Interpolating",  
  "SaveFields:=" , False,  
  "InterpTolerance:=" , 0.5,  
  "InterpMaxSolns:=" , 50,  
  "InterpMinSolns:=" , 0,  
  "InterpMinSubranges:=" , 1,  
  "ExtrapToDC:=" , False,  
  "InterpUseS:=" , True,  
  "InterpUsePortImped:=" , False,  
  "InterpUsePropConst:=" , True,  
  "UseDerivativeConvergence:=", False,  
  "InterpDerivTolerance:=", 0.2,  
  "UseFullBasis:=" , True,  
  "EnforcePassivity:=" , True,  
  "PassivityErrorTolerance:=", 0.0001  
])
```

```
#Create a relative coordinate system centered on the array for radiation pattern calculations

#-----

oEditor.CreateRelativeCS(

[
    "NAME:RelativeCSParameters",
    "Mode:=" , "Axis/Position",
    "OriginX:=" , "-a/2+NumX*a/2+(NumX-1)*Lambda/4",
    "OriginY:=" , "-b/2+NumY*b/2+(NumY-1)*Lambda/4",
    "OriginZ:=" , "0mm",
    "XAxisXvec:=" , "1mm",
    "XAxisYvec:=" , "0mm",
    "XAxisZvec:=" , "0mm",
    "YAxisXvec:=" , "0mm",
    "YAxisYvec:=" , "1mm",
    "YAxisZvec:=" , "0mm"
],
[
    "NAME:Attributes",
    "Name:=" , "RelativeCS1"
])

#Create an infinite sphere with fine theta resolution and phi cuts at 0 and 90 degrees

#-----

oModule = oDesign.GetModule("RadField")
oModule.InsertFarFieldSphereSetup(

[
    "NAME:Infinite Sphere1",
    "UseCustomRadiationSurface:=", False,
```

```
"ThetaStart:=" , "-90deg",
"ThetaStop:=" , "90deg",
"ThetaStep:=" , "1deg",
"PhiStart:=" , "0deg",
"PhiStop:=" , "90deg",
"PhiStep:=" , "90deg",
"UseLocalCS:=" , True,
"CoordSystem:=" , "RelativeCS1"
])
```

```
#Create output plots for Ephi/Etheta real and imaginary components
for Phi = 0 and separately for Phi = 90
```

```
#-----
-----
```

```
oModule = oDesign.GetModule("ReportSetup")
```

```
#For phi = 0
```

```
oModule.CreateReport("Etheta/Ephi Re/Im Components for Phi=0", "Far
Fields",
```

```
    "Rectangular Plot", "Setup1 : LastAdaptive",
```

```
    [
```

```
        "Context:=" , "Infinite Sphere1"
```

```
    ],
```

```
    [
```

```
        "Theta:=" , ["All"],
```

```
        "Phi:=" , ["0deg"],
```

```
        "Freq:=" , ["All"],
```

```
        "a:=" , ["Nominal"],
```

```
        "b:=" , ["Nominal"],
```

```
        "NumX:=" , ["Nominal"],
```

```
        "NumY:=" , ["Nominal"],
```

```
        "WaveguideLength:=" , ["Nominal"],
```

```
"Frequency:=" , ["Nominal"]
],
[
  "X Component:=" , "Theta",
  "Y Component:=" , ["im(rEPhi)", "re(rEPhi)", "im(rETheta)", "re
(rETheta)"]
], [])

#For phi = 90
#-----

oModule.CreateReport("Etheta/Ephi Re/Im Components for Phi=90", "Far
Fields",
  "Rectangular Plot", "Setup1 : LastAdaptive",
  [
    "Context:=" , "Infinite Sphere1"
  ],
  [
    "Theta:=" , ["All"],
    "Phi:=" , ["90deg"],
    "Freq:=" , ["All"],
    "a:=" , ["Nominal"],
    "b:=" , ["Nominal"],
    "NumX:=" , ["Nominal"],
    "NumY:=" , ["Nominal"],
    "WaveguideLength:=" , ["Nominal"],
    "Frequency:=" , ["Nominal"]
  ],
  [
    "X Component:=" , "Theta",
    "Y Component:=" , ["im(rEPhi)", "re(rEPhi)", "im(rETheta)", "re
(rETheta)"]
  ]
]
```


], [])

Index

3

3D Modeler editor commands 8-1

AlignFaces 8-144

AssignMaterial 8-145

AssignSurfaceMaterial 8-275

Chamfer 8-148

ChangeProperty 6-12, 8-279,
15-8

CleanUpModel 8-150

CloseAllWindows 8-285

Connect 8-151

Copy 8-123

CoverLines 8-152

CoverSurfaces 8-153

Create3DComponent 8-6

CreateBondwire 8-11

CreateBox 8-15

CreateCircle 8-18

CreateCone 8-20

CreateCutplane 8-23

CreateCylinder 8-25

CreateEllipse 8-28

CreateEntityList 8-154

CreateEquationCurve 8-30

CreateEquationSurface 8-34

CreateFaceCS 8-155, 8-161

CreateGroup 8-160

CreateHelix 8-37

CreateObjectFromEdges 8-168

CreateObjectFromFace 8-170

CreateObjectFromFaces 8-172

CreatePoint 8-39

CreatePolyline 8-41

CreateRectangle 8-46

CreateRegularPolygon 8-49

CreateRegularPolyhedron 8-52

CreateRelativeCS 8-173

CreateSphere 8-55

CreateSpiral 8-57

CreateTorus 8-59

CreateUserDefinedModel 8-62

CreateUserDefinedPart 8-71

Defeature 8-286

Delete 8-287

DeleteEmptyGroups 8-175

DeleteLastOperation 8-176

DeleteOperation 8-177

DeletePolylinePoint 8-124

DetachEdges 8-179

DetachFaces 8-180

DuplicateAlongLine 8-125	GetEdgeIDsfromFace 8-304
DuplicateAroundAxi 8-127	GetEdgeIDsfromObject 8-304
DuplicateMirror 8-130	GetEdgeLength 8-305
EditEntityList 8-182	GetEdgePositionAtNormalizedParameter 7-31
EditFaceCS 8-183	GetEntityListIDByName 8-306
EditObjectCS 8-189	GetFaceArea 8-308
EditPolyline 8-88	GetFaceByPosition 8-309
EditRelativeCS 8-196	GetFaceCenter 8-308
Export 8-198	GetFaceIDFromNameForFirstOperation 8-311
ExportModelImagetoFile 8-200, 10-52	GetFacelDs 8-311
Fillet 8-204	GetFacelDsOfSheet 8-312
FlattenGroup 8-207	GetModelBoundingBox 8-313
GenerateAllUserDefinedModels 8-288	GetPartsForUserDefinedModel 8-323
GenerateHistory 8-207	GetPoints 8-324
GenerateUserDefinedModel 8-289	GetProperties 2-45, 6-52, 7-40
Get3DComponentParameters 8-95	GetPropertyValue 2-46, 6-53, 7-41, 8-325, 10-77, 21-105
Get3DComponentPartNames 8-96	GetPropEvaluatedValue 6-48, 8-325
GetActiveCoordinateSystem 8-208	GetPropSIValue 6-50, 8-328
GetAct- iveCoordin- ateSystemTransform 8-209	GetRelativeCoordinateSystems 8-330
GetBodyNamesByPosition 8-292	GetUser Position 8-332
GetCoordinateSystems 8-210	GetVertexIDFromNameForFirstOperation 8-333
GetEdgeByPosition 8-301	GetVertexIDsFromEdge 8-334
GetEdgeIDFromNameForFirstOperation 8-303	GetVertexIDsFromFace 8-335
	GetVertexIDsFromObject 8-335
	GetVertexPosition 8-336
	GetWireBodyNames 8-337

HealObject 8-210	SetModelUnits 8-244
Import 8-215	SetPropValue Modeler 8-342
ImportDXF 8-218	SetTopDownViewDir- ectionForActiveView 8-343
ImportFromClipboard 8-223	SetTopDownViewDir- ectionForAllViews 8-344
ImportGDSII 8-223	SetWCS 8-245
Imprint 8-227	Simplify 8-247
ImprintProjection 8-228	Split 8-249
Insert3DComponent 8-96	Stitch 8-252
InsertComponent 8-97	Subtract 8-253
InsertNativeComponent 8-99	SweepAlongPath 8-110
Intersect 8-230	SweepAlongVector 8-112
Mirror 8-133	SweepAroundAxis 8-114
Move 8-135	SweepFacesAlongNormal 8-116, 8-116, 8-255, 8-255
MoveCSToEnd 8-231	Sweep- FacesAlongNormalWithAt- tributes 8-118
MoveEntityToGroup 8-232	ThickenSheet 8-257
MoveFaces 8-233	UncoverFaces 8-258
OffsetFaces 8-137	Ungroup 8-262
PageSetup 4-50, 8-337	Unite 8-260
ProjectSheet 8-236	UpdateComponentDefinition 8- 121
RemoveBadEdges 8-339	UpdatePriorityList 8-345
RemoveBadFaces 8-340	UpgradeVersion 8-345
RemoveBadVertices 8-340	Validate3DComponent 8-347
RenamePart 8-341	WrapSheet 8-262
Replacewith3DComponent 8- 238	WriteHistoryTreeLayoutForTest 8- 348
Rotate 8-139	
Scale 8-140	
Section 8-241	
SeparateBody 8-243	

3D Modeler editor object commands

GetNumObjects 8-315
GetObjectIDByName 8-315
GetObjectName 8-316
GetObjectNameByEdgeID 8-317
GetObjectNameByFaceID 8-317
GetObjectNameByID 8-318
GetObjectNameByVertexID 8-319
GetObjectShapeType 8-320
GetObjectVolume 8-322
GetObjPath 8-323

A

Aborting Scripts 1-11
Add 21-2, 21-32, 21-39, 21-126, 21-200, 21-251, 21-285, 21-317
AddAddSolverOnDemandModel 21-150
AddAllEyeMeasurements 10-3
AddAssignmentToBoundary 11-3
AddCartesianLimitLine 10-4
AddCartesianLimitLineFromCurve 10-6
AddCartesianLimitLineFromEquation 10-8
AddCartesianXMarker 10-10
AddCartesianYMarker 10-10
AddCartesianYMarkerToStack 10-11

AddDataset 6-4, 7-4, 21-20
AddDefinitionFromBlock 8-266, 21-23
AddDefinitionFromLibFile 8-271
AddDeltaMarker 10-12
AddDiffPair 19-4
Additional Property Scripting Example 2-52
AddMarker 10-13
AddMarker[Fields Reporter] 15-1
AddMarkerToPlot 15-2
AddMaterial 6-7, 21-28, 21-53
AddMessage 4-6
AddModelingProperties 7-7
AddNote 10-14
AddNPortData 21-144
AddTraceCharacteristics 10-16
AddTraces 10-18
AlignFaces 8-144
Analysis module commands
 AnalyzeAll 7-7, 7-23, 7-48
 AnalyzeAllNominal 7-8
 CopyDrivenSetup 13-4
 CopyEigenSetup 13-5
 CopySetup 13-3, 14-6, 14-45
 DeleteSetups 13-6
 EditSetup 13-7, 14-8, 14-48
 GetSetupCount 13-21
 GetSetups 13-22
 GetSweeps 13-23

InsertSetup 13-23	AssignCylindricalSupport 11-203
PasteDrivenSetup 13-42	AssignCylindricalWave 11-35
PasteEigenSetup 13-43	AssignDielectricCavity 11-38
PasteSetup 13-43	AssignEMLoss 11-205, 11-208
RenameSetup 13-44	AssignFEBl 11-39
ResetToTimeZero 13-44	AssignFiniteCond 11-40
RevertAllToInitial 13-45	AssignFixedSupport 11-211
RevertAllToZeroDisplacement 13-46	AssignFloquet 11-43
RevertSetupToInitial 13-47	AssignForce 11-213
RevertSetupToInitialConditions 13-48	AssignFresnel 11-50
RevertSetupToZeroDisplacement 13-49	AssignFrictionlessSupport 11-217
RevertToInitialCondition 7-55	AssignGaussianBeam 11-51
SolveSetup 13-49	AssignGradientSurfaceRoughness 11-55, 11-136
Analysis Setup Module Script Com- mands 13-1	AssignHalfSpace 11-57
Ansoft Application Object com- mands 3-1	AssignHeatFlux 11-219
GetAppDesktop 3-2	AssignHeatGeneration 11-222
ApplyReportTemplate 10-20	AssignHertzianDipoleWave 11-58
AreMaterialPropertiesEqual 21-49	AssignHybridRegion 11-60
AreSurfaceMaterialPropertiesEqual 21-50	AssignIERegion 11-62
AreThereSimulationsRunning 4-7	AssignImpedance 11-63
arithmetic operators 1-6	AssignIncidentWave 11-65
array variables 1-4	AssignInitialTemperature 11-224
AssignCircuitPort 11-32	AssignInteriorNearFieldSource 11-62
AssignConvection 11-195, 11-199	AssignLatticePair 11-69
AssignCurrent 11-34	AssignLayeredImp 11-71
	AssignLinearAntennaWave 11-74
	AssignLinkedImpedance 11-78
	AssignLinkedRegion 11-80

AssignLumpedPort 11-82
AssignLumpedRLC 11-85
AssignMagneticBias 11-88
AssignMaterial 8-145
AssignMultipactionChargeRegion 11-90
AssignMultipactionDCBias 11-92
AssignMultipactionSEE 11-96
AssignPartialDischargeDCBias 11-106
AssignPerfectE 11-97
AssignPerfectH 11-99
AssignPlaneWave 11-100
AssignPointMonitor 12-1
AssignPressure 11-226
AssignRadiation 11-104
AssignRotatingFluid 11-230
AssignScreeningImpedance 11-107
AssignSurfaceMaterial 8-275
AssignSymmetry 11-111
AssignTemperature 11-232
AssignTerminal 11-112
AssignThermalCondition 11-234
AssignVoltage 11-112
AutoCreatePECCapforWavePort 11-114
AutoidentifyLatticePair 11-4
AutoidentifyNets 11-4
AutoidentifyPorts 11-5

AutoidentifyTerminals 11-6

B

Boundary and Excitation Module Script Commands 11-1

Boundary module commands

EditAperture 11-119

Boundary/Excitation module commands

AddAssignmentToBoundary 11-3

AssignCircuitPort 11-32

AssignCurrent 11-34

AssignCylindricalWave 11-35

AssignDielectricCavity 11-38

AssignFEBI 11-39

AssignFiniteCond 11-40

AssignFloquet 11-43

AssignFresnel 11-50

AssignGaussianBeam 11-51

AssignGradientSurfaceRoughness 11-55, 11-136

AssignHalfSpace 11-57

AssignHertzianDipoleWave 11-58

AssignHybridRegion 11-60

AssignIERegion 11-62

AssignImpedance 11-63

AssignIncidentWave 11-65

AssignInteriorNearFieldSource 11-62

AssignLatticePair 11-69

AssignLayeredImp 11-71

AssignLinearAntennaWave 11-74

AssignLinkedImpedance 11-78	ver-
AssignLinkedRegion 11-80	tNportCircuitElementsToPorts 11-8
AssignLumpedPort 11-82	CreateNportCircuitElements 11-9
AssignLumpedRLC 11-85	DeleteAllBoundaries 11-11
AssignMagneticBias 11-88	DeleteAllExcitations 11-12
AssignMultipactionChargeRegion 11-90	DeleteBoundaries 11-12
AssignMultipactionDCBias 11- 92	EditAnisotropicImpedance 11-116
AssignMultipactionSEE 11-96	EditCircuitPort 11-120
AssignPerfectE 11-97	EditDielectricCavity 11-122
AssignPerfectH 11-99	EditDiffPairs 11-123
AssignPlaneWave 11-100	EditFEBl 11-125
AssignRadiation 11-104	EditFiniteCond 11-126
AssignRFDDischargeDCBias 11- 106	EditFloquetPort 11-127
AssignScreeningImpedance 11- 107	EditFresnel 11-134
AssignSymmetry 11-111	EditGlobalMatEnv 11-135
AssignTerminal 11-112	EditHalfSpace 11-137
AssignVoltage 11-112	EditHybridRegion 11-139
AutoCreatePECCapforWavePort 11-114	EditImpedance 11-140
AutoIdentifyLatticePair 11-4	EditIncidentWave 11-141
AutoIdentifyNets 11-4	EditInteriorNearFieldSource 11- 144
AutoIdentifyPorts 11-5	EditLatticePair 11-145
AutoIdentifyTerminals 11-6	EditLayeredImp 11-146
ChangeImpedanceMult 11-7	EditLinkedImpedance 11-149
CircuitPortToLumpedPort 11- 115	EditLinkedRegion 11-151
Con-	EditLumpedPort 11-153
	EditLumpedRLC 11-156
	EditMagneticBias 11-158

EditMultipactionChargeRegion 11-160	GetNumHybridRegions 11-23
EditMultipactionDCBias 11-161	GetNumHybridRegionsOfType 11-24
EditMultipactionSEE 11-165	GetPortExcitationCounts 11-24
EditNportCircuitElement 11-13	LumpedPortToCircuitPort 11-180
EditPerfectE 11-168	ReassignBoundaries 11-2, 11-25
EditPerfectH 11-169	ReassignBoundary 11-25
EditRadiation 11-170	RenameBoundary 11-26
EditRFDISchargeDCBias 11-167	ReprioritizeBoundaries 11-27
EditSymmetry 11-172	SetAllHybridRegionsToOneWayCoupled 11-181
EditTerminal 11-173	SetAllHybridRegionsToTwoWayCoupled 11-181
EditVoltage 11-174	SetDefaultBaseName 11-28
EditVoltageDrop 11-176	SetHybridRegionCoupledGroup 11-182
EditWavePort 11-176	SetHybridRegionsCoupling 11-185
GetBoundaries 11-14	SetSBRCreepingWaveSettings 11-186
GetBoundariesOfType 11-15	SetSBRTxRxSettings 11-188
GetBoundaryAssignment 11-16	SetScatteredFieldFormulation 11-186
GetDefaultBaseName 11-16	SetTerminalReferenceImpedances 11- 190
GetDiffPairs 11-17	SetTotalFieldFormulation 11-191
GetExcitations 11-18	SwapCircuitPortDirection 11-191
GetExcitationsOfType 11-18	SwapLatticePair 11-192
GetHybridRegions 11-19	UnassignIERegions 11-193
GetHybridRegionssofType 11- 20	boundary/excitation script commands 11-2
GetNumBoundaries 11-20	BreakUDMConnection 8-278
GetNumBoundariesOfType 11- 21	BringToFront 21-327
GetNumExcitations 11-22	
GetNumExcitationsOfType 11- 22	

C

CalcOp 15-5

- CalcStack 15-5
- CalculatorRead 15-6
- CalculatorWrite 15-7
- Cascade 19-5
- Chamfer 8-148
- ChangeGeomSettings 15-13
- ChangeImpedanceMult 11-7
- ChangeProperty 6-12, 8-279, 15-8
- CircuitPortToLumpedPort 11-115
- ClcEval 15-13
- ClcMaterial 15-14
- ClcMaterialValue 15-15
- CleanUpModel 8-150
- ClearAllMarkers 10-21
- ClearAllMarkers[Fields Reporter] 15-16
- ClearAllNamedExpr 15-16
- ClearAllTraceCharacteristics 10-22
- ClearDiffPairs 19-6
- ClearMessages 4-7, 6-18
- ClearSolutionCache 21-150
- Clone 19-7
- CloneMaterial 6-19, 21-51
- CloneReportsFromDatasetSolution 10-22
- Close 6-19, 19-8
- CloseAllWindows 4-11, 8-285
- CloseProject 4-12
- CloseProjectNoForce 4-13
- Combine 19-9
- comment lines 1-12, 1-14
- comparison operators 1-7
- ComplInstance Functions 60
- ComplInstance Script Commands (multiple)
- Component Manager Script Commands 21-125
- conditional statements
 - If...Then... Else 1-8
 - Select Case 1-8
 - types of 1-8
- Connect 8-151
- ConstructVariationString 7-10
- conventions
 - scripting help 1-1
- converting data types 1-10
- ConvertNportCircuitElementsToPorts 11-8
- ConvertToDynamic 21-262
- ConvertToParametric 21-262
- Copy 8-123
- CopyArray 7-10
- CopyDesign 6-20
- CopyDrivenSetup 13-4
- CopyEigenSetup 13-5
- CopyItemCommand 7-11
- CopyNamedExprToStack 15-17
- CopyPlotSettings 10-23
- CopyReportDefinition 10-24
- CopyReportsData 10-25

Copyright and Trademark Information 2

CopySetup

- Analysis module command 13-3, 14-6, 14-45
- Optimetrics module command 13-3, 14-6, 14-45

CopyTraceDefinitions 10-25

CopyTracesData 10-26

Core Global Script Context Commands 22-1

Count 4-14

CoverLines 8-152

CoverSurfaces 8-153

CPython 1-70

Create3DComponent 8-6

CreateBondwire 8-11

CreateBox 8-15

CreateCircle 8-18

CreateCone 8-20

CreateCutplane 8-23

CreateCylinder 8-25

CreateEllipse 8-28

CreateEntityList 8-154

CreateEquationCurve 8-30

CreateEquationSurface 8-34

CreateFaceCS 8-155, 8-161

CreateFieldPlot 15-18

CreateGroup 8-160

CreateHelix 8-37

CreateNportCircuitElements 11-9

CreateObjectFromEdges 8-168

CreateObjectFromFace 8-170

CreateObjectFromFaces 8-172

CreateOutputVariable 9-1

CreatePoint 8-39

CreatePolyline 8-41

CreateRectangle 8-46

CreateRegularPolygon 8-49

CreateRegularPolyhedron 8-52

CreateRelativeCS 8-173

CreateReport 10-27

CreateReportFromFile 10-36

CreateReportFromTemplate 10-37

CreateReportofAllQuantities 10-38

CreateSphere 8-55

CreateSpiral 8-57

CreateTorus 8-59

CreateUserDefinedModel 8-62

CreateUserDefinedPart 8-71

CreateUserDefinedSolution 18-1

CutDesign 6-21

D

dataset commands

- AddDataset 6-4, 7-4, 21-20
- DeleteDataset 6-22, 7-11, 21-52
- EditDataset 6-24, 21-81
- ExportDataset 6-36, 7-24, 21-99

- ImportDataset 6-57, 7-49, 21-107
- Deembed 19-10
- DeembedBack 19-11
- DeembedFront 19-13
- Defeature 8-286
- Definition Manager commands
 - AddDefinitionFromBlock 8-266, 21-23
 - AddDefinitionFromLibFile 8-271
 - GetExtendedDefinitionObject 8-307
 - GetProjectMaterialNames 21-105
 - UpdateDefFromBlock 21-119, 21-121
- Definition Manager Script Commands 21-1
- Delete 8-287
- DeleteAllBoundaries 11-11
- DeleteAllExcitations 11-12
- DeleteAllReports 10-40
- DeleteBoundaries 11-12
- DeleteDataset 6-22, 7-11, 21-52
- DeleteDesign 6-23
- DeleteEmptyGroups 8-175
- DeleteFieldPlot 15-30
- DeleteFieldVariation 7-12
- DeleteFullVariation 7-13
- DeleteLastOperation 8-176
- DeleteLinkedDataVariation 7-14
- DeleteMarker 10-39
- DeleteMarker[Fields Reporter] 15-31
- DeleteNamedExpr 15-31
- DeleteOperation 8-177
- DeleteOutputVariable 7-15, 7-18, 9-2
- DeletePlotEntities 7-15
- DeletePolylinePoint 8-124
- DeleteReport 10-41
- DeleteSetups 13-6
 - Analysis module command 13-5
 - Optimetrics module command 14-7, 14-46
- DeleteSolutionVariation 7-17
- DeleteTraceCharacteristics 10-41
- Deletetraces 10-42
- DeleteUneditablePlot 15-32
- DeleteUserDefinedSolutions 18-3
- Design object commands 10-64, 10-65, 10-67, 10-67, 10-68, 10-83
 - AddCartesianLimitLine 10-4
 - AddCartesianXMarker 10-10
 - AddCartesianYMarker 10-10
 - AddDeltaMarker 10-12
 - AddMarker 10-13
 - AddMarkerToPlot 15-2
 - AddNote 10-14
 - AddTraceCharacteristics 10-16
 - AddTraces 10-18
 - CalculatorRead 15-6

ClearAllMarkers 10-21	GetLibraryDirectory 4-30
ClearAllTraceCharacteristics 10-22	GetMatchedObjectName 8-313
CloseAllWindows 4-11	GetModelUnits 8-314
CopyReportDefinition 10-24	GetModule 7-33, 7-34
CreateOutputVariable 9-1	GetName 7-35, 10-75, 14-33, 14-76
CreateReportFromTemplate 10-37	GetObjectsByMaterial 8-319
DeleteAllReports 10-40	GetObjectsInGroup 8-321
DeleteOutputVariable 7-15, 7-18, 9-2	GetObjPath 7-37, 10-76, 14-33, 14-77
DeleteReport 10-41	GetOutputVariableValue 7-38, 9-7
DeleteTraces 10-42	GetProjectDirectory 4-36
DoesOutputVariableExist 9-3	GetProperties 2-45, 6-52, 7-40
Edit Layout 7-20	GetPropertyValue 2-46, 6-53, 7-41, 8-325, 10-77, 21-105
EditMarker 10-46	GetPropEvaluatedValue 6-48, 8-325
EditOutputVariable 9-4	GetPropNames 7-43
ExportConvergence 7-23	GetPropSIValue 6-50, 8-328
ExportModelMeshToFile 8-204, 10-56	GetRegistryInt 4-38, 4-80
ExportPlot3DToFile 10-57	GetRegistryString 4-39, 4-81
ExportPlotImageToFile 15-48	GetSelections 8-331
ExportPlotImageWithViewToFile 15-49	GetSolutionType 7-45
ExportProfile 7-25	GetSolveInsideThreshold 7-45
ExportReport 10-58	GetSubGroupsInGroup 8-331
ExportToFile 10-61	GetTempDirectory 4-43
Fast Transformation 7-62, 7-63	GetVariables 2-48, 6-56, 7-46
GeometryCheckAndAutofix 8-290	GetVariableValue 2-48, 6-55, 7-47
GetDisplayType 10-72	GetVariationVariableValue 7-48
	GetVersion 4-44
	ImportIntoReport 10-100
	IsFeaturEnabled 4-45

PasteDesign 7-52	ValidateDesign 7-77
PasteReports 10-106	Design Object Script Commands 7-1
PasteReportsWithLegacyNames 10-106	Desktop Commands For Registry Val- ues 4-78
PasteTracesWithLegacyNames 10-108	Desktop object commands
Redo 7-53	AddMessage 4-6
RenameDesignInstance 7-54	AreThereSimulationsRunning 4-7
RenameReport 10-108	ClearMessages 4-7
RenameTraces 10-109	CloseProject 4-12
SARSetup 7-56	CloseProjectNoForce 4-13
SavePlotSettingsAsDefault 10- 111	Count 4-14
SetActiveEditor 7-56	DownloadJobResults 4-15
SetAllowMaterialOverride 7-57	EnableAutosave 4-18
SetDesignSettings 7-59	ExportOptionsFiles 4-19
SetLibraryDirectory 4-68	GetActiveProject 4-19
SetProjectDirectoryVBCommand> 4-69	GetActiveScheduler 4-20
SetPropertyValue 2-49, 6-73, 7- 67, 21-118	GetActiveSchedulerInfo 4-21
SetSolutionType 7-70	GetAutosaveEnabled 4-22
SetSolveInsideThreshold 7-72	GetBuildTimeDateString 4-23
SetTempDirectory 4-73	GetChildNames 6-39, 7-26
SetVariableValue 2-51, 6-75, 7- 74	GetChildNames Modeler 8-293
Show Layout 7-68, 7-69	GetChildTypes 6-41, 7-28, 8-299
ShowWindow 8-246	GetChildTypes Optimetrics 14-32, 14-76
Solve 7-74, 7-76	GetCustomMenuSet 4-24
StopSimLink 7-75	GetDesigns 6-44
UpdateAllReports 10-114	GetDesktopConfiguration 4-26
	GetDistributedAnalysisMachines 4-27

GetDis-
trib-
utedAna-
lysisMachinesForDesignType
4-28

GetMonitorData 4-33

GetProjectList 4-36

GetProjects 4-32, 4-37

GetPropNames 6-49, 14-36, 14-
79

GetSchematicEnvironment 4-40

KeepDesktopResponsive 4-46

LaunchJobMonitor 4-47

OpenMultipleProjects 4-48

OpenProject 4-49

PauseRecording 4-52

PauseScript 4-53

Print 4-54

QuitApplication 4-54

RefreshJobMonitor 4-55

ResetLogging 4-56

RestoreWindow 4-58

ResumeRecording 4-58

RunProgram 4-59

RunScript 4-60

SelectScheduler 4-63

SetActiveProject 4-65

SetActiveProjectByPath 4-65

SetCustomMenuSet 4-66

SetDesktopConfiguration 4-67

SetSchematicEnvironment 4-72

Sleep 4-74

StopSimulations 4-75

SubmitJob 4-76

TileWindows 4-77

Desktop Object Script Commands 4-1, 11-
194

Desktop Scripting Conventions 1-85

DetachEdges 8-179

DetachFaces 8-180

DisableDiffPairs 19-14

DistributedAnalyzeSetup, Optimetrics mod-
ule command 14-7, 14-47

DoesNamedExpressionExists 15-33

DoesOutputVariableExist 9-3

DoesSupportTraceCharacteristics 10-43

Draw Menu commands 8-4

DumpAllReportsData 10-44

DuplicateAlongLine 8-125

DuplicateAroundAxis 8-127

DuplicateMirror 8-130

E

Edit 21-54, 21-92, 21-151, 21-223, 21-262,
21-291, 21-328

Edit Menu Commands 8-122

Edit3DComponent 8-79

Edit3DComponentDefinition 8-81

EditAnisotropicImpedance 11-116

EditAperture 11-119

EditCartesianXMarker 10-45

EditCartesianYMarker 10-45	EditLayeredImp 11-146
EditCircuitPort 11-120	EditLayoutForLayoutComponent 7-20
EditConvection 11-195, 11-199	EditLinkedImpedance 11-149
EditCoSimulationOptions 7-19	EditLinkedRegion 11-151
EditCylindricalSupport 11-203	EditLumpedPort 11-153
EditDataset 6-24, 21-81	EditLumpedRLC 11-156
EditDielectricCavity 11-122	EditMagneticBias 11-158
EditDiffPairs 11-123	EditMarker 10-46
EditEMLoss 11-205, 11-208	EditMaterial 6-27, 21-83
EditEntityList 8-182	EditMultipactionChargeRegion 11-160
EditFaceCS 8-183	EditMultipactionDCBias 11-161
EditFEBI 11-125	EditMultipactionSEE 11-165
EditFieldsSummarySetting 16-1	EditNativeComponentDefinition 8-83
EditFiniteCond 11-126	EditNportCircuitElement 11-13
EditFixedSupport 11-211	EditObjectCS 8-189
EditFloquetPort 11-127	Editor object commands
EditForce 11-213	SetPropertyValue 2-49, 6-73, 7-67, 21-118
EditFresnel 11-134	EditOutputVariable 9-4
EditFrictionlessSupport 11-217	EditPartialDischargeDCBias 11-167
EditGlobalMatEnv 11-135	EditPerfectE 11-168
EditHalfSpace 11-137	EditPerfectH 11-169
EditHeatFlux 11-219	EditPolyline 8-88
EditHeatGeneration 11-222	EditPressure 11-226
EditHybridRegion 11-139	EditRadiation 11-170
EditImpedance 11-140	EditRelativeCS 8-196
EditIncidentWave 11-141	EditRotatingFluid 11-230
EditInfiniteArray 7-19	
EditInteriorNearFieldSource 11-144	
EditLatticePair 11-145	

EditSetup 13-7, 14-8, 14-48
 optimization command 14-96
 parametric command 14-87
 sensitivity command 14-109
 statistical command 14-120
EditSolverOnDemandModel 21-178
EditSymmetry 11-172
EditTemperature 11-232
EditTerminal 11-173
EditThermalCondition 11-234
EditUserDefinedSolution 18-4
EditVoltage 11-174
EditVoltageDrop 11-176
EditWavePort 11-176
EditWithComps 21-178, 21-223, 21-263, 21-298, 21-328
EMDesignOptions 7-20
EnableAutoSave 4-18
EnableDiffPairs 19-15
EnableSetup 14-21, 14-61
EnterComplex 15-34
EnterComplexVector 15-34
EnterCoord 15-35
EnterEdge 15-36
EnterLine 15-37
EnterOutputVar 15-37
EnterPoint 15-38
EnterQty 15-39
EnterScalar 15-40
EnterScalarFunc 15-40
EnterSurf 15-41
EnterVector 15-42
EnterVectorFunc 15-42
EnterVol 15-43
Event Callback Scripting 1-88
Example Scripts 23-1
Export 8-198, 21-100, 21-102, 21-103, 21-190, 21-241, 21-274, 21-305, 21-339
ExportCitiFile 19-16
ExportConvergence 7-23
ExportDataset 6-36, 7-24, 21-99
ExportDOELocalSensitivity 14-72
ExportDOELocalSensitivityCurve 14-73
ExportDOEResponseCurve 14-69
ExportDOEResponseCurveSlices 14-70
ExportDOEResponseSurface 14-71
ExportDXConfigFile, Optimetrics module command 14-22, 14-61
ExportEyeMaskViolation 10-47
ExportFieldPlot 15-44
ExportFieldsSummary 16-2
ExportFullWaveSpice 19-18, 21-281
ExportImageToFile 10-48
ExportMarkerTable 15-45
ExportMaterial 6-36, 21-101
ExportMatlab 19-19
ExportModelMeshToFile 8-204, 10-56
ExportOptimetricsProfile, Optimetrics module command 14-23, 14-62

ExportOptimetricsResults, Optimetrics module command 14-24, 14-63

ExportOptionsFiles 4-19

ExportOutputVariables 9-5

ExportParametricResults, Optimetrics module command 14-25, 14-64

ExportPlot3DToFile 10-57

ExportPlotImageToFile 15-48

ExportPlotImageWithViewToFile 15-49

ExportProfile 7-25

ExportReport 10-58

ExportReportDataToFile 10-60

ExportRespSurfaceMinMaxTable 14-27, 14-66

ExportRespSurfaceRefinePoints 14-27, 14-67

ExportRespSurfaceResponsePoints 14-28, 14-68

ExportRespSurfaceVerificationPoints 14-29, 14-69

ExportScript 21-103, 21-313

ExportSpreadsheet 19-24

ExportTableToFile 10-60

ExportToFile 15-51

ExportTouchstone 19-26

ExportTouchstone2 19-28

ExportUniformPointsToFile 10-62

Extract 19-31

F

Field Calculator commands

DoesNamedExpressionExists 15-33

Field Overlay module commands, GetFieldPlotNames 15-53

field overlay script commands 15-1

Field Overlays module commands

CalcOp 15-5

CalcStack 15-5

ChangeGeomSettings 15-13

ClcMaterial 15-14

ClearAllNamedExpr 15-16

CopyNamedExprToStack 15-17

CreateFieldPlot 15-18

DeleteFieldPlot 15-30

DeleteNamedExpr 15-31

DeleteUneditablePlot 15-32

EnterComplex 15-34

EnterComplexVector 15-34

EnterCoord 15-35

EnterEdge 15-36

EnterLine 15-37

EnterOutputVar 15-37

EnterPoint 15-38

EnterQty 15-39

EnterScalar 15-40

EnterScalarFunc 15-40

EnterSurf 15-41

- EnterVector 15-42
 - EnterVectorFunc 15-42
 - EnteVol 15-43
 - ExportFieldPlot 15-44
 - ExportOnGrid 15-45
 - GetFieldFolderNames 15-52
 - GetFieldPlotQuantityName 15-53
 - GetMeshPlotNames 15-54
 - ModifyFieldPlot 15-57
 - ReassignFieldPlot 15-59
 - RenameFieldPlot 15-62
 - RenamePlotFolder 15-63
 - SaveFieldsPlots 15-64
 - SetFieldPlotSettings 15-66
 - SetPlotFolderSettings 15-68
 - UpdateAllFieldsPlots 15-74
 - Fields Calculator commands
 - AddNamedExpr 15-4
 - CalcOp 15-5
 - CalcStack 15-5
 - CalculatorWrite 15-7
 - ChangeGeomSettings 15-13
 - ClcEval 15-13
 - ClcMaterial 15-14
 - ClcMaterialValue 15-15
 - ClearAllNamedExpr 15-16
 - CopyNamedExprToStack 15-17
 - DeleteNamedExpr 15-31
 - EnterComplex 15-34
 - EnterComplexVector 15-34
 - EnterCoord 15-35
 - EnterEdge 15-36
 - EnterLine 15-37
 - EnterOutputVar 15-37
 - EnterPoint 15-38
 - EnterQty 15-39
 - EnterScalar 15-40
 - EnterScalarFunc 15-40
 - EnterSurf 15-41
 - EnterVector 15-42
 - EnterVectorFunc 15-42
 - EnterVol 15-43
 - Fields reporter module commands
 - AddMarker[Fields Reporter] 15-1
 - ClearAllMarkers[Fields Reporter] 15-16
 - DeleteMarker[Fields Reporter] 15-31
 - ExportMarkerTable 15-45
 - Fields Summary Script Commands 16-1
 - Fillet 8-204
 - FlattenGroup 8-207
 - For...Next loop 1-9
 - functions
 - VBScript procedures 1-10
- G**
- GenerateAllUserDefinedModels 8-288
 - GenerateHistory 8-207

GenerateUserDefinedModel 8-289	GetBodyNamesByPosition 8-292
GenerateVariationData Parametric, parametric command 14-30, 14-74, 14-89	GetBoundaries 11-14
GeometryCheckAndAutofix 8-290	GetBoundariesOfType 11-15
Get3DComponentDefinitionNames 8-92	GetBoundaryAssignment 11-16
Get3DComponentInstanceNames 8-93	GetBuildDateTimeString 4-23
Get3DComponentMaterialNames 8-94	GetChild Object (Report Setup), Report module command 10-69
Get3DCom- ponentMaterialProperties 8- 94	GetChildNames 6-39, 7-26
Get3DComponentParameters 8-95	GetChildNames (Optimetrics), Report module command 10-69
Get3DComponentPartNames 8-96	GetChildNames Modeler 8-293
GetActiveCoordinateSystem 8-208	GetChildObject Design 6-40, 7-26
GetAct- iveCoordin- ateSystemTransform 8-209	GetChildObject Modeler 8-296
GetActiveDesign 6-37	GetChildTypes 6-41, 7-28, 8-299
GetActiveProject 4-19	GetChildTypes Optimetrics 14-32, 14-76
GetActiveScheduler 4-20	GetChildTypes ReportSetup 10-71
GetActiveSchedulerInfo 4-21	GetComponentName 60
GetAllCategories 10-64	GetCoordinateSystems 8-210
GetAllQuantities 10-65	GetCurvePropServerName 10-71
GetAllReportNames 10-66	GetData 21-191, 21-242, 21-246, 21- 275, 21-307, 21-340
GetArrayVariables 2-44, 6-38	GetDataExpressions 10-85
GetAutoSaveEnabled 4-22	GetDataUnits 10-86
GetAvailableDisplayTypes 10-67	GetDefaultBaseName 11-16
GetAvailableReportTypes 10-68	GetDefinitionManager 6-42
GetAvailableSolutions 10-68	GetDependentFiles 6-43
	GetDesignID 7-29
	GetDesigns 6-44
	GetDesignType 7-30

GetDesignVariableNames 10-87	GetFaceArea 8-308
GetDesignVariableUnits 10-88	GetFaceByPosition 8-309
GetDesignVariableValue 10-89	GetFaceCenter 8-308
GetDesignVariationKey 10-90	GetFaceIDFromNameForFirstOperation 8-311
GetDiffPairs 11-17	GetFaceIDs 8-311
GetDisplayType 10-72	GetFaceIDsOfSheet 8-312
GetDistributedAnalysisMachines 4-27	GetFieldFolderNames 15-52
GetDis-trib-utedAnalysisMachinesForDesignType 4-28	GetFieldPlotNames, Field Overlay module command 15-53
GetDocumentNames 17-8	GetFieldPlotQuantityName 15-53
GetDynLinkIntrinsicVariables 10-73	GetFrequencies 19-32
GetDynLinkQtyValueState 10-73	GetFrequencyCount 19-33
GetDynLinkTraces 10-74	GetHybridRegions 11-19
GetDynLinkVariableValues 10-75	GetHybridRegionsOfType 11-20
GetEdgeByPosition 8-301	GetImagDataValues 10-90
GetEdgeIDFromNameForFirstOperation 8-303	GetInstanceID 61
GetEdgeIDsFromFace 8-304	GetInstanceName 62
GetEdgeIDsFromObject 8-304	GetLibraryDirectory 4-30
GetEdgeLength 8-305	GetMatchedObjectName 8-313
GetEdgePositionAtNormalizedParameter 7-31	GetMeshPlotNames 15-54
GetEntityListIDByName 8-306	GetModelBoundingBox 8-313
GetExcitations 11-18	GetModelUnits 8-314
GetExcitationsOfType 11-18	GetModule 7-33, 7-34
GetExeDir 4-29	GetMonitorData 4-33
GetExtendedDefinitionObject 8-307	GetName 6-46, 7-35, 10-75, 14-33, 14-76, 19-34
	GetNames 21-192, 21-243, 21-247, 21-276, 21-307, 21-341
	GetNPortData 21-195

GetNumBoundaries 11-20	GetPoints 8-324
GetNumBoundariesOfType 11-21	GetPortCount 19-34
GetNumExcitations 11-22	GetPortExcitationCounts 11-24
GetNumExcitationsOfType 11-22	GetPortNumber 19-35
GetNumHybridRegions 11-23	GetPostProcSettings 19-37
GetNumHybridRegionsOfType 11-24	GetProcessID 4-35
GetNumObjects 8-315	GetProjectDirectory 4-36
GetObjectIDByName 8-315	GetProjectList 4-36
GetObjectName 8-316	GetProjectMaterialNames 21-105
GetObjectNameByEdgeID 8-317	GetProjects 4-32, 4-37
GetObjectNameByFaceID 8-317	GetProperties 2-45, 6-52, 7-40, 21-248
GetObjectNameByID 8-318	GetPropertyValue 2-46, 6-53, 7-41, 8-325, 10-77, 21-105
GetObjectNameByVertexID 8-319	GetPropEvaluatedValue 6-48, 8-325
GetObjectsByMaterial 8-319	GetPropNames 6-49, 7-43, 10-78, 14-36, 14-79
GetObjectShapeType 8-320	GetPropNames Modeler 8-327
GetObjectsIngroup 8-321	GetPropServerName 63
GetObjectVolume 8-322	GetPropSIValue 6-50, 8-328
GetObjPath 6-47, 7-37, 8-323, 10-76, 14-33, 14-77	GetPropValue Modeler 8-329
GetOptimetricsResults, Optimetrics module command 14-35	GetPropValue Optimetrics 14-37, 14-79
GetOutputVariableValue 7-38, 9-7	GetPropValue Project 6-51, 7-44
GetParentDesign 62	GetPropValue Reporter 10-79
GetPartsForUserDefinedModel 8-323	GetQtyExpressionsForSourceTrace 10-80
GetPath 6-48	GetRealDataValues 10-92
GetPer-QuantityPrimarySweepValues 10-91	GetRegistryInt 4-38, 4-80
GetPersonalLibDirectory 4-34	GetRegistryString 4-39, 4-81

- GetRelativeCoordinateSystems 8-330
 - GetReportSum-
maryForRegressionTesting 10-81
 - GetReportTraceNames 10-80
 - GetSelections 7-45, 8-331
 - GetSetupCount 13-21
 - GetSetupCount, Analysis module command 13-21
 - GetSetupNames (Optimetrics), Optimetrics module command 14-31, 14-31, 14-37, 14-74, 14-75, 14-80
 - GetSetupNamesByType (Optimetrics), Optimetrics module command 14-38, 14-81
 - GetSetups 13-22
 - Analysis module command 13-22
 - GetSolutionContexts 10-82
 - GetSolutionDataPerVariation 10-83
 - GetSolutionType 7-45
 - GetSolutionVariation 19-38
 - GetSolveInsideThreshold 7-45
 - GetSolverOnDemandData 21-196
 - GetSolverOnDemandModelList 21-196
 - GetSubGroupsInGroup 8-331
 - GetSweepNames 10-93
 - GetSweeps 13-23
 - Analysis module command 13-22
 - GetSweepUnits 10-94
 - GetSweepValues 10-95
 - GetSysLibDirectory 4-42
 - GetTempDirectory 4-43
 - GetTool 4-84
 - GetTopDesignList 6-55
 - GetTopEntryValue 15-55
 - GetUserDefinedSolutionNames 18-6
 - GetUserDefinedSolutionProperties 18-6
 - GetUserLibDirectory 4-43
 - GetUserPosition 8-332
 - GetVariables 2-48, 6-56, 7-46
 - GetVariableValue 2-48, 6-55, 7-47
 - GetVariation 19-39
 - GetVariationVariableValue 7-48
 - GetVersion 4-44
 - GetVertexIDFromNameForFirstOperation 8-333
 - GetVertexIDsFromEdge 8-334
 - GetVertexIDsFromFace 8-335
 - GetVertexIDsFromObject 8-335
 - GetVertexPosition 8-336
 - GetWireBodyNames 8-337
 - GroupPlotCurvesByGroupingStrategy 10-99
- H**
- HasSameData 19-40
 - HealObject 8-210
 - hierarchy of variables in HFSS 1-76

I

Icepak boundary condition commands

ExportDOELocalSensitivity 14-72

ExportDOELocalSensitivityCurve 14-73

ExportDOEResponseCurve 14-69

ExportDOEResponseCurveSlices 14-70

ExportDOEResponseSurface 14-71

ImportIDX 4-99

If...Then... Else statement 1-8

Import (3D Modeler command) 8-215

ImportANF 4-85, 4-87

ImportAutoCAD 4-88

ImportAWRMicrowaveOffice 4-89

ImportDataset 6-57, 7-49, 21-107

ImportDXF 8-218

ImportEDB 4-91

ImportExport Tool 4-84

ImportExtracta 4-92

ImportFromClipboard 8-223

ImportGDSII 4-93, 4-94, 8-223

ImportIDF 4-95, 4-98

ImportIDX 4-99

ImportIntoReport 10-100

ImportIPC 4-102

ImportODB 4-103

ImportOutputVariables 9-8

ImportReportDataIntoReport 10-101

ImportSetup, Optimetrics module command 14-39, 14-82

ImportXFL 4-104

Imprint 8-227

ImprintProjection 8-228

include files

scripts 1-10

indentation, IronPython 1-21

InputBox function 1-11

Insert3DComponent 8-96

InsertComponent 8-97

InsertDesignWithWorkflow 6-59

InsertNativeComponent 8-99

InsertSetup

Analysis module command 13-23

optimization command 14-101

parametric command 14-89

sensitivity command 14-116

statistical command 14-123

Intersect 8-230

IronPython 1-15

indentation in 1-21

IsDataComplex 10-96

IsFeatureEnabled 4-45

IsPerQuantityPrimarySweep 10-97

IsUsed 21-196, 21-243, 21-249, 21-277, 21-308, 21-342

K

KeepDesktopResponsive 4-46

keywords, VBScript 1-12, 1-14

L

LaunchJobMonitor 4-47

Layout Scripting 1-88

LoadNamedExpressions 15-56

LoadSolution 19-41

logical operators 1-7

looping through code

 Do ... Loop 1-8

 For ... Next 1-8

LumpedPortToCircuitPort 11-180

M

material commands

 AddDefinitionFromBlock 8-266, 21-23

 AddDefinitionFromLibFile 8-271

 AddMaterial 6-7, 21-28, 21-53

 AreMaterialPropertiesEqual 21-49

AreSurfaceMaterialPropertiesEqual 21-50

CloneMaterial 6-19, 21-51

EditMaterial 6-27, 21-83

ExportMaterial 6-36, 21-101

GetExtendedDefinitionObject 8-307

RemoveMaterial 6-63, 21-112

UpdateDefFromBlock 21-119

UpdateDefFromBlockEx 21-121

Materials Scripting Support 2-22

Mechanical initial condition commands

 AssignInitialTemperature and EditInitialTemperature 11-224

Microsoft

 VBScript user's guide 1-11

Mirror 8-133

Model Manager Script Commands 21-250

Modeler Menu Commands 8-142

ModifyFieldPlot 15-57

ModifyLibraries 21-313

module script commands

 boundary/excitation 11-2

 field overlay 15-1

modules in HFSS scripting 1-76

Move 8-135

MoveCSToEnd 8-231

MoveEntityToGroup 8-232

MoveFaces 8-233

MovePlotCurveToGroup 10-102

MovePlotCurveToNewGroup 10-103

MsgBox function 1-11

N

Named Arguments 1-85

NdExplorer

Script Commands 19-1

Network Data Explorer Manager
Commands 21-280

O

oAnsoftApp object 1-76

Object Oriented Property
Scripting 2-1

oDesign object 1-76

oDesktop object 1-76

oEditor object 1-76

OffsetFaces 8-137

oModule object 1-76

Open 19-43

OpenAndConvertProject 4-47

OpenMultipleProjects 4-48

OpenProject 4-49

OpenWindowForAllReports 10-103

OpenWindowForReports 10-104

operators

arithmetic 1-6

categories in VBScript 1-5

comparison 1-7

concatenation 1-7

logical 1-7

precedence of 1-5

oProject object 1-76

Optimetrics module commands

DeleteSetups 14-7, 14-46

DistributeAnalyzeSetup 14-7, 14-47

EnableSetup 14-21, 14-61

ExportDXConfigFile 14-22, 14-61

ExportOptimetricsProfile 14-23, 14-62

ExportOptimetricsResults 14-24, 14-63

ExportParametricResults 14-25, 14-64

ExportRespSurfaceMinMaxTable 14-27, 14-66

ExportRespSurfaceRefinePoints 14-27, 14-67

ExportRespSurfaceResponsePoints 14-28, 14-68

ExportRespSurfaceVerificationPoints 14-29, 14-69

GetChildNames 14-31, 14-31, 14-75, 14-75

GetOptimetricResult 14-34, 14-78

GetOptimetricsResults 14-35

GetSetupNames 14-37, 14-80

GetSetupNamesByType 14-38, 14-81

ImportSetup 14-39, 14-82

RenameSetup 14-41, 14-84

SolveSetup 13-3, 14-6, 14-43, 14-44, 14-45, 14-85, 14-86

Optimetrics Module Script
Commands 14-1

optimization commands

EditSetup 14-96

- InsertSetup 14-101
- Optimization Script Commands 14-96
- Other oEditor Commands 8-264
- output variable commands
 - CreateOutputVariable 9-1
 - DeleteOutputVariable 7-15, 7-18, 9-2
 - DoesOutputVariableExist 9-3
 - EditOutputVariable 9-4
 - GetOutputVariableValue 7-38, 9-7
- Output variable commands
 - ExportOutputVariables 9-5
 - ImportOutputVariables 9-8
- Output Variable Script Commands 9-1

P

- PageSetup 4-50, 8-337
- parametric commands
 - EditSetup 14-87
 - GenerateVariationData Parametric 14-30, 14-74, 14-89
 - InsertSetup 14-89
- Parametric Script Commands 14-87
- Paste 6-61
- Paste (Model Editor) 8-138
- Paste (Project Object) 6-61
- PasteArray 7-51
- PasteDesign 7-52

- PasteDrivenSetup 13-42
- PasteEigenSetup 13-43
- PastePlotSettings 10-105
- PasteReports 10-106
- PasteReportsWithLegacyNames 10-106
- PasteSetup
 - Analysis module command 13-43
 - Optimetrics module command 14-40, 14-83
- PasteTraces 10-107
- PasteTracesWithLegacyNames 10-108
- PauseRecording 4-52
- PauseScript 4-53
- Print 4-54
- Project object commands
 - AddDataset 6-4, 7-4, 21-20
 - AddMaterial 6-7, 21-28, 21-53
 - AreMaterialPropertiesEqual 21-49
 - AreSurfaceMaterialPropertiesEqual 21-50
 - CloneMaterial 6-19, 21-51
 - Close 6-19
 - CopyArray 7-10
 - CopyDesign 6-20
 - CutDesign 6-21
 - DeleteDataset 6-22, 7-11, 21-52
 - DeleteDesign 6-23
 - EditDataset 6-24, 6-36, 6-57, 7-24, 7-49, 21-81, 21-99, 21-107
 - EditMaterial 6-27, 21-83

ExportMaterial 6-36, 21-101	GetVariableValue 2-48, 6-55, 7-47
GetActiveDesign 6-37	LoadSolution 19-41
GetArrayVariables 2-44, 6-38	Paste 6-61, 6-61
GetChildObject Design 6-40, 7-26	PasteArray 7-51
GetChildObject Modeler 8-296	Redo 6-62
GetDependentFiles 6-43	RemoveAllUnusedDefinitions 6-62
GetDesignID 7-29	RemoveMaterial 6-63, 21-112
GetDesignType 7-30	RemoveUnusedDefinitions 6-64, 21-116
GetName 6-46	Rename 6-65
GetObjPath 6-47	RestoreProjectArchive 4-57
GetPath 6-48	Save 6-66
GetProjectMaterialNames 21-105	SaveAs 6-67
GetProperties 2-45, 6-52, 7-40	SaveAsStandAloneProject 6-69
GetPropertyValue 2-46, 6-53, 7-41, 8-325, 10-77, 21-105	SaveProjectArchive 6-70
GetPropEvaluatedValue 6-48, 8-325	SetActiveDesign 6-72
GetPropNames Modeler 8-327	SetPropertyValue 2-49, 6-73, 7-67, 21-118
GetPropSIValue 6-50, 8-328	SetPropValue Design 7-66
GetPropvalue Modeler 8-329	SetPropValue Optimetrics 14-42, 14-85
GetPropvalue Optimetrics 14-37, 14-79	SetPropValue Project 6-72, 10-112
GetPropvalue Project 6-51, 7-44	SetVariableValue 2-51, 6-75, 7-74
GetPropvalue Reporter 10-79	SimulateAll 6-11, 6-76
GetSolutionVariation 19-38	Undo 6-76
GetTopDesignList 6-55	UpdateDefinitions 6-77, 21-124
GetTopEntryValue 15-55	Project Object Script Commands 6-1
GetVariables 2-48, 6-56, 7-46	ProjectSheet 8-236

property commands

GetArrayVariables 2-44, 6-38
GetProjectMaterialNames 21-105
GetProperties 2-45, 6-52, 7-40
GetPropertyValue 2-46, 6-53, 7-41, 8-325, 10-77, 21-105
GetPropEvaluatedValue 6-48, 8-325
GetPropSIValue 6-50, 8-328
GetTopEntryValue 15-55
GetVariables 2-48, 6-56, 7-46
GetVariableValue 2-48, 6-55, 7-47
SetPropertyValue 2-49, 6-73, 7-67, 21-118
SetVariableValue 2-51, 6-75, 7-74

Property Script Commands 2-30

Conventions uSed in thie Chapter 2-41
Object Script Property Function Summary 2-32

PurgeHistory 8-237

Q

QuitApplication 4-54

R

ReassignBoundaries 11-2, 11-25
ReassignBoundary 11-25
ReassignFieldPlot 15-59
ReassignPointMonitor 12-3

Record Script and Edit Properties 2-51

Redo

design-level command 7-53
project-level command 6-62

references, for VBScript 1-11

RefreshJobMonitor 4-55

ReleaseData 10-98

Remove 21-109, 21-111, 21-113, 21-115, 21-197, 21-244, 21-278, 21-308, 21-342

RemoveAllUnusedDefinitions 6-62

RemoveBadEdges 8-339

RemoveBadFaces 8-340

RemoveBadVertices 8-340

RemoveMaterial 6-63, 21-112

RemoveScript 21-114, 21-315

RemoveSolverOnDemandModel 21-198

RemoveUnused 21-199, 21-245, 21-250, 21-279, 21-309, 21-343

RemoveUnusedDefinitions 6-64, 21-116

Rename 6-65, 19-43

RenameBoundary 11-26

RenameDesignInstance 7-54

RenameFieldPlot 15-62

RenamePart 8-341

RenamePlotFolder 15-63

RenameReport 10-108

RenameSetup 13-44

Analysis module command 13-44

- Optimetrics module
 - command 14-41, 14-84
- RenameSource [Interface Source] 7-55
- RenameTraces 10-109
- Renormalize 19-45
- Reorder 19-46, 19-47
- Repeating a Statement Until a Condition Becomes True 1-9
- Repeating Statements While a Condition is True 1-9
- Replacewith3DComponent 8-238
- Report module commands
 - GetChildNames 10-69
- Reporter editor commands
 - AddAllEyeMeasurements 10-3
 - AddCartesianLimitLine 10-4
 - AddCartesianLimitLineFromCurve 10-6
- AddCartesianLimitLineFromEquation 10-8
- AddCartesianXMarker 10-10, 10-10
- AddCartesianYMarkerToStack 10-11
- AddDeltaMarker 10-12
- AddMarker 10-13
- AddMarkerToPlot 15-2
- AddNote 10-14
- AddTraceCharacteristics 10-16
- AddTraces 10-18
- ApplyReportTemplate 10-20
- ClearAllMarkers 10-21
- ClearAllTraceCharacteristics 10-22
- CloneReportsFromDatasetSolution 10-22
- CopyPlotSettings 10-23
- CopyReportDefinition 10-24
- CopyReportsData 10-25
- CopyTraceDefinitions 10-25
- CopyTracesData 10-26
- CreateReport 10-27
- CreateReportFromFile 10-36
- CreateReportFromTemplate 10-37
- CreateReportOfAllQuantities 10-38
- DeleteAllReports 10-40
- DeleteReport 10-41
- DeleteTraceCharacteristics 10-41
- DeleteTraces 10-42
- DoesSupportTraceCharacteristics 10-43
- DumpAllReportsData 10-44
- EditCartesianXMarker 10-45
- EditCartesianYMarker 10-45
- EditMarker 10-46
- ExportEyeMaskViolation 10-47
- ExportImageToFile 10-48
- ExportPlot3DToFile 10-57
- ExportPlotImageToFile 15-48, 15-49
- ExportReport 10-58

ExportReportDataToFile 10-60	83
ExportTableToFile 10-60	GroupPlotCurvesByGroupingStrategy 10-99
ExportToFile 10-61, 10-100	ImportReportDataIntoReport 10-101
ExportUniformPointsToFile 10-62	MovePlotCurveToGroup 10-102
GetAllCategories 10-64	MovePlotCurveToNewGroup 10-103
GetAllQuantities 10-65	OpenWindowForAllReports 10-103
GetAllReportNames 10-66	OpenWindowForReports 10-104
GetAvailableDisplayTypes 10-67	PastePlotSettings 10-105
GetAvailableReportTypes 10-68	PasteReports 10-106
GetAvailableSolutionss 10-68	PasteReportsWithLegacyNames 10-106
GetChildTypes ReportSetup 10-71	PasteTraces 10-107
GetCurvePropServerName 10-71	PasteTracesWithLegacyNames 10-108
GetDisplayType 10-72	RenameReport 10-108
GetDynLinkIntrinsicVariables 10-73	RenameTraces 10-109
GetDynLinkQtyValueState 10-73	ResetPlotSettings 10-110
GetDynLinkTraces 10-74	SavePlotSettingsAsDefault 10-111
GetDynLinkVariableValues 10-75	SetLinkOutputTraces 10-111
GetPropNames 10-78	UnGroupPlotCurvesInGroup 10-113
GetQtyExpressionsForSourceTrace 10-80	UpdateAllReports 10-114
GetRe- portSum- maryForRegressionTesting 10-81	UpdateReports 10-115
GetReportTraceNames 10-80	UpdateTraces 10-115
GetSolutionContexts 10-82, 10-	UpdateTracesContextAndSweeps 10-118
	Reporter Editor Script Commands 10-1
	Reporter module commands
	GetChildObject 10-70
	ReprioritizeBoundaries 11-27
	Reset 19-48

ResetLogging 4-56
ResetPlotSettings 10-110
RestoreWindow 4-58
RestToTimeZero 13-44
ResumeRecording 4-58
RevertAllToInitial 13-45
RevertAllToZeroDisplacement 13-46
RevertSetupToInitial 13-47
RevertSetupToInitialCondition 13-48
RevertSetupToZeroDisplacement 13-49
Rotate 8-139
Running Instance Manager Script Commands 5-1
RunProgram 4-59
RunScript 4-60
RunScriptWithArguments 4-62

S

sample scripts
 simple HFSS 1-12
 simple Q3D Extractor 1-14
 variable helix 23-16
 waveguide 23-18
SARSetup 7-56
Save 6-66
SaveAs 6-67
SaveAsStandAloneProject 6-69
SaveFieldsPlots 15-64

SaveNamedExpressions 15-65
SavePlotSettingsAsDefault 10-111
Scale 8-140
Scope and Lifetime of Variables 1-4
Script and Library Scripts 21-310
script commands
 boundary/excitation 11-2
 field overlay 15-1
Script Commands for Creating and Modifying Boundaries 11-29
scripts
 CPython 1-70
 IronPython 1-15
 overview 1-1
 pausing 1-81
 recording 1-81
 running 1-80
 stopping 1-80
 VBScript 1-2
Scripts and Locked Layers 1-88
Section 8-241
Select Case statement 1-8
SelectScheduler 4-15, 4-63
sensitivity commands
 EditSetup 14-109
 InsertSetup 14-116
Sensitivity Script Commands 14-109
SeparateBody 8-243
SetActiveDefinitionEditor 6-71

SetActiveDesign 6-72	SetProjectDirectory 4-69
SetActiveEditor 7-56	SetPropertyValue 2-49, 6-73, 7-67, 21-118
SetActiveProject 4-65	SetPropValue Design 7-66
SetActiveProjectByPath 4-65	SetPropValue Modeler 8-342
SetAllHybridRegionsToOneWayCoupled 11-181	SetPropValue Optimetrics 14-42, 14-85
SetAllHybridRegionsToTwoWayCoupled 11-181	SetPropValue Project 6-72, 10-112
SetAllowMaterialOverride 7-57	SetSBRCreepingWaveSettings 11-186
SetAllPortImpedance 19-49	SetSBRTxRxSettings 11-188
SetBackgroundMaterial 7-58	SetScatteredFieldFormulation 11-186
SetDefaultBaseName 11-28	SetShowLayoutForLayoutComponent 7-68, 7-69
SetDesignSettings 7-59	SetSolutionType 7-70
SetFastTransformationForLayoutComponent 7-62, 7-63	SetSolveInsideThreshold 7-72
SetFieldPlotSettings 15-66	SetSourceContexts 7-73
SetHybridRegionCoupledGroup 11-182	SetTempDirectory 4-73
SetHybridRegionsCoupling 11-185	SetTerminalReferenceImpedances 11-190
SetLengthSettings 7-63	Setting Numerical Values 1-87
SetLibraryDirectory 4-68	SetTopDownViewDirectionForActiveView 8-343
SetLinkOutputTraces 10-111	SetTopDownViewDirectionForAllViews 8-344
SetModelUnits 8-244	SetTotalFieldFormulation 11-191
SetPlotFolderSettings 15-68	SetVariableValue 2-51, 6-75, 7-74
SetPortDeembedDistance 19-50	SetWCS 8-245
SetPortImpedance 19-51	SGetAppDesktop 3-1
SetPostProcSettings 19-52	ShowWindow 8-246
	simple and composite names 1-3
	Simplify 8-247
	SimulateAll 6-11, 6-76

- SimValueContext 9-9
 - Sleep 4-74
 - Smooth 19-53
 - Solution module commands
 - SetSourceContexts 7-73
 - Solutions module commands
 - ConstructVariationString 7-10
 - DeleteFieldVariation 7-12
 - DeleteFullVariation 7-13
 - DeleteLinkedDataVariation 7-14
 - DeleteLotEntities 7-15
 - DeleteSolutionVariation 7-17
 - Solve 7-74
 - SolveAllSetup, Optimetrics module command 14-43, 14-85
 - SolveSetup 13-49
 - SolveSetup, Optimetrics module command 14-44, 14-86
 - Split 8-249
 - statistical commands
 - EditSetup 14-120
 - InsertSetup 14-123
 - Statistical Script Commands 14-120
 - Stitch 8-252
 - StopSimLink 7-75
 - StopSimulations 4-75
 - Stretch 19-55
 - string concatenation operator 1-7
 - Sub procedures 1-12, 1-14
 - Sub Procedures 1-10
 - SubmitJob 4-76
 - Subtract 8-253
 - SwapCircuitPortDirection 11-191
 - SwapLatticePair 11-192
 - SweepAlongPath 8-110
 - SweepAlongVector 8-112
 - SweepAroundAxis 8-114
 - SweepFacesAlongNormal 8-116, 8-116, 8-255, 8-255
 - Sweep-FacesAlongNormalWithAttributes 8-118
 - Symbol Manager Script Commands 21-316
- T**
- Terminate 19-56
 - Thermal Monitor 12-1
 - ThickenSheet 8-257
- U**
- UnassignIERegions 11-193
 - UncoverFaces 8-258
 - underscore (_) character 1-12, 1-14
 - Undo
 - project-level command 6-76
 - Ungroup 8-262
 - UnGroupPlotCurvesInGroup 10-113
 - Unite 8-260
 - UpdateAllFieldsPlots 15-74
 - UpdateAllReports 10-114

UpdateComponentDefinition 8-121
UpdateDefFromBlock 21-119
UpdateDefFromBlockEx 21-121
UpdateDefinitions, project-level
 command 6-77, 21-124
UpdateDynamicLink 21-200
UpdatePriorityList 8-345
UpdateReports 10-115
UpdateTraces 10-115
UpdateTracesContextAndSweeps
 10-118
UpgradeVersion 8-345
User defined documents module
 commands
 GetDocumentNames 17-8
User defined solution module com-
 mands
 CreateUserDefinedSolution 18-
 1
 DeleteUserDefinedSolutions 18-
 3
 EditUserDefinedSolution 18-4
 GetUserDefinedSolutionNames
 18-6
 GetUserDefinedSolutionProperties
 18-6
User Defined Solutions
 Commands 18-1
Using a Do Loop 1-9

V

Validate3DComponent 8-347
ValidateDesign 7-77

Variable Naming Conventions 1-4

variables

 array 1-4
 assigning information 1-3
 declaring 1-3
 hierarchy in HFSS 1-76
 used as objects 1-12
 used in HFSS scripts 1-76

VBScript 1-2

 Microsoft user's guide 1-11
 operators 1-5
 references 1-11
 Sub procedures 1-12, 1-14

VBScript Procedures 1-9

W

WrapSheet 8-262

WriteHistoryTreeLayoutForTest 8-348