



Icepak 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 2023 R2
July 2023

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

Copyright and Trademark Information

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

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

Disclaimer Notice

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

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

U.S. Government Rights

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

Third-PartySoftware

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

Table of Contents

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	1-6
Arithmetic Operators	1-6
String Concatenation Operator	1-7
Comparison Operators	1-7
Logical Operators	1-8
Controlling Program Execution	1-8
Using If...Then...Else	1-8
Using Select Case	1-8
Looping Through Code	1-9
Using a For...Next Loop	1-9
Using a Do Loop	1-9
Repeating Statements While a Condition is True	1-9
Repeating a Statement Until a Condition Becomes True	1-10
VBScript Procedures	1-10
Function Procedures	1-10

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

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

Return Values for UDM and UDP Functions	1-41
Constants	1-42
Methods	1-42
Output Parameters	1-42
Comparison with C function:	1-43
'List Size' Parameters	1-43
Comparison with C function:	1-44
Added Parameters	1-45
Developing a UDM/UDP	1-46
Creation	1-46
Location	1-46
Organize	1-46
Edit/Reload	1-46
UDPExtension	1-47
Import	1-47
Main class: UDPExtension	1-47
IUDPExtension methods	1-47
Mandatory methods	1-47
GetLengthParameterUnits()	1-47
GetPrimitiveTypeInfo()	1-47
GetPrimitiveParametersDefinition2()	1-47
AreParameterValuesValid2(errorMsg, udpParams)	1-48
CreatePrimitive2(funcLib, udpParams)	1-48
Optional methods	1-48
GetPrimitiveParameters()	1-48
GetRegisteredFaceNames()	1-48
GetRegisteredEdgeNames()	1-48
GetRegisteredVertexNames()	1-48
MapParametersDefinitionVersions2(oldVersion, oldUDPPParams)	1-49
GetOldPrimitiveParametersDefinition2(version)	1-49

Example UDP	1-49
UDMExtension	1-49
Import	1-49
Main class: UDMExtension	1-50
IUDMExtension methods	1-50
Mandatory methods	1-50
GetInfo()	1-50
IsAttachedToExternalEditor()	1-50
CreateInstance(funcLib)	1-50
GetUnits(instanceld)	1-50
ReleaseInstance(instanceld)	1-51
GetAttribNameForEntityId()	1-51
GetAttribNameForPartId()	1-51
Optional methods	1-51
GetInstanceSourceInfo(instanceld)	1-52
ShouldAttachDefinitionFilesToProject()	1-52
Example UDM	1-52
UDMFunctionLibrary	1-53
Functions list:	1-54
UDM/UDP Functions	1-55
Return Values for Each UDM and UDP Function	1-55
UDP/UDM Structures and Constants	1-58
UDP/UDM Structures	1-58
List of structures	1-59
UDP/UDM Constants	1-65
Enum constants:	1-65
UDP Python Example	1-67
Introduction to CPython (Beta)	1-72
Ansys Electronics Desktop Scripting	1-76
Overview of Electronics Desktop Scripting Objects	1-76

oAnsoftApp	1-77
oDesktop	1-77
oProject	1-77
oDesign	1-77
oEditor	1-78
oModule	1-78
Example Script Opening	1-79
GetActiveProject and GetActiveDesign for Wider Use	1-80
Running a Script	1-80
Within Electronics Desktop	1-80
From the Command Line	1-81
Direct Launch	1-81
Recording a Script	1-82
Recording a Script to File	1-82
Recording a Script to a Project	1-83
Working with Project Scripts	1-84
Executing a Script from Within a Script	1-85
Electronics Desktop Scripting Conventions	1-85
Named Arguments	1-86
VBscript Example	1-86
IronPython Example	1-87
Setting Numerical Values	1-88
Layout Scripts and the Active Layer	1-89
Scripts and Locked Layers	1-89
Event Callback Scripting	1-89
2 - Application Object Script Commands	2-1
GetAppDesktop	2-2
3 - Desktop Object Script Commands	3-1
AddMessage	3-6
ClearMessages	3-7

CloseAllWindows	3-8
CloseProject	3-8
CloseProjectNoForce	3-9
Count	3-10
DeleteProject	3-11
DownloadJobResults	3-12
DeleteRegistryEntry	3-13
DoesRegistryValueExist	3-14
EnableAutoSave	3-15
ExportOptionsFiles	3-16
GetActiveProject	3-16
GetAutoSaveEnabled	3-17
GetBuildDateTimeString	3-18
GetCustomMenuSet	3-19
GetDefaultUnit	3-19
GetDesktopConfiguration	3-22
GetDistributedAnalysisMachines	3-22
GetDistributedAnalysisMachinesForDesignType	3-23
GetExeDir	3-24
GetGDIObjectCount	3-25
GetLibraryDirectory	3-25
GetLocalizationHelper	3-27
GetMessages	3-28
GetPersonalLibDirectory	3-29
GetPPELicensingEnabled	3-29
GetProcessID	3-30
GetProjectDirectory	3-31
GetProjectList	3-32
GetProjects	3-32
GetRegistryInt	3-33

GetRegistryString	3-34
GetRunningInstancesMgr	3-35
GetSchematicEnvironment	3-35
GetScriptingToolsHelper	3-36
GetSysLibDirectory	3-37
GetTempDirectory	3-38
GetUserLibDirectory	3-38
GetVersion	3-39
IsFeatureEnabled	3-40
KeepDesktopResponsive	3-41
LaunchJobMonitor	3-42
NewProject	3-42
OpenMultipleProjects	3-43
OpenProject	3-44
OpenProjectWithConversion	3-45
PauseRecording	3-45
PauseScript	3-46
Print	3-46
QuitApplication	3-47
RefreshJobMonitor	3-48
ResetLogging	3-49
RestoreProjectArchive	3-50
RestoreWindow	3-51
ResumeRecording	3-51
RunACTWizardScript	3-52
RunProgram	3-52
RunScript	3-53
RunScriptWithArguments	3-55
SelectScheduler	3-56
SetActiveProject	3-58

SetActiveProjectByPath	3-59
SetCustomMenuSet	3-59
SetDesktopConfiguration	3-61
SetLibraryDirectory	3-62
SetProjectDirectory	3-62
SetRegistryFromFile	3-63
SetRegistryInt	3-64
SetRegistryString	3-65
SetSchematicEnvironment	3-65
SetTempDirectory	3-66
ShowDockingWindow	3-67
Sleep	3-68
SubmitJob	3-69
TileWindows	3-70
Desktop Commands For Registry Values	3-71
DeleteRegistryEntry	3-72
DoesRegistryValueExist	3-72
GetRegistryInt	3-73
GetRegistryString	3-74
SetRegistryFromFile	3-75
SetRegistryInt	3-76
SetRegistryString	3-76
ImportExport Tool Commands	3-77
ImportANF	3-78
ImportANFv2	3-79
ImportAutoCAD	3-81
ImportAWRMicrowaveOffice	3-82
ImportEDB	3-84
ImportExtracta	3-85
ImportGDSII	3-86

ImportGerber	3-87
ImportIDF	3-88
ImportIDFandMerge	3-91
ImportIDX	3-92
ImportIPC	3-95
ImportODB	3-96
ImportXFL	3-97
4 - Running Instances Manager Script Commands	4-1
GetAllRunningInstances	4-1
GetRunningInstanceByProcessID	4-2
GetRunningInstancesMgr	4-2
5 - Project Object Script Commands	5-1
AddDataset	5-3
AddMaterial	5-6
AnalyzeAll [project]	5-10
ClearMessages	5-11
Close	5-12
CopyDesign	5-12
CutDesign	5-13
DeleteDataset	5-14
DeleteDesign	5-15
DeleteToolObject	5-16
EditDataset	5-16
EditMaterial	5-19
ExportDataset	5-28
ExportMaterial	5-29
GetActiveDesign	5-30
GetArrayVariables	5-31
GetChildNames [Project]	5-31
GetChildObject [Project]	5-32

GetChildTypes [Project]	5-33
GetConfigurableData (Project)	5-34
GetDefinitionManager	5-35
GetDependentFiles	5-35
GetDesign	5-36
GetDesigns	5-37
GetEDBHandle	5-37
GetLegacyName	5-38
GetName [Project]	5-39
GetObjPath [Project]	5-40
GetPath	5-40
GetPropNames [Project]	5-41
GetPropValue [Project]	5-42
GetProperties	5-42
GetPropertyValue	5-44
GetTopDesignList	5-45
GetVariableValue	5-46
GetVariables	5-47
ImportDataset	5-48
Note About File Types:	5-49
InsertDesign	5-50
InsertDesignWithWorkflow	5-51
InsertToolObject	5-53
Paste (Project Object)	5-53
Redo [Project Level]	5-54
RemoveAllUnusedDefinitions	5-54
RemoveMaterial	5-55
RemoveUnusedDefinitions	5-56
Rename	5-57
Save	5-58

SaveAs	5-59
SaveAsStandAloneProject	5-61
SaveProjectArchive	5-62
SetActiveDefinitionEditor	5-63
SetActiveDesign	5-64
SetPropValue [Project]	5-64
SetPropertyValue	5-65
SetVariableValue	5-67
SimulateAll	5-68
Undo [Project]	5-69
UpdateDefinitions	5-69
6 - Object Oriented Property Scripting	6-1
Object-Oriented Scripting	6-1
Material Properties and Examples	6-9
Body Properties and Modification	6-11
Retrieving Variables	6-11
Retrieve Datasets and Values	6-12
GetSolutionData API	6-14
Summary	6-14
Additional Details Specific to AEDT Solvers	6-18
Additional details on Boundaries/Excitations	6-20
Additional Details Specific to Icepak Monitor and Mesh Modules	6-21
3D component encapsulation	6-22
Additional details on Solve setup	6-22
Materials Scripting Support	6-23
Object Oriented Scripting for Materials	6-27
Examples showing change to material property type:	6-28
Examples showing change to a vector component value	6-28
Change choice property value	6-29
Change choice property value	6-30

7 - Property Script Commands	7-1
Object Script Property Function Summary	7-3
Object Path	7-3
Property Object	7-3
Project Object	7-5
Design Object	7-6
3D Modeler Object	7-7
Variable Object	7-7
Optimetrics Module Object:	7-9
Optimetrics Setup Object	7-9
ReportSetup(Results) Module Object:	7-10
ReportSetup(Results) Module Child Objects:	7-11
Radiation Module Object:	7-12
Radiation Module Child Objects:	7-12
Conventions Used in this Chapter	7-13
GetArrayVariables	7-16
GetProperties	7-17
GetPropertyValue	7-18
GetVariables	7-20
GetVariableValue	7-20
SetPropertyValue	7-21
SetVariableValue	7-23
Example Use of Record Script and Edit Properties	7-24
Additional Property Scripting Examples	7-25
8 - Dataset Script Commands	8-1
AddDataset	8-1
DeleteDataset	8-4
EditDataset	8-5
ExportDataset	8-7
HasDataset	8-8

ImportDataset	8-9
Note About File Types:	8-10
9 - Design Object Script Commands	9-1
AddDataset	9-4
AddModelingProperties	9-7
AnalyzeAll [design]	9-7
AnalyzeAllNominal	9-8
ConstructVariationString	9-8
CopyItemCommand	9-9
CreateReport	9-10
DeleteDataset	9-25
DeleteFieldVariation	9-25
DeleteFullVariation	9-26
DeleteLinkedDataVariation	9-27
DeleteOutputVariable	9-28
DeleteSolutionVariation	9-29
DeleteOutputVariable	9-30
EditCoSimulationOptions	9-30
EditDesignSettings	9-31
EditInfiniteArray	9-33
EMDesignOptions	9-33
ExportConvergence	9-35
ExportDataset	9-36
ExportLPVROM	9-37
EcxmlExport	9-38
ExportProfile	9-38
GetChildNames [Design]	9-39
GetChildObject [Design]	9-40
GetChildTypes [Design]	9-41
GetConfigurableData	9-42

GetData	9-43
GetDesignID	9-43
GetDesignName	9-44
GetDesignType	9-44
GetEdgePositionAtNormalizedParameter	9-45
GetManagedFilesPath	9-46
GetModule	9-47
GetName	9-48
GetNominalVariation	9-49
GetNoteText	9-49
GetObjPath [Design]	9-50
GetOutputVariableValue	9-51
GetOutputVariables	9-52
GetPostProcessingVariables	9-52
GetProperties	9-53
GetPropertyValue	9-54
GetPropNames [Design]	9-56
GetPropValue [Design]	9-57
GetSelections [Design]	9-58
GetSolutionType	9-58
GetSolveInsideThreshold	9-59
GetVariables	9-59
GetVariableValue	9-60
GetVariationVariableValue	9-61
ImportDataset	9-62
Note About File Types:	9-63
InsertDesign	9-64
PasteDesign (Design Object)	9-65
Redo [Design]	9-66
RenameDesignInstance	9-67

RenameSource [Interface Source]	9-68
RunToolkit	9-68
SARSetup	9-69
SetActiveEditor	9-70
SetAllowMaterialOverride	9-70
SetBackgroundMaterial	9-71
SetLengthSettings	9-72
SetPropValue [Design]	9-73
SetPropertyValue	9-74
SetSolutionType	9-75
SetSolveInsideThreshold	9-77
SetSourceContexts	9-78
SetVariableValue	9-79
Solve	9-80
StopSimLink	9-81
Undo [Design]	9-81
ValidateLink	9-82
ValidateDesign	9-82
ValidateLink	9-83
10 - 3D Modeler Editor Script Commands	10-1
Conventions Used in this Chapter:	10-1
<AttributesArray>	10-1
<SelectionsArray>	10-2
Draw Menu Commands	10-4
CreateBondwire	10-6
CreateBox	10-9
CreateCircle	10-12
CreateCone	10-15
CreateCutplane	10-18
CreateCylinder	10-19

CreateEllipse	10-22
CreateEquationCurve	10-25
CreateEquationSurface	10-28
CreateHelix	10-31
CreatePoint	10-34
CreatePolyline	10-35
CreateRectangle	10-41
CreateRegularPolygon	10-43
CreateRegularPolyhedron	10-46
CreateSpiral	10-49
CreateTorus	10-52
CreateUserDefinedModel	10-54
CreateUserDefinedPart	10-57
Edit3DComponent	10-64
Edit3DComponentDefinition	10-67
EditPolyline	10-69
Get3DComponentDefinitionNames	10-73
Get3DComponentInstanceNames	10-74
Get3DComponentMaterialNames	10-75
Get3DComponentMaterialProperties	10-75
Get3DComponentParameters	10-76
Get3DComponentPartNames	10-77
Insert3DComponent	10-77
InsertComponent	10-78
InsertPolylineSegment	10-80
SweepAlongPath	10-82
SweepAlongVector	10-84
SweepAroundAxis	10-86
SweepFacesAlongNormal	10-88
SweepFacesAlongNormalWithAttributes	10-90

UpdateComponentDefinition	10-93
Edit Menu Commands	10-94
Copy	10-95
DeletePolylinePoint	10-96
DuplicateAlongLine	10-97
DuplicateAroundAxis	10-99
DuplicateMirror	10-102
Mirror	10-105
Move	10-107
OffsetFaces	10-109
Paste (Model Editor)	10-110
Rotate	10-111
Scale	10-112
Modeler Menu Commands	10-114
AlignFaces	10-116
AssignMaterial	10-117
Chamfer	10-120
CleanUpModel	10-122
Connect	10-123
CoverLines	10-124
CoverSurfaces	10-125
CreateEntityList	10-126
CreateFaceCS	10-127
CreateGroup	10-132
CreateObjectCS	10-133
CreateObjectFromEdges	10-140
CreateObjectFromFace	10-142
CreateObjectFromFaces	10-144
CreateRelativeCS	10-145
DeleteEmptyGroups	10-147

DeleteLastOperation	10-148
DeleteOperation	10-149
DetachEdges	10-151
DetachFaces	10-152
EditEntityList	10-154
EditFaceCS	10-155
EditObjectCS	10-161
EditRelativeCS	10-168
Export	10-170
ExportModelImageToFile	10-172
ExportModelMeshToFile	10-175
Fillet	10-176
FlattenGroup	10-178
GenerateHistory	10-179
GetActiveCoordinateSystem	10-180
GetActiveCoordinateSystemTransform	10-180
GetCoordinateSystems	10-181
HealObject	10-182
Import	10-186
ImportDXF [Modeler]	10-190
ImportFromClipboard	10-194
ImportGDSII [Modeler]	10-195
Imprint	10-198
ImprintProjection	10-199
Intersect	10-201
MoveCStoEnd	10-203
MoveEntityToGroup	10-204
MoveFaces	10-205
ProjectSheet	10-207
PurgeHistory	10-208

ReplaceWith3DComponent	10-209
Section	10-212
SeparateBody	10-214
SetModelUnits	10-215
SetWCS	10-216
ShowWindow	10-217
Simplify	10-218
Split	10-221
Stitch	10-223
Subtract	10-225
SweepFacesAlongNormal	10-226
ThickenSheet	10-228
UncoverFaces	10-230
Unite	10-231
Ungroup	10-232
WrapSheet	10-233
Other oEditor Commands	10-234
AddDefinitionFromBlock	10-237
AddDefinitionFromLibFile	10-242
AddViewOrientation	10-246
AssignSurfaceMaterial	10-253
BreakUDMConnection	10-256
CloseAllWindows[Editor]	10-257
Defeature	10-258
Delete	10-259
FitAll	10-260
GenerateAllUserDefinedModels	10-260
GenerateUserDefinedModel	10-261
GeometryCheckAndAutofix	10-262
GetBodyNamesByPosition	10-264

GetChildNames [Modeler]	10-265
GetChildObject [Modeler]	10-269
GetChildTypes [Modeler]	10-272
GetEdgeByPosition	10-274
GetEdgeIDFromNameForFirstOperation	10-276
GetEdgeIDsFromFace	10-276
GetEdgeIDsFromObject	10-277
GetEdgeLength	10-278
GetEntityListIDByName	10-278
GetExtendedDefinitionObject	10-279
GetFaceArea	10-280
GetFaceCenter	10-281
GetFaceByPosition	10-282
GetFaceIDFromNameForFirstOperation	10-283
GetFaceIDs	10-284
GetFaceIDsOfSheet	10-285
GetGeometryModelerMode	10-285
GetGroupSubmodelNames	10-286
GetMatchedObjectName	10-287
GetModelBoundingBox	10-287
GetModelUnits	10-288
GetNumObjects	10-289
GetObjectIDByName	10-289
GetObjectName	10-290
GetObjectNameByEdgeID	10-291
GetObjectNameByFaceID	10-291
GetObjectNameByID	10-292
GetObjectNameByVertexID	10-293
GetObjectsByMaterial	10-293
GetObjectShapeType	10-294

GetObjectsInGroup	10-295
GetObjectVolume	10-296
GetObjPath [Editor]	10-296
GetPartsForUserDefinedModel	10-297
GetPoints [3D Modeler Editor]	10-298
GetPropEvaluatedValue	10-298
GetPropertyValue	10-299
GetPropNames [Modeler]	10-301
GetPropSIValue	10-302
GetPropValue [Modeler]	10-303
GetRelativeCoordinateSystems	10-304
GetSelections [Model Editor]	10-304
GetSubGroupsInGroup	10-305
GetUserPosition	10-306
GetVertexIDFromNameForFirstOperation	10-307
GetVertexIDsFromEdge	10-308
GetVertexIDsFromFace	10-308
GetVertexIDsFromObject	10-309
GetVertexPosition	10-310
GetWireBodyNames	10-310
OpenExternalEditor	10-311
PageSetup	10-312
RemoveBadEdges	10-314
RemoveBadFaces	10-314
RemoveBadVertices	10-315
RenamePart	10-316
SetPropValue [Modeler]	10-317
SetTopDownViewDirectionForActiveView	10-318
SetTopDownViewDirectionForAllViews	10-319
UpdatePriorityList	10-319

UpgradeVersion	10-320
Validate3DComponent	10-321
WriteHistoryTreeLayoutForTest	10-322
Cable Modeling Commands	10-324
AddCableToBundle	10-325
CreateCableBundle	10-326
CreateCableHarness	10-328
CreateClockSource	10-330
CreatePWLSource	10-332
CreateStraightWireCable	10-334
CreateTwistedPairCable	10-335
ExportCableLibrary	10-337
ImportCableLibrary	10-338
RemoveCable	10-339
UpdateCableHarness	10-339
11 - Output Variable Script Commands	11-1
CreateOutputVariable	11-1
DeleteOutputVariable	11-2
DoesOutputVariableExist	11-3
EditOutputVariable	11-4
ExportOutputVariables	11-5
GetOutputVariables	11-6
GetOutputVariableValue	11-7
ImportOutputVariables	11-8
SimValueContext	11-9
12 - Reporter Editor Script Commands	12-1
AddAllEyeMeasurements	12-4
AddCartesianLimitLine	12-4
AddCartesianLimitLineFromCurve	12-6
AddCartesianLimitLineFromEquation	12-8

AddCartesianXMarker	12-10
AddCartesianYMarker	12-10
AddCartesianYMarkerToStack	12-11
AddDeltaMarker	12-12
AddMarker	12-13
AddNote	12-14
AddTraceCharacteristics	12-16
AddTraces	12-18
ApplyReportTemplate	12-20
ClearAllMarkers	12-21
ClearAllTraceCharacteristics	12-22
CloneReportsFromDatasetSolution	12-22
CopyPlotSettings	12-23
CopyReportDefinitions	12-24
CopyReportsData	12-25
CopyTraceDefinitions	12-25
CopyTracesData	12-26
CreateReport	12-27
CreateReport [Designer]	12-42
CreateReportFromTemplate	12-47
CreateReportOfAllQuantities	12-48
DeleteMarker	12-50
DeleteAllReports	12-51
DeleteReports	12-51
DeleteTraceCharacteristics	12-52
DeleteTraces	12-53
DoesSupportTraceCharacteristics	12-54
DumpAllReportsData	12-55
EditCartesianXMarker	12-55
EditCartesianYMarker	12-56

EditMarker	12-57
ExportEyeMaskViolation	12-58
ExportImageToFile [Reporter]	12-59
ExportModelImageToFile	12-60
ExportModelMeshToFile	12-63
ExportPlot3DToFile [Reporter]	12-63
ExportReport	12-65
ExportReportDataToFile	12-66
ExportTableToFile	12-67
ExportToFile	12-68
ExportToFile [Reporter]	12-69
ExportUniformPointsToFile	12-70
GetAllCategories	12-72
GetAllQuantities	12-73
GetAllReportNames	12-74
GetAvailableDisplayTypes	12-75
GetAvailableReportTypes	12-76
GetAvailableSolutions	12-76
GetChildNames [Report Setup]	12-77
GetChildObject [Report Setup]	12-78
GetChildTypes [ReportSetup]	12-79
GetCurvePropServerName	12-79
GetDisplayType	12-80
GetDynLinkIntrinsicVariables	12-81
GetDynLinkQtyValueState	12-81
GetDynLinkTraces	12-82
GetDynLinkVariableValues	12-83
GetName	12-83
GetObjPath [Design]	12-84
GetPropertyValues	12-85

GetPropNames [Reporter]	12-86
GetPropValue [Report Setup]	12-87
GetQtyExpressionsForSourceTrace	12-88
GetReportTraceNames	12-88
GetReportSummaryForRegressionTesting	12-89
GetSolutionContexts	12-90
GetSolutionDataPerVariation	12-91
GetDataExpressions	12-93
GetDataUnits	12-94
GetDesignVariableNames	12-95
GetDesignVariableUnits	12-96
GetDesignVariableValue	12-97
GetDesignVariationKey	12-98
GetImgDataValues	12-98
GetPerQuantityPrimarySweepValues	12-99
GetRealDataValues	12-100
GetSweepNames	12-101
GetSweepUnits	12-102
GetSweepValues	12-103
IsDataComplex	12-104
IsPerQuantityPrimarySweep	12-105
Release Data	12-106
GroupPlotCurvesByGroupingStrategy	12-107
ImportIntoReport	12-108
ImportReportDataIntoReport	12-109
MovePlotCurvesToGroup	12-109
MovePlotCurvesToNewGroup	12-110
OpenWindowForAllReports	12-111
OpenWindowForReports	12-112
PastePlotSettings	12-113

PasteReports	12-113
PasteReportsWithLegacyNames	12-114
PasteTraces	12-115
PasteTracesWithLegacyNames	12-115
RenameReport	12-116
RenameTrace	12-117
ResetPlotSettings	12-118
SavePlotSettingsAsDefault	12-118
SetLinkOutputTraces	12-119
SetPropValue [Report Setup]	12-120
UnGroupPlotCurvesInGroup	12-121
UpdateAllReports	12-122
UpdateReports	12-122
UpdateTraces	12-123
UpdateTracesContextAndSweeps	12-126
13 - Boundary and Excitation Module Script Commands	13-1
General Commands Recognized by the Boundary Conditions Module	13-1
AddAssignmentToBoundary	13-2
DeleteAllBoundaries	13-3
DeleteBoundaries	13-3
GetBoundaries	13-4
GetBoundariesOfType	13-4
GetBoundaryAssignment	13-5
GetDefaultBaseName	13-5
GetNumBoundaries	13-6
GetNumBoundariesOfType	13-6
ReassignBoundary	13-7
RemoveAssignmentFromBoundary	13-7
RenameBoundary	13-8
ReprioritizeBoundaries	13-9

SetDefaultBaseName	13-10
Icepak Boundary Condition Commands	14-1
AssignBlockBoundary	14-1
AssignGrilleBoundary	14-3
AssignNetworkBoundary	14-5
AssignOpeningBoundary	14-6
AssignRecircBoundary	14-7
AssignAdiabaticPlateBoundary	14-8
AssignConductingPlateBoundary	14-10
AssignResistanceBoundary	14-13
AssignSourceBoundary	14-17
AssignStationaryWallBoundary	14-18
AssignSymmetryWallBoundary	14-20
AssignEMLoss	14-21
AssignBlowerBoundary	14-22
14 - Mesh Region Module Script Commands	14-24
Script Commands for Creating and Modifying Mesh Regions	14-24
EditGlobalMeshRegion	14-24
AssignVirtualMeshRegion	14-27
EditMeshRegion	14-29
General Commands Recognized by the Mesh Operations Module	14-30
DeleteOp	14-30
GetOperationNames	14-30
ReassignOp	14-31
RenameOp	14-32
15 - Analysis Setup Module Script Commands	15-1
ClearLinkedData (Module)	15-2
CopySetup	15-2
DeleteSetups	15-3
EditSetup	15-4

GetSetupCount	15-16
GetSetups	15-17
InsertSetup (Icepak - Steady State)	15-17
InsertSetup (Icepak - Transient)	15-28
PasteSetup	15-39
RenameSetup	15-39
RevertAllToInitial	15-40
RevertSetupToInitial	15-41
16 - Optimetrics Module Script Commands	16-1
CopySetup	16-6
DeleteSetups [Optimetrics]	16-7
DistributedAnalyzeSetup	16-7
EditSetup	16-8
EnableSetup	16-20
ExportDXConfigFile	16-21
ExportOptimetricsProfile	16-22
ExportOptimetricsResult	16-23
ExportParametricResults	16-24
ExportParametricSetupTable	16-25
ExportRespSurfaceMinMaxTable	16-26
ExportRespSurfaceRefinePoints	16-27
ExportRespSurfaceResponsePoints	16-27
ExportRespSurfaceVerificationPoints	16-28
GenerateVariationData [Parametric]	16-29
GetChildNames [Optimetrics]	16-30
GetChildObject [Optimetrics]	16-30
GetChildTypes [Optimetrics]	16-31
GetName	16-32
GetObjPath [Design]	16-32
GetOptimetricResult	16-33

GetOptimetricsResult	16-34
GetPropNames [Optimetrics]	16-35
GetPropValue [Optimetrics]	16-36
GetSetupNames [Optimetrics]	16-36
GetSetupNamesByType [Optimetrics]	16-37
ImportSetup	16-38
PasteSetup [Optimetrics]	16-39
RenameSetup [Optimetrics]	16-40
SetPropValue [Optimetrics]	16-41
SolveAllSetup	16-42
SolveSetup [Optimetrics]	16-42
General Commands Recognized by the Optimetrics Module	16-43
CopySetup	16-44
DeleteSetups [Optimetrics]	16-45
DistributedAnalyzeSetup	16-46
EditSetup	16-46
EnableSetup	16-59
ExportDXConfigFile	16-59
ExportOptimetricsProfile	16-60
ExportOptimetricsResult	16-61
ExportParametricResults	16-62
ExportParametricSetupTable	16-63
ExportRespSurfaceMinMaxTable	16-64
ExportRespSurfaceRefinePoints	16-65
ExportRespSurfaceResponsePoints	16-66
ExportRespSurfaceVerificationPoints	16-67
GenerateVariationData [Parametric]	16-67
GetChildNames [Optimetrics]	16-68
GetChildObject [Optimetrics]	16-69
GetChildTypes [Optimetrics]	16-69

GetName	16-70
GetObjPath [Design]	16-70
GetOptimetricResult	16-71
GetPropNames [Optimetrics]	16-72
GetPropValue [Optimetrics]	16-73
GetSetupNames [Optimetrics]	16-74
GetSetupNamesByType [Optimetrics]	16-74
ImportSetup	16-75
PasteSetup [Optimetrics]	16-77
RenameSetup [Optimetrics]	16-77
SetPropValue [Optimetrics]	16-78
SolveAllSetup	16-79
SolveSetup [Optimetrics]	16-79
Parametric Script Commands	16-80
EditSetup [Parametric]	16-80
ExportParametricSetupTable	16-81
GenerateVariationData [Parametric]	16-82
InsertSetup [Parametric]	16-82
Optimization Script Commands	16-87
EditSetup [Optimization]	16-87
InsertSetup [Optimization]	16-93
Sensitivity Script Commands	16-104
EditSetup [Sensitivity]	16-104
InsertSetup [Sensitivity]	16-111
Statistical Script Commands	16-116
EditSetup [Statistical]	16-116
InsertSetup [Statistical]	16-119
17 - Solutions Module Script Commands	17-1
DeleteSolutionVariation	17-1
EditSources	17-2

ExportNMFDData [HFSS]	17-2
ExportTransientData	17-3
FFTOnReport	17-4
GetAdaptiveSettings	17-5
GetAllSourceMagnitudes	17-6
GetAllSourceModes	17-7
GetAllSourcePhases	17-7
GetAllSources	17-8
GetAntennaParameters	17-9
GetFieldType	17-9
GetIncludePortPostProcessing	17-10
GetMultipactionBreakdown	17-11
GetNetworkDataSolution	17-11
GetNetworkDataSolutionDefinition	17-12
GetNetworkPostprocSetup	17-13
GetSourceContexts	17-13
GetTerminalExcitationType	17-14
GetTransientSolveTimes	17-15
SetSourceContexts	17-15
TDROnReport	17-16
18 - Field Overlays Module Script Commands	18-1
AddMarker[Fields Reporter]	18-1
AddMarkerToPlot	18-2
AddNamedExpr	18-4
AddNamedExpression	18-4
CalcOp	18-5
CalcStack	18-6
CalcWrite	18-6
CalculatorRead	18-7
CalculatorWrite	18-8

ChangeGeomSettings	18-9
ClcEval	18-10
ClcMaterial	18-11
ClcMaterialValue	18-12
ClearAllMarkers[Fields Reporter]	18-13
ClearAllNamedExpr	18-13
CopyNamedExprToStack	18-14
CreateFieldPlot	18-15
Maxwell Field Line Trace Plot Examples	18-37
DeleteFieldPlot	18-46
DeleteMarker[Fields Reporter]	18-47
DeleteNamedExpr	18-47
DeleteUneditablePlot	18-48
DoesNamedExpressionExists	18-49
EditSurfaceMeshSummaryData	18-50
EnterComplex	18-54
EnterComplexVector	18-54
EnterCoord	18-55
EnterEdge	18-56
EnterLine	18-57
EnterOutputVar	18-58
EnterPoint	18-58
EnterQty	18-59
EnterScalar	18-60
EnterScalarFunc	18-60
EnterSurf	18-61
EnterVector	18-62
EnterVectorFunc	18-62
EnterVol	18-63
ExportFieldPlot	18-64

ExportMarkerTable	18-65
ExportOnGrid [Field Overlays]	18-65
ExportPlotImageToFile	18-68
ExportPlotImageWithViewToFile [Reporter]	18-70
ExportSurfaceMeshSummary	18-71
ExportToFile	18-72
GetFieldFolderNames	18-73
GetFieldPlotNames	18-74
GetFieldPlotQuantityName	18-75
GetMeshPlotNames	18-75
GetTopEntryValue	18-76
HidePolarPlot	18-77
HideRadiatedPlotOverlay	18-78
LoadNamedExpressions	18-78
ModifyFieldPlot	18-79
ReassignFieldPlot	18-82
RenameFieldPlot	18-85
RenamePlotFolder	18-86
SaveFieldsPlots	18-87
SaveNamedExpressions	18-87
SetFieldPlotSettings	18-88
SetPlotFolderSettings	18-91
ShowPolarPlot	18-97
UpdateAllFieldsPlots	18-97
UpdateQuantityFieldsPlots	18-97
19 - Fields Calculator Script Commands	19-1
AddNamedExpression	19-2
AddNamedExpr	19-3
CalcOp	19-4
CalcRead(deprecated)	19-4

CalculatorRead	19-5
CalcStack	19-6
CalculatorWrite	19-7
CalcWrite	19-8
ChangeGeomSettings	19-9
ClcEval	19-10
ClcMaterial	19-11
ClcMaterialValue	19-11
ClearAllNamedExpr	19-12
CopyNamedExprToStack	19-13
DeleteNamedExpr	19-13
DoesNamedExpressionExists	19-14
EnterComplex	19-15
EnterComplexVector	19-16
EnterCoord	19-17
EnterEdge	19-17
EnterLine	19-18
EnterOutputVar	19-19
EnterPoint	19-19
EnterQty	19-20
EnterScalar	19-21
EnterScalarFunc	19-22
EnterSurf	19-22
EnterVector	19-23
EnterVectorFunc	19-24
EnterVol	19-24
ExportOnGrid [Fields Calculator]	19-25
ExportToFile [Fields Calculator]	19-28
GetTopEntryValue	19-31
LoadNamedExpressions	19-32

SaveNamedExpressions	19-33
20 - Fields Summary Script Commands	20-1
EditFieldsSummarySetting	20-1
ExportFieldsSummary	20-2
21 - User Defined Document Script Commands	21-1
AddDocument	21-1
DeleteAllDocuments	21-4
DeleteDocument	21-5
EditDocument	21-5
GetDocumentDefinitionNames	21-7
GetDocumentNames	21-8
RenameDocument	21-8
SaveHtmlDocumentAs	21-9
SavePdfDocumentAs	21-10
UpdateAllDocuments	21-10
UpdateDocument	21-11
ViewHtmlDocument	21-12
ViewPdfDocument	21-13
Explication of a Sample UDD Script	21-13
Example Python Script: Defining a Document	21-15
22 - User Defined Solutions Commands	22-1
CreateUserDefinedSolution	22-1
DeleteUserDefinedSolutions	22-3
EditUserDefinedSolution	22-4
GetUserDefinedSolutionNames	22-6
GetUserDefinedSolutionProperties	22-6
23 - Definition Manager Script Commands	23-1
Add [component manager]	23-2
AddDataset	23-20
AddDefinitionFromBlock	23-23

AddMaterial	23-28
Add [padstack manager]	23-32
Add [symbol manager]	23-39
DeleteDataset	23-50
Edit [component manager]	23-50
EditDataset	23-77
EditMaterial	23-80
Edit [padstack manager]	23-89
ExportDataset	23-96
Export [footprint manager]	23-97
ExportMaterial	23-98
Export [padstack manager]	23-99
ExportScript	23-100
Export [symbol manager]	23-100
GetProjectMaterialNames	23-101
GetPropertyJsonValue	23-102
ImportDataset	23-104
Note About File Types:	23-105
Remove [component manager]	23-106
Remove [footprint manager]	23-108
RemoveMaterial	23-109
Remove [padstack manager]	23-110
RemoveScript	23-111
Remove [symbol manager]	23-112
RemoveUnusedDefinitions	23-113
SetPropertyValue	23-115
UpdateDefFromBlock	23-116
Material Manager Script Commands	23-118
GetData [material manager]	23-119
GetNames [material manager]	23-119

GetProperties [material manager]	23-121
IsUsed [material manager]	23-122
RemoveUnused [material manager]	23-122
Model Manager Script Commands	23-123
Add [model manager]	23-124
ConvertToDynamic	23-134
ConvertToParametric	23-135
Edit [deprecated]	23-135
EditWithComps [model manager]	23-135
Export [model manager]	23-147
GetData [model manager]	23-148
GetNames [model manager]	23-149
IsUsed [model manager]	23-150
Remove [model manager]	23-150
RemoveUnused [model manager]	23-152
Script and Library Scripts	23-153
AddScript	23-153
EditScript	23-154
ExportScript	23-155
ModifyLibraries	23-156
RemoveScript	23-157
Symbol Manager Script Commands	23-158
Add [symbol manager]	23-159
BringToFront [symbol manager]	23-170
Edit [deprecated]	23-170
EditWithComps [symbol manager]	23-170
Export [symbol manager]	23-182
GetData [symbol manager]	23-183
GetNames [symbol manager]	23-184
IsUsed [symbol manager]	23-184

Remove [symbol manager]	23-185
RemoveUnused [symbol manager]	23-186
24 - Core Global Script Context Commands	24-1
AddErrorMessage	24-1
AddFatalMessage	24-2
AddInfoMessage	24-2
AddWarningMessage	24-3
LogDebug	24-3
.LogError	24-4
25 - Example Scripts	25-1
VBScript Example Scripts	25-1
Data Export Script	25-1
Variable Helix Script	25-4
IronPython Example Scripts	25-8
BH Coordinates Python Script	25-8
Script Contents	25-9
Posco_BH_Curve.tab Contents	25-11
Equation Based Curve Python Script	25-13
HFSS Waveguide Array Python Script	25-16
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" 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:=", _  
    0.40000005960464, "PartCoordinateSystem:=", _  
    "Global", "MaterialName:=", "vacuum", "SolveInside:=", true)  
' oEditor.CreateBox is a Sub procedure that takes two array variables as input. The  
' first array is for the box's geometric parameters and the second array is for the box's
```

' attributes. You can modify the italicized entries to create a different box. In VBScript, ' **Array** is a function that returns a variant containing an array. The underscore ' character (_) here indicates that the statement continues to the next line. The underscore character must be placed outside of string constants, or else VBScript will ' recognize the character as part of the string constant rather than an indication that the ' string continues on the next line. Following is an example of proper use of the underscore ' character:

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

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

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

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

Sample Q3D Extractor Script

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

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

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

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

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

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

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

```
oDesktop.NewProject
```

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

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

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

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

```
Set oDesign = oProject.SetActiveDesign("Q3DDesign1")
```

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

```
oEditor.CreateBox Array("NAME:BoxParameters", "XPosition:=", __  
"0mm", "YPosition:=", "0mm", "ZPosition:=", "0mm", __  
"XSize:=", "1.6mm", "YSize:=", "1.2mm", "ZSize:=", __  
"0.8mm"), Array("NAME:Attributes", "Name:=", "Box1", "Flags:=", __  
"", "Color:=", "(132 132 193)", "Transparency:=", __  
0.40000005960464, "PartCoordinateSystem:=", __  
"Global", "MaterialName:=", "vacuum", "SolveInside:=", true)
```

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

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

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

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

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

Introduction to IronPython

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

Scope

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

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

Python compatibility

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

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

Advantages of IronPython

The advantages that IronPython use provides are significant:

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

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

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

 IronPython Command Window

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

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

[Introduction to IronPython](#)

[IronPython Mini Cookbook](#)

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

[Scripting Using Iron Python](#)

[Standalone IronPython and Desktop IronPython](#)

[IronPython Examples](#)

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

IronPython Mini Cookbook

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

consult Python documentation for additional information.

Comments

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

Assigning/Creating Variables

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

Create Lists/Arrays

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

Create Dictionaries/Maps

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

Boolean Values

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

Converting Numbers to Strings and Vice Versa

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

String Formatting/Concatenation

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

Looping over Lists

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

Looping over a Range

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

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

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

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

Note:

Python recommends the use of spaces over tabs.

Indenting Functions

Define a function that starts at 0 indentation:

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

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

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

Indenting If Conditions

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

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

Methods in IronPython

Finding Methods

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

```
import string
```

Then get the directory listing:

```
dir(string)
```

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

Help

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

Related Topics

[Introduction to IronPython](#)

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

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

[IronPython Samples](#)

Translating Script Commands from VBScript to IronPython

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

Related Topics

[Script Method Argument](#)

[VBscript Method Call Types](#)

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

[Introduction to IronPython](#)

[IronPython Mini Cookbook](#)

[Scripting Using Iron Python](#)

[IronPython Samples](#)

Script Method Argument

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

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

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

- Primitive Types
- Named Arrays
- Named Functions

Primitive Types

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

Named Arrays

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

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

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

Named Functions

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

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

Related Topics

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

VBScript Method Call Types

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

VBScript Functions

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

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

Note:

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

VBScript Sub-Routines

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

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

Related Topics

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

Converting VBScript Function calls to IronPython Syntax

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

So:

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

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

Return Values

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

Primitive Method Arguments

Replace each VBScript primitive with an equivalent IronPython primitive.

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

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

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

Named Array Arguments

Formatting helps readability immensely but is not required.

All that *must* be done is:

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

Named Array Values with All Key Value Pairs

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

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

Named Arrays with Nested Named Arrays

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

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

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

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

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

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

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

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

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

```

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

```

Function Blocks

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

Important:

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

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

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

Scripting Using Iron Python

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

Translating a script in VBScript to IronPython

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

Writing an IronPython script from scratch

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

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

[IronPython Script Execution Environment](#)

[Scripting using Embedded VBScript or JavaScript](#)

[Scripting with IronPython](#)

[Standalone IronPython and Desktop IronPython](#)

Related Topics

[Introduction to IronPython](#)

[IronPython Mini-cookbook](#)

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

[Appendix: IronPython Samples](#)

IronPython Script Execution Environment

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

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

 IronPython Command Window

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

- **Tools > Run Script** menu item, select "IronPython" from the file type drop-down list.
- Launch the product with a script argument.
- 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.

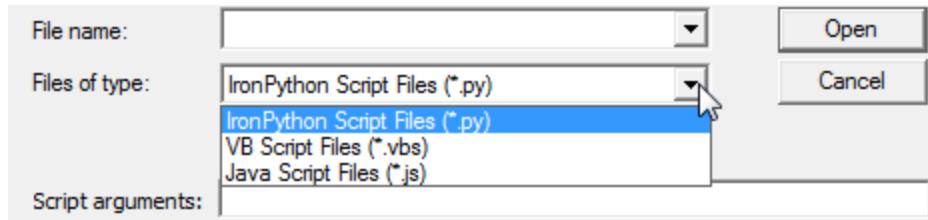


Figure 1: Run Script dialog and script arguments

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 From (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 GetAp-pDesktop() is not needed (or even supported on all platforms). Since all scripting occurs in the context of a running workbench, the available Desktop object is always provided and expected to be used directly.

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

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

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

Related Topics

[IronPython Script Execution Environment](#)

[Standalone IronPython and Desktop IronPython](#)

Standalone IronPython

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

Running Standalone IronPython

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

It is located in:

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

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

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

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

Using a Recorded Script

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

Creating an External Script

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

- ```
import sys
```

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

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

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

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

**or**

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

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

You must end the script with:

- ```
ScriptEnv.Shutdown()
```

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

Example Script

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

```
import ScriptEnv

ScriptEnv.Initialize("Ansoft.ElectronicsDesktop")
oDesktop.RestoreWindow()

oProject = oDesktop.NewProject()

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

oDesign = oProject.SetActiveDesign("HFSSDesign1")

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

oEditor.CreateRectangle(
[

    "NAME:RectangleParameters",
    "IsCovered:= ", True,
    "XStart:= ", "-0.2mm",
    "YStart:= ", "-3mm",
    "ZStart:= ", "0mm",
    "Width:= ", "0.8mm",
    "Height:= ", "1.2mm",
    "WhichAxis:= ", "Z"
], ,
[

    "NAME:Attributes",
    "Name:= ", "Rectangle1",
    "Flags:= ", "",
    "Color:= ", "(132 132 193)",
    "Transparency:= ", 0,
    "PartCoordinateSystem:= ", "Global",
    "UDMID:= ", "",
    "MaterialValue:= ", "\\"vacuum\\\"",
    "SolveInside:= ", True
])
```

```
oDesign.SetDesignSettings(['NAME:Design Settings Data', 'Allow Material Override:=' , True, 'Calculate Lossy Dielectrics:=' , True])  
oEditor.SetModelUnits(['NAME:Units Parameter', 'Units:=' , 'mil', 'Rescale:=' , False ])  
ScriptEnv.Shutdown()
```

IronPython Samples

Change property

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

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

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

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

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

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

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

```
[ "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 [UDMFuctionLibrary](#) listed further in this document. A separate chapter includes a [UDP IronPython example of fillet and chamfer](#).

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

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

Output Parameters

The rule for the output parameters is as follows:

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

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

```

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

(lRet, partIdsList) = udpFunctionLibrary.DetachFaces(nPartIds, faceId-
sList)

# Two output parameter; return value must be preserved

# the return tuple is (returnVal, output1, output2)

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

```

Comparison with C function:

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

'List Size' Parameters

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

- If function has input 'List' parameter and input 'list size' parameter, 'list size' parameter will be omitted.
- If function has output 'List' parameter and output 'list size' parameter, 'list size' parameter will be omitted.

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

Example:

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

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

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

Comparison with C function:

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

C	Python
<pre>long* partIds, void* callbackData);</pre> <p>where partIds is an output list and nSize is an input parameter and nSize is the size of partIds.</p>	(nSize is not ignored, callbackData is omitted)

Added Parameters

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

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

Example:

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

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

Here cloneIDsSize is a new integer parameter.

Comparison with C function:

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

C	Python
void* callbackData);	

Developing a UDM/UDP

Creation

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

Location

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

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

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

Organize

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

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

For UDM:extension:

```
from UDM import *
```

For UDP:extension:

```
from UDP import *
```

Edit/Reload

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

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

UDPExtension

Import

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

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

```
from UDP import *
```

Main class: UDPExtension

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

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

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

IUDPExtension methods

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

Mandatory methods.

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

GetLengthParameterUnits()

- returns string.

GetPrimitiveTypeInfo()

- returns UDPPrimitiveTypeInfo.

GetPrimitiveParametersDefinition2()

- returns a list of UDPPrimitiveParameterDefinition2 or None on failure

AreParameterValuesValid2(errorMsg, udpParams)

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

CreatePrimitive2(funcLib, udpParams)

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

Optional methods

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

GetPrimitiveParameters()

- returns Python list of strings or NULL

GetRegisteredFaceNames()

- returns Python list of strings or NULL

GetRegisteredEdgeNames()

- returns Python list of strings or NULL

GetRegisteredVertexNames()

- returns Python list of strings or NULL

ProjectParametersOnToValidPlane2(currentUDPPParams, projectedUDPPParams)

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

MapParametersDefinitionVersions2(oldVersion, oldUDPPParams)

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

GetOldPrimitiveParametersDefinition2(version)

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

Example UDP

```
import sys

class UDPExtension(IUDPExtension):


    def GetLengthParameterUnits(self):
        return "mm"

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

        return typeInfo

    ...
    ...
```

UDMExtension**Import**

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

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

```
from UDM import *
```

Main class: UDMExtension

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

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

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

IUDMExtension methods

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

Mandatory methods.

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

GetInfo()

- returns UDMInfo object populated with appropriate UDM information.

IsAttachedToExternalEditor()

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

CreateInstance(funcLib)

- funcLib is UDMFunctionLibrary
- returns UDMPARAMETERS.

GetUnits(instanceId)

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

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

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

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

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

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

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

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

returns True on success, False on failure.

ReleaseInstance(instancId)

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

GetAttribNameForEntityId()

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

GetAttribNameForPartId()

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

Optional methods

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

DialogForDefinitionOptionsAndParams(self, defData, optData, params):

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

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

DialogForDefinitionAndOptions(self, defData, optData) [Deprecated]

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

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

GetInstanceSourceInfo(instanceld)

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

ShouldAttachDefinitionFilesToProject()

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

Example UDM

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

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

    return udmInfo
...
...
```

UDMFunctionLibrary

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

- CreateInstance
- Refresh

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

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

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

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

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

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

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

Functions list:

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

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

Example:

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

UDM/UDP Functions

Return Values for Each UDM and UDP Function

ID – *ID of created Object*

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

Functions list:

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

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

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

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

UDP/UDM Structures and Constants

The following sections describe:

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

UDP/UDM Structures

Differences compared to C API

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

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

Example (creating UDMParameter list):

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

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

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

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

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

Example:

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

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

- **UDP3x3Matrix**

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

Example:

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

List of structures

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

Example:

```

pos1 = UDPPosition(1,2,3)

pos2 = UDPPosition(x=1,y=10,z=0)

pos2.Z = pos1.Z

udpParam = UDPParam(ParamDataType.Double,1)

value = udpParam.Data

```

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

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

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

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

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

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

UDP/UDM Constants

Full names of enum constants must be used in scripts.

Example:

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

Enum constants:

enum Constant	Parameters
UnitType	NoUnit LengthUnit AngleUnit
ParamDataType	Int

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

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

UDP Python Example

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

```
import sys

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

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

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

#####
```

```
# Class Implementation

#####
class UDPExtension(IUDPExtension):

    def CreatePrimitive2(self, func_lib, param_values):
        """
            Inbuilt function that is called to generate a UDP after successful validation

            :param func_lib: drawing inbuilt class, see in Help: UDMFunctionLibrary

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

            :return: None
        """

        param_dict = self.get_param_dict(param_values)

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

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

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

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

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

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

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

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

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

def GetPrimitiveTypeInfo(self):
    return primitive_info
```

```
def GetLengthParameterUnits(self):
    return length_units

def GetPrimitiveParametersDefinition2(self):
    return primitive_param_definitions

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

# Custom Functions

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

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

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

Introduction to CPython (Beta)

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

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

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

Important:

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

Important:

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

Creating an External Script

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

```

import sys

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

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

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

```

Those lines are followed by:

```

import ScriptEnv

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

```

Follow that with:

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

Launching Electronics Desktop

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

```

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

```

Where:

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

Note:

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

Connecting with a Running Instance of Electronics Desktop

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

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

Where:

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

Closing Electronics Desktop/Ending the Script

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

```
ScriptEnv.Shutdown()
```

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

-grpcsrv Flag

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

On Windows:

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

On Linux:

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

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

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

Related Topics:

[Standalone Scripting in Iron Python](#)

This page intentionally
left blank.

Ansys Electronics Desktop Scripting

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

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

[Running a Script](#)

[Recording a Script](#)

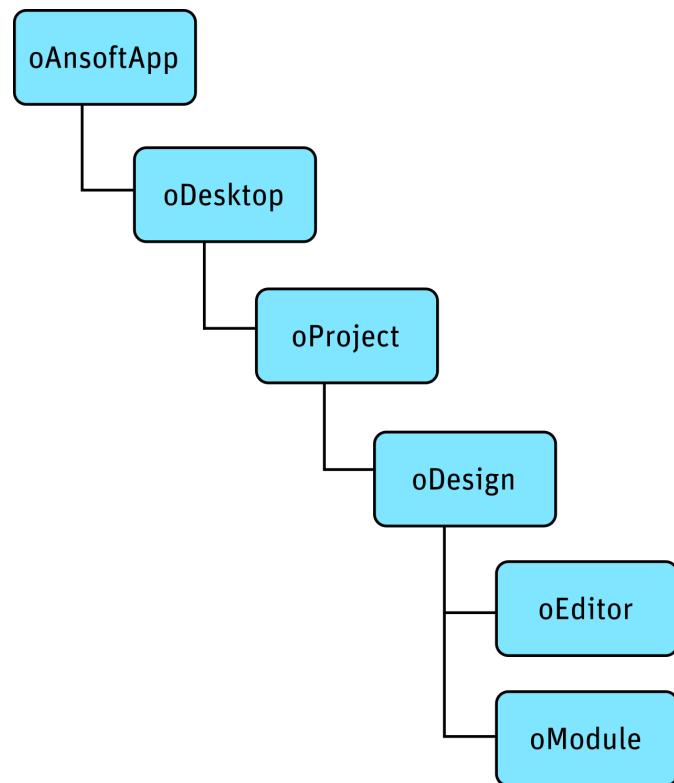
[Working with Project Scripts](#)

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

[Ansys Electronics Desktop Scripting Conventions](#)

Overview of Electronics Desktop Scripting Objects

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



The commands are described below, followed by examples.

oAnsoftApp

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

In Iron Python, for example:

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

In VBscript, for example:

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

oDesktop

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

In Iron Python, for example:

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

In VBscript, for example:

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

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

oProject

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

In Iron Python, for example:

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

In VBscript, for example:

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

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

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

oDesign

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

In Iron Python, for example:

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

In VBscript, for example:

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

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

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

oEditor

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

In Iron Python, for example:

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

In VBscript, for example:

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

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

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

Important:

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

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

oModule

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

In IronPython, for example:

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

In VBscript, for example:

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

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

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

Example Script Opening

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

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

In VBscript, this would be:

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

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

GetActiveProject and GetActiveDesign for Wider Use

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

In IronPython:

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

In VBscript:

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

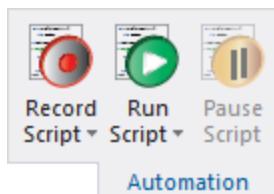
Running a Script

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

Within Electronics Desktop

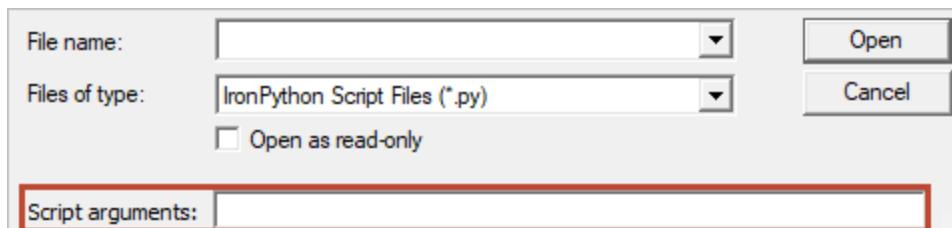
To run scripts in Electronics Desktop:

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



The **Run Script** file browser appears.

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



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

4. Click **Open**.

Electronics Desktop executes the script.

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

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

From the Command Line

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

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

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

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

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

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

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

Direct Launch

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

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

```

End if

on error goto 0

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

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

```

Recording a Script

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

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

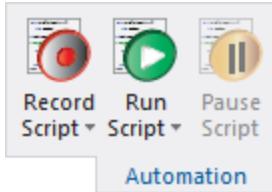
Important:

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

Recording a Script to File

To record a script to file:

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

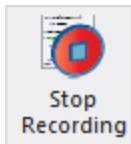


A **Save As** file browser appears.

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

5. Click **Save**.

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



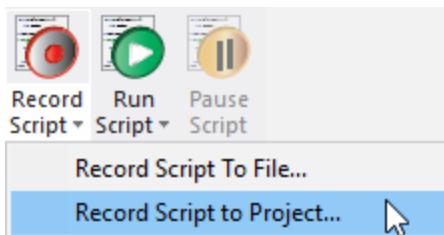
6. Perform the steps you want to record.
7. When you have finished recording the script, click **Stop Recording**, or select **Tools > Stop Script Recording**.

The recorded script is saved to the folder you specified.

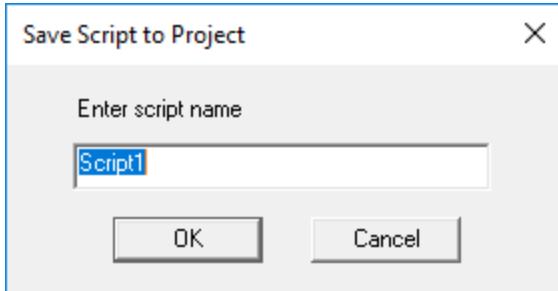
Recording a Script to a Project

To record a script to a project:

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



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



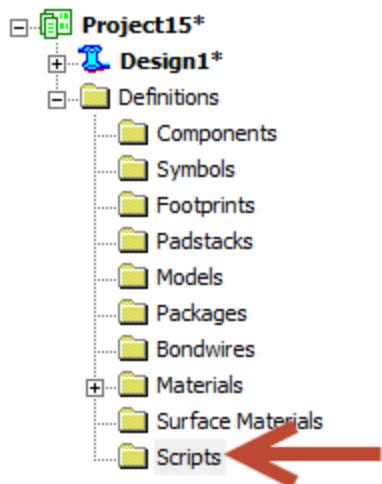
2. Enter a name for the script in the text box, then click **OK**.
3. Perform the steps you want to record.
4. When you have finished, click **Stop Recording**, or select **Tools > Stop Script Recording**.

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

Working with Project Scripts

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

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



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



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

Note:

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

Executing a Script from Within a Script

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

In IronPython:

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

In VBscript:

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

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

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

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

Electronics Desktop Scripting Conventions

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

Consult the following topics:

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

- [Event Callback Scripting](#)

Named Arguments

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

1. Named data, where name precedes data.

For example:

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

2. Named Array, where name precedes array.

For example:

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

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

For example:

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

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

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

VBscript Example

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

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

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

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

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

IronPython Example

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

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

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

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

```
]  
)
```

Setting Numerical Values

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

- Pass in the number directly.

In IronPython:

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

In VBscript:

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

- Pass in a string containing the number with units.

In IronPython:

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

In VBscript:

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

- Pass in a variable name.

In IronPython:

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

In VBscript:

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

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

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

This page intentionally
left blank.

2 - Application Object Script Commands

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

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

General Application Script Commands

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

- [GetAppDesktop](#)

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

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

- SetNumberOfProcessors (deprecated)
- SetUseHPCForMP (deprecated)

GetAppDesktop

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

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>

3 - Desktop Object Script Commands

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

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

[AddMessage](#)

[AreSimulationsRunning](#)

[ClearMessages](#)

[CloseAllWindows](#)

[CloseProject](#)

[CloseProjectNoForce](#)

[Count](#)

[DeleteProject](#)

[DownloadJobResults](#)

[EnableAutoSave](#)

[ExportOptionsFiles](#)

[GetActiveProject](#)

[GetActiveScheduler](#)

[GetActiveSchedulerInfo](#)

[GetAutoSaveEnabled](#)

[GetBuildDateTimeString](#)

[GetCustomMenuSet](#)

[GetDefaultUnit](#)

[GetDesktopConfiguration](#)

[GetDistributedAnalysisMachines](#)

[GetDistributedAnalysisMachinesForDesignType](#)

[GetExeDir](#)

[GetGDIObjectCount](#)

[GetLibraryDirectory](#)

[GetLocalizationHelper](#)

[GetMessages](#)

[GetPersonalLibDirectory](#)

[GetPPELicensingEnabled](#)

[GetProcessID](#)

[GetProjectDirectory](#)

[GetProjectList](#)

[GetProjects](#)

[GetRegistryInt](#)

[GetRegistryString](#)

[GetRunningInstancesMgr](#)

[GetSchematicEnvironment](#)

[GetScriptingToolsHelper](#)

[GetSysLibDirectory](#)

[GetTempDirectory](#)

[GetUserLibDirectory](#)

[GetVersion](#)

[IsFeatureEnabled](#)

[KeepDesktopResponsive](#)

[LaunchJobMonitor](#)

[NewProject](#)

[OpenAndConvertProject](#)

[OpenMultipleProjects](#)

[OpenProject](#)

[OpenProjectWithConversion](#)

[PauseRecording](#)

[PauseScript](#)

[Print](#)

[QuitApplication](#)

[RefreshJobMonitor](#)

[ResetLogging](#)

[RestoreProjectArchive](#)

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

[StopSimulations](#)

[SubmitJob](#)

[Tile Windows](#)

Related Topics:

[Desktop Commands For Registry Values](#)

[ImportExport Tool Commands](#)

AddMessage

Add a message with severity and context to message window.

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	<code>oDesktop.AddMessage("Project1", "Icepak", 0, "This is a test message", "")</code>

VB Syntax	AddMessage <projectName>, <designName>, <severity>, <msg>, <category>
VB Example	oDesktop.AddMessage "Project1", "Icepak1", 0, "This is a test message", ""

ClearMessages

Clears messages, optionally specifying severity and design.

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

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

VB Syntax	ClearMessages <projectName>, <designName>, <severity>
------------------	---

VB Example

```
Set oAnsoftApp = CreateObject("Ansoft.ElectronicsDesktop")
Set oDesktop = oAnsoftApp.GetAppDesktop()
oDesktop.ClearMessages "", "", 3
```

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

```
oDesktop.CloseAllWindows ()
```

VB Syntax

```
CloseAllWindows
```

VB Example

```
oDesktop.CloseAllWindows
```

CloseProject

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

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

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

VB Syntax	CloseProject <ProjectName>
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 <ProjectName>	Type String	Description The name of the project already on the Desktop that is to be closed, without path or extension
Return Value	None		

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

VB Syntax	CloseProjectNoForce <ProjectName>
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</pre>

```
Dim oDesktop
Dim oProject
Dim oDesign
Dim oEditor
Dim oModule
Dim oProjects
Set oAnsoftApp = CreateObject("Ansoft.ElectronicsDesktop")
Set oDesktop = oAnsoftApp.GetAppDesktop()
oDesktop.RestoreWindow
Dim projects
set projects = oDesktop.GetProjects()
for i = 0 to projects.Count - 1
    msgbox projects(i).GetName()
    dim designs
    set designs = projects(i).GetDesigns()
    for j = 0 to designs.Count - 1
        msgbox designs(j).GetName()
    next
next
```

DeleteProject

Deletes a project from disk.

UI Access	Edit > Delete								
Parameters	<table border="1"><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><ProjectName></td><td>String</td><td>Name of the project</td></tr></table>			Name	Type	Description	<ProjectName>	String	Name of the project
Name	Type	Description							
<ProjectName>	String	Name of the project							
Return Value	None								

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

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

DownloadJobResults

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

UI Access	Select Scheduler								
Parameters	<table border="1"><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><jobID></td><td>String</td><td>Provide the job ID of the target job. The job ID must be able to be found in cur-</td></tr></table>			Name	Type	Description	<jobID>	String	Provide the job ID of the target job. The job ID must be able to be found in cur-
Name	Type	Description							
<jobID>	String	Provide the job ID of the target job. The job ID must be able to be found in cur-							

		rent.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	boolDownloadCompleted = oDesktop.DownloadJobResults ("012345678901234567890", "C:\\\\projects\\\\basic.aedt", "C:\\\\projects\\\\DownloadResults\\\\", "*")

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

DeleteRegistryEntry

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

UI Access	N/A								
Parameters	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><pathToRegistry></td> <td>String</td> <td>Path to Registry entry.</td> </tr> </tbody> </table>			Name	Type	Description	<pathToRegistry>	String	Path to Registry entry.
Name	Type	Description							
<pathToRegistry>	String	Path to Registry entry.							

Return Value	Boolean: <ul style="list-style-type: none">• True – Key has been deleted.• False – Key does not exist.
---------------------	---

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

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

DoesRegistryValueExist

Determines whether a registry value exists.

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

Python Syntax	DoesRegistryValueExist(<KeyName>)
Python Example	Exist = oDesktop.DoesRegistryValueExist('Desktop/ActiveDSOConfigurations/Icepak')

VB Syntax	DoesRegistryValueExist(<KeyName>)
VB Example	bExist = oDesktop.DoesRegistryValueExist("Desktop/ActiveDSOConfigurations/Icepak")

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	oDesktop.EnableAutoSave(True)

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 <DestinationDirectory>	Type String	Description 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>

GetAutoSaveEnabled

Checks whether the autosave feature is enabled.

UI Access	N/A
Parameters	None.
Return Value	Integer: <ul style="list-style-type: none"> • 1 – Autosave is enabled.

	<ul style="list-style-type: none">• 0 – Autosave is not enabled.
--	--

Python Syntax	GetAutoSaveEnabled()
Python Example	Enabled = oDesktop.GetAutoSaveEnabled()

VB Syntax	GetAutoSaveEnabled
VB Example	Enabled = oDesktop.GetAutoSaveEnabled()

GetBuildDateString

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	GetBuildDateString()
Python Example	oDesktop.GetBuildDateString()

VB Syntax	GetBuildDateTimeString
VB Example	dnt = oDesktop.GetBuildDateTimeString

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 <code><type></code>	Type String	Description String containing a type of measurement. Valid strings are (case insensitive): <ul style="list-style-type: none">• "Acceleration"• "Angle"• "AngularAcceleration"• "AngularDamping"• "AngularSpeed"• "Capacitance"• "Conductance"• "Current"• "CurrentChangeRate"• "DataRate"• "DeltaH" (Magnetic Field Strength)• "Density"• "Flux"• "Force"• "Frequency"• "Inductance"• "Length"

			<ul style="list-style-type: none"> • "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	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><code><designTypeName></code></td> <td>String</td> <td>The name of the type of design, such as "Twin Builder", "HFSS", HFSS-IE", "Maxwell 3D", "Maxwell 2D", "RMxprt", "EM Design", "Circuit", "System", "Q3D Extractor", "2D Extractor"</td> </tr> </tbody> </table>	Name	Type	Description	<code><designTypeName></code>	String	The name of the type of design, such as "Twin Builder", "HFSS", HFSS-IE", "Maxwell 3D", "Maxwell 2D", "RMxprt", "EM Design", "Circuit", "System", "Q3D Extractor", "2D Extractor"
Name	Type	Description					
<code><designTypeName></code>	String	The name of the type of design, such as "Twin Builder", "HFSS", HFSS-IE", "Maxwell 3D", "Maxwell 2D", "RMxprt", "EM Design", "Circuit", "System", "Q3D Extractor", "2D Extractor"					
Return Value	Object; returns a collection of machine names.						

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

VB Syntax	GetDistributedAnalysisMachinesForDesignType <designTypeName>
VB Example	<pre>Set machineNames =oDesktop. GetDistributedAnalysisMachinesForDesignType ("Icepak")</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/AnsysEM/v232/Win64/'

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

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

GetGDIObjectCount

Note:

This command is for internal Ansys use only.

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

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

GetLibraryDirectory

Get the path to the SysLib directory.

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

	The path to the SysLib directory.
--	-----------------------------------

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

VB Syntax	GetLibraryDirectory
VB Example	MsgBox oDesktop.GetLibraryDirectory

VB Example:

```
-----  
message box returns the path in this example  
-----
```

```
Dim oAnsoftApp  
Dim oDesktop  
Dim oProject  
Dim oDesign  
Dim oEditor  
Dim oModule  
  
Set oAnsoftApp = CreateObject("Ansoft.ElectronicsDesktop")
```

```

Set oDesktop = oAnsoftApp.GetAppDesktop()
oDesktop.RestoreWindow
libdir = oDesktop.GetLibraryDirectory
msgbox(oDesktop.GetLibraryDirectory())

```

GetLocalizationHelper

Note:

This command is for internal Ansys use only.

Returns the object for the localization helper.

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

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

VB Syntax	GetLocalizationHelper
------------------	-----------------------

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

GetMessages

Get the messages from a specified project and design.

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

Python Syntax	<code>GetMessages (<ProjectName>, <DesignName>, <Severity>)</code>
Python Example	<code>Messages = oDesktop.GetMessages ("MyProject", "Icepak1", 1)</code>

VB Syntax	GetMessages <ProjectName>, <DesignName>, <Severity>
VB Examples	Messages = oDesktop.GetMessages "MyProject", "Icepak1", 1

GetPersonalLibDirectory

Get the path to the PersonalLib directory.

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

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

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

GetPPELicensingEnabled

Returns whether the PPE licensing is enabled.

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

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

VB Syntax	<code>GetPPELicensingEnabled</code>
VB Example	<code>oDesktop.GetPPELicensingEnabled</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	GetProcessID ()
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	<code>oDesktop.GetProjects ()</code>

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

GetRegistryInt

Obtains registry key integer value.

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

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

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

GetRegistryString

Obtains registry key string value.

UI Access	N/A						
Parameters	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td><KeyName></td><td>String</td><td>Full name of registry key, including path.</td></tr></tbody></table>	Name	Type	Description	<KeyName>	String	Full name of registry key, including path.
Name	Type	Description					
<KeyName>	String	Full name of registry key, including path.					
Return Value	String if the string value is found. Return as Bad-Argument-Value if registry key does not exist or it is not a string value.						

Python Syntax	GetRegistryString(<KeyName>)
Python Example	<pre>activeDSO = oDesktop.GetRegistryString('Desktop/ActiveDSOConfigurations/Icepak')</pre>

VB Syntax	GetRegistryString(<KeyName>)
VB Example	<pre>activeDSO = oDesktop.GetRegistryString("Desktop/ActiveDSOConfigurations/Icepak")</pre>

GetRunningInstancesMgr

Returns the object of the Running Instances Manager.

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

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

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

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:

	<ul style="list-style-type: none">• 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	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>None</td><td></td><td></td></tr></tbody></table>			Name	Type	Description	None		
Name	Type	Description							
None									
Return Value	Object								

	ScriptingTools helper object
--	------------------------------

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

VB Syntax	<code>GetScriptingToolsHelper</code>
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 SysLib directory.

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

VB Syntax	<code>GetSysLibDirectory</code>
------------------	---------------------------------

VB Example

```
oDesktop.GetSysLibDirectory
```

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

```
oDesktop.GetTempDirectory()
```

VB Syntax

```
GetTempDirectory
```

VB Example

```
oDesktop.GetTempDirectory
```

GetUserLibDirectory

Gets the path to the UserLib directory.

UI Access	N/A
Parameters	None.

Return Value	Stringpath to the UserLib directory.
---------------------	--------------------------------------

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

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

GetVersion

Returns a string representing the version.

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

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

VB Syntax	GetVersion()
------------------	--------------

VB Example

```
oDesktop.GetVersion
```

IsFeatureEnabled

Returns a Boolean for whether a queried feature is enabled.

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

Python Syntax	IsFeatureEnabled()
Python Example	<pre>import ScriptEnv ScriptEnv.Initialize("Ansoft.ElectronicsDesktop") oDesktop.RestoreWindow() feature_strs = ["SF3519", "F195709_EXPORT_TO_EMIT", "F362235_EMIT_RESULTS_WINDOW_IMPROVEMENTS",_ "F136736_SBR_Rough_Surface", "F353006_VOLUMETRIC_SBR", "F359673_SBR_MULTISTATE_ARRAY",_ "F393115_SWE_ANTENNA",_ "S196592_SBR_Directivity", "F432541_VSBR_IMPROVEMENTS", "F353007_VRT_FILTERS_ENHANCE",_ "S540337_SBR_REGION_LOSS_DIRECTIVITY",_ "F11941_VRT_CURRENT_DENSITY", "S544593_SBR_3D_COMPONENT_ARRAY", "F539850_SBR_GO_BLOCKAGE", "F540275_SBR_RAYSTATS"]</pre>

```

results = [oDesktop.IsFeatureEnabled(str) for str in feature_strs]

result_txt = open("C:/Users/MyResults/Downloads/results.txt", "w")

for i in range(len(feature_strs)):

    result_txt.write('%s : %s\n' % (feature_strs[i], results[i]))

result_txt.close()

```

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 <i><MinTimeInMilliseconds></i>	Type Integer	Description 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 (<i><TimeInMilliseconds></i>)
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	<table border="1"><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><projectPath></td><td>String</td><td>Path to the project file to be monitored</td></tr></table>	Name	Type	Description	<projectPath>	String	Path to the project file to be monitored
Name	Type	Description					
<projectPath>	String	Path to the project file to be monitored					
Return Value	None						

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

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

NewProject

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

UI Access	File > New.
------------------	-------------

Parameters	None.
Return Value	Object, the project that is added.

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

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

OpenMultipleProjects

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

UI Access	N/A									
Parameters	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><directory></td> <td>String</td> <td>Path to the projects.</td> </tr> <tr> <td><fileType></td> <td>String</td> <td>Type of projects to open.</td> </tr> </tbody> </table>	Name	Type	Description	<directory>	String	Path to the projects.	<fileType>	String	Type of projects to open.
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	<pre>oProject = oDesktop.OpenAndConvertProject("c:\files\optimtee.icepak", "*.aedt")</pre>

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

OpenProject

Opens a specified project.

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

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

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

OpenProjectWithConversion

Note:

This command is for internal Ansys use only.

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

PauseRecording

Temporarily stop script recording.

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

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

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

PauseScript

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

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

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

VB Syntax	PauseScript <Message>
VB Example	<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: <ul style="list-style-type: none">• "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.
Return Value	None.		

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

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

RestoreProjectArchive

Restores a previously archived project to a specified path.

UI Access	File > Restore Archive.		
Parameters	Name	Type	Description
	<ArchiveFilePath>	String	Path to archived file
	<ProjectFilePath>	String	Path to restore location
	<OverwriteExistingFiles>	Boolean	True to overwrite an existing file of the same name; else False.
	<OpenProjectAfterRestore>	Boolean	True to open the project after it is restored; else False.
Return Value	None.		

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

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

RestoreWindow

Restores a minimized Desktop window.

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

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

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

ResumeRecording

Resume recording a script.

UI Access	N/A
Parameters	None.
Return Value	None

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

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

RunACTWizardScript

Note:

This command is for internal Ansys use only.

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

RunProgram

Runs an external program.

UI Access	NA			
Parameters	<table><tr><td>Name</td><td>Type</td><td>Description</td></tr></table>	Name	Type	Description
Name	Type	Description		

	<i><ProgName></i>	String	Name of the program to run.
	<i><ProgPath></i>	String	Location of the program. Pass in an empty string to use the system path.
	<i><WorkPath></i>	String	Working directory in which program will start.
	<i><ArgArray></i>	Array of Strings	Arguments to pass to the program. If no arguments, pass in <code>None</code>
Return Value	<code>None</code>		

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

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

RunScript

Launches another script from within the script currently being executed.

UI Access	Tools>Run Script		
Parameters	Name	Type	Description

	<SScriptPath>	String	<p>Name or full path of the script to execute.</p> <p>If the full path to the script is not specified, Twin Builder searches for the specified script in the following locations, in this order:</p> <ol style="list-style-type: none"> 1. Personal library directory. <p>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.</p> <ol style="list-style-type: none"> 2. User library directory. <p>This is the userlib 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.</p> <ol style="list-style-type: none"> 3. System library directory. <p>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.</p> <ol style="list-style-type: none"> 4. HFSS installation directory.
Return Value	<p>Long</p> <p>the return code for the script method.</p>		

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

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

RunScriptWithArguments

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

UI Access	NA		
Parameters	Name <ScriptPath>	Type String	Description The name or full path of the script to execute. If the full path to the script is not specified, the software looks for the script in the following locations: <ul style="list-style-type: none">• Personal library directory: "PersonalLib". The PersonalLib directory can be specified in Tools>Options>General Options on the 'Project Options' tab.• User library directory: directory "userlib". The UserLib directory can be specified in Tools>Options>General Options on the 'Project Options' tab.• System library directory: directory "syslib". The SysLib directory can be specified in Tools>Options>General Options on the 'Project Options' tab.

		<ul style="list-style-type: none"> • 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):

			<ul style="list-style-type: none"> • 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
	<code><address (optional)></code>	String	String specifying the IP address or hostname of the head node or for the remote host running the RSM service.
	<code><username (optional)></code>	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.
	<code><forcePasswordEntry (optional)></code>	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	<code>Select Scheduler(<option>, <address>, <username>, <forcePasswordEntry>)</code>
Python Example	<pre>result = oDesktop.SelectScheduler("Windows HPC", "headnode.win.example.com")</pre>

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

SetActiveProject

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

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

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

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

SetActiveProjectByPath

Specifies the name of the project that should become active in the desktop. Returns that project. If a user has two projects open with the same name, the result of `SetActiveProject` is ambiguous (the first one listed is selected). This command permits unambiguous specification of the active project.

UI Access	N/A		
Parameters	Name <i><ProjectName></i>	Type String	Description 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	<code>SetActiveProjectByPath(<ProjectName>)</code>
Python Example	<code>oProject = oDesktop.SetActiveProjectByPath ("c:\Projects\MyProject.aedt")</code>

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

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: <ul style="list-style-type: none">• 'Default'• 'EM'• 'RF'• 'RF.0'• 'SI'• 'SI1.0'• 'SI2.0'• 'Twin Builder'
Return Value	None		

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

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

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	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><configName></td> <td>String</td> <td> Name of desired Desktop configuration. Can be one of: <ul style="list-style-type: none"> • 'All' • 'EM' • 'RF' • 'SI' • 'Twin Builder' </td> </tr> </tbody> </table>			Name	Type	Description	<configName>	String	Name of desired Desktop configuration. Can be one of: <ul style="list-style-type: none"> • 'All' • 'EM' • 'RF' • 'SI' • 'Twin Builder'
Name	Type	Description							
<configName>	String	Name of desired Desktop configuration. Can be one of: <ul style="list-style-type: none"> • 'All' • '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 <code><DirectoryPath></code>	Type String	Description The path to the SysLib Directory
Return Value	None		

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

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

SetProjectDirectory

Sets the project directory path.

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

Return Value	None.
---------------------	-------

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

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

SetRegistryFromFile

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

UI Access	N/A							
Parameters	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><filePath></td> <td>String</td> <td>Full file path of registry file.</td> </tr> </tbody> </table>		Name	Type	Description	<filePath>	String	Full file path of registry file.
Name	Type	Description						
<filePath>	String	Full file path of registry file.						
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	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td><KeyName></td><td>String</td><td>Full name of registry key, including path.</td></tr><tr><td><int></td><td>Integer</td><td>Integer value to be assigned to registry key.</td></tr></tbody></table>			Name	Type	Description	<KeyName>	String	Full name of registry key, including path.	<int>	Integer	Integer value to be assigned to registry key.
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	<code>oDesktop.SetRegistryInt('Desktop/Settings/ProjectOptions/Icepak/UpdateReportsDynamicallyOnEdits', 0)</code>

VB Syntax	SetRegistryInt <KeyName> <int>
VB Example	<code>oDesktop.SetRegistryInt "Desktop/Settings/ProjectOptions/Icepak/UpdateReportsDynamicallyOnEdits", 0</code>

SetRegistryString

Sets registry key to a string value.

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

VB Syntax	SetRegistryString <KeyName>, <value>
VB Example	<code>oDesktop.SetRegistryString "Desktop/ActiveDSOConfigurations/Icepak", "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: <ul style="list-style-type: none">• 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	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><windowName></td> <td>String</td> <td>The window name (for example, "Message Manager", "Component Libraries", "Properties")</td> </tr> <tr> <td><show></td> <td>Boolean</td> <td>True to show; False to hide.</td> </tr> </tbody> </table>	Name	Type	Description	<windowName>	String	The window name (for example, "Message Manager", "Component Libraries", "Properties")	<show>	Boolean	True to show; False to hide.
Name	Type	Description								
<windowName>	String	The window name (for example, "Message Manager", "Component Libraries", "Properties")								
<show>	Boolean	True to show; False to hide.								
Return Value	None.									

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

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

Sleep

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

UI Access	NA		
Parameters	Name	Type	Description
	<code><TimeInMilliseconds></code>	Integer	The time that the execution should be suspended in milliseconds
Return Value	None		

Python Syntax	Sleep (<TimeInMilliseconds>)
Python Example	<code>oDesktop.Sleep (1000)</code>

VB Syntax	Sleep <TimeInMilliseconds>
VB Example	<code>oDesktop.Sleep 1000</code>

SubmitJob

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

UI Access	Tools > Job Management > Submit Job.		
Parameters	Name	Type	Description
	<settingsPath>	String	Path to the settings file (exported from the Submit Job GUI) to use for submission.
	<projectPath>	String	Path to the project file to use in the batch job. This could be an archive (.aedtz file) or an un-archived project.
	<design (optional)>	String	Name of the design to use for batch solve.
	<setup (optional)>	String	Name of the specific setup to solve.
Return Value	Array of job ID strings (empty if no jobs submitted).		

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

VB Syntax	SubmitJob <settingsPath>, <projectPath>, <design>, <setup>
------------------	--

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

TileWindows

Arrange all open windows in a tiled format.

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

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

VB Syntax	TileWindows <i><TilingFlag></i>
VB Example	<code>oDesktop.TileWindows 0</code>

Desktop Commands For Registry Values

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

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

1. Start Icepak.
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/Icepak", "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	<table border="1"><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><pathToRegistry></td><td>String</td><td>Path to Registry entry.</td></tr></table>			Name	Type	Description	<pathToRegistry>	String	Path to Registry entry.
Name	Type	Description							
<pathToRegistry>	String	Path to Registry entry.							
Return Value	<p>Boolean:</p> <ul style="list-style-type: none">• 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	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><KeyName></td> <td>String</td> <td>Full name of registry key, including path.</td> </tr> </tbody> </table>	Name	Type	Description	<KeyName>	String	Full name of registry key, including path.
Name	Type	Description					
<KeyName>	String	Full name of registry key, including path.					
Return Value	<p>Boolean:</p> <ul style="list-style-type: none"> • True – Key exists. • False – Key does not exist. 						

Python Syntax	DoesRegistryValueExist(<KeyName>)
Python Example	<pre>Exist = oDesktop.DoesRegistryValueExist ('Desktop/ActiveDSOConfigurations/Icepak')</pre>

VB Syntax	DoesRegistryValueExist(<KeyName>)
VB Example	<pre>bExist = oDesktop.DoesRegistryValueExist ("Desktop/ActiveDSOConfigurations/Icepak")</pre>

GetRegistryInt

Obtains registry key integer value.

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

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

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

GetRegistryString

Obtains registry key string value.

UI Access	N/A							
Parameters	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td><KeyName></td><td>String</td><td>Full name of registry key, including path.</td></tr></tbody></table>		Name	Type	Description	<KeyName>	String	Full name of registry key, including path.
Name	Type	Description						
<KeyName>	String	Full name of registry key, including path.						
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	activeDSO = oDesktop.GetRegistryString('Desktop/ActiveDSOConfigurations/Icepak')

VB Syntax	GetRegistryString(<KeyName>)
VB Example	activeDSO = oDesktop.GetRegistryString("Desktop/ActiveDSOConfigurations/Icepak")

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	<table border="1"> <tr> <td>Name</td> <td>Type</td> <td>Description</td> </tr> <tr> <td><filePath></td> <td>String</td> <td>Full file path of registry file.</td> </tr> </table>	Name	Type	Description	<filePath>	String	Full file path of registry file.
Name	Type	Description					
<filePath>	String	Full file path of registry file.					
Return Value	Success if configuration is imported. Bad argument value if the file is not found or does not contain valid analysis configuration data.						

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

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

SetRegistryInt

Sets registry key to an integer value.

UI Access	N/A		
Parameters	Name	Type	Description
	<KeyName>	String	Full name of registry key, including path.
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	<code>oDesktop.SetRegistryInt('Desktop/Settings/ProjectOptions/Icepak/UpdateReportsDynamicallyOnEdits', 0)</code>

VB Syntax	SetRegistryInt <KeyName> <int>
VB Example	<code>oDesktop.SetRegistryInt "Desktop/Settings/ProjectOptions/Icepak/UpdateReportsDynamicallyOnEdits", 0</code>

SetRegistryString

Sets registry key to a string value.

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

Python Syntax	<code>SetRegistryString(<KeyName>, <value>)</code>
Python Example	<code>oDesktop.SetRegistryString ('Desktop/ActiveDSOConfigurations/Icepak', 'Local')</code>

VB Syntax	<code>SetRegistryString <KeyName>, <value></code>
VB Example	<code>oDesktop.SetRegistryString "Desktop/ActiveDSOConfigurations/Icepak", "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	<table border="1"><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><ANFfilename></td><td>String</td><td>Full path of ANF file.</td></tr></table>	Name	Type	Description	<ANFfilename>	String	Full path of ANF file.
Name	Type	Description					
<ANFfilename>	String	Full path of ANF file.					
Return Value	None.						

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

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

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

ImportANFv2

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

UI Access	File > Import > ANF.
-----------	-----------------------------------

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

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

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

```

Dim oEditor
Dim oModule
Dim oProjects
Dim omachine
Dim oTool
Set oAnsoftApp = CreateObject("Ansoft.ElectronicsDesktop")
Set oDesktop = oAnsoftApp.GetAppDesktop()
oDesktop.RestoreWindow
Set oTool = oDesktop.GetTool("ImportExport")
oTool.ImportANFV2 "C:/Users/jdoe/Documents/SAMPLEFILES/my_model.anf", _
"C:/Users/jdoe/Documents/Ansoft/my_model.aedb", _
"C:/Users/jdoe/Documents/Ansoft/my_model.xml", ""

```

ImportAutoCAD

Imports an AutoCAD file into a new project.

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

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

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

ImportAWRMicrowaveOffice

Imports an AWRMicrowaveOffice file into a new project.

UI Access	File > Import > AWRMicrowaveOffice.												
Parameters	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><xmlFileName></td> <td>String</td> <td>Full path of XML file.</td> </tr> <tr> <td><outputPathName></td> <td>String</td> <td>Full path of output file.</td> </tr> <tr> <td><logFileName></td> <td>String</td> <td>Full path of log file.</td> </tr> </tbody> </table>	Name	Type	Description	<xmlFileName>	String	Full path of XML file.	<outputPathName>	String	Full path of output file.	<logFileName>	String	Full path of log file.
Name	Type	Description											
<xmlFileName>	String	Full path of XML file.											
<outputPathName>	String	Full path of output file.											
<logFileName>	String	Full path of log file.											
Return Value	None.												

Python Syntax	ImportAWRMicrowaveOffice (<xmlFileName>, <outputPathName>, <logFileName>)
----------------------	---

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	<pre> ImportAWRMicrowaveOffice <xmlFileName>, <outputPathName>, <logFileName> </pre>
VB Example	<pre> Dim oAnsoftApp Dim oDesktop Dim oProject Dim oDesign Dim oEditor Dim oModule Dim oProjects Dim omachine Dim oTool Set oAnsoftApp = CreateObject("Ansoft.ElectronicsDesktop") Set oDesktop = oAnsoftApp.GetAppDesktop() oDesktop.RestoreWindow Set oTool = oDesktop.GetTool("ImportExport") </pre>

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

ImportEDB

Imports an EDB file into a new project.

UI Access	File > Import > EDB.						
Parameters	<table border="1"><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><edbFileName></td><td>String</td><td>Full path of EDB file.</td></tr></table>	Name	Type	Description	<edbFileName>	String	Full path of EDB file.
Name	Type	Description					
<edbFileName>	String	Full path of EDB file.					
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.
Return Value	None.		

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

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

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

ImportGDSII

Imports a GDSII file into a new project.

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

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

VB Syntax	<code>ImportGDSII <gdsiiFileName>, <outputPathName>, <controlFileName>, <propertyFileName></code>
VB Example	<code>Set oTool = oDesktop.GetTool("ImportExport")</code>

```
oTool.ImportGDSII "C:/Files/test.gds", "C:/Files/test.aedb.edb", _
"C:/Files/test.xml", "C:/Files/test.txt"
```

ImportGerber

Imports a GERBER file into a new project.

UI Access	File > Import > GERBER.		
Parameters	Name	Type	Description
	<gerberFileName>	String	Full path of GERBER file.
	<outputPathName>	String	Full path of EDB file to create during import.
Return Value	None.		

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

VB Syntax	ImportGerber <gerberFileName>, <outputPathName>, <controlFileName>
VB Example	<pre> Set oTool = oDesktop.GetTool("ImportExport")</pre>

```

oTool.ImportGERBER "C:/Files/test.gbr", "C:/Files/test.aedb.edb", _
"C:/Files/test.xml"

```

ImportIDF

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

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

	<ModelDeviceAsRect>	bool	True or False
	<Cutoff>	bool	True or False
	<IncludeDrilledHoles>	bool	True or False
	<ReplaceDevices>	bool	True or False
Return Value	None.		

Python Syntax	ImportIDF_1 (<NAME>, <Board>, <Library>, <Control>, <Filters>, <CreateFilteredAsNonModel>, <NAME>, <NAME>, <HighSurfThickness>, <LowSurfThickness>, <InternalLayerThickness>, <NumInternalLayer>, <HighSurfaceCopper>, <LowSurfaceCopper>, <InternalLayerCopper>, <TraceMaterial>, <SubstrateMaterial>, <CreateBoard>, <ModelBoardAsRect>, <ModelDeviceAsRect>, <Cutoff>, <IncludeDrilledHoles>, <ReplaceDevices>)
Python Example	<pre> oDesign.ImportIDF(["NAME:Settings", "Board:=" , "C:\\\\Users\\\\Model Files\\\\brd_board.emn", "Library:=" , "C:\\\\Users\\\\Model Files\\\\brd_board.emp", "Control:=" , "C:\\\\Users\\\\Model Files\\\\brd_board.xml", "Filters:=" , ["HeightExclude2D"], "CreateFilteredAsNonModel:=", False, ["NAME:definitionOverridesMap"], ["NAME:instanceOverridesMap"], "HighSurfThickness:=" , "0.07mm", "LowSurfThickness:=" , "0.07mm", "InternalLayerThickness:=" , "0.07mm",]) </pre>

```

    "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
  ]
)

```

VB Syntax	<code>ImportIDF <idfFileName>, <outputPathName>, <controlFileName>, <libFileName></code>
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", </pre>

```
"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)
```

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	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><code><idfFileName></code></td> <td>String</td> <td>Full path of GERBER file.</td> </tr> <tr> <td><code><libFileName></code></td> <td>String</td> <td>Optional. Full path of library file. Pass empty string if none.</td> </tr> <tr> <td><code><controlFileName></code></td> <td>String</td> <td>Optional. Full path of XML control file. Pass empty string if none.</td> </tr> <tr> <td><code><project></code></td> <td>String</td> <td>Name of project to merge with.</td> </tr> <tr> <td><code><design></code></td> <td>String</td> <td>Name of design to merge with.</td> </tr> </tbody> </table>	Name	Type	Description	<code><idfFileName></code>	String	Full path of GERBER file.	<code><libFileName></code>	String	Optional. Full path of library file. Pass empty string if none.	<code><controlFileName></code>	String	Optional. Full path of XML control file. Pass empty string if none.	<code><project></code>	String	Name of project to merge with.	<code><design></code>	String	Name of design to merge with.
Name	Type	Description																	
<code><idfFileName></code>	String	Full path of GERBER file.																	
<code><libFileName></code>	String	Optional. Full path of library file. Pass empty string if none.																	
<code><controlFileName></code>	String	Optional. Full path of XML control file. Pass empty string if none.																	
<code><project></code>	String	Name of project to merge with.																	
<code><design></code>	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', </pre>

	'C:/Files/test.xml', 'Project1', 'Design12')
--	--

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

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

	<code><HighSurfaceCopper></code>	int	High surface percent coverage value
	<code><LowSurfaceCopper></code>	int	Low surface percent coverage value
	<code><InternalLayerCopper></code>	int	Internal layer percent coverage value
	<code><TraceMaterial></code>	string	Trace material name
	<code><SubstrateMaterial></code>	int	Substrate material name
	<code><CreateBoard></code>	bool	True or False
	<code><Model/BoardAsRect></code>	bool	True or False
	<code><Model/DeviceAsRect></code>	bool	True or False
	<code><Cutoff></code>	bool	True or False
	<code><ReplaceDevices></code>	bool	True or False
Return Value	None		

Python Syntax	<code>ImportIDX (<NAME>, <Board>, <Library>, <Control>, <Filters>, <CreateFilteredAsNonModel>, <NAME>, <NAME>, <HighSurfThickness>, <LowSurfThickness>, <InternalLayerThickness>, <NumInternalLayer>, <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"]]) </pre>

```

        ],
        "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
    )
)

```

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", </pre>

```
"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)
```

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

ImportODB

Imports an ODB++ file into a new project.

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

Python Syntax	ImportODB(<odbFileName>, <outputPathName>, <controlFileName>)
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	ImportODB <odbFileName>, <outputPathName>, <controlFileName>
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	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><xflFileName></td> <td>String</td> <td>Full path of XFL file.</td> </tr> <tr> <td><outputPathName></td> <td>String</td> <td>Full path of EDB file to create during import.</td> </tr> <tr> <td><controlFileName></td> <td>String</td> <td>Optional. Full path of XML control file. Pass empty string if none.</td> </tr> </tbody> </table>	Name	Type	Description	<xflFileName>	String	Full path of XFL file.	<outputPathName>	String	Full path of EDB file to create during import.	<controlFileName>	String	Optional. Full path of XML control file. Pass empty string if none.
Name	Type	Description											
<xflFileName>	String	Full path of XFL file.											
<outputPathName>	String	Full path of EDB file to create during import.											
<controlFileName>	String	Optional. Full path of XML control file. Pass empty string if none.											
Return Value	None.												

Python Syntax	ImportXFL(<xflFileName>, <outputPathName>, <controlFileName>)
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	ImportXFL <xflFileName>, <outputPathName>, <controlFileName>
------------------	--

VB Example

```
Set oTool = oDesktop.GetTool("ImportExport")
oTool.ImportXFL "C:/Files/test.xfl", "C:/Files/test.aedb", _
"C:/Files/test.xml"
```

4 - Running Instances Manager Script Commands

The Running Instances Manager is a scripting object that lets you identify and connect to all running instances of Electronics Desktop. oDesktop objects that are returned provide full scripting functionality. Running Instances Manager commands should be executed by the oDesktop object. For example:

```
Set oRunningInstances = oDesktop.GetRunningInstancesMgr()
```

[GetAllRunningInstances](#)

[GetRunningInstanceByProcessID](#)

[GetRunningInstanceByProject](#)

GetAllRunningInstances

Returns a list of running instances of Ansys Electronics Desktop.

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

Python Syntax	GetAllRunningInstances()
Python Example	obj = oRunningInstances.GetAllRunningInstances()

VB Syntax	GetAllRunningInstances
VB Example	set obj = oRunningInstances.GetAllRunningInstances()

GetRunningInstanceByProcessID

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

UI Access	N/A								
Parameters	<table border="1"><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><processID></td><td>Integer</td><td>Process ID</td></tr></table>			Name	Type	Description	<processID>	Integer	Process ID
Name	Type	Description							
<processID>	Integer	Process ID							
Return Value	String of the returned instance.								

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

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

GetRunningInstancesMgr

Returns the object of the Running Instances Manager.

UI Access	N/A								
Parameters	<table border="1"><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>None</td><td></td><td></td></tr></table>			Name	Type	Description	None		
Name	Type	Description							
None									

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>

This page intentionally
left blank.

5 - Project Object Script Commands

Project commands should be executed by the oProject object. One example of accessing this object is:

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

[AddMaterial](#)

[AnalyzeAll](#)

[ClearMessages](#)

[Close](#)

[CopyDesign](#)

[CutDesign](#)

[DeleteDesign](#)

[DeleteToolObject](#)

[GetActiveDesign](#)

[GetChildNames \[Project\]](#)

[GetChildObject \[Project\]](#)

[GetChildTypes \[Project\]](#)

[GetConfigurableData](#)

[GetDefinitionManager](#)

[GetDependentFiles](#)

[GetDesign](#)

[GetDesigns](#)

[GetEDBHandle](#)
[GetLegacyName](#)
[GetName \[Project\]](#)
[GetObjPath \[Project\]](#)
[GetPath](#)
[GetPropEvaluatedValue](#)
[GetPropNames \[Project\]](#)
[GetPropSIValue](#)
[GetPropValue \[Project\]](#)
[GetProperties](#)
[GetPropertyValue](#)
[GetTopDesignList](#)
[InsertDesign](#)
[InsertDesignWithWorkflow](#)
[InsertToolObject](#)
[Paste \[Project Object\]](#)
[Redo \[Project Level\]](#)
[Rename](#)
[RunToolkit](#)
[Save](#)

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

AddDataset

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

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

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

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

VB Syntax	AddDataset <DatasetdataArray>
VB Example	<pre>oProject.AddDatasetArray("NAME:ds1", Array("NAME:Coordinates",</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))))
```

AddMaterial

Adds a local material.

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

	<i><MatProperty></i>	Array	<p>For simple material:</p> <pre>"<PropertyName>:=", <value></pre> <p>For anisotropic material:</p> <pre>["NAME:<PropertyName>", "property_type:=", "AnisoProperty", "unit:=", <Unit>, "component1:=", <value>, "component2:=", <value>, "component3:=", <value>)]</pre>
	<i><PropertyName></i>	String	<p>Should be one of the following (depending on the material, design, and solution types):</p> <p>Electromagnetic (Maxwell-exclusive material properties omitted, see Maxwell Scripting help):</p> <pre>"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"</pre>

		<p>Thermal (including solids, Icepak fluid flow, and Mechanical rotating fluid modeling):</p> <pre>"thermal_conductivity", "mass_density", "specific_heat", "thermal_expansion_coefficient", "thermal_material_type", "viscosity", "diffusivity", "molecular_mass", "clarity_type"</pre> <p>Structural:</p> <pre>"mass_density", "youngs_modulus", "poissons_ratio", "thermal_expansion_coefficient"</pre>
<Unit>	String	<p>Possible values (Maxwell-exclusive properties omitted, see Maxwell Scripting Help; other missing entries are unitless):</p> <pre>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"</pre>

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

```
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")
```

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>

ClearMessages

Clears information in the **Messages** window.

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>

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 <i><DesignName></i>	Type String	Description Name of the design to copy from.
Return Value	None.		

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

VB Syntax	CopyDesign <i><DesignName></i>
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 <i><DesignName></i>
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	<table border="1"><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><DatasetName></td><td>String</td><td>Name of the dataset found in the project.</td></tr></table>	Name	Type	Description	<DatasetName>	String	Name of the dataset found in the project.
Name	Type	Description					
<DatasetName>	String	Name of the dataset found in the project.					
Return Value	None.						

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

VB Syntax	DeleteDataset < <i>DatasetName</i> >
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	<table border="1"> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> <tr> <td><<i>DesignName</i>></td> <td>String</td> <td>Name of the design.</td> </tr> </table>	Name	Type	Description	< <i>DesignName</i> >	String	Name of the design.
Name	Type	Description					
< <i>DesignName</i> >	String	Name of the design.					
Return Value	None.						

Python Syntax	DeleteDesign (< <i>DesignName</i> >)
Python Example	<pre>oProject.DeleteDesign("IcepakDesign2")</pre>

VB Syntax	DeleteDesign < <i>DesignName</i> >
VB Example	<pre>oProject.DeleteDesign "IcepakDesign2"</pre>

DeleteToolObject

Note:

This command is for internal Ansys use only.

UI Access	N/A		
Parameters	Name <i><ObjectName></i>	Type String	Description 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>)
	<pre> oProject.EditDataset ("ds1" ["NAME:ds2", ["NAME:Coordinates", ["NAME:Coordinate", "X:=", 1, "Y:=", 2], ["NAME:Coordinate", "X:=", 3, "Y:=", 4]]]) oDesign.EditDataset ("ds1" </pre>
Python Example	

```
[ "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)))</pre>

```

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

```

EditMaterial

Modifies an existing material.

UI Access	View/Edit Materials command in the material editor		
Parameters	Name <i><OriginalName></i>	Type String	Description Name of the material before editing.

Name	Type	Description
<i><OriginalName></i>	String	Name of the material before editing.
<i><MatProperties></i>	Array	Structured array containing material properties: ["NAME:<New material name>", "CoordinateSystemType:=" , <string>, "BulkOrSurfaceType:=" , <integer>, ["NAME:PhysicsTypes", "set:=" , <array containing string physics types>], <Optional ModifierdataArray>, "permeability:=" , <string containing value>,]]

		<pre> "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 con- taining value>] </pre>
<code><Modi- fierdataArray></code>	Array	<p>Optional structured array containing thermal or spatial modifiers:</p> <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>", </pre>

		<pre> "Property::=" , <string property being modified>, "Index::=" , <integer>, "prop_modifier::=" , <"thermal_modifier" or "spatial_modifier">, "use_free_form::=" , <Boolean>, "free_form_value::=" , <string modifier value>,]]]] </pre>
Return Value	None.	

Python Syntax	EditMaterial (<OriginalName>, <MatProperties>)
Python Example	<p>Without Modifiers:</p> <pre> oDefinitionManager.EditMaterial("alumina_92pct", ["NAME:alumina_92pct", "CoordinateSystemType::=" , "Cartesian", "BulkOrSurfaceType::=" , 1,]) </pre>

```
[  
    "NAME:PhysicsTypes",  
    "set:=", ["Electromagnetic", "Thermal", "Structural"]  
,  
    "permittivity:=", "9.3",  
    "dielectric_loss_tangent:=", "0.008",  
    "thermal_conductivity:=", "26",  
    "mass_density:=", "3720",  
    "specific_heat:=", "790",  
    "youngs_modulus:=", "267000000000",  
    "poissons_ratio:=", "0.26",  
    "thermal_expansion_coeffcient:=", "7.2e-006"  
]  
)
```

With Thermal Modifier:

```
oDefinitionManager>EditMaterial("copper",  
[  
    "NAME:copper",  
    "CoordinateSystemType:=", "Cartesian",  
    "BulkOrSurfaceType:=", 1,
```

```
[  
    "NAME:PhysicsTypes",  
    "set:=" , ["Electromagnetic","Thermal","Structural"]  
],  
[  
    "NAME:ModifierData",  
    [  
        "NAME:ThermalModifierData",  
        "modifier_data:=" , "thermal_modifier_data",  
        [  
            "NAME:all_thermal_modifiers",  
            [  
                "NAME:one_thermal_modifier",  
                "Property::=" , "permittivity",  
                "Index::=" , 0,  
                "prop_modifier:=" , "thermal_modifier",  
                "use_free_form:=" , True,  
                "free_form_value:=" , "if(Temp > 1000cel, 1, if(Temp < -273.15cel,  
1, 1))"  
            ]  
        ]  
    ]  
]
```

```
],
"permeability:=" , "0.999991",
"conductivity:=" , "58000000",
"thermal_conductivity:=" , "400",
"mass_density:=" , "8933",
"specific_heat:=" , "385",
"youngs_modulus:=" , "120000000000",
"poissons_ratio:=" , "0.38",
"thermal_expansion_coefficient:=" , "1.77e-05"
])
```

Transient Solve, Non-linear Drude Data Plasma

```
# -----
# Script Recorded by Ansys Electronics Desktop Version 2023.2.0
# 14:23:56 Jun 17, 2022
# -----
import ScriptEnv
ScriptEnv.Initialize("Ansoft.ElectronicsDesktop")
oDesktop.RestoreWindow()
oProject = oDesktop.SetActiveProject("Drude_plasma_parameters_r231")
oDefinitionManager = oProject.GetDefinitionManager()
```

```
oDefinitionManager>EditMaterial("Drude",
[
    "NAME:Drude",
    "CoordinateSystemType:=", "Cartesian",
    "BulkOrSurfaceType:=" , 1,
    [
        "NAME:PhysicsTypes",
        "set:=" , ["Electromagnetic"]
    ],
    [
        "NAME:AttachedData",
        [
            "NAME:MatNonLinearDrudeFreqDepData",
            "property_data:=" , "nonlinear_drude_data",
            "EpsilonInfinity:=" , "1",
            "PlasmaFrequency:=" , "4.62348462366278GHz",
            "CollisionFrequency:=" , "0.00054491190162662GHz",
            "FieldBreakdown:=" , "10000V_per_meter",
            "PlasmaMaintainFrequency:=" , "2.31174231183139GHz",
            "NeutralDensity:=" , 2.65164580488373E+20,
            "ElectronDensity:=" , 2.65164580488373E+17,
```

```

    "CollisionRateConstant:=", 2.05499505485618E-15
]
]
])

```

VB Syntax	EditMaterial <OriginalName>, <MatProperties>
VB Example	<p>Without Modifiers:</p> <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", "specific_heat:=", "790", </pre>

```

"youngs_modulus:=", "267000000000",
"poissons_ratio:=", "0.26",
"thermal_expansion_coeffcient:=", "7.2e-006"
)

```

With Thermal Modifier:

```

oDefinitionManager>EditMaterial "copper",
    Array("NAME:copper",
        "CoordinateSystemType:=", "Cartesian",
        "BulkOrSurfaceType:=" , 1,
            Array("NAME:PhysicsTypes",
                "set:=" , Array("Electromagnetic","Thermal","Structural")),
        Array("NAME:ModifierData",
            Array("NAME:ThermalModifierData",
                "modifier_data:=" , "thermal_modifier_data",
                    Array("NAME:all_thermal_modifiers",
                        Array("NAME:one_thermal_modifier",
                            "Property:::=" , "permittivity",
                            "Index:::=" , 0,
                            "prop_modifier:=" , "thermal_modifier",
                            "use_free_form:=" , True,
                            "free_form_value:=" , "if(Temp > 1000cel, 1, if(Temp < -273.15cel,
1, 1))" )
                )
            )
        )
    )
)

```

```
        )
    )
)
),
"permeability:=" , "0.999991",
"conductivity:=" , "58000000",
"thermal_conductivity:=" , "400",
"mass_density:=" , "8933",
"specific_heat:=" , "385",
"youngs_modulus:=" , "120000000000",
"poissons_ratio:=" , "0.38",
"thermal_expansion_coefficient:=" , "1.77e-05"
)
```

ExportDataset

Exports a dataset to a named file. This can be executed by the oProject, or oDesign variables.

UI Access	Project > Datasets > Export.		
Parameters	Name <i><datasetFilePath></i>	Type String	Description The full path to the file.
Return Value	None.		

Python Syntax	ExportDataset (<datasetFileFullPath>)
Python Example	<pre>oProject.ExportDataset('e:/tmp/dsdata.txt') oDesign.ExportDataset('e:/tmp/dsdata.txt')</pre>

VB Syntax	ExportDataset <datasetFileFullPath>
VB Example	<pre>oProject.ExportDataset "e:/tmp/dsdata.txt" oDesign.ExportDataset "e:/tmp/dsdata.txt"</pre>

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	<pre>oDesign = oProject.GetActiveDesign()</pre>

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	<code>GetArrayVariables()</code>
Python Example	<code>oProject.GetArrayVariables()</code> <code>oDesign.GetArrayVariables()</code>

VB Syntax	<code>GetArrayVariables</code>
VB Example	<code>oProject.GetArrayVariables</code> <code>oDesign.GetArrayVariables</code>

GetChildNames [Project]

Returns the names of the project's child objects.

UI Access	N/A
------------------	-----

Parameters	Name <code><type></code>	Type String	Description (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	<code>GetChildNames (<type>)</code>
Python Example	<code>arrDesignNames = oProject.GetChildNames ()</code> <code>arrVarbleNames = oProject.GetChildNames ("Variable")</code>

VB Syntax	<code>GetChildNames (<type>)</code>
VB Example	<code>arrDesignNames = oProject.GetChildNames ()</code> <code>arrVarbleNames = oProject.GetChildNames ("Variable")</code>

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 = oDesktop.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	DefinitionManager object.

Python Syntax	GetDefinitionManager()
Python Example	<code>oDefinitionManager = oProject.GetDefinitionManager()</code>

VB Syntax	GetDefinitionManager
VB Example	<code>Set oDefinitionManager = oProject.GetDefinitionManager</code>

GetDependentFiles

Provides a list of the external files referenced in the project, including characteristic (for example, MDX) and coupled project files.

UI Access	N/A
Parameters	None.
Return Value	List of referenced files.

Python Syntax	GetDependentFiles()
Python Example	files = oProject.GetDependentFiles()

VB Syntax	GetDependentFiles
VB Example	files = oProject.GetDependentFiles

GetDesign

Returns the interface to a specific design in a given project.

UI Access	N/A						
Parameters	<table border="1"><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><designName></td><td>String</td><td>Name of the design.</td></tr></table>	Name	Type	Description	<designName>	String	Name of the design.
Name	Type	Description					
<designName>	String	Name of the design.					
Return Value	Object of the specified design.						

Python Syntax	GetDesign (<designName>)
Python Example	oProject.GetDesign ("HFSSDesign1")

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

VB Syntax	<code>GetDesigns</code>
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	<code>GetEDBHandle()</code>
Python Example	<code>oProject.GetEDBHandle()</code>

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

GetLegacyName

Obtains the legacy name of a project.

Note:

This command is for internal Ansys use only.

UI Access	N/A
Parameters	None.
Return Value	String containing the legacy project name.

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

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

GetName [Project]

Obtains the project name

UI Access	N/A
Parameters	None.
Return Value	String containing the project name, not including the path or extension.

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

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

GetObjPath [Project]

Obtains the project name from the full path.

UI Access	N/A
Parameters	None.
Return Value	String containing only the project name.

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

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

GetPath

Returns the location of the project on disk.

UI Access	N/A
Parameters	None.
Return Value	String containing the path to the project, not including the project name.

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

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

GetPropNames [Project]

Obtains the property name of the object. At the project level, GetPropNames always returns empty because the project is not associated with any property.

UI Access	N/A		
Parameters	Name <i><includeReadOnly></i>	Type Boolean	Description (Optional) <ul style="list-style-type: none">• True – Include read only props.• False – Do not include read only props.
Return Value	Empty array.		

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

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

GetPropertyValue [Project]

Returns the property value for the active project object, or specified property values.

UI Access	N/A		
Parameters	Name	Type	Description
	<code><propPath></code>	String	A child object's property path. See: Property Function Summary .
Return Value	String of property value.		

Python Syntax	GetProperty(<propPath>)
Python Example	<code>oProject.GetProperty("TeeModel/offset")</code>

VB Syntax	GetProperty <propPath>
VB Example	<code>oProject.GetProperty "TeeModel/offset"</code>

GetProperties

Gets a list of all the properties belonging to a specific <PropServer> and <PropTab>. This can be executed by the oProject, oDesign, or oEditor variables.

UI Access	N/A		
Parameters	Name	Type	Description
	<PropTab>	String	<p>One of the following, where tab titles are shown in parentheses:</p> <ul style="list-style-type: none"> • 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")
Return Value	Array of strings containing the names of the appropriate properties.		

Python Syntax	GetProperties(<PropTab>, <PropServer>)
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	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><PropTab></td> <td>String</td> <td> One of the following, where tab titles are shown in parentheses: <ul style="list-style-type: none"> • 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") </td> </tr> </tbody> </table>	Name	Type	Description	<PropTab>	String	One of the following, where tab titles are shown in parentheses: <ul style="list-style-type: none"> • 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") 		
Name	Type	Description							
<PropTab>	String	One of the following, where tab titles are shown in parentheses: <ul style="list-style-type: none"> • 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") 							

		<ul style="list-style-type: none"> • 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.GetPropertyValues("PassedParameterTab", k, "R") ... </pre>

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

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

GetVariableValue

Gets the value of a single specified variable. To get the value of project variables, execute this command using `oProject`. To get the value of local variables, use `oDesign`.

UI Access	N/A						
Parameters	<table border="1"><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><code><VarName></code></td><td>String</td><td>Name of the variable to access.</td></tr></table>	Name	Type	Description	<code><VarName></code>	String	Name of the variable to access.
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>

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>
--	-----------------------------------

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><datasetFilePath></code>	String	The full path to the file containing the dataset values. *.tab files recommended (see note below).
Return Value	None.		

Python Syntax	<code>ImportDataset (<datasetFilePath>, <optionalDatasetName>)</code>
Python Example	<code>oProject.ImportDataset('e:\tmp\dsdata.tab')</code> <code>oDesign.ImportDataset('e:\tmp\dsdata.tab')</code> <code>oProject.ImportDataset('e:\tmp\dsdata.tab', 'MyDatasetName')</code> <code>oDesign.ImportDataset('e:\tmp\dsdata.tab', 'MyDatasetName')</code>

VB Syntax	<code>ImportDataset <datasetFilePath>, <optionalDatasetName></code>
-----------	---

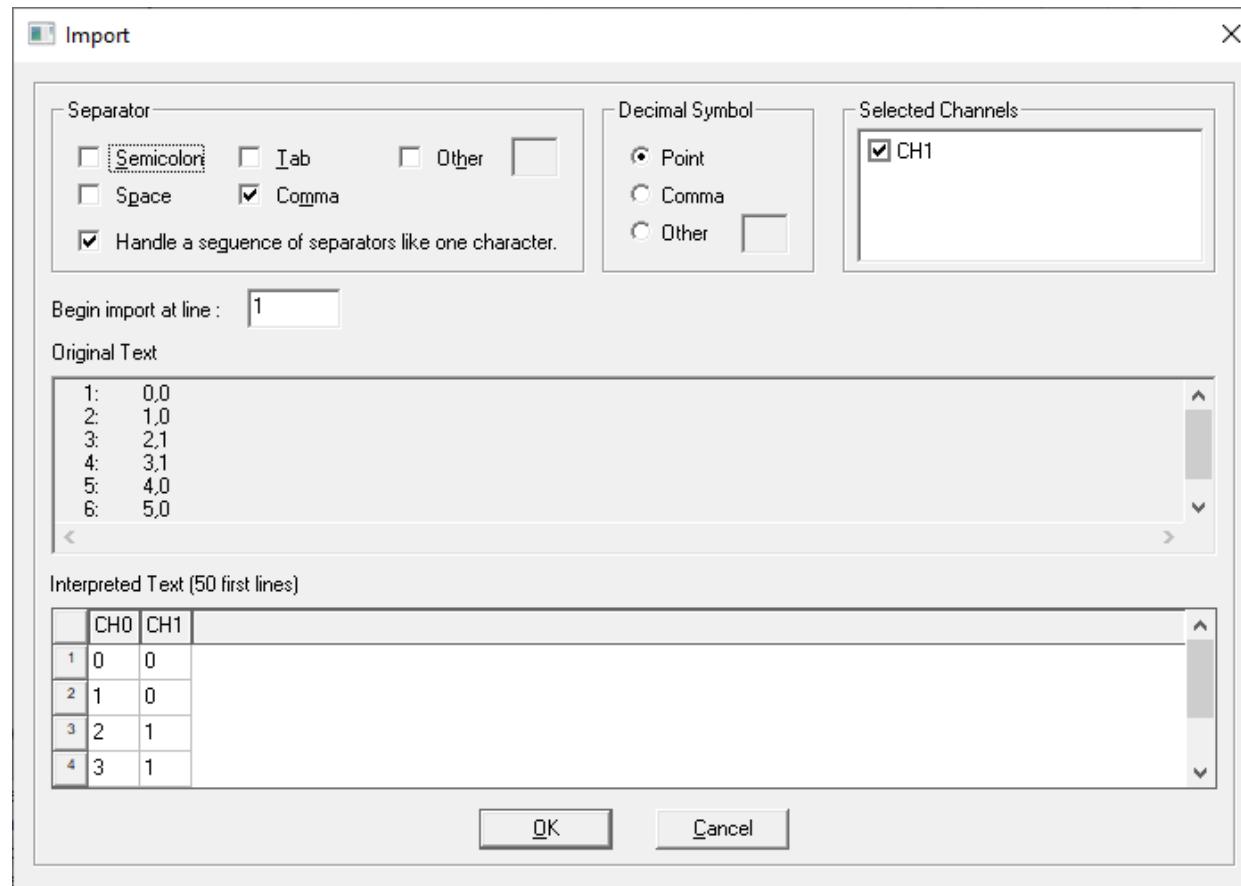
VB Example

```
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"
```

Note About File Types:

Tab-delimited or space-delimited files with the extension *.tab are the recommended file type. When using ImportDataset at the Design level, *.tab is the only file type supported.

At the Project level, other file types are supported (for example, *.csv). However, after calling the command, you must configure the file import format manually through the Electronics Desktop GUI by selecting **Project > Datasets** and clicking **Import**.



InsertDesign

Inserts a new design in the project. For Icepak scripts, the last argument will always be empty.

UI Access	Project > [Insert Icepak Design].		
Parameters	Name	Type	Description
	< <i>DesignType</i> >	String	Design type."Icepak".
	< <i>DesignName</i> >	String	Design name.
	< <i>SolutionType</i> >< <i>ProblemType</i> >	String	Can be "SteadyState TemperatureAndFlow", "SteadyState TemperatureOnly", "SteadyState FlowOnly", "Transient TemperatureAndFlow", "Transient TemperatureOnly", or "Transient FlowOnly".
	""	None	Empty argument.
Return Value	None.		

Python Syntax	InsertDesign (< <i>DesignType</i> >, < <i>DesignName</i> >, < <i>SolutionType</i> >,")
Python Example	<pre>oProject.InsertDesign('Icepak', 'MyDesign', 'TemperatureAndFlow', '')</pre>

VB Syntax	InsertDesign < <i>DesignType</i> >, < <i>DesignName</i> >, < <i>SolutionType</i> >,""
VB Example	<pre>oProject.InsertDesign "Icepak", "MyDesign", "TemperatureAndFlow", ""</pre>

InsertDesignWithWorkflow

Inserts a design with a named workflow and returns an IDispatch string.

UI Access	N/A		
Parameters	Name	Type	Description
	< <i>type</i> >	String	Type of design.

	<code><workflowName></code>	String	Name of the workflow.
	<code><specName></code>	String	Name of the spec.
	<code><fileName></code>	String	Name of the file.
	<code><libLoc></code>	String	Type of library, such as SysLib.
	<code><stationaryPath></code>	String	Path.
Return Value	IDispatch string, such as 'IDispatch(IAltraSimScript)'		

Python Syntax	<code>InsertDesignWithWorkflow(<type>, <workflowName>, <specName>, <fileName>, <libLoc>, <stationaryPath>)</code>
Python Example	<pre>oProject.InsertDesignWithWorkflow ("Circuit Design", "Serial Design", "PCIe3 Stressed", "LongChannel", "SysLib", "C:\Program Files\AnsysEM\v232\Win64\syslib\MS - RT_duroid 6010 (Er=10.2) 0.010 inch, 0.5 oz copper.asty")</pre>

VB Syntax	<code>InsertDesignWithWorkflow <type>, <workflowName>, <specName>, <fileName>, <libLoc>, <stationaryPath></code>
VB Example	<pre>oProject.InsertDesignWithWorkflow "Circuit Design", "Serial Design", "PCIe3 Stressed", "LongChannel", "SysLib", "C:\Program Files\AnsysEM\v232\Win64\syslib\MS - RT_duroid 6010" & " (Er=10.2) 0.010 inch, 0.5 oz copper.asty"</pre>

InsertToolObject

Note:

This command is for internal Ansys use only.

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

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

RemoveAllUnusedDefinitions

Removes all unused project definitions.

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

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

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

RemoveMaterial

Removes a material from a library.

UI Access	Remove Material(s) command in the material editor		
Parameters	Name	Type	Description
	<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	RemoveMaterial (<MaterialName>, <IsProjectMaterial>, <LibraryName>, <LibraryLocation>)
Python Example	<code>oDefinitionManager.RemoveMaterial ([</code>

	"Material1", false, "mo0907", "UserLib"])
--	---

VB Syntax	RemoveMaterial <MaterialName>, <IsProjectMaterial>, <LibraryName>, <LibraryLocation>
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	<table border="1"><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><Definitions></td><td>Array</td><td>Definitions to be removed, such as materials and surface materials.</td></tr></table>	Name	Type	Description	<Definitions>	Array	Definitions to be removed, such as materials and surface materials.
Name	Type	Description					
<Definitions>	Array	Definitions to be removed, such as materials and surface materials.					
Return Value	None.						

Python Syntax	RemoveUnusedDefinitions(<Definitions>)
Python Example	<pre>oProject.RemoveUnusedDefinitions() [["NAME:Materials",</pre>

```

    "Al-Extruded"
],
[
    "NAME:SurfaceMaterials",
    "Steel-oxidised-surface"
]
)

```

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	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><NewName></td> <td>String</td> <td>Desired name of the project. The path is optional.</td> </tr> <tr> <td><OverWriteOk></td> <td>Boolean</td> <td> <ul style="list-style-type: none"> True - overwrite the file on disk if it exists. False - prevent overwrite. </td> </tr> </tbody> </table>	Name	Type	Description	<NewName>	String	Desired name of the project. The path is optional.	<OverWriteOk>	Boolean	<ul style="list-style-type: none"> True - overwrite the file on disk if it exists. False - prevent overwrite. 	
Name	Type	Description									
<NewName>	String	Desired name of the project. The path is optional.									
<OverWriteOk>	Boolean	<ul style="list-style-type: none"> True - overwrite the file on disk if it exists. False - prevent overwrite. 									
Return Value	None.										

Python Syntax	Rename(<NewName>,<OverWriteOK>)
Python Example	<code>oProject.Rename ("c:\projects\MyProject.aedt", True)</code>

VB Syntax	Rename <NewName>,<OverWriteOK>
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	Save()
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
	<code><NewName></code>	String	The desired name of the project, with directory and extension.
	<code><OverWriteOK></code>	Boolean	True to overwrite the file of the same name, if it exists. False to prevent over-write.
	<code><DefaultAction></code>	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.
	<code><OverwriteActions></code>	Array	For Schematic/Layout projects only. Otherwise omit. See note below. Structured array: Array("Name: <Action>", <FileName>, <FileName>, ...) Valid actions: ef_overwrite , ef_copy_no_overwrite, ef_make_path_absolute, or empty string.
Return Value	None.		

Python Syntax	For non-Schematic/Layout project: <code>SaveAs (<NewName> <OverWriteOK>)</code> For Schematic/Layout project: <code>SaveAs (<NewName> <OverWriteOK> <DefaultAction> <OverrideActions>)</code>
----------------------	--

Python Example	<pre> oProject.SaveAs('D:/projects/project1.aedt', True) ----- oProject.SaveAs('D:/Projects/Project1.aedt', True, 'ef_overwrite', ['NAME:OverrideActions', ['NAME:ef_copy_no_overwrite'], ['NAME:Files', '\$PROJECTDIR/circuit_models.inc']], ['NAME:ef_make_path_absolute', ['NAME:Files', '\$PROJECTDIR/SL_6s.sp']]) </pre>
-----------------------	--

VB Syntax	<p>For non-Schematic/Layout project: SaveAs <NewName> <OverWriteOK></p> <p>For Schematic/Layout project: SaveAs <NewName> <OverWriteOK> <DefaultAction> <OverrideActions></p>
VB Examples	<pre> oProject.SaveAs "D:/projects/project1.aedt", true ----- oProject.SaveAs "F:\Designer Projects\TA33097\HighSpeedChannel.aedt", true, "ef_overwrite", Array ("NAME:OverrideActions", Array("NAME:ef_copy_no_overwrite", Array("NAME:Files", "\$PROJECTDIR/circuit_mod- eles.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 <projectName>	Type String	Description 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.
Return Value	<ArchiveNotes>	String	String describe the archive.
	None.		

Python Syntax	SaveProjectArchive(<archivefilepath>, <IncludeExternalFiles>, <IncludeResultsFiles>, <AdditionalFiles>, <ArchiveNotes>)
Python	<code>oProject.SaveProjectArchive("C:\\\\Users\\\\Documents\\\\Ansoft\\\\Project27.aedtz", True,</code>

Example	<code>False, [], "")</code>
----------------	-----------------------------

VB Syntax	<code>SaveProjectArchive <archivefilepath>, <IncludeExternalFiles>, <IncludeResultsFiles>, <AdditionalFiles>, <ArchiveNotes></code>
VB Example	<code>oProject.SaveProjectArchive "C:\Documents\OptimTee.aedtz", 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".
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",_ "Simpler Elements\Basic Elements\Circuit\Passive Elements:R"</pre>

SetActiveDesign

Sets a design to be the active design.

UI Access	N/A						
Parameters	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td><DesignName></td><td>String</td><td>Name of the design to set as the active design.</td></tr></tbody></table>	Name	Type	Description	<DesignName>	String	Name of the design to set as the active design.
Name	Type	Description					
<DesignName>	String	Name of the design to set as the active design.					
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	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><code><propPath></code></td> <td>String</td> <td>A child object's property path. See: Property Function.</td> </tr> <tr> <td><code><newValue></code></td> <td>String</td> <td>New property value.</td> </tr> </tbody> </table>	Name	Type	Description	<code><propPath></code>	String	A child object's property path. See: Property Function .	<code><newValue></code>	String	New property value.
Name	Type	Description								
<code><propPath></code>	String	A child object's property path. See: Property Function .								
<code><newValue></code>	String	New property value.								
Return Value	<p>Boolean:</p> <ul style="list-style-type: none"> • True – property found. • False – property not found. 									

Python Syntax	<code>SetPropValue(<propPath>, <newValue>)</code>
Python Example	<pre>oProject.SetPropValue("TeeModel/offset", "2mm") oProject.SetPropValue("TeeModel/Results/S Parameter Plot 1/Display Type", "Data Table")</pre>

VB Syntax	<code>SetPropValue <propPath>, <newValue></code>
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	<p>One of the following, where tab titles are shown in parentheses:</p> <ul style="list-style-type: none"> • 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 ComplInst@R;2;3
	<propName>	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>

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	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><code><VarName></code></td> <td>String</td> <td>Variable name.</td> </tr> <tr> <td><code><VarValue></code></td> <td>Value</td> <td>New value for the variable.</td> </tr> </tbody> </table>			Name	Type	Description	<code><VarName></code>	String	Variable name.	<code><VarValue></code>	Value	New value for the variable.
Name	Type	Description										
<code><VarName></code>	String	Variable name.										
<code><VarValue></code>	Value	New value for the variable.										
Return Value	None.											

Python Syntax	<code>SetVariableValue (<VarName>, <VarValue>)</code>
----------------------	---

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

VB Syntax	<code>SetVariableValue <VarName>, <VarValue></code>
------------------	---

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>

6 - Object Oriented Property Scripting

Scripting in AEDT has been markedly enhanced via the convenient use of Object-Oriented access to retrieve or modify properties of objects in AEDT. This feature allows for much less code to be written to access object properties and enables much more readable code for our users, avoiding complex array input.

The primary gains of the use of Object-Oriented scripting are the ease with which properties of various existing objects in an Ansys Electronics DesktopProject/Design can be read, modified, and set. Along with this ease of implementation comes much more 'readable' code to aid in others' interpretation of custom scripts.

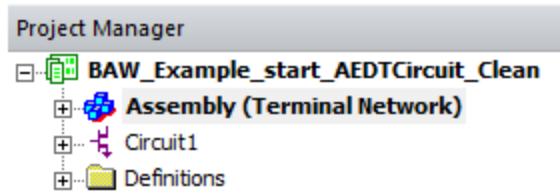
This App Note will discuss the following: The logic for syntax of access, Basic Attributes of Project and Design properties and examples of use.

Object-Oriented Scripting

There are five basic functions that you use in Object Oriented scripting to retrieve and set properties:

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

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



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

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

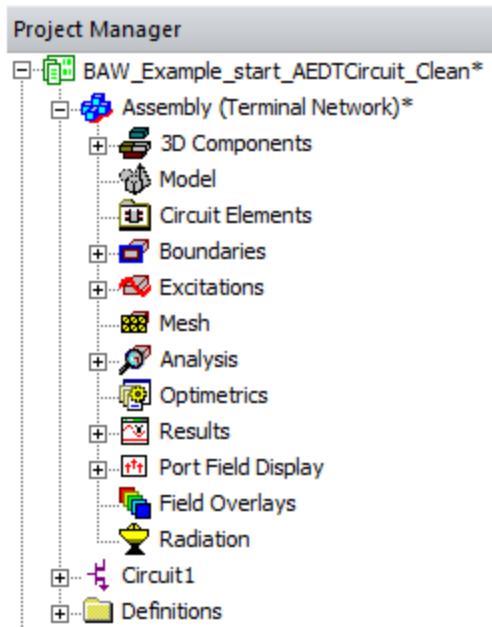
Once the objects have been defined, you can use GetChildNames() to learn what object instances exist for each. As an example, observe the Child Names of oProject, and you see a list of the Designs in the AEDT Project, per the GUI.

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

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

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

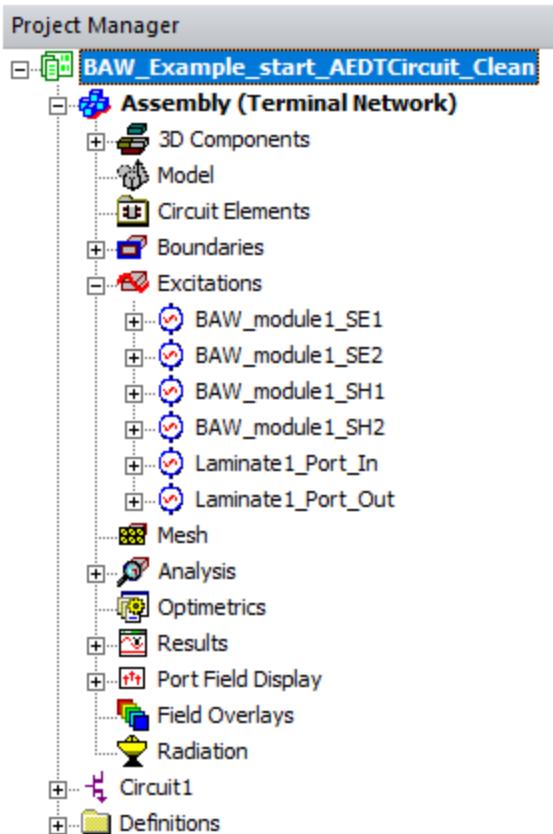
These names are what you would expect based on the Project Manager Layout:



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

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

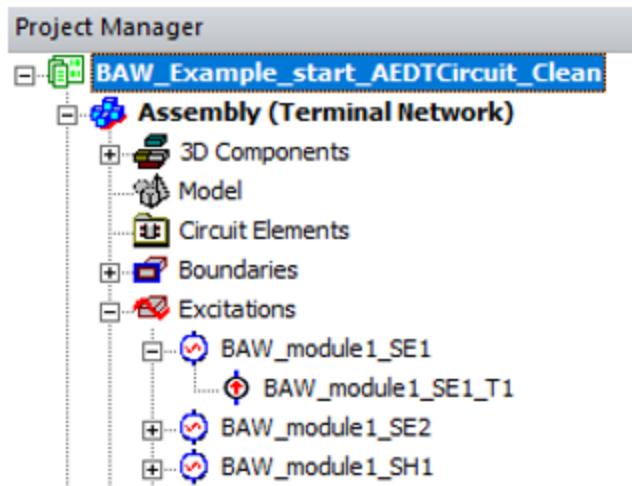
You now have an object, oExcitations defined to be the oDesign Child Object 'Excitations'. What does this mean? If you expand the Excitations dialogue in the Project Manager, you expect that the Child Names of this object would be the names of the Excitations as defined, in this case six ports:



```
>>> oExcitations.GetChildNames()
['Laminate1_Port_In', 'Laminate1_Port_Out', 'BAW_module1_SE1', 'BAW_module1_SE2', 'BAW_module1_SH1',
'BAW module1 SH2']
```

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

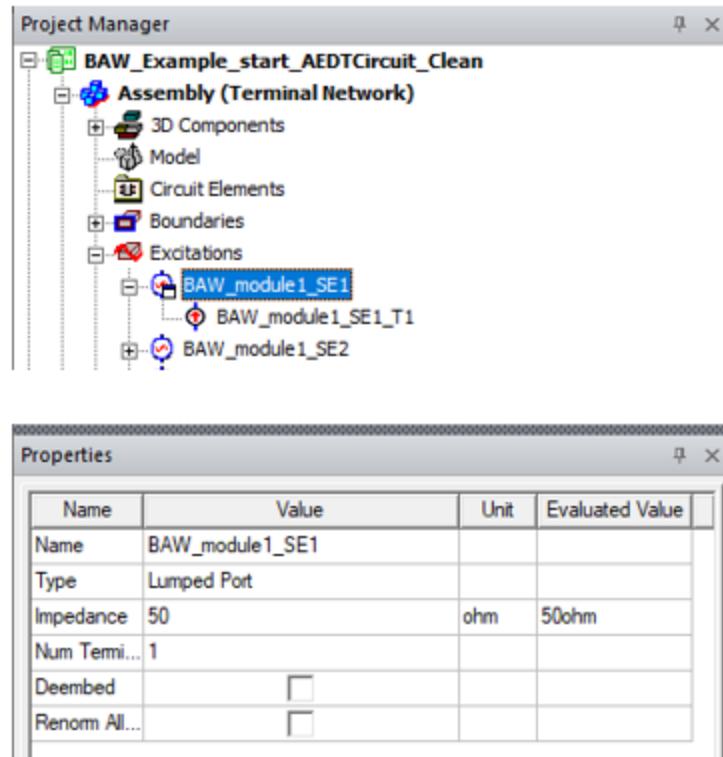
Expand the first port to see its expected Child Object, its Terminal, in the **Project Manager** Window:



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

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

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



Name	Value	Unit	Evaluated Value
Name	BAW_module1_SE1		
Type	Lumped Port		
Impedance	50	ohm	50ohm
Num Termi... Deembed	1		
Renom All...	<input type="checkbox"/>		

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

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

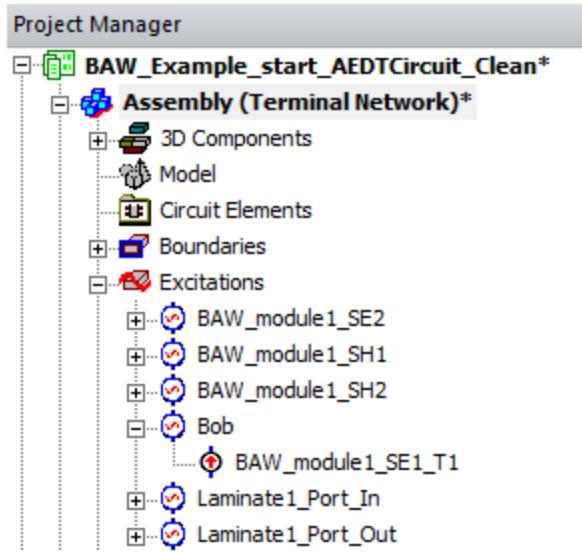
Once you identify the desired object and you know the desired property, you can access the value via GetPropValue(). For example, if you want to retrieve the name of the object oPort:

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

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

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

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



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

Material Properties and Examples

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

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

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

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

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

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

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

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

Body Properties and Modification

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

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

Retrieving Variables

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

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

Retrieve Datasets and Values

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

```
>>>oDesktop.GetChildTypes()
['Projects']

>>>oDesktop.GetChildNames()
['OptimTee']

>>>arrProjectNames = oDesktop.GetChildNames()

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

>>>tp.GetChildTypes()
```

```
['Design', 'Project Data']

>>>tp.GetChildNames('Project Data')

['Variables', 'Materials', 'Surface Materials', 'Datasets']

>>>ds = tp.GetChildObject('datasets')

>>>ds.GetChildNames()

 ['$ds1']

>>>ds1=ds.GetChildObject('$ds1')

>>>ds1.GetPropValue('[:, :, :]')

[[1.0, 4.0], [2.0, 5.0], [3.0, 6.0]]

>>>ds1.GetPropSIValue()

[[1.0, 4.0], [2.0, 5.0], [3.0, 6.0]]

>>>ds1.DimUnits

>>>ds1.DimUnits = ['mm', 'mm']

>>>ds1.DimUnits

['mm', 'mm']

>>>ds1.GetPropSIValue()

[[0.001, 0.004000000000000001], [0.002, 0.005000000000000001], [0.003000000000000001,
0.006000000000000001]]

>>>ds1.GetPropValue('[:, :, :]')

[[1.0, 4.0], [2.0, 5.0], [3.0, 6.0]]
```

GetSolutionData API

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

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

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

Summary

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

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

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

Each exposed script object supports the following COM functions:

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

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

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

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

For example, a Mechanical design object has these children.

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

Four of the children are of “Module” type

```
>>> design.GetChildNames("module")
['Boundaries', 'Excitations', 'Optimetrics', 'Results']
```

- **GetChildObject**

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

For example,

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

- **GetPropNames**

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

For example,

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

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

For example,

```
>>> geom.GetPropertyValue("material")
'"vacuum"'
>>> geom.GetPropertyValue("xsize")
'3mm'
>>> op.GetPropertyValue("attach to original object")
False
```

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

For example,

```
>>> geom.SetPropertyValue("model", False)
>>> boxcmd.SetPropertyValue("ysize", "4mm")
```

- GetPropEvaluatedValue (<PropName>)

For example,

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

- GetPropSIValue (<PropName>)

For example,

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

Additional Details Specific to AEDT Solvers

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

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

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

HFSS

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

HFSS 3D Layout

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

Maxwell 3D/2D

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

RMxprt

- Analysis, Field Overlays, Optimetrics, Results

Q3D

- Boundaries, Nets, Analysis, Optimetrics,

Q2D

- Boundaries, Conductors, Analysis, Field Overlays, Optimetrics,

Icepak

- Thermal, Monitor, Mesh, Analysis, Field Overlays, Optimetrics, Results

Mechanical

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

Circuit

- Optimetrics, Results

Circuit Netlist

- Results

EMIT

- Coupling

Simplorer/Twin Builder

- Analysis, Optimetrics, Results

Additional details on Boundaries/Excitations

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

These module script objects do not have properties

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

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

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

GetChildNames of these module object returns its immediate children

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

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

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

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

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

You can query/edit these properties

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

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

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

Additional Details Specific to Icepak Monitor and Mesh Modules

GetChildTypes of the Monitor module returns the two types of its children, which are Face and Point. These refer to the two types of monitors (i.e., face and point monitors that can be created by the user).

```
>>> design.GetChildObject("monitor").GetChildTypes()
['Face', 'Point']
```

GetChildTypes of the Mesh module returns the two types of its children, which are Operation and Region. These refer to the Mesh operations specified and the Mesh Regions created in the design.

```
>>> design.GetChildObject("mesh").GetChildTypes()  
['Operation', 'Region']
```

3D component encapsulation

These script interfaces are compliant with encapsulation. For example,

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

Additional details on Solve setup

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

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

This module script object has no property

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

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

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

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

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

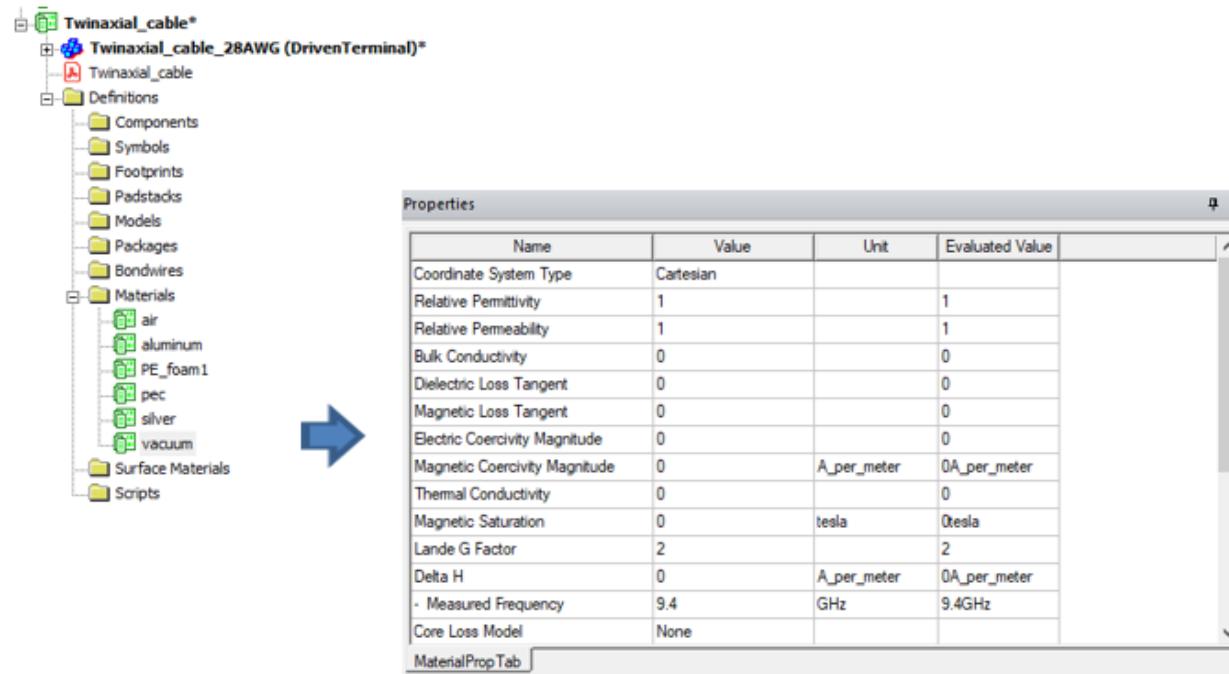
```
>>> setup.GetChildNames()  
['Sweep', 'Sweep1', 'Sweep2']  
  
>>> setup.GetChildTypes()  
['Discrete', 'Interpolating']  
  
>>> sweep1 = setup.GetChildObject("sweep")
```

Related Topics

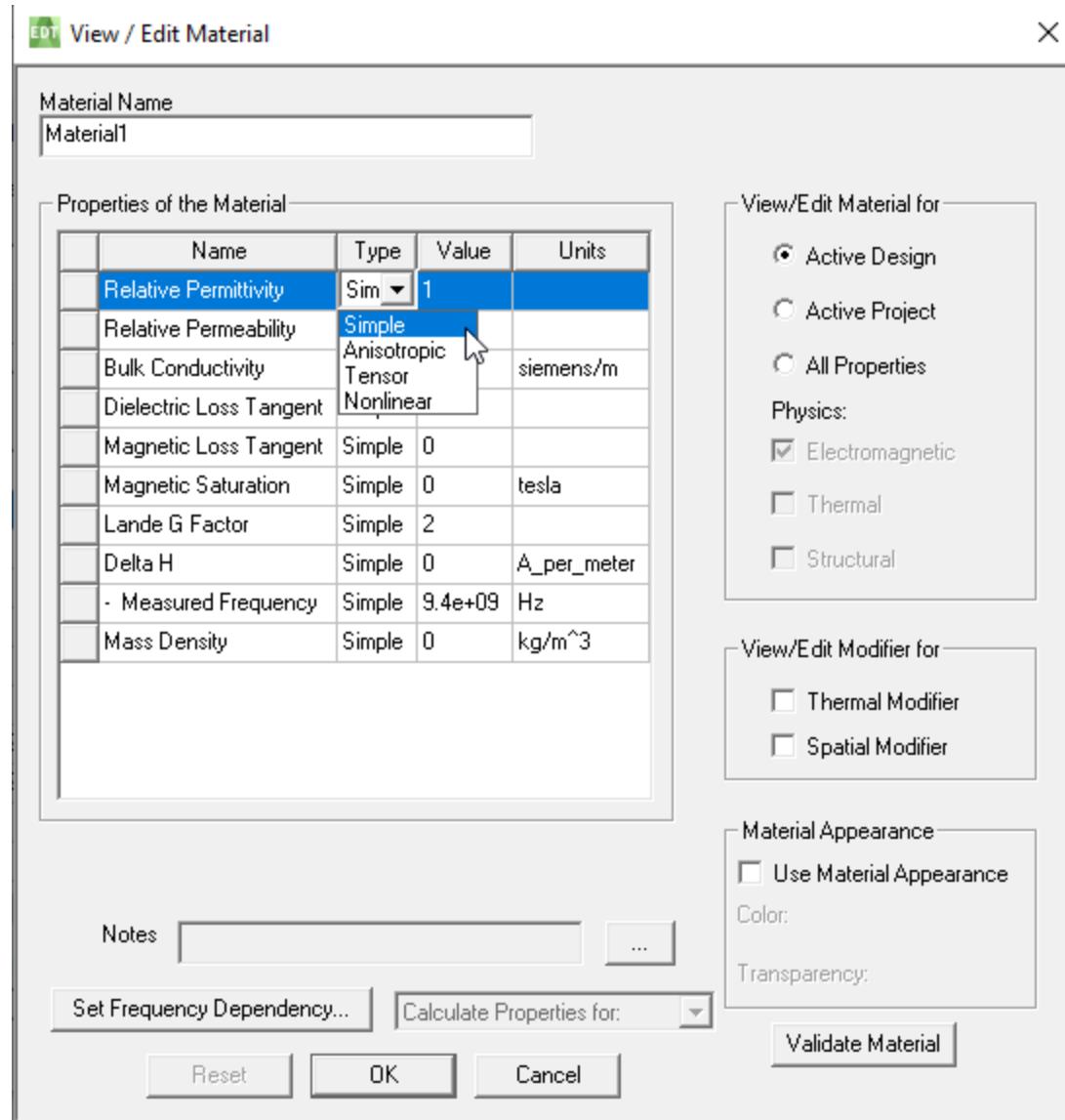
[ExampleGetLayeredImpedanceBoundaryPropertyNamesandValues.htm](#)

Materials Scripting Support

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



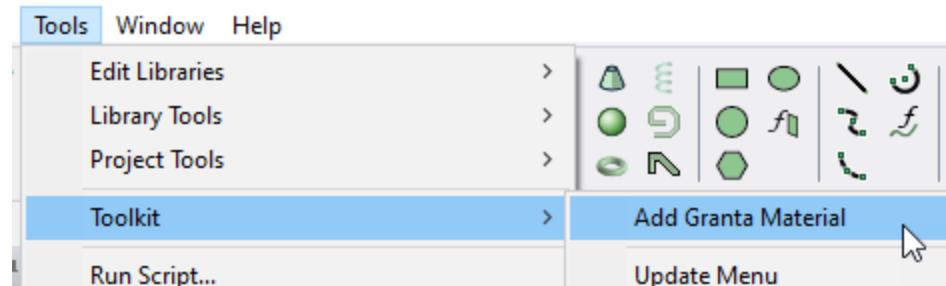
Some Material Properties like Relatively Permittivity may have values assigned as BH Curves or Tensors, as discussed in the Assigning Materials chapter of the online help.

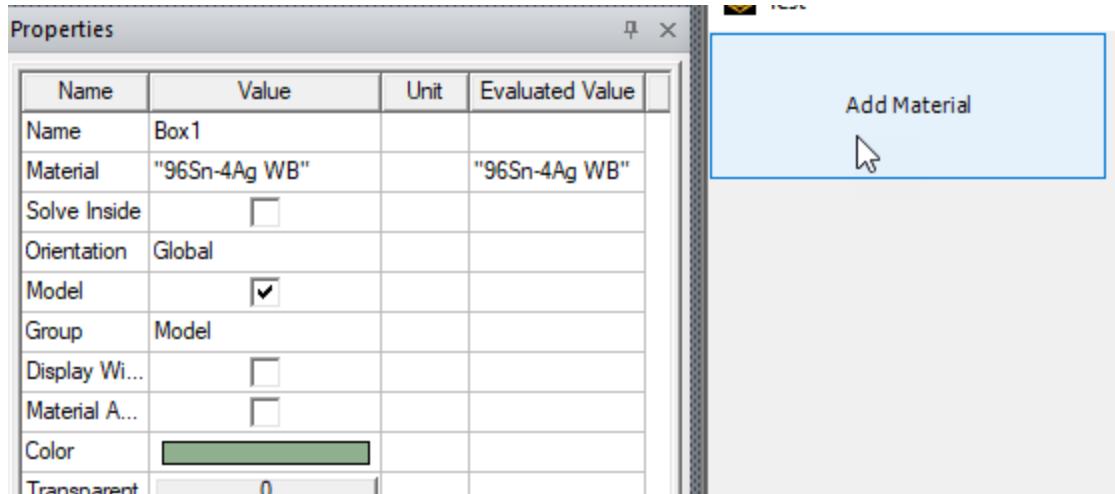


With this feature enabled you can then:

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

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





What is not supported:

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

Object Oriented Scripting for Materials

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

```
>>> omats = oDesktop.GetActiveProject().GetChildObject("Materials")
>>> omat = omats.GetChildObject("vacuum")
>>> omat.GetPropNames()
['Coordinate System Type', 'Coordinate System Type/Choices', 'Relative Permeability Type',
'Relative Permeability Type/Choices', 'Relative Permeability', 'Relative Permeability/SIValue',
```

```
'Relative Permeability/EvaluatedValue', 'Bulk Conductivity Type', 'Bulk Conductivity Type/Choices', 'Bulk Conductivity', 'Bulk Conductivity/SIValue', 'Bulk Conductivity/EvaluatedValue', 'Magnetic Coercivity Type', 'Magnetic Coercivity Magnitude', 'Magnetic Coercivity Magnitude/SIValue', 'Magnetic Coercivity Magnitude/EvaluatedValue', 'Composition', 'Composition/Choices', "Young's Modulus Type", "Young's Modulus Type/Choices", "Young's Modulus", "Young's Modulus/SIValue", "Young's Modulus/EvaluatedValue", "Poisson's Ratio Type", "Poisson's Ratio", "Poisson's Ratio Type/Choices", "Poisson's Ratio", "Poisson's Ratio/SIValue", "Poisson's Ratio/EvaluatedValue"]
```

Examples showing change to material property type:

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

Examples showing change to a vector component value

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

```
True
```

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

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

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

```
True
```

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

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

Change choice property value

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

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

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

```
True
```

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

```
['Coordinate System Type', 'Coordinate System Type/Choices', 'Relative Permeability Type',
'Relative Permeability Type/Choices', 'Relative Permeability', 'Relative Permeability/SIValue',
'Relative Permeability/EvaluatedValue', 'Bulk Conductivity Type', 'Bulk Conductivity
Type/Choices', 'Bulk Conductivity', 'Bulk Conductivity/SIValue', 'Bulk Con-
ductivity/EvaluatedValue', 'Magnetic Coercivity Type', 'Magnetic Coercivity Magnitude', 'Magnetic
Coercivity Magnitude/SIValue', 'Magnetic Coercivity Magnitude/EvaluatedValue', 'Magnetic Coer-
civity Components', 'Magnetic Coercivity Components/Component1', 'Magnetic Coercivity Com-
ponents/Component2', 'Magnetic Coercivity Components/Component3', 'Composition',
'Composition/Choices', '- Stacking Factor Type', '- Stacking Factor', '- Stacking Fact-
or/SIValue', '- Stacking Factor/EvaluatedValue', '- Stacking Direction', '- Stacking Dir-
ection/Choices', "Young's Modulus Type", "Young's Modulus Type/Choices", "Young's Modulus",
"Young's Modulus/SIValue", "Young's Modulus/EvaluatedValue", "Poisson's Ratio Type", "Poisson's
Ratio Type/Choices", "Poisson's Ratio", "Poisson's Ratio/SIValue", "Poisson's Ratio/E-
valuatedValue"]
```

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

```
True
```

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

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

Change choice property value

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

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

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

```
True
```

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

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

>>>

This page intentionally
left blank.

7 - Property Script Commands

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

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

GetName ()

- Returns the name of the object.

GetChildTypes ()

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

GetChildNames (<type>)

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

GetChildObject (<objPath>)

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

GetPropNames (<bIncludeReadOnly>)

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

GetPropValue (<propertyPath>)

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

```
SetPropValue(<propertyPath>, <data>)
```

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

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

Note:

Older property commands should be executed by the oProject object.

```
Set oProject = oDesktop.SetActiveProject ("Project1")  
oProject.CommandName <args>
```

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

[Conventions Used in this Chapter](#)

[GetArrayVariables](#)

[GetProperties](#)

[GetPropertyValues](#)

[GetVariables](#)[GetVariableValue](#)[SetPropertyValue](#)[SetVariableValue](#)[Additional Property Scripting Example](#)[Example Use of Record Script and Edit Properties](#)

Object Script Property Function Summary

Object Path

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

- An Object path consisted of one or multiple Object-ID-Nodes separated by "/" .
- Object-ID-Node; may exist in the following forms:
 - A simple object name or property name.
 - Type[Name] for object; Tab[name] for property.
 - Name[attr1="v1", attr2 = "v2", ...]. When more than one child object have the same name, use attributes to specify the difference.
 - ArrayName[index]. For example, in an Optimetric setup with multiple calculations, "Calculation[0]" could be used to identify the first calculation.
 - Name beginning with '@' character denoted as a property name, when an object has a child and property with the same name.

Property Object

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

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

- GetPropValue(<propertyPath>)
 - <propertyPath> – the property's full path. A property name or child object's path appended with a property name, like “TeeModel/Offset/SIValue”
 - Returns the property value if the property is found; otherwise causes script error.
- SetPropValue(<propertyPath>, <data>)
 - <propertyPath> – the property's full path. A property name or child object's path appended with a property name, like “TeeModel/Offset/Value”
 - <data> – new data, type is dependent on property type.
 - Returns True if property data is updated successfully; False if the new data is invalid.

Project Object

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

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

```
oDesign = oProject("TeeModel")  
  
oVariable = oProject.GetChildObject("VariableName")  
  
oReport = oProject.GetChildObject("TeeModel/Results/S Parameter Plot 1")
```
- GetPropNames(bnIncludeReadOnly) always return empty array since the project has no property.

- GetPropValue(propertyPath)

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

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

Design Object

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

- GetChildTypes() always return ['Module', 'Editor', 'Variable'].
GetChildNames(type) will return modules & editor child names.
GetChildNames("Variable") will return all variable names
GetChildNames("Module") will return all module names that support property-object-script like ['Optimetrics', 'RadField', 'Results']
GetChildNames("Editor") will return a 3D editor name for all 3D Designs
- GetChildObject()

```
oVariable = oDesign.GetChildObject("VariableName")  
oReport = oDesign.GetChildObject("Results/S Parameter Plot 1")  
oRptModule = oDesign.GetChildObject("ReportSetup")
```
- GetPropNames(bIncludeReadOnly) always return empty array since the design has no property.

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

3D Modeler Object

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

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

Variable Object

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

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

- GetChildTypes() always return empty array.
- GetChildNames() always return empty array , since variable has no child.
- GetChildObject(objPath) it has no child.
- GetPropNames(bIncludeReadOnly) ['EvaluatedValue', 'SIValue'] are read-only properties
 - Independent variable :["Value", 'EvaluatedValue', 'SIValue', 'Description', 'ReadOnly', 'Hidden', 'Sweep', 'Optimization/Included', 'Optimization/Min', 'Optimization/Max', 'Sensitivity/Included', 'Sensitivity/Min', 'Sensitivity/Max', 'Sensitivity/IDisp', 'Statistical', 'Statistical/Included', 'Tuning/Included', 'Tuning/Step', 'Tuning/Min', 'Tuning/Max'].
 - Dependent variable ["Value", 'EvaluatedValue', 'SIValue', 'Description', 'ReadOnly', 'Hidden', 'Sweep']
- GetPropValue(propName)
oVar.GetPropValue() return the variable value as text string.
oVar.GetPropValue("Value") return the variable value as text string.
oVar.GetPropValue("SIValue") return the SI-value of variable as number.
oVar.Get_SIValue() also return the SI value.
- SetPropValue(propName, newValue)
oVar.SetPropValue("Value", 888)
oVar.SetPropValue("Sensitivity/Included", True)
oVar.SetPropValue("Sensitivity/Max", '1.8pF'])
oVar.Set_Sensitivity_Max('1.8pF') also works as last call.
oVar.SetPropValue("Sensitivity", ['Min:=' , '0.8pF', 'Max:=' , '1.8pF'])
//set multiple attributes at one call:
oVar.SetPropValue("@", ["Value:='288, 'Sensitivity', ['Included', True,'Min', '0.0']]])
oVar.SetPropValue("", ["Value:='288, 'Sensitivity', ['Included', True,'Min', '0.0']]])

Optimetrics Module Object:

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

- GetChildTypes() there are six type of children, they are ['OptiParametric', 'OptiOptimization', 'OptiSensitivity', 'OptiStatistical', 'OptiDesignExplorer', 'OptiDXDOE']. But the return array only included those that have setup defined, so it may be an empty array if no optimetrics setup is defined. The GetChildNames(type) function also recognized the type name without the prefix "Opti".
- GetChildNames(type)
GetChildNames() will return all setup for all types.
GetChildNames("OptiOptimization") & GetChildNames("Optimization") will return all Optimization setup.
- GetChildObject()
oParamSetup = oOptModule.GetChildObject('ParametricSetup1') get the
oOptSetup = oOptModule.GetChildObject('OptimizationSetup1')
- GetPropNames(bIncludeReadOnly) always return empty array since the it has no property.
- GetPropValue(propPath) may be used to get its child's property value
oOptModule.GetPropValue("OptimizationSetup1\Optimizer") get the optimizer name for OptimizationSetup1
- SetPropValue(propPath, newValue) may be used to set its child's property value
oOptModule.SetPropValue(ParametricSetup1\Enabled", False) //disable ParametricSetup1

Optimetrics Setup Object

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

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

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

ReportSetup(Results) Module Object:

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

- GetChildTypes() always empty array.
- GetChildNames(type)
 - GetChildNames() return all report names
- •GetChildObject(objPath)
 - oRpt = oRptModule.GetChildObject("S Parameter Plot 1") return the report property object

```
oTrace = oRptModule.GetChildObject("S Parameter Plot 1/dB(S(Port1,Port1))") return the trace property object  
oAxisX = oRptModule.GetChildObject("S Parameter Plot 1/AxisX") return the axis X property object
```

- GetPropNames(bIncludeReadOnly) always return empty array since the it \has no property.
- GetPropValue()
oRptModule.GetPropValue("S Parameter Plot 1/Display Type")
- SetPropValue()
oRptModule.SetPropValue("S Parameter Plot 1/Display Type", "DataTable")

ReportSetup(Results) Module Child Objects:

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

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

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

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

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

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

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

- SetPropValue(propName, newValue)
oTrace.SetPropValue("Primary sweep", "Freq")

Radiation Module Object:

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

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

Radiation Module Child Objects:

These are Property Objects. It also provides quick function call to get/set its properties by adding functions with property name appended to Get_ & Set_ prefix. To find what functions it provides enter dir(oVar) the command window.

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

```
oRadSetup = oRadModule.GetChildObject(setupName)  
oRadSetup = oDesign.GetChildObject(RadField/setupName)
```

- GetChildTypes() always return empty array.
- GetChildNames() always return empty array , since Radiation setup has no child.
- GetChildObject(objPath) it has no child.
- GetPropNames(bIncludeReadOnly) will return the property names listed in the property window when the setup is selected.
- GetPropValue(propName)
 - oRadSetup.GetPropValue("Num Points") return the line setup's "Num Points" property value.
 - oRadSetup.Get_NumPoints() will also get the same value.
- SetPropValue(propName, newValue)
 - oRadSetup.SetPropValue('Num Points', 888)
 - oRadSetup.Set_NumPoints(888)

Conventions Used in this Chapter

General Definitions:

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": Set oModule = oDesign.GetModule("ReportSetup") oModule.ChangeProperty Array("NAME:AllTabs", _ Array("NAME:Header", _ Array("NAME:PropServers", _ "XY Plot1:Header"), Array("NAME:ChangedProps", _ Array("NAME:Company Name", "Value:=", "My Company"))))

Note:

For scripted property changes in the various modules and editors, refer to the chapters on the System, HFSS 3D Layout, and Nexxim tools, as well as the Layout and Schematic editors.

GetArrayVariables

Returns a list of array variables. To get a list of indexed project variables, execute with oProject. To get a list of indexed local variables, use oDesign.

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

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

VB Syntax	GetArrayVariables
VB Example	<pre>oProject.GetArrayVariables oDesign.GetArrayVariables</pre>

GetProperties

Gets a list of all the properties belonging to a specific <PropServer> and <PropTab>. This can be executed by the oProject, oDesign, or oEditor variables.

UI Access	N/A		
Parameters	Name	Type	Description
	<PropTab>	String	<p>One of the following, where tab titles are shown in parentheses:</p> <ul style="list-style-type: none"> • 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 ComplInst@R;2;3
Return Value	Array of strings containing the names of the appropriate properties.		

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

VB Syntax	GetProperties <PropTab>, <PropServer>
VB Example	<code>oEditor.GetProperties "PassedParameterTab", "k"</code>

GetProperty Value

Returns the value of a single property belonging to a specific <PropServer> and <PropTab>. This function is available with the Project, Design or Editor objects, including definition editors.

Tip:

Use the script recording feature and edit a property, and then view the resulting script to see the format for that property.

UI Access	N/A		
Parameters	Name <PropTab>	Type String	Description One of the following, where tab titles are shown in parentheses: <ul style="list-style-type: none">• PassedParameterTab ("Parameter Values")• DefinitionParameterTab (Parameter Defaults")• LocalVariableTab ("Variables" or "Local Variables")• ProjectVariableTab ("Project variables")• ConstantsTab ("Constants")• BaseElementTab ("Symbol" or "Footprint")

		<ul style="list-style-type: none"> • 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.GetPropertyValues("PassedParameterTab", k, "R") ... </pre>

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

GetVariables

Returns a list of all defined variables. To get a list of project variables, execute this command using `oProject`. To get a list of local variables, use `oDesign`.

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

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

VB Syntax	GetVariableValue <i><VarName></i>
VB Example	<code>oProject.GetVariableValue "var_name"</code>

SetPropertyValue

Sets the value of a single property belonging to a specific PropServer and PropTab. This function is available with the Project, Design or Editor objects, including definition editors. This is not supported for properties of the following types: ButtonProp, PointProp, V3DPointProp, and VPointProp. Only the ChangeProperty command can be used to modify these properties.

Use the script recording feature and edit a property, and then view the resulting script entry or use GetPropertyValue for the desired property to see the expected format.

UI Access	N/A		
Parameters	Name <i><propTab></i>	Type String	Description One of the following, where tab titles are shown in parentheses: <ul style="list-style-type: none">• PassedParameterTab ("Parameter Values")

			<ul style="list-style-type: none"> • 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 ComplInst@R;2;3
	<i><propName></i>	String	Name of the property.
	<i><propValue></i>	String	The value for the property
Return Value	None.		

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

VB Syntax	SetPropertyValue <propTab>, <propServer>, <propName>, <propValue>
VB Example	<code>oEditor SetPropertyValue "PassedParameterTab", "k", "R", "2200"</code>

SetVariableValue

Sets the value of a variable. To set the value of a project variable, execute this command using `oProject`. To set the value of a local variable, use `oDesign`.

UI Access	N/A									
Parameters	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><VarName></td> <td>String</td> <td>Variable name.</td> </tr> <tr> <td><VarValue></td> <td>Value</td> <td>New value for the variable.</td> </tr> </tbody> </table>	Name	Type	Description	<VarName>	String	Variable name.	<VarValue>	Value	New value for the variable.
Name	Type	Description								
<VarName>	String	Variable name.								
<VarValue>	Value	New value for the variable.								
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("HFSSModell1")  
Set oEditor = oDesign.SetActiveEditor("3D Modeler")  
  
oEditor.ChangeProperty Array("NAME:AllTabs", Array("NAME:Geometry3DAttributeTab", Array  
("NAME:PropServers", _  
"Polyline1"), Array("NAME:ChangedProps", Array("NAME:Display Wireframe", "Value:=", true), _  
Array("NAME:Display Wireframe", "Value:=", false), Array("NAME:Transparent", "Value:=", 0.2))))
```

Additional Property Scripting Examples

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

```
Dim all_props
Dim prop
all_props = oEditor.GetProperties("Geometry3DCmdTab",_
"Box1>CreateBox:1")
For Each prop In all_props
    val = oEditor.GetProperty("Geometry3DCmdTab",_
    "Box1>CreateBox:1", prop)
    new_val = InputBox("New Value of " + prop + ":" ,_
    "Current Value of '" + prop + "' is " + val, val)
    If new_val <> val Then
        oEditor SetProperty "Geometry3DCmdTab",_
        "Box1>CreateBox:1", prop, new_val
        val = _
        oEditor SetProperty "Geometry3DCmdTab",_
        "Box1>CreateBox:1", prop)
    MsgBox("Now the value of '" + prop + "' is " + val)
End If
Next
```

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

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

'
oDesktop.RestoreWindow
oDesktop.NewProject
Set oProject = oDesktop.GetActiveProject
'

' CREATE A RECTANGLE IN HFSS_3D_LAYOUT
'

oProject.InsertDesign "HFSS 3D Layout", "HFSS 3D Layout1", _
"C:\testinstall\Designer\syslib\PCB - SingleSided.asty", ""
```

```
Set oDesign = oProject.SetActiveDesign("HFSS 3D Layout1")
Set oEditor = oDesign.SetActiveEditor("Layout")
oEditor.CreateRectangle Array("NAME:Contents", _
"rectGeometry:=", Array("Name:=", _
"rect_1", "LayerName:=", "Top", "lw:=", _
"0mm", "Ax:=", "-22mm", "Ay:=", "20mm", "Bx:=", _
"29mm", "By:=", "-4mm", "ang:=", "0deg"))
'

' GET ALL PROPERTIES OF THE RECTANGLE
'

Dim all_props
Dim prop
Dim val
Dim new_val
'

all_props = oEditor.GetProperties("BaseElementTab", "rect_1")
'

' LOOP OVER ALL PROPERTIES
'

For Each prop in all_props
val = oEditor.GetProperty("BaseElementTab", "rect_1", prop)
```

```
'  
  
' DISPLAY VALUE TO THE USER  
  
'  
  
new_val = InputBox("New Value of "+prop+":",_  
"Current Value of "+prop+" is "+val,val)  
  
'  
  
' CHANGE THE VALUE IF DESIRED  
  
'  
  
If new_val <> val Then  
oEditor SetPropertyValue _  
"BaseElementTab","rect_1",prop,new_val  
val = _  
oEditor.GetPropertyValue("BaseElementTab","rect_1",prop)  
MsgBox("Now the value of "+prop+" is "+val)  
End If  
  
'  
  
Next  
'
```

8 - Dataset Script Commands

Dataset commands should be executed by the oProject object:

```
Set oProject = oDesktop.SetActiveProject("Project1")
oProject.CommandName <args>
```

[AddDataSet](#)

[DeleteDataSet](#)

[EditDataSet](#)

[ExportDataSet](#)

[HasDataSet](#)

[ImportDataSet](#)

AddDataset

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

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.
Return Value	None.		

Python Syntax	AddDataset < <i>DatasetdataArray</i> >
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

```
oProject.AddDatasetArray("NAME:ds1",
    Array("NAME:Coordinates",
        Array("NAME:Coordinate", "X:=", 1, "Y:=", 2,
            Array("NAME:Coordinate", "X:=", 3, "Y:=", 4),
            Array("NAME:Coordinate", "X:=", 5, "Y:=", 7),
            Array("NAME:Coordinate", "X:=", 6, "Y:=", 20)))
oDesign.AddDatasetArray("NAME:ds1",
    Array("NAME:Coordinates",
        Array("NAME:Coordinate", "X:=", 1, "Y:=", 2,
            Array("NAME:Coordinate", "X:=", 3, "Y:=", 4),
            Array("NAME:Coordinate", "X:=", 5, "Y:=", 7),
            Array("NAME:Coordinate", "X:=", 6, "Y:=", 20))))
```

DeleteDataset

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

UI Access	Project > Datasets > Remove.		
Parameters	Name <i><DatasetName></i>	Type String	Description 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>

EditDataset

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

UI Access	Project > Datasets > Edit.									
Parameters	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><OriginalName></td> <td>String</td> <td>Name of the original dataset.</td> </tr> <tr> <td><DatasetdataArray></td> <td>Array</td> <td>Data for the modified dataset.</td> </tr> </tbody> </table>	Name	Type	Description	<OriginalName>	String	Name of the original dataset.	<DatasetdataArray>	Array	Data for the modified dataset.
Name	Type	Description								
<OriginalName>	String	Name of the original dataset.								
<DatasetdataArray>	Array	Data for the modified dataset.								
Return Value	None.									

Python Syntax	EditDataset (<OriginalName> <DatasetdataArray>)
Python Example	<pre>oProject.EditDataset ("ds1" ["NAME:ds2", ["NAME:Coordinates",</pre>

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

```

    "X:=", 3, "Y:=", 4
]
]
]
)

```

VB Syntax	EditDataset < <i>OriginalName</i> > < <i>DatasetdataArray</i> >
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>

ExportDataset

Exports a dataset to a named file. This can be executed by the oProject, or oDesign variables.

UI Access	Project > Datasets > Export.								
Parameters	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><<i>datasetFilePath</i>></td> <td>String</td> <td>The full path to the file.</td> </tr> </tbody> </table>			Name	Type	Description	< <i>datasetFilePath</i> >	String	The full path to the file.
Name	Type	Description							
< <i>datasetFilePath</i> >	String	The full path to the file.							

Return Value	None.
---------------------	-------

Python Syntax	ExportDataset (<datasetFileFullPath>)
Python Example	<pre>oProject.ExportDataset ('e:/tmp/dsdata.txt') oDesign.ExportDataset ('e:/tmp/dsdata.txt')</pre>

VB Syntax	ExportDataset <datasetFileFullPath>
VB Example	<pre>oProject.ExportDataset "e:/tmp/dsdata.txt" oDesign.ExportDataset "e:/tmp/dsdata.txt"</pre>

HasDataset

Determines whether a specified dataset exists.

UI Access	N/A						
Parameters	<table border="1"><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><datasetName></td><td>String</td><td>Name of the specified dataset.</td></tr></table>	Name	Type	Description	<datasetName>	String	Name of the specified dataset.
Name	Type	Description					
<datasetName>	String	Name of the specified dataset.					
Return Value	<p>Integer:</p> <ul style="list-style-type: none">• 1 - dataset exists.• 0 - dataset does not exist.						

Python Syntax	HasDataset (<datasetName>)
Python Example	<code>oDesign.HasDataset ('\$ds1')</code>

VB Syntax	HasDataset <datasetName>
VB Example	<code>oDesign.HasDataset "\$ds1"</code>

ImportDataset

Imports a dataset from a named file. This can be executed by the oProject, or oDesign variables. The name of the dataset is file-name+index number (e.g., dsdata1) unless the filename ends with a trailing number. When there is a trailing number at the end, we will remove the number and use first unused index. Alternatively, the name of the dataset can be explicitly defined by providing a string as an optional second argument.

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).
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')</pre>

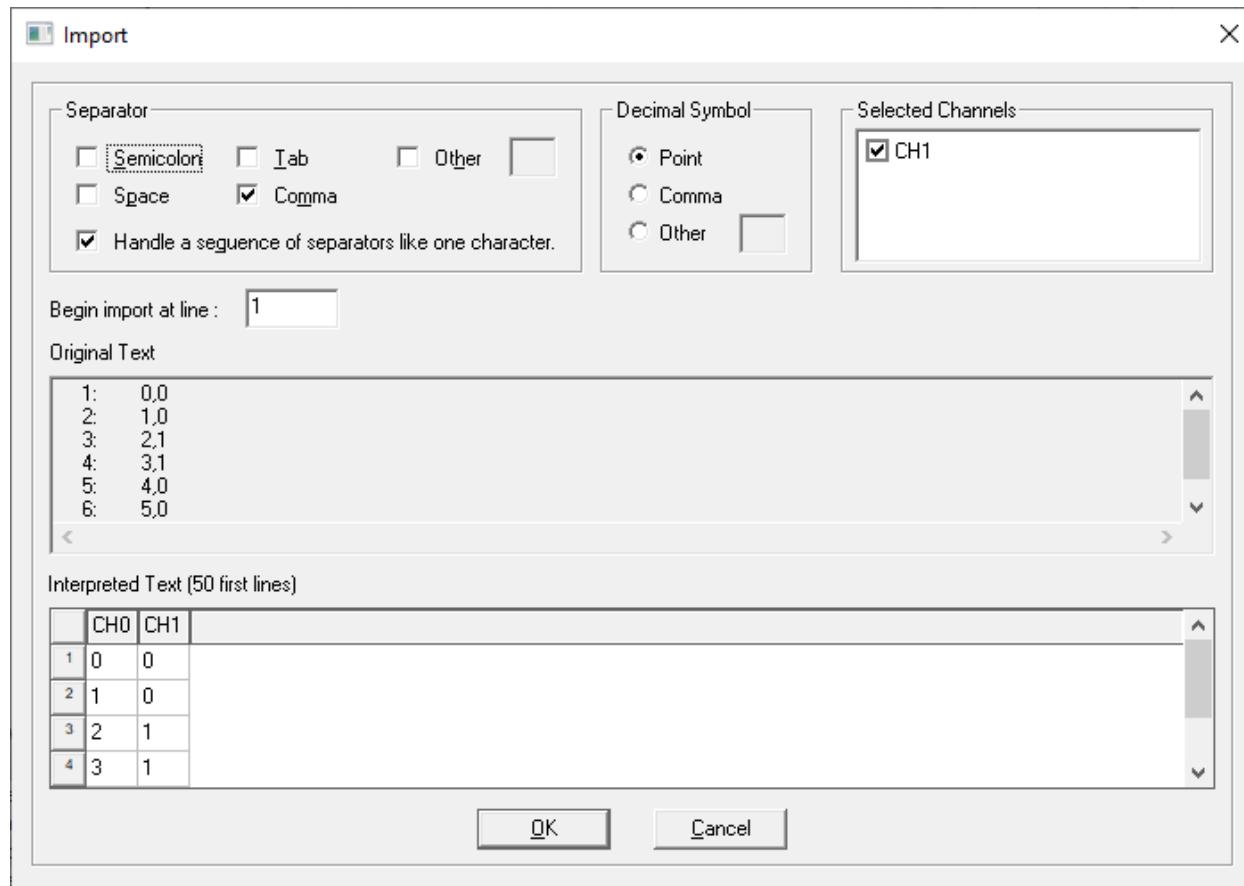
	<code>oDesign.ImportDataset('e:\tmp\dsdata.tab', 'MyDatasetName')</code>
--	--

VB Syntax	<code>ImportDataset <datasetFilePath>, <optionalDatasetName></code>
VB Example	<code>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"</code>

Note About File Types:

Tab-delimited or space-delimited files with the extension *.tab are the recommended file type. When using ImportDataset at the Design level, *.tab is the only file type supported.

At the Project level, other file types are supported (for example, *.csv). However, after calling the command, you must configure the file import format manually through the Electronics Desktop GUI by selecting **Project > Datasets** and clicking **Import**.



This page intentionally
left blank.

9 - Design Object Script Commands

Design object commands should be executed by the oDesign object.

```
oDesign.CommandName <args>
```

For example:

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"
- [Reporter Module](#) – "ReportSetup"
- [Solutions Module](#) – "Solutions"

[ApplyMeshOps](#)

[ClearLinkedData](#)

[ConstructVariationString](#)

[DeleteFieldVariation](#)

[DeleteFullVariation](#)

[DeleteLinkedDataVariation](#)

[ExportLPVROM](#)

[EcxmlExport](#)

[EditDesignSettings](#)

[EditNotes](#)

[ExportConvergence](#)

[ExportProfile](#)

[ExportReport](#)

[GenerateMesh](#)

[GetActiveEditor](#)

[GetAllPorts](#)

[GetChildNames \[Design\]](#)

[GetChildObject \[Design\]](#)

[GetChildTypes \[Design\]](#)

[GetConfigurableData](#)

[GetData](#)

[GetDesignName](#)

[GetDesignType](#)

[GetManagedFilesPath](#)

[GetModule](#)

[GetName](#)

[GetNominalVariation](#)

[GetNoteText](#)

[GetPostProcessingVariables](#)

[GetPropNames \[Design\]](#)

[GetPropValue \[Design\]](#)

[GetSelections](#)

[PasteDesign](#)

[Redo](#)

[RenameDesignInstance](#)

[RenameSource](#)

[RunToolkit](#)

[SetActiveEditor](#)

[SetDesignSettings](#)

[SetPropValue \[Design\]](#)

[SetPropertyValue](#)

[SetShowLayoutForLayoutComponent](#)

[Solve](#)

[StartAnalysis](#)

[StopSimLink](#)

[Undo](#)

[ValidateDesign](#)

[ValidateLink](#)

AddDataset

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

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

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

```
[  
    "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,</pre>

	<pre>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	None.
Return Value	None.

Python Syntax	AnalyzeAll()
Python Example	oDesign.AnalyzeAll()

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	None.
Return Value	None.

Python Syntax	AnalyzeAllNominal()
Python Example	<code>oDesign.AnalyzeAllNominal()</code>

VB Syntax	AnalyzeAllNominal
VB Example	<code>oDesign.AnalyzeAllNominal</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.
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 ConstructionVariationString 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>

CopylItemCommand

Use: Copy tree items, such as Altrasim Solution Setups in Nexxim, or Solve Setups and Frequency Sweeps in Ensemble.

Command: None

Syntax: CopylItemCommand <ItemPathList>

Return Value: None

Parameters: <ItemPathList>

Type: Array of strings

VB Example: `oDesign.CopyItemCommand Array ("Project1|Nexxim1|Analysis|DCAnalysis2")
y,|`

CreateReport

Creates a new report with a single trace and adds it to the **Results** branch in the project tree.

UI Access	Right-click on Results > Create [Type] Report		
	Name	Type	Description
	<code><ReportName></code>	String	Name of report.
Parameters	<code><ReportType></code>	String	Type of report. Possible values are: "Modal S Parameters" - Only for Driven Modal solution-type problems with ports. "Terminal S Parameters" - Only for Driven Terminal solution-type problems with ports. "Eigenmode Parameters" - Only for Eigenmode solution-type problems. "Fields" "Far Fields" - Only for problems with radiation or PML boundaries. "Near Fields" - Only for problems with radiation or PML boundaries. "Emission Test"
	<code><DisplayType></code>	String	Type of display. If ReportType is "Modal S Parameters", "Terminal S Parameters", or

		<p>"Eigenmode Parameters", then set to one of the following:</p> <p>"Rectangular Plot", "Polar Plot", "Radiation Pattern", "Smith Chart", "Data Table", "3D Rectangular Plot", "3D Rectangular Bar Plot" or "3D Polar Plot".</p> <p>If <ReportType> is "Fields", then set to one of the following:</p> <p>"Rectangular Plot", "Polar Plot", "Radiation Pattern", "Rectangular Contour Plot", "Data Table", "3D Rect- angular Plot", or "3D Rectangular Bar Plot".</p> <p>If <ReportType> is "Far Fields" or "Near Fields", then set to one of the fol- lowing:</p> <p>"Rectangular Plot", "Radiation Pattern", "Rectangular Contour Plot" "Data Table", "3D Rectangular Plot", "3D Rectangular Bar Plot" or "3D Polar Plot"</p> <p>If <ReportType> is "Emission Test", then set to one of the following:</p> <p>"Rectangular Plot" or "Data Table"</p>
<SolutionName>	String	Name of the solution as listed in the Modify Report dialog box.
<ContextArray>	Array	<p>Context for which the expression is being evaluated. This can be an empty string if there is no context.</p> <p>Array("Domain:=", <DomainType>) <DomainType> ex. "Sweep" or "Time" Array("Context:=", <GeometryType>) <GeometryType> ex. "Infinite Spheren", "Spheren", "Polylinen"</p>
<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>	<p>Array</p> <p>This array contains the report quantity and X, Y, and (Z) axis definitions.</p> <pre>Array("X Component:=", <VariableName>, "Y Component:=", <VariableName> <ReportQuantityArray> <ReportQuantityArray> ex. Array("dB(S(Port1, Port1))")</pre>
Return Value	None.	

Python Syntax	<pre>CreateReport(<ReportName>, <ReportType>, <DisplayType>, <SolutionName>, <ContextArray>, <FamiliesArray>, <ReportdataArray>, <ExtendedInfo>)</pre>
Python Example	<p>Three examples are given below: (nominal, Optimetrics, spectral)</p> <p>Nominal Example:</p> <pre>oModule.CreateReport("XY Plot 2", "Standard", _ "Rectangular Plot", "TR", ["NAME:Context", _ "SimValueContext:=", [] 1, 0, 2, 0, false,_ false, -1, 1, 0, 1, 1, "", 0, 0]], ["Time:=", _</pre>

```

        ["All"], "aaa:=", ["Nominal"]],_
        ["X Component:=", "Time", "Y Component:=",_
        ["E1.I"]]][])

```

Optimetrics Example:

```

oModule.CreateReport ("XY Plot 3", "Standard",_
        "Rectangular Plot", "TR", ["NAME:Context",_
        "OptiSetup:=", "ParametricSetup1", "SimValueCor_
        [1, 0, 2, 0, false, false, 0, 1,_
        0, 1, 1, "", 0, 0]], ["Time:=", ["All"],_
        "aaa:=", ["Nominal"]], ["X Component:=",_
        "Time", "Y Component:=", ["E1.I"]], [])

```

Spectral Example:

```

oModule.CreateReport ("XY Plot 4", "Standard",_
        "Rectangular Plot", "TR", ["NAME:Context",_
        "SimValueContext:=", [ 2, 0, 2, 0, false, false,_
        -1, 1, 0, 1, 1, "", 0, 0, "CG", false, "0", "K",_
        false, "0", "MH", false, "100", "TE", false, "4",_
        "TH", false, "40", "TS", false, "0ns", "UF", fa_
        "0", "WT", false, "0", "WW", false, "100"]],_
        ["Spectrum:=", ["All"], "aaa:=",_
        ["Nominal"]], ["X Component:=", "Spectrum", _]

```

```
"Y Component:=", ["mag(E1.I)"]], [])
```

Partial Discharge Example

```
oModule.CreateReport("Partial Discharge Plot 2", "Partial Discharge", "Rectangular Plot",
"PartialDischarge1 : PartialDischarge", [],
[
    "GasPressure:="          , ["All"],
    "Freq:="                 , ["All"],
    "bend_angle:="           , ["All"]
],
[
    "X Component:="         , "GasPressure",
    "Y Component:="          , ["Power"]
])
```

Example 3D Rectangular Bar Plot with Change Bar properties

```
oModule = oDesign.GetModule("ReportSetup")

oModule.CreateReport("S Parameter Plot 4", "Modal Solution Data", "3D Rectangular Bar
Plot", "Setup1 : Sweep", [],
[
    "Freq:="                 , ["All"],
    "UU:="                   , ["All"],
    "ZZZ:="                  , ["Nominal"]]
```

```
        ],
        [
            "X Component:=" , "Freq",
            "Y Component:=" , "UU",
            "Z Component:=" , ["dB(S(wDipole1_1_p1,wDipole1_1_p1))"]
        ])
oModule.ChangeProperty(
    [
        "NAME:AllTabs",
        [
            "NAME:Bar",
            [
                "NAME:PropServers",
                "S Parameter Plot 4:Traces-dB(S(wDipole1_1_p1,wDipole1_1_p1))"
            ],
            [
                "NAME:ChangedProps",
                [
                    "NAME:Transparency",
                    "Value:=" , "0.5"
                ]
            ]
        ]
    ]
)
```

```
        ]
    ]
])

oModule.ChangeProperty(
[
    "NAME:AllTabs",
    [
        "NAME:Bar",
        [
            "NAME:PropServers",
            "S Parameter Plot 4:Traces-dB(S(wDipole1_1_p1,wDipole1_1_p1))"
        ],
        [
            "NAME:ChangedProps",
            [
                "NAME:Filled",
                "Value:=" , False
            ]
        ]
    ]
]
```

```
        ])
    oModule.ChangeProperty(
        [
            "NAME:AllTabs",
            [
                "NAME:Bar",
                [
                    "NAME:PropServers",
                    "S Parameter Plot 4:Traces-dB(S(wDipole1_1_p1,wDipole1_1_p1))"
                ],
                [
                    "NAME:ChangedProps",
                    [
                        "NAME:Filled",
                        "Value:=" , True
                    ]
                ]
            ]
        ])
    oModule.ChangeProperty(
        [
```

```
"NAME:AllTabs",
[
    "NAME:Bar",
    [
        "NAME:PropServers",
        "S Parameter Plot 4:Traces-dB(S(wDipole1_1_p1,wDipole1_1_p1))"
    ],
    [
        "NAME:ChangedProps",
        [
            "NAME>Show Outline",
            "Value:=" , True
        ]
    ]
]
))

oModule.ChangeProperty(
[
    "NAME:AllTabs",
    [
```

```
"NAME:Bar",
[
    "NAME:PropServers",
    "S Parameter Plot 4:Traces-dB(S(wDipole1_1_p1,wDipole1_1_p1))"

],
[
    "NAME:ChangedProps",
    [
        "NAME:Outline Color",
        "R:=" , 0,
        "G:=" , 0,
        "B:=" , 160
    ]
]
])

oModule.ChangeProperty(
[
    "NAME:AllTabs",
    [
        "NAME:Bar",
```

```
[  
    "NAME:PropServers",  
    "S Parameter Plot 4:Traces-dB(S(wDipole1_1_p1,wDipole1_1_p1))"  
,  
[  
    "NAME:ChangedProps",  
    [  
        "NAME:Outline Width",  
        "Value:=" , "2"  
    ]  
]  
])  
oModule.ChangeProperty(  
[  
    "NAME:AllTabs",  
    [  
        "NAME:Bar",  
        [  
            "NAME:PropServers",
```

```
        "S Parameter Plot 4:Traces-dB(S(wDipole1_1_p1,wDipole1_1_p1))"

    ] ,
    [
        "NAME:ChangedProps",
        [
            "NAME:Bar Width",
            "Value:=" , "Narrow"
        ]
    ]
])

oModule.ChangeProperty(
[
    "NAME:AllTabs",
    [
        "NAME:Bar",
        [
            "NAME:PropServers",
            "S Parameter Plot 4:Traces-dB(S(wDipole1_1_p1,wDipole1_1_p1))"

        ],
        [

```

```
        "NAME:ChangedProps",
        [
            "NAME:Bar Width",
            "Value:=" , "Wide"
        ]
    ]
]
)

oModule.ChangeProperty(
[
    "NAME:AllTabs",
    [
        "NAME:Bar",
        [
            "NAME:PropServers",
            "S Parameter Plot 4:Traces-dB(S(wDipole1_1_p1,wDipole1_1_p1))"
        ],
        [
            "NAME:ChangedProps",
            [

```

```

        "NAME:Bar Infinity",
        "Value:=" , True
    ]
]
]
)
```

VB Syntax	<pre>CreateReport <ReportName>, <ReportType>, <DisplayType>, <SolutionName>, <ContextArray>, <FamiliesArray>, <ReportdataArray>, <ExtendedInfo></pre>
VB Example	<pre> oModule.CreateReport "3D Cartesian Plot1", "Far Fields", _ "3D Cartesian Plot", "Setup1 : LastAdaptive", _ Array("Context:=", "Infinite Sphere1", "Domain:=", "Sweep"), _ Array("Theta:=", Array("All")), "Phi:=", Array("All"), _ "Freq:=", Array("10GHz")), _ Array("X Component:=", "Theta", _ "Y Component:=", "Phi", _ "Z Component:=", Array("rETotal")), _ Array() oModule.CreateReport "ReptSmithFreq", _ "Modal Solution Data", "Smith Plot", "Setup1 : Sweep1", _</pre>

```
        Array("Domain:=", "Sweep"), _
        Array("Freq:=", Array("All")), _
        Array("Polar Component:=", _
        Array("ln(Y(LumpPort1,LumpPort1))")), _
        Array()

oModule.CreateReport "Rectangular Contour Plot 2", "Far Fields", _
    "Rectangular Contour Plot", "Setup1 : LastAdaptive", Array("Context:=", _
    "Infinite Sphere1"), Array("_u:=", Array("All"), "_v:=", Array("All"), "Freq:=", _
    Array( _
        "5GHz")), Array("X Component:=", "_u", "Y Component:=", "_v", "Z Component:=", Array(
        (
            "rETotal")), Array()

oModule.CreateReport "Rept2DRectFreq", _
    "Modal Solution Data", "XY Plot", _
    "Setup1 : Sweep1", _
    Array("Domain:=", "Sweep"), _
    Array("Freq:=", Array("All")), _
    Array("X Component:=", "Freq", _
    "Y Component:=", _ Array("dB(S(LumpPort1,LumpPort1))")), _
    Array()
```

DeleteDataset

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

UI Access	Project > Datasets > Remove.								
Parameters	<table border="1"> <tr> <td>Name</td> <td>Type</td> <td>Description</td> </tr> <tr> <td><DatasetName></td> <td>String</td> <td>Name of the dataset found in the project.</td> </tr> </table>			Name	Type	Description	<DatasetName>	String	Name of the dataset found in the project.
Name	Type	Description							
<DatasetName>	String	Name of the dataset found in the project.							
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>

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	<table border="1"> <tr> <td>Name</td> <td>Type</td> <td>Description</td> </tr> <tr> <td><variationKeys></td> <td>Array</td> <td>Array containing either "All" string to select all variations, or strings of vari-</td> </tr> </table>			Name	Type	Description	<variationKeys>	Array	Array containing either "All" string to select all variations, or strings of vari-
Name	Type	Description							
<variationKeys>	Array	Array containing either "All" string to select all variations, or strings of vari-							

		ations to include.
<deleteMesh>	Boolean	When true, deletes mesh data.
<deleteLinkedData>	Boolean	When true, deletes linked data.
Return Value	None.	

Python Syntax	DeleteFieldVariation(<variationKeys>, <deleteMesh>, <deleteLinkedData>)
Python Example	<pre> oProject = oDesktop.GetActiveProject() oDesign = oProject.GetActiveDesign() Design.DeleteFieldVariation(['All'], True, False) </pre>

VB Syntax	DeleteFieldVariation <variationKeys>, <deleteMesh>, <deleteLinkedData>
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	[Icepak] > 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	<code>DeleteFullVariation(<variationKeys>, <deleteLinkedData>)</code>
Python Example	<code>oDesign.DeleteFullVariation(['All']), false)</code>

VB Syntax	<code>DeleteFullVariation <variationKeys>, <deleteLinkedData></code>
VB Example	<code>oDesign.DeleteFullVariation Array('All'), false</code>

DeleteLinkedDataVariation

Deletes the linked data of specified variations.

UI Access	[Icepak] > Results > Clean Up Solutions.		
Parameters	Name	Type	Description
	<code><DesignVariationKeys></code>	Array	Array containing strings of the variations keys whose linked data are going to be deleted.
Return Value	None.		

Python Syntax	<code>DeleteLinkedDataVariation(<DesignVariationKeys>)</code>
Python Example	<code>oDesign.DeleteLinkedDataVariation(["current='0.9mA'", "current='1.0mA'"])</code>

VB Syntax	DeleteLinkedDataVariation < <i>DesignVariationKeys</i> >
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	Icepak > Results > Output Variables. In the Output Variables window, click Delete .					
Parameters	<table border="1"><tr><td>Name <i><OutputVarName></i></td><td>Type String</td><td>Description Name of the output variable.</td></tr></table>			Name <i><OutputVarName></i>	Type String	Description Name of the output variable.
Name <i><OutputVarName></i>	Type String	Description Name of the output variable.				
Return Value	None.					

Python Syntax	DeleteOutputVariable (< <i>OutputVarName</i> >)
Python Example	<pre>oModule = oDesign.GetModule("OutputVariable") oModule.DeleteOutputVariable ("testNew")</pre>

VB Syntax	DeleteOutputVariable < <i>OutputVarName</i> >
VB Example	<pre>Set oModule = oDesign.GetModule("OutputVariable") oModule.DeleteOutputVariable "testNew"</pre>

DeleteSolutionVariation

Deletes all solution data for specific solutions and design variations. This is obsolete and is supported only for backward compatibility. You should use DeleteFullVariation.

UI Access	Right-click on Results , select Browse Solutions... , click Delete button in the dialog.		
Parameters	Name <i><SoluParam></i>	Type Array	Description Structured array. Array (<DataSpecifierArray>, ...)
	<i><DataSpecifierArray></i>	Array	Structured array. Array (<DesignVariationKey>, <SetupName>, <SolutionName>)
Return Value	None.		

Python Syntax	DeleteSolutionVariation(<SoluParam>)
Python Example	<pre>oModule.DeleteSolutionVariation([["width='2in'", "Setup1", "Adaptive_1"], ["width='2in'", "Setup1", "Sweep1"]])</pre>

VB Syntax	DeleteSolutionVariation <SoluParam>
VB Example	<pre>oModule.DeleteSolutionVariation Array(_</pre>

	Array("width='2in'", "Setup1", "Adaptive_1"), _ Array("width='2in'", "Setup1", "Sweep1"))
--	--

DeleteOutputVariable

Deletes an existing output variable. The variable can only be deleted if it is not in use by any traces.

UI Access	Icepak > Results > Output Variables. In the Output Variables window, click Delete .								
Parameters	<table border="1"><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><OutputVarName></td><td>String</td><td>Name of the output variable.</td></tr></table>			Name	Type	Description	<OutputVarName>	String	Name of the output variable.
Name	Type	Description							
<OutputVarName>	String	Name of the output variable.							
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")
```

EditDesignSettings

Edits design settings.

UI Access	[Icepak] > Design Settings.		
Parameters	Name	Type	Description
	<overrideArray>	Array	Structured array.
Return Value	None.		

Python Syntax	EditDesignSettings(<overrideArray>)
Python Example	<pre>oDesign.EditDesignSettings (["NAME:Design Settings Data", "Allow Material Override:=", True, "Perform Minimal validation:=", False, "EnabledObjects:=" , [],], ["NAME:Model Validation Settings", "EntityCheckLevel:=" , "Strict", "IgnoreUnclassifiedObjects:=", False, "SkipIntersectionChecks:=", False, "SkipMeshChecks:=", True])</pre>

VB Syntax	EditDesignSettings <overrideArray>
------------------	------------------------------------

VB Example	
------------	--

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>

EMDesignOptions

Use: Set options for an EM Design.

Command: Right click on design and select **EM Design Options**

Syntax: DesignOptions <Options Array>

Return Value: None

Parameters:

<Options Array>

```
Array("NAME:options",
"SaveSolFilesAsBinary:=", <boolean>,
"LowPriorityForSimulations:=", <boolean>,
"SaveNearFieldSolutions:=", <boolean>,
```

```
"SchematicEnabled:=", <boolean>,  
"UseGlobalNumProc:=", <boolean>,  
"ComputeBothEvenAndOddCPWModes:=", <boolean>,  
"NumProcessors:=", <int>,  
"NumProcessorsDistrib:=", <int>,  
"CausalMaterials:=", <boolean>,  
"UseHPCForMP:=", <boolean>,  
"HPCLicenseType:=", <int>)
```

SaveSolFilesAsBinary - if true, solutions files are saved using a binary format.

LowPriorityForSimulations - if true, run simulations at a lower CPU priority.

SaveNearFieldSolutions - if true, save near field solutions

SchematicEnabled - if true, enable schematics

UseGlobalNumProc - if true, use global number of processors and ignore NumProcessors

ComputeBothEvenAndOddCPWModes - if true, compute both even and odd cpw modes

NumProcessors - number of processors

NumProcessorsDistrib - number of distributed processors

CausalMaterials - if true, use causal materials

UseHPCForMP - if true, use hpc for mp

HPCLicenseType - number indicating hpc license type

VB Example:

```

oDesign.DesignOptions Array("NAME:options",
"SaveSolFilesAsBinary:=", true,
"LowPriorityForSimulations:=", false,
"SaveNearFieldSolutions:=", false,
"SchematicEnabled:=", true,
"UseGlobalNumProc:=", true,
"ComputeBothEvenAndOddCPWModes:=", false,
"NumProcessors:=", 1,
"NumProcessorsDistrib:=", 1,
"CausalMaterials:=", true,
"UseHPCForMP:=", true,
"HPCLicenseType:=", 1)

```

ExportConvergence

For a given variation, exports convergence data (max mag delta S, E, freq) to *.conv file.

UI Access	Results > Solution Data. Select Convergence tab and click Export.		
Parameters	Name	Type	Description
	<Setup>	String	Setup name.
	<VariationKeys>	String	The variation. Pass empty string for the current nominal variation.
	<FilePath>	String	Full path to desired *.conv file location.
	<OverwritelfExists>	Boolean	<i>Optional.</i> If True, overwrites any existing file at the same path.

Return Value	None.
---------------------	-------

Python Syntax	
Python Example	

VB Syntax	
VB Example	

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 <i><datasetFilePath></i>	Type String	Description 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 <datasetFileFullPath>
VB Example	<pre>oProject.ExportDataset "e:/tmp/dsdata.txt" oDesign.ExportDataset "e:/tmp/dsdata.txt"</pre>

ExportLPVROM

Exports LPV ROM files for use in TwinBuilder.

UI Access	Icepak > Export ROM > LPV ROM		
Parameters	Name	Type	Description
	<Parametric Setup>	string	Parametric Setup
	<Transient Setup>	string	Transient Setup
	<File path>	string	The file path to the AEDT project export location
Return Value	None		

Python Syntax	ExportLPVROM (<C>)
Python Example	<pre>oDesign.ExportLPVROM("ParametricSetup", "TransientSetup", "C:/Model/LPVProject_LPVROM.aedtexport/")</pre>

VB Syntax	ExportLPVROM (<C>)
VB Example	<pre>oDesign.ExportLPVROM "ParametricSetup", "TransientSetup", _ "C://Model/LPVProject_LPVROM.aedtexport/"</pre>

EcxmlExport

Exports and Electronics Cooling (EC XML) file.

UI Access	Icepak > Export ECXML		
Parameters	Name	Type	Description
	NA	NA	NA
Return Value	None		

Python Syntax	EcxmlExport
Python Example	<code>oDesign.EcxmlExport()</code>

VB Syntax	EcxmlExport
VB Example	<code>oDesign = oProject.SetActiveDesign("IcepakDesign1") oDesign.EcxmlExport</code>

ExportProfile

Exports a solution profile to file.

UI Access	N/A		
Parameters	Name	Type	Description
	<code><SetupName></code>	String	Text of the design's notes.

	<code><VariationString></code>	String	Variation name. Pass empty string for the current nominal variation.
	<code><filePath></code>	String	Full file path, including extension *.prof.
	<code><overwriteIfExists></code>	Boolean	Optional. <ul style="list-style-type: none"> • True - overwrites any existing file of the same name. • False - no overwrite.
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	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><code><type></code></td> <td>String</td> <td>Optional. Valid options are "Module", "Editor", and "Variable". Default returns Module and Editor names.</td> </tr> </tbody> </table>			Name	Type	Description	<code><type></code>	String	Optional. Valid options are "Module", "Editor", and "Variable". Default returns Module and Editor names.
Name	Type	Description							
<code><type></code>	String	Optional. Valid options are "Module", "Editor", and "Variable". Default returns Module and Editor names.							
Return Value	Names of children for the queried object.								
Python Syntax	<code>GetChildNames (<type>)</code>								
Python Example	<code>oDesign.GetChildNames ("Variable")</code>								

VB Syntax	GetChildNames <type>
VB Example	<code>oDesign.GetChildNames "Variable"</code>

GetChildObject [Design]

Returns the design's child objects.

UI Access	N/A		
Parameters	Name	Type	Description
	<code><path></code>	String	The path may include multiple generations (for example, "designObject/moduleObj/SetupObject"). See: Object Path .
Return Value	A child object if one is found. Otherwise script error.		

Python Syntax	GetChildObject(<path>)
Python Example	<pre>oDesign = oProject.GetActiveDesign() oOptimModule = oDesign.GetChildObject("Optimetrics") oEditor = oDesign.GetChildObject("3D Model") oBox = oDesign.GetChildObject("3D Model/Box1") oRpt = oDesign.GetChildObject("Results/S Parameter Plot 1") oVariable= oDesign.GetChildObject("Offset")</pre>

VB Syntax	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 <code><type></code>	Type String	Description Specified type.
Return Value	None.		

Python Syntax	<code>GetConfigurableData(<type>)</code>
Python Example	<code>oDesign.GetConfigurableData ("model")</code>

VB Syntax	<code>GetConfigurableData <type></code>
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	<code>oDesign = oProject.GetActiveDesign() oDesign.GetDesignID()</code>

VB Syntax	GetDesignType
------------------	---------------

VB Example

```
Set oDesign = oProject.GetActiveDesign  
oDesign.GetDesignID
```

GetDesignName

Returns the name of the active design.

UI Access	N/A
Parameters	None.
Return Value	String indicating the name of the active design.

Python Syntax

```
GetDesignName()
```

Python Example

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

VB Syntax

```
GetDesignName
```

VB Example

```
Set oDesign = oProject.GetActiveDesign  
oDesign.GetDesignName
```

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", "RMxprt", or "Twin Builder").

Python Syntax	<code>GetDesignType()</code>
Python Example	<code>oDesign = oProject.GetActiveDesign() oDesign.GetDesignType()</code>

VB Syntax	<code>GetDesignType</code>
VB Example	<code>Set oDesign = oProject.GetActiveDesign oDesign.GetDesignType</code>

GetEdgePositionAtNormalizedParameter

Gets the position on an edge with normalized parameter.

UI Access	N/A											
Parameters	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><code><EdgeID></code></td> <td>Integer</td> <td>Edge ID.</td> </tr> <tr> <td><code><NormParam></code></td> <td>Double</td> <td>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.</td> </tr> </tbody> </table>			Name	Type	Description	<code><EdgeID></code>	Integer	Edge ID.	<code><NormParam></code>	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.
Name	Type	Description										
<code><EdgeID></code>	Integer	Edge ID.										
<code><NormParam></code>	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	GetEdgePositionAtNormalizedParameter <EdgeID>, <NormParam>
VB Example	<pre>oEditor.GetEdgePositionAtNormalizedParameter 5, 0 oEditor.GetEdgePositionAtNormalizedParameter 5, 1 oEditor.GetEdgePositionAtNormalizedParameter 5, 0.5</pre>

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	oDesign.GetManagedFilePath()
----------------------	------------------------------

Python Example	<code>oDesign.GetManagedFilesPath()</code>
-----------------------	--

VB Syntax	<code>oDesign.GetManagedFilesPath</code>
VB Example	<code>oDesign.GetManagedFilesPath</code>

GetModule

Returns the IDispatch for the specified module.

UI Access	N/A		
Parameters	Name <code><modulename></code>	Type String	Description One of the following: <ul style="list-style-type: none">• Analysis Module – "AnalysisSetup"• Boundary Module – "BoundarySetup"• Field Overlays Module – "FieldsReporter"• Mesh Module – "MeshSetup"• Optimetrics Module – "Optimetrics"• Reporter Module – "ReportSetup"• Solutions Module – "Solutions"
Return Value	Module IDispatch		

Python Syntax	<code>GetModule (<modulename>)</code>
----------------------	---

Python Example

```
oModule = oDesign.GetModule("SimSetup")
```

VB Syntax

```
GetModule <modulename>
```

VB Example

```
set oModule = oDesign.GetModule "SimSetup"
```

GetName

Returns the name of the active design.

UI Access	N/A
------------------	-----

Parameters	None.
-------------------	-------

Return Value	String indicating the name of the active design.
---------------------	--

Python Syntax

```
GetName()
```

Python Example

```
design_name = oDesign.GetName()
```

VB Syntax

```
GetName
```

VB Example

```
design_name = oDesign.GetName
```

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>, <ContextArray>
------------------	--

VB Example

```
Val = oDesign.GetOutputVariableValue  
"test", "Freq = '20Ghz' Theta='20deg'  
Phi='30deg'", "TR", "Standard", Array()
```

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

```
oDesign.GetOutputVariables()
```

VB Syntax

```
GetOutputVariables
```

VB Example

```
oDesign.GetOutputVariables
```

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	<p>One of the following, where tab titles are shown in parentheses:</p> <ul style="list-style-type: none"> • PassedParameterTab ("Parameter Values") • DefinitionParameterTab (Parameter Defaults") • LocalVariableTab ("Variables" or "Local Variables") • ProjectVariableTab ("Project variables") • ConstantsTab ("Constants")

			<ul style="list-style-type: none"> • 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 ComplInst@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	<p>One of the following, where tab titles are shown in parentheses:</p> <ul style="list-style-type: none"> • 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.GetPropertyValues("PassedParameterTab", k, "R") ... </pre>

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

GetPropNames [Design]

Returns array containing names of property. For designs, always returns an empty array because the design has no property.

UI Access	N/A		
Parameters	Name <includeReadOnly>	Type Boolean	Description (Optional). <ul style="list-style-type: none">• True - include read only property.• False - do not include read only property.

Return Value	Empty array.
Python Syntax	GetPropNames(<includeReadOnly>)
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	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><propPath></td> <td>String</td> <td>A child object's property path. See: Object Property Script Function Summary.</td> </tr> </tbody> </table>			Name	Type	Description	<propPath>	String	A child object's property path. See: Object Property Script Function Summary .
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

```
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")
```

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

```
oDesign.GetSolutionType()
```

VB Syntax

```
GetSolutionType
```

VB Example

```
oDesign.GetSolutionType
```

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	GetVariables ()
Python Example	<pre>oProject.GetVariables() oDesign.GetVariables()</pre>

VB Syntax	GetVariables
VB Example	<pre>oProject.GetVariables oDesign.GetVariables</pre>

GetVariableValue

Gets the value of a single specified variable. To get the value of project variables, execute this command using `oProject`. To get the value of local variables, use `oDesign`.

UI Access	N/A						
Parameters	<table border="1"><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><VarName></td><td>String</td><td>Name of the variable to access.</td></tr></table>	Name	Type	Description	<VarName>	String	Name of the variable to access.
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	<pre>oProject.GetVariableValue("var_name")</pre>

VB Syntax	GetVariableValue <VarName>
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.
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	<code>oDesign.GetVariationVariableValue ('x_size = 2mm y_size = 1mm', 'y_size')</code>

VB Syntax	GetVariationVariableValue <VariationString>, <VariableName>
VB Example	<code>varval = oDesign.GetVariationVariableValue "x_size = 2mm y_size = 1mm", "y_size"</code>

ImportDataset

Imports a dataset from a named file. This can be executed by the oProject, or oDesign variables. The name of the dataset is file-name+index number (e.g., dsdata1) unless the filename ends with a trailing number. When there is a trailing number at the end, we will remove the number and use first unused index. Alternatively, the name of the dataset can be explicitly defined by providing a string as an optional second argument.

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).
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>

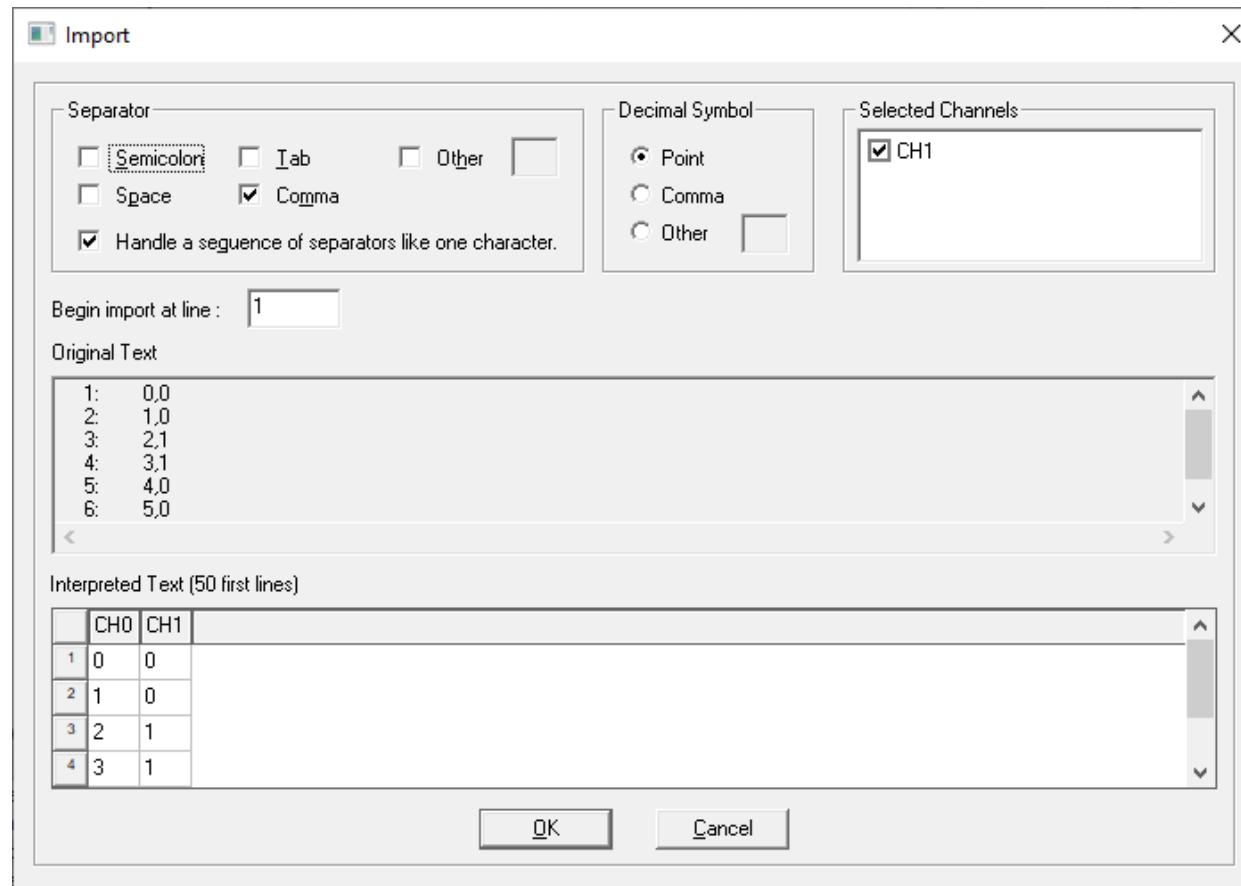
VB Syntax	ImportDataset <datasetFilePath>, <optionalDatasetName>
VB Example	<pre>oProject.ImportDataset "e:\tmp\dsdata.tab"</pre>

```
oDesign.ImportDataset "e:\tmp\dsdata.tab"  
oProject.ImportDataset "e:\tmp\dsdata.tab", "MyDatasetName"  
oDesign.ImportDataset "e:\tmp\dsdata.tab", "MyDatasetName"
```

Note About File Types:

Tab-delimited or space-delimited files with the extension *.tab are the recommended file type. When using ImportDataset at the Design level, *.tab is the only file type supported.

At the Project level, other file types are supported (for example, *.csv). However, after calling the command, you must configure the file import format manually through the Electronics Desktop GUI by selecting **Project > Datasets** and clicking **Import**.



InsertDesign

Inserts a new design in the project. For Icepak scripts, the last argument will always be empty.

UI Access	Project > [Insert Icepak Design].		
Parameters	Name	Type	Description
	< <i>DesignType</i> >	String	Design type."Icepak".
	< <i>DesignName</i> >	String	Design name.
	< <i>SolutionType</i> >< <i>ProblemType</i> >	String	Can be "SteadyState TemperatureAndFlow", "SteadyState TemperatureOnly", "SteadyState FlowOnly", "Transient TemperatureAndFlow", "Transient TemperatureOnly", or "Transient FlowOnly".
	""	None	Empty argument.
Return Value	None.		

Python Syntax	InsertDesign (< <i>DesignType</i> >, < <i>DesignName</i> >, < <i>SolutionType</i> >,")
Python Example	<pre>oProject.InsertDesign('Icepak', 'MyDesign', 'TemperatureAndFlow', '')</pre>

VB Syntax	InsertDesign < <i>DesignType</i> >, < <i>DesignName</i> >, < <i>SolutionType</i> >,""
VB Example	<pre>oProject.InsertDesign "Icepak", "MyDesign", "TemperatureAndFlow", ""</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
	< <i>pasteOption</i> >	Integer	One of the following:

			<ul style="list-style-type: none">• 0 – Link to the existing design that was copied• 1 – Create a new copy of the design and keep the original layers of the design being copied• 2 – Create a new copy of the design and merge the layers of the design being copied.
Return Value	None.		

Python Syntax	PasteDesign(<pasteOption>)
Python Example	<pre>oProject.CopyDesign('DesignB') oDesign = oProject.SetActiveDesign('DesignA') oDesign.PasteDesign(1)</pre>

VB Syntax	PasteDesign <pasteOption>
VB Example	<pre>oProject.CopyDesign "B" Set oDesign = oProject.SetActiveDesign ("A") oDesign.PasteDesign 1</pre>

Redo [Design]

Reapplies the last design-level command.

UI Access	Edit > Redo.
------------------	--------------

Parameters	None.
Return Value	None.

Python Syntax	Redo()
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
	<code><OldName></code>	String	The current name of the design, which must be the design on which this command is invoked.
Return Value	None.		

Python Syntax	RenameDesignInstance (<OldName>, <NewName>)
Python Example	<code>oDesign.RenameDesignInstance ("Design1", "Design2")</code>

VB Syntax	RenameDesignInstance <OldName>, <NewName>
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"`

RunToolkit

Runs a Python toolkit script, applying it to the active design. The toolkit script itself may have prerequisites, such as sweeps defined, or specific model characteristics, such as port definitions.

Important:

Full scripting for cable modeling is not supported and no arguments are allowed in the script. The **RunToolkit** command will not automatically create a cable bundle, but will simply open the **Cable Modeling** dialog box. You will have to manually input your parameters.

UI Access	[Icepak] > Toolkit > [Script Name].		
Parameters	Name	Type	Description
	<code><LibraryType></code>	String	Name of the library in which the script is located.
	<code><ToolkitName></code>	String	Name of the Python script.
	<code><ToolkitArg></code>	Array	Structured array containing arguments required by the toolkit script.

Return Value	User-defined solution(s) and/or output(s).
---------------------	--

Python Syntax	RunToolkit(<LibraryType>, <ToolkitName>, <ToolkitArg>)
Python Example	<code>oDesign.RunToolkit("SysLib", "Geometry/Heatsinks/AngledFinHS", [])</code>

VB Syntax	RunToolkit <LibraryType>, <ToolkitName>, <ToolkitArg>
VB Example	<code>oDesign.RunToolkit "SysLib", "Geometry/Heatsinks/AngledFinHS", Array()</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 <code><EditorName></code>	Type String	Description 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

	<code><allowMaterialOverride></code>	Integer	1 - allow material override. 0 - does not allow material override.
Return Value	None.		

Python Syntax	<code>SetAllowMaterialOverride(<allowMaterialOverride>)</code>
Python Example	<code>oDesign.SetAllowMaterialOverride(1)</code>

VB Syntax	<code>SetAllowMaterialOverride <allowMaterialOverride></code>
VB Example	<code>oDesign.SetAllowMaterialOverride 1</code>

SetBackgroundMaterial

Sets the design's background material.

Important:

You can only use this script in 2D Extractor if there is no surface ground in the design *and* the problem type is "open".

UI Access	From the Project Manager , right-click the design and select Set Background Material .		
Parameters	Name	Type	Description
	<code><matName></code>	String	Material name.
Return Value	None.		

Python Syntax	<code>SetBackgroundMaterial(<matName>)</code>
----------------------	---

Python Example	<code>oDesign.SetBackgroundMaterial('vacuum')</code>
-----------------------	--

VB Syntax	<code>SetBackgroundMaterial <matName></code>
------------------	--

VB Example	<code>oDesign.SetBackgroundMaterial "vacuum"</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
	<code><DistributedUnits></code>	String	Length units used for post-processing.
	<code><LumpedLength></code>	String	Length of design in post-processing.
	<code><RiseTime></code>	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>
-------------------	---

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 .
Return Value	Boolean: <ul style="list-style-type: none"> • 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	<p>One of the following, where tab titles are shown in parentheses:</p> <ul style="list-style-type: none">• 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")

		<ul style="list-style-type: none"> • Signals ("Signals")
<propServer>	String	An object identifier, generally returned from another script method, such as ComplInst@R;2;3
<propName>	String	Name of the property.
<propValue>	String	The value for the property
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"

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 .

			<ul style="list-style-type: none"> • True - turn on auto open mode. • False - turn off auto open mode.
	<code><ModelExteriorAsIE></code>	String	Only Applies for Driven Solution Type with Auto Open Mode as True. Possible values are: Radiation, FEBI, PML
Return Value	None.		

Python Syntax	<code>SetSolutionType(<SolutionType>, <AutoOpenMode>, <ModelExteriorAsIE>)</code>
	<pre> oDesign.SetSolutionType("HFSS Modal Network", ["NAME:Options", "EnableAutoOpen:=" , False]) </pre>
Python Example	<pre> oDesign.SetSolutionType("Transient Network", ["NAME:Options", "EnableAutoOpen:=" , False]) oDesign.SetSolutionType("HFSS Hybrid Modal Network", ["NAME:Options", "EnableAutoOpen:=" , False]) </pre>

```

    [
        "NAME:Options",
        "EnableAutoOpen:=" , False
    ] )

oDesign.SetSolutionType("SBR+",
    [
        "NAME:Options",
        "EnableAutoOpen:=" , False
    ] )

```

VB Syntax	SetSolutionType <SolutionType>, <AutoOpenMode>, <ModelExteriorAsIE>
VB Example	<pre> oDesign.SetSolutionType "Transient Network", Array("NAME:Options", "EnableAutoOpen:=", true, "BoundaryType:=", "Radiation") oDesign.SetSolutionType "SBR+", Array("NAME:Options", "EnableAutoOpen:=", false) oDesign.SetSolutionType "HFSS Terminal Composite", Array("NAME:Options", "EnableAutoOpen:=", false) </pre>

SetSolveInsideThreshold

Sets the solve inside threshold to the specified double.

UI Access	N/A
------------------	-----

Parameters	Name <i><threshold></i>	Type Double	Description Siemens/m.
Return Value	None.		

Python Syntax	SetSolveInsideThreshold(<i><threshold></i>)
Python Example	<code>oDesign.SetSolveInsideThreshold(100000)</code>

VB Syntax	SetSolveInsideThreshold <i><threshold></i>
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 <i><SourceId></i>	Type Array	Description 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	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><VarName></td> <td>String</td> <td>Variable name.</td> </tr> <tr> <td><VarValue></td> <td>Value</td> <td>New value for the variable.</td> </tr> </tbody> </table>	Name	Type	Description	<VarName>	String	Variable name.	<VarValue>	Value	New value for the variable.
Name	Type	Description								
<VarName>	String	Variable name.								
<VarValue>	Value	New value for the variable.								
Return Value	None.									

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

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	<table border="1"><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><SimulationNames></td><td>Array</td><td>Array containing string simulation names.</td></tr></table>	Name	Type	Description	<SimulationNames>	Array	Array containing string simulation names.
Name	Type	Description					
<SimulationNames>	Array	Array containing string simulation names.					
Return Value	<p>Integer:</p> <ul style="list-style-type: none">• 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.
Return Value	None.		

Python Syntax	StopSimLink(<simId>, <abort>)
Python Example	oDesign.StopSimLink(18, True)

VB Syntax	StopSimLink <simId>, <abort>
VB Example	oDesign.StopSimLink 18, true

Undo [Design]

Cancels the last design-level command.

UI Access	Edit > Undo
------------------	-------------

Parameters	None.
Return Value	None.

Python Syntax	Undo()
Python Example	<code>oDesign.Undo()</code>

VB Syntax	Undo
VB Example	<code>oDesign.Undo</code>

ValidateLink

Note:

This command is for internal Ansys use only.

Python Syntax	ValidateLink()
Python Example	<code>oDesign.ValidateLink()</code>

ValidateDesign

Returns whether a design is valid.

UI Access	Icepak > Validation Check.
Parameters	None.
Return Value	<p>Integer:</p> <ul style="list-style-type: none"> • 1 – Validation passed. • 0 – Validation failed.
Python Syntax	<code>ValidateDesign()</code>
Python Example	<code>oDesign.ValidateDesign()</code>
VB Syntax	<code>ValidateDesign</code>
VB Example	<code>oDesign.ValidateDesign</code>

ValidateLink

Note:

This command is for internal Ansys use only.

Python Syntax	<code>ValidateLink()</code>
Python Example	<code>oDesign.ValidateLink()</code>

This page intentionally
left blank.

10 - 3D Modeler Editor Script Commands

3D Modeler commands should be executed by the "3D Modeler" editor:

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

Conventions Used in this Chapter:

<AttributesArray>

<AttributesArray> takes the following structure:

```
Array("NAME:Attributes",
      "Name:=", <string>,
      "Flags:=", <string>,
      "Color:=", <string>,
      "Transparency:=", <value>,
      "PartCoordinateSystem:=", <string>,
      "UDMId:=", <string>,
      "MaterialValue:=", <string>,
      "SurfaceMaterialValue:=", <string>,
      "Solveinside:=", <boolean>,
      "ShellElement:=", <boolean>,
      "ShellElementThickness:=", <string>,
      "IsMaterialEditable:=", <boolean>,
```

```
"UseMaterialAppearance:=", <boolean>,  
"IsLightweight:=", <boolean>)
```

Where:

- **Flags** – Takes a string containing "NonModel" and/or "Wireframe", separated by the # character. For example, "NonModel#Wireframe".
- **Color** – Takes a string containing an RGB triple, formatted as "<RGB>". For example, "(255 255 255)".
- **Transparency** – Takes a value between 0 and 1.
- **PartCoordinateSystem** – Orientation of the primitive. The name of one of the defined coordinate systems should be specified.
- **UDMId** – Takes a string containing an ID.
- **MaterialValue** – Takes a string of the material name.
- **SurfaceMaterialValue** – Takes a string of the surface material name.
- **Solveinside** – Takes a boolean value.
- **ShellElement** – Takes a boolean value specifying whether or not a shell element is present.
- **ShellElementThickness** – Takes a string containing the shell element thickness. If element is not present, pass empty string.
- **IsMaterialEditable** – Takes a boolean value.
- **IsLightweight** – Takes a boolean value.

<SelectionsArray>

<SelectionsArray> typically takes the following structure:

```
Array ("NAME:Selections",  
"Selections:=", <string>)
```

Where:

- **Selections** – Takes a comma-separated list of parts on which to perform the action. For example, "Rect1, Rect2, Rect3".

In some cases, <SelectionsArray> takes additional parameters:

```
Array ("NAME:Selections",
      "AllowRegionDependentPartSelectionForPMLCreation:=", <boolean>,
      "AllowRegionSelectionForPMLCreation:=", <boolean>,
      "Selections:=", <string>,
      "NewPartsModelFlag:=", <string>,
      "UseCurrentCS:=", <boolean>)
```

Where:

- **AllowRegionDependentPartSelectionForPMLCreation** – Takes a boolean value. See individual script for whether this parameter is required.
- **AllowRegionSelectionForPMLCreation** – Takes a boolean value. See individual script for whether this parameter is required.
- **Selections** – Takes a comma-separated list of parts on which to perform the action. For example, "Rect1, Rect2, Rect3".
- **NewPartsModelFlag** – Takes either string "Model" or string "Nonmodel". See individual script for whether this parameter is required.
- **UseCurrentCS** – Takes a boolean value. See individual script for whether this parameter is required. Use [GetActiveCoordinateSystem](#) to determine the current CS.

Note:

Selections is the only parameter required in *all* 3D Modeler Editor scripts. See individual scripts for additional required parameters.

Organization

3D Modeler editor scripts are organized into the following categories:

[Draw Menu Commands](#)

[Edit Menu Commands](#)

[Modeler Menu Commands](#)

[Cable Modeling Commands](#)

[Other oEditor Commands](#)

Draw Menu Commands

[CreateBondwire](#)

[CreateBox](#)

[CreateCircle](#)

[CreateCone](#)

[CreateCutplane](#)

[CreateCylinder](#)

[CreateEllipse](#)

[CreateEquationCurve](#)

[CreateEquationSurface](#)

[CreateHelix](#)

[CreatePoint](#)

[CreatePolyline](#)

[CreateRectangle](#)

CreateRegion

[CreateRegularPolygon](#)

[CreateRegularPolyhedron](#)

CreateSphere

[CreateSpiral](#)

[CreateTorus](#)

[CreateUserDefinedPart](#)

[Edit3DComponent](#)

[EditPolyline](#)

[Get3DComponentDefinitionNames](#)

[Get3DComponentInstanceNames](#)

[Get3DComponentMaterialNames](#)

[Get3DComponentMaterialProperties](#)

[Get3DComponentParameters](#)

[Insert3DComponent](#)

[InsertPolylineSegment](#)

[SweepAlongPath](#)

[SweepAlongVector](#)

[SweepAroundAxis](#)

[SweepFacesAlongNormal](#)

[SweepFacesAlongNormalWithAttributes](#)

[UpdateComponentDefinition](#)**CreateBondwire**

Creates a bondwire.

UI Access	Draw > Bondwire.		
	Name	Type	Description
Parameters	<Parameters>	Array	<p>Structured array.</p> <pre>Array("NAME:BondwireParameters", "WireType:=", <string("JEDEC_4Points", "JEDEC_5Points", or "LOW")>, "WireDiameter:=", <string>, "NumSides:=", <value>, "XPadPos:=", <value>, "YPadPos:=", <value>, "ZPadPos:=", <value>, "XDir:=", <value>, "YDir:=", <value>, "ZDir:=", <value>, "Distance:=", <value>, "h1:=", <value>, "h2:=", <value>,</pre>

			<pre>"alpha:=", <value>, "beta:=", <value>, "WhichAxis:=", <string("X", "Y", or "Z")>, "ReverseDirection:=", <boolean>)</pre>
	<AttributesArray>	Array	Structured array. See: AttributesArray .
Return Value	None.		

Python Syntax	CreateBondwire(<Parameters>, <Attributes>)
Python Example	<pre>oEditor.CreateBondwire(["NAME:BondwireParameters", "WireType:=" , "JEDEC_4Points", "WireDiameter:=" , "0.025mm", "NumSides:=" , "6", "XPadPos:=" , "1.6mm", "YPadPos:=" , "-0.2mm", "ZPadPos:=" , "0mm", "XDir:=" , "-2.2mm", "YDir:=" , "-1.4mm", "ZDir:=" , "0mm", "Distance:=" , "2.60768096208106mm", "h1:=" , "0.2mm",</pre>

```
"h2:=" , "0mm",
"alpha:=" , "80deg",
"beta:=" , "0",
"WhichAxis:=" , "Z",
"ReverseDirection:=" , True
] ,
["NAME:Attributes",
 "Name:=" , "Bondwire1",
 "Flags:=" , "",
 "Color:=" , "(143 175 143)",
 "Transparency:=" , 0,
 "PartCoordinateSystem:=" , "Global",
 "UDMId:=" , "",
 "MaterialValue:=" , "\\"vacuum\\\"",
 "SurfaceMaterialValue:=" , "\\"\\\"",
 "SolveInside:=" , True,
 "ShellElement:=" , False,
 "ShellElementThickness:=" , "0mm",
 "IsMaterialEditable:=" , True,
 "UseMaterialAppearance:=" , False,
```

	"IsLightweight:=" , False])
--	---------------------------------

VB Syntax	CreateBondwire <Parameters>, <Attributes>
VB Example	<pre> oEditor.CreateBondwire Array("NAME:BondwireParameters", "WireType:=", "LOW", "WireDiameter:=", _ "0.025mm", "NumSides:=", "6", "XPadPos:=", "6mm", "YPadPos:=", "-2.5mm", "ZPadPos:=", _ "0mm", "XDir:=", "-10mm", "YDir:=", "4mm", "ZDir:=", "0mm", "Distance:=", _ "10.770329614269mm", "h1:=", "0.2mm", "h2:=", "0mm", "alpha:=", "80deg", "beta:=", _ "80deg", "WhichAxis:=", "Z", "ReverseDirection:=", false), Array("NAME:Attributes", "Name:=", _ "Bondwire2", "Flags:=", "", "Color:=", "(143 175 143)", "Transparency:=", 0, "PartCoordinateSystem:=", _ "Global", "UDMID:=", "", "MaterialValue:=", "" & Chr(34) & "copper" & Chr(34) & "", "SurfaceMaterialValue:=", _ "" & Chr(34) & "" & Chr(34) & "", "SolveInside:=", false, "ShellElement:=", _ false, "ShellElementThickness:=", "0mm", "IsMaterialEditable:=", true, "UseMaterialAppearance:=", _ false, "IsLightweight:=", false) </pre>

CreateBox

Creates a box.

UI Access	Draw > Box.		
Parameters	Name	Type	Description
	<Parameters>	Array	<p>Structured array.</p> <pre>Array ("NAME:BoxParameters", "XPosition:=", <string>, "YPosition:=", <string>, "ZPosition:=", <string>, "XSize:=" , <string>, "YSize:=" , <string>, "ZSize:=" , <string>)</pre>
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",</pre>

```
"YSize:=" , "1.5mm",
"ZSize:=" , "1.5mm"
],
[ "NAME:Attributes",
  "Name:=" , "Box3",
  "Flags:=" , "",
  "Color:=" , "(143 175 143)",
  "Transparency:=" , 0,
  "PartCoordinateSystem:=" , "Global",
  "UDMID:=" , "",
  "MaterialValue:=" , "\"copper\"",
  "SurfaceMaterialValue:=" , "\"\"",
  "SolveInside:=" , False,
  "ShellElement:=" , False,
  "ShellElementThickness:=" , "0mm",
  "IsMaterialEditable:=" , True,
  "UseMaterialAppearance:=" , False,
  "IsLightweight:=" , False
])
```

VB Syn- CreateBox <Parameters>, <Attributes>

tax	
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	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><Parameters></td> <td>Array</td> <td> <p>Structured array.</p> <p>Array ("NAME:CircleParameters", "IsCovered:=", <boolean>, "XCenter:=", <value>, "YCenter:=", <value>,</p> </td></tr> </tbody> </table>	Name	Type	Description	<Parameters>	Array	<p>Structured array.</p> <p>Array ("NAME:CircleParameters", "IsCovered:=", <boolean>, "XCenter:=", <value>, "YCenter:=", <value>,</p>		
Name	Type	Description							
<Parameters>	Array	<p>Structured array.</p> <p>Array ("NAME:CircleParameters", "IsCovered:=", <boolean>, "XCenter:=", <value>, "YCenter:=", <value>,</p>							

			<pre>"ZCenter:=", <value>, "Radius:=", <value>, "WhichAxis:=", <string> "NumSegments:=", <string containing integer>)</pre>
	<code><AttributesArray></code>	Array	Structured array. See: AttributesArray .
Return Value	None.		

Python Syntax	<code>CreateCircle(<Parameters>, <Attributes>)</code>
Python Example	<pre>oEditor.CreateCircle([["NAME:CircleParameters", "IsCovered:=" , True, "XCenter:=" , "5.5mm", "YCenter:=" , "-3mm", "ZCenter:=" , "0mm", "Radius:=" , "0.707106781186548mm", "WhichAxis:=" , "Z", "NumSegments:=" , "0"], ["NAME:Attributes", "Name:=" , "Circle1", "Flags:=" , ""]])</pre>

```

    "Color:="           , "(143 175 143)",
    "Transparency:="   , 0,
    "PartCoordinateSystem:=", "Global",
    "UDMID:="          , "",

    "MaterialValue:="   , "\"copper\",
    "SurfaceMaterialValue:=", "\",
    "SolveInside:="     , False,
    "ShellElement:="    , False,
    "ShellElementThickness:=", "0mm",
    "IsMaterialEditable:=" , True,
    "UseMaterialAppearance:=", False,
    "IsLightweight:="    , False
  ]
)

```

VB Syntax	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:=", "") </pre>

```
"Global", "UDMID:=", "", "MaterialValue:=", "" & Chr(34) & "copper" & Chr(34) & "", "SurfaceMaterialValue:=", _
"" & Chr(34) & "" & Chr(34) & "", "SolveInside:=", false, "ShellElement:=", _
false, "ShellElementThickness:=", "0mm", "IsMaterialEditable:=", true, "UseMaterialAppearance:=", _
false, "IsLightweight:=", false)
```

CreateCone

Creates a cone.

UI Access	Draw > Cone.		
Parameters	Name <i><Parameters></i>	Type Array	Description Structured array. Array("NAME:ConeParameters", "XCenter:=", <string>, "YCenter:=", <string>, "ZCenter:=", <string>, "WhichAxis:=", <string>, "Height:=", <string>, "BottomRadius:=", <string>, "TopRadius:=", <string>)
	<i><AttributesArray></i>	Array	Structured array. See: AttributesArray .
Return Value	None.		

Python Syntax	CreateCone(<Parameters>, <Attributes>)
Python Example	<pre> oEditor.CreateCone(["NAME:ConeParameters", "XCenter:=" , "3mm", "YCenter:=" , "-4.5mm", "ZCenter:=" , "0mm", "WhichAxis:=" , "Z", "Height:=" , "2.5mm", "BottomRadius:=" , "2.82842712474619mm", "TopRadius:=" , "2.23606797749979mm"], ["NAME:Attributes", "Name:=" , "Cone1", "Flags:=" , "", "Color:=" , "(143 175 143)", "Transparency:=" , 0, "PartCoordinateSystem:=" , "Global", "UDMID:=" , "", "MaterialValue:=" , "\"copper\"", "SurfaceMaterialValue:=" , "\"\""]) </pre>

	<pre> "SolveInside:=" , False, "ShellElement:=" , False, "ShellElementThickness:=", "0mm", "IsMaterialEditable:=" , True, "UseMaterialAppearance:=", False, "IsLightweight:=" , False]) </pre>
--	---

VB Syntax	CreateCone <Parameters>, <Attributes>
VB Example	<pre> oEditor.CreateCone Array("NAME:ConeParameters", "XCenter:=", "3mm", "YCenter:=", _ "ZCenter:=", "0mm", "WhichAxis:=", "Z", "Height:=", "4mm", "BottomRadius:=", _ "1.11803398874989mm", "TopRadius:=", "2.06155281280883mm"), Array("NAME:Attributes", "Name:=", _ "Cone1", "Flags:=", "", "Color:=", "(143 175 143)", "Transparency:=", 0, "PartCoordinateSystem:=", _ "Global", "UDMID:=", "", "MaterialValue:=", "" & Chr(34) & "copper" & Chr(34) & "",_ "SurfaceMaterialValue:=", _ "" & Chr(34) & "" & Chr(34) & "", "SolveInside:=", false, "ShellElement:=", _ false, "ShellElementThickness:=", "0mm", "IsMaterialEditable:=", true, "UseMaterialAppearance:=", _ false, "IsLightweight:=", false) </pre>

CreateCutplane

Creates a cutplane.

UI Access	Draw > Plane.		
Parameters	Name <i><Parameters></i>	Type Array	Description Structured array. Array ("NAME:PlaneParameters", "PlaneBaseX:=", <string>, "PlaneBaseY:=", <string>, "PlaneBaseZ:=", <string>, "PlaneNormalX:=", <string>, "PlaneNormalY:=", <string>), "PlaneNormalZ:=", <string>)
	<i><AttributesArray></i>	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",</pre>

```

    "PlaneBaseY:="           , "-0.8mm",
    "PlaneBaseZ:="           , "0mm",
    "PlaneNormalX:="         , "1.2mm",
    "PlaneNormalY:="         , "0.2mm",
    "PlaneNormalZ:="         , "0mm"
],
[ "NAME:Attributes",
  "Name:="                 , "Plane1",
  "Color:="                , "(143 175 143)"
])

```

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	<p>Structured array.</p> <pre>Array ("NAME:CylinderParameters", "XCenter:=", <string>, "YCenter:=", <string>, "ZCenter:=", <string>, "Radius:=", <string>, "Height:=", <string>, "WhichAxis:=", <string>, "NumSides:=", <string containing integer>)</pre>
	<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",</pre>

```
"Radius:=" , "0.5mm",
"Height:=" , "4.5mm",
"WhichAxis:=" , "Z",
"NumSides:=" , "0"
],
["NAME:Attributes",
 "Name:=" , "Cylinder1",
 "Flags:=" , "",
 "Color:=" , "(143 175 143)",
 "Transparency:=" , 0,
 "PartCoordinateSystem:=", "Global",
 "UDMID:=" , "",
 "MaterialValue:=" , "\"copper\"",
 "SurfaceMaterialValue:=" , "\"\"",
 "SolveInside:=" , False,
 "ShellElement:=" , False,
 "ShellElementThickness:=" , "0mm",
 "IsMaterialEditable:=" , True,
 "UseMaterialAppearance:=" , False,
 "IsLightweight:=" , False
])
```

VB Syntax	CreateCylinder <Parameters>, <Attributes>
VB Example	<pre> oEditor.CreateCylinder Array("NAME:CylinderParameters", "XCenter:=", "1.5mm", "YCenter:=", _ "6.5mm", "ZCenter:=", "0mm", "Radius:=", "1mm", "Height:=", "1mm", "WhichAxis:=", _ "Z", "NumSides:=", "0"), Array("NAME:Attributes", "Name:=", "Cylinder2", "Flags:=", _ "", "Color:=", "(143 175 143)", "Transparency:=", 0, "PartCoordinateSystem:=", _ "Global", "UDMID:=", "", "MaterialValue:=", "" & 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>

CreateEllipse

Creates an ellipse.

UI Access	Draw > Ellipse.		
Parameters	Name <Parameters>	Type Array	Description Structured array. Array ("NAME:EllipseParameters", "IsCovered:=", <string>,

			<pre>"XCenter:=", <string>, "YCenter:=", <string>, "ZCenter:=", <string>, "MajRadius:=", <string>, "Ratio:=", <string>, "WhichAxis:=", <string>, "NumSegments:=", <string>)</pre>
	<code><AttributesArray></code>	Array	Structured array. See: AttributesArray .
Return Value	None.		

Python Syntax	<code>CreateEllipse(<Parameters>, <Attributes>)</code>
Python Example	<pre>oEditor.CreateEllipse(["NAME:EllipseParameters", "IsCovered:=" , True, "XCenter:=" , "0.6mm", "YCenter:=" , "-0.6mm", "ZCenter:=" , "0mm", "MajRadius:=" , "0.2mm", "Ratio:=" , "7", "WhichAxis:=" , "Z", "NumSegments:=" , "0"])</pre>

```
        ],
        [
            "NAME:Attributes",
            "Name:=" , "Ellipse1",
            "Flags:=" , "",
            "Color:=" , "(143 175 143)",
            "Transparency:=" , 0,
            "PartCoordinateSystem:=", "Global",
            "UDMID:=" , "",
            "MaterialValue:=" , "\"copper\"",
            "SurfaceMaterialValue:=" , "\"\"",
            "SolveInside:=" , False,
            "ShellElement:=" , False,
            "ShellElementThickness:=" , "0mm",
            "IsMaterialEditable:=" , True,
            "UseMaterialAppearance:=" , False,
            "IsLightweight:=" , False
        ]
    )
```

**VB
Syntax**

```
CreateEllipse <Parameters>, <Attributes>
```

VB Example	<pre> oEditor.CreateEllipse Array("NAME:EllipseParameters", "IsCovered:=", true, "XCenter:=", - "-0.4mm", "YCenter:=", "-3.2mm", "ZCenter:=", "0mm", "MajRadius:=", "0.4mm", "Ratio:=", - "0.5", "WhichAxis:=", "Z", "NumSegments:=", "0"), Array("NAME:Attributes", "Name:=", - "Ellipse2", "Flags:=", "", "Color:=", "(143 175 143)", "Transparency:=", 0, "PartCoordinateSystem:=", - "Global", "UDMID:=", "", "MaterialValue:=", "" & 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	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 25%;">Name</th> <th style="width: 25%;">Type</th> <th style="width: 50%;">Description</th> </tr> </thead> <tbody> <tr> <td style="height: 150px;"><i><Parameters></i></td> <td style="height: 150px;">Array</td> <td> <p>Structured array.</p> <pre> Array("NAME:EquationBasedCurveParameters", "XtFunction:=", <string>, "YtFunction:=", <string>, "ZtFunction:=", <string>, "tStart:=", <string>, </pre> </td> </tr> </tbody> </table>	Name	Type	Description	<i><Parameters></i>	Array	<p>Structured array.</p> <pre> Array("NAME:EquationBasedCurveParameters", "XtFunction:=", <string>, "YtFunction:=", <string>, "ZtFunction:=", <string>, "tStart:=", <string>, </pre>		
Name	Type	Description							
<i><Parameters></i>	Array	<p>Structured array.</p> <pre> Array("NAME:EquationBasedCurveParameters", "XtFunction:=", <string>, "YtFunction:=", <string>, "ZtFunction:=", <string>, "tStart:=", <string>, </pre>							

			<pre>"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", "tEnd:=" , "3", "NumOfPointsOnCurve:=" , "0", "Version:=" , 1, ["NAME:PolylineXSection", "XSectionType:=" , "None", "XSectionOrient:=" , "Auto",</pre>

```
        "XSectionWidth:="      , "0",
        "XSectionTopWidth:="   , "0",
        "XSectionHeight:="    , "0",
        "XSectionNumSegments:=" , "0",
        "XSectionBendType:="   , "Corner"
    ]
],
[ "NAME:Attributes",
  "Name:="                  , "EquationCurve1",
  "Flags:="                 , "",
  "Color:="                 , "(143 175 143)",
  "Transparency:="          , 0,
  "PartCoordinateSystem:="   , "Global",
  "UDMID:="                 , "",
  "MaterialValue:="         , "\"copper\"",
  "SurfaceMaterialValue:="   , "\"\"",
  "SolveInside:="            , False,
  "ShellElement:="           , False,
  "ShellElementThickness:="  , "0mm",
  "IsMaterialEditable:="     , True,
  "UseMaterialAppearance:="  , False,
```

	"IsLightweight:=" , False])
--	---------------------------------

VB Syntax	CreateEquationCurve <Parameters>, <Attributes>
VB Example	<pre> oEditor.CreateEquationCurve Array("NAME:EquationBasedCurveParameters", "XtFunction:=", _ "1", "YtFunction:=", "3", "ZtFunction:=", "32", "tStart:=", "1", "tEnd:=", "3", "NumOfPointsOnCurve:=", _ "0", "Version:=", 1, Array("NAME:PolylineXSection", "XSectionType:=", "None", "XSectionOrient:=", _ "Auto", "XSectionWidth:=", "0", "XSectionTopWidth:=", "0", "XSectionHeight:=", _ "0", "XSectionNumSegments:=", "0", "XSectionBendType:=", "Corner")), Array("NAME:Attributes", "Name:=", _ "EquationCurve2", "Flags:=", "", "Color:=", "(143 175 143)", "Transparency:=", _ 0, "PartCoordinateSystem:=", "Global", "UDMID:=", "", "MaterialValue:=", _ "" & Chr(34) & "copper" & Chr(34) & "", "SurfaceMaterialValue:=", "" & Chr(34) & "" & Chr(34) & "", "SolveInside:=", _ false, "ShellElement:=", false, "ShellElementThickness:=", "0mm", "IsMaterialEditable:=", _ true, "UseMaterialAppearance:=", false, "IsLightweight:=", false) </pre>

CreateEquationSurface

Creates an equation-based surface.

UI Access	Draw > Equation-Based Surface.		
Parameters	Name	Type	Description
	<Parameters>	Array	<p>Structured array.</p> <pre>Array ("NAME:EquationBasedSurfaceParameters", "XuvFunction:=" , <string equation containing Function, Operators and/or quantities _u, _v, or PI>, "YuvFunction:=" , <string equation containing Function, Operators and/or quantities _u, _v, or PI>, "ZuvFunction:=" , <string equation containing Function, Operators and/or quantities _u, _v, or PI>, "uStart:=" , <string>, "uEnd:=" , <string>, "vStart:=" , <string>, "vEnd:=" , <string>, "Version:=" , <integer>)</pre>
Return Value	None.		

Python Syntax	CreateEquationSurface(<Parameters>, <Attributes>)
Python Example	<pre>oEditor.CreateEquationSurface (["NAME:EquationBasedSurfaceParameters", "XuvFunction:=" , "_u", "YuvFunction:=" , "_v",</pre>

```
"ZuvFunction:=" , "sin(_u)+cos(_v)",
"uStart:=" , "1",
"uEnd:=" , "10",
"vStart:=" , "1",
"vEnd:=" , "10",
"Version:=" , 1
],
["NAME:Attributes",
 "Name:=" , "EquationSurface1",
 "Flags:=" , "",
 "Color:=" , "(143 175 143)",
 "Transparency:=" , 0,
 "PartCoordinateSystem:=" , "Global",
 "UDMID:=" , "",
 "MaterialValue:=" , "\"copper\"",
 "SurfaceMaterialValue:=" , "\"\"",
 "SolveInside:=" , False,
 "ShellElement:=" , False,
 "ShellElementThickness:=" , "0mm",
 "IsMaterialEditable:=" , True,
```

```

    "UseMaterialAppearance:=", False,
    "IsLightweight:=", False
])

```

VB Syntax	CreateEquationSurface <Parameters>, <Attributes>
VB Example	<pre> oEditor.CreateEquationSurface Array("NAME:EquationBasedSurfaceParameters", "XuvFunction:=", _ "_u", "YuvFunction:=", "_v", "ZuvFunction:=", "sin(_u)+cos(_v)", "uStart:=", "1", "uEnd:=", _ "10", "vStart:=", "1", "vEnd:=", "10", "Version:=", 1), Array("NAME:Attributes", "Name:=", _ "EquationSurface2", "Flags:=", "", "Color:=", "(143 175 143)", "Transparency:=", _ 0, "PartCoordinateSystem:=", "Global", "UDMID:=", "", "MaterialValue:=", _ "" & Chr(34) & "copper" & Chr(34) & "", "SurfaceMaterialValue:=", "" & Chr(34) & "" & 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.

	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: SelectionsArray .
Parameters	<Parameters>	Array	<p>Structured array.</p> <pre>Array("NAME:HelixParameters", "XCenter:=" , <string>, "YCenter:=" , <string>, "ZCenter:=" , <string>, "XStartDir:=" , <string>, "YStartDir:=" , <string>, "ZStartDir:=" , <string>, "NumThread:=" , <string>, "RightHand:=" , <boolean>, "RadiusIncrement:=" , <string>, "Thread:=" , <string>)</pre>
Return Value	None.		

Python Syntax	CreateHelix(<SelectionsArray>, <Parameters>)
Python Example	<pre>oEditor.CreateHelix (["NAME:Selections", "Selections:=" , "EquationSurface2",</pre>

```

    "NewPartsModelFlag:=", "Model"
],
[ "NAME:HelixParameters",
    "XCenter:=", "10000mm",
    "YCenter:=", "40000mm",
    "ZCenter:=", "0mm",
    "XStartDir:=", "0mm",
    "YStartDir:=", "10000mm",
    "ZStartDir:=", "0mm",
    "NumThread:=", "1",
    "RightHand:=", True,
    "RadiusIncrement:=", "0mm",
    "Thread:=", "1mm"
])

```

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:=", _ </pre>

```
"-2.39945573144418mm", "NumThread:=", "1", "RightHand:=", true, "RadiusIncrement:=", _  
"0mm", "Thread:=", "1mm")
```

CreatePoint

Creates a point.

UI Access	Draw > Point.		
Parameters	Name <i><Parameters></i>	Type Array	Description Structured array. Array ("NAME:PointParameters", "PointX:=", <value>, "PointY:=", <value>, "PointZ:=", <value>)
	<i><AttributesArray></i>	Array	Structured array. See: AttributesArray . CreatePoint takes only the Name and Color attributes.
Return Value	None.		

Python Syntax	CreatePoint(<i><Parameters></i> , <i><Attributes></i>)
Python Example	<pre>oEditor.CreatePoint (["NAME:PointParameters", "PointX:=" , "0.2mm",</pre>

```

    "PointY:=" , "-0.2mm",
    "PointZ:=" , "0mm"
],
[ "NAME:Attributes",
  "Name:=" , "Point1",
  "Color:=" , "(143 175 143)"
])

```

VB Syntax	CreatePoint <Parameters>, <Attributes>
VB Example	<pre> oEditor.CreatePoint Array("NAME:PointParameters", "PointX:=" , "0.2mm", "PointY:=" , "-0.2mm", "PointZ:=" , "0mm") , Array("NAME:Attributes", "Name:=" , "Point1", "Color:=" , "(143 175 143)")) </pre>

CreatePolyline

Creates a polyline.

UI Access	Draw > Line.		
	Name	Type	Description
Parameters	<Parameters>	Array	<p>Structured array.</p> <pre>Array ("NAME:PolylineParameters", "IsPolylineCovered:=", <bool>, "IsPolylineClosed:=", <bool>, <PolylinePointsArray>, <PolylineSegmentsArray>)</pre>
	<PolylinePointsArray>	Array	<pre>Array ("NAME:PolylinePoints", <OnePointArray>, <OnePointArray>, ...)</pre>
	<OnePointArray>	Array	<pre>Array ("NAME:PLPoint", "X:=", <value>, "Y:=", <value>, "Z:=", <value>))</pre>
	<PolylineSegmentsArray>	Array	<pre><PolylineSegmentsArray> Array ("NAME:PolylineSegments", <OneSegmentArray>, <OneSegmentArray>, ...)</pre>
	<OneSegmentArray>	Array	<pre>Array ("NAME:PLSegment", "SegmentType:=", <"Line", "Arc", "Spline", or "AngularArc">, "StartIndex:=", <value>, "NoOfPoints:=", <value>)</pre>

	<code><AttributesArray></code>	Array	Structured array. See: AttributesArray .
Return Value	None.		

Python Syntax	<code>CreatePolyline(<Parameters>, <Attributes>)</code>
	<pre> oEditor.CreatePolyline(["NAME:PolylineParameters", "IsPolylineCovered:=" , True, "IsPolylineClosed:=" , False, ["NAME:PolylinePoints", ["NAME:PLPoint", "X:=" , "20000mm", "Y:=" , "-20000mm", "Z:=" , "0mm"], ["NAME:PLPoint", "X:=" , "-90000mm", "Y:=" , "20000mm", "Z:=" , "0mm"], ["NAME:PLPoint", "X:=" , "10000mm", "Y:=" , "10000mm"]]]) </pre>
Python Example	<pre> oEditor.CreatePolyline(["NAME:PolylineParameters", "IsPolylineCovered:=" , True, "IsPolylineClosed:=" , False, ["NAME:PolylinePoints", ["NAME:PLPoint", "X:=" , "20000mm", "Y:=" , "-20000mm", "Z:=" , "0mm"], ["NAME:PLPoint", "X:=" , "-90000mm", "Y:=" , "20000mm", "Z:=" , "0mm"], ["NAME:PLPoint", "X:=" , "10000mm", "Y:=" , "10000mm"]]]) </pre>

```
        "Y:="                  , "-14000mm",
        "Z:="                  , "0mm"
    ]
],
[ "NAME:PolylineSegments",
  [ "NAME:PLSegment",
    "SegmentType:="      , "Line",
    "startIndex:="       , 0,
    "noOfPoints:="       , 2
  ],
  [ "NAME:PLSegment",
    "SegmentType:="      , "Line",
    "startIndex:="       , 1,
    "noOfPoints:="       , 2
  ]
],
[ "NAME:PolylineXSection",
  "xSectionType:="      , "None",
  "xSectionOrient:="     , "Auto",
  "xSectionWidth:="      , "0mm",
```

```
"XSectionTopWidth:=", "0mm",
"XSectionHeight:=", "0mm",
"XSectionNumSegments:=", "0",
"XSectionBendType:=", "Corner"
]] ,
["NAME:Attributes",
 "Name:=", "Polyline1",
 "Flags:=", "",
 "Color:=", "(143 175 143)",
 "Transparency:=", 0,
 "PartCoordinateSystem:=", "Global",
 "UDMID:=", "",
 "MaterialValue:=", "\copper\",
 "SurfaceMaterialValue:=", "\",
 "SolveInside:=", False,
 "ShellElement:=", False,
 "ShellElementThickness:=", "0mm",
 "IsMaterialEditable:=", True,
 "UseMaterialAppearance:=", False,
 "IsLightweight:=", False
])
```

VB Syntax	CreatePolyline <Parameters>, <Attributes>
VB Example:	<pre> oEditor.CreatePolyline Array("NAME:PolylineParameters", "IsPolylineCovered:=", true, "IsPolylineClosed:=", _ false, Array("NAME:PolylinePoints", Array("NAME:PLPoint", "X:=", "40000mm", "Y:=", _ "50000mm", "Z:=", "0mm"), Array("NAME:PLPoint", "X:=", "-150000mm", "Y:=", "-50000mm", "Z:=", _ "0mm")), Array("NAME:PolylineSegments", Array("NAME:PLSegment", "SegmentType:=", "Line", "StartIndex:=", _ 0, "NoOfPoints:=", 2)), Array("NAME:PolylineXSection", "XSectionType:=", "None", "XSectionOrient:=", _ "Auto", "XSectionWidth:=", "0mm", "XSectionTopWidth:=", "0mm", "XSectionHeight:=", _ "0mm", "XSectionNumSegments:=", "0", "XSectionBendType:=", "Corner")), Array("NAME:Attributes", "Name:=", _ "Polyline2", "Flags:=", "", "Color:=", "(143 175 143)", "Transparency:=", 0, "PartCoordinateSystem:=", _ "Global", "UDMID:=", "", "MaterialValue:=", "" & 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>

CreateRectangle

Creates a rectangle.

UI Access	Draw > Rectangle.		
Parameters	Name <i><Parameters></i>	Type Array	Description Structured array. Array ("NAME:RectangleParameters", "IsCovered:=" , <boolean>, "XStart:=" , <string>, "YStart:=" , <string>, "ZStart:=" , <string>, "Width:=" , <string>, "Height:=" , <string>, "WhichAxis:=" , <string "X", "Y", or "Z">)
	<i><AttributesArray></i>	Array	Structured array. See: AttributesArray .
Return Value	None.		

Python Syntax	CreateRectangle(<Parameters>, <Attributes>)
Python Example	<pre>oEditor.CreateRectangle(["NAME:RectangleParameters", "IsCovered:=" , True, "XStart:=" , "-80000mm",</pre>

```
"YStart:=" , "-90000mm",
"ZStart:=" , "0mm",
"Width:=" , "20000mm",
"Height:=" , "30000mm",
"WhichAxis:=" , "Z"
],
[ "NAME:Attributes",
  "Name:=" , "Rectangle1",
  "Flags:=" , "",
  "Color:=" , "(143 175 143)",
  "Transparency:=" , 0,
  "PartCoordinateSystem:=" , "Global",
  "UDMId:=" , "",
  "MaterialValue:=" , "\"copper\"",
  "SurfaceMaterialValue:=" , "\"\"",
  "SolveInside:=" , False,
  "ShellElement:=" , False,
  "ShellElementThickness:=" , "0mm",
  "IsMaterialEditable:=" , True,
  "UseMaterialAppearance:=" , False,
```

	"IsLightweight:=" , False])
--	---------------------------------

VB Syntax	CreateRectangle <Parameters>, <Attributes>
VB Example	<pre> oEditor.CreateRectangle Array("NAME:RectangleParameters", "IsCovered:=", true, "XStart:=", _ "10000mm", "YStart:=", "-40000mm", "ZStart:=", "0mm", "Width:=", "40000mm", "Height:=", - "10000mm", "WhichAxis:=", "Z"), Array("NAME:Attributes", "Name:=", "Rectangle2", "Flags:", "", "Color:=", "(143 175 143)", "Transparency:=", 0, "PartCoordinateSystem:=", _ "Global", "UDMID:=", "", "MaterialValue:=", "" & Chr(34) & "copper" & Chr(34) & "", "SurfaceMaterialValue:=", _ "" & Chr(34) & "" & Chr(34) & "", "SolveInside:=", false, "ShellElement:=", _ false, "ShellElementThickness:=", "0mm", "IsMaterialEditable:=", true, "UseMaterialAppearance:", _, false, "IsLightweight:=", false) </pre>

CreateRegularPolygon

Creates a regular polygon.

UI Access	Draw > Regular Polygon.		
Parameters	Name <Parameters>	Type Array	Description Structured array.

			<pre>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")</pre>
	<code><AttributesArray></code>	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",</pre>

```
"ZCenter:=" , "0mm",
"XStart:=" , "-50000mm",
"YStart:=" , "-80000mm",
"ZStart:=" , "0mm",
"NumSides:=" , "12",
"WhichAxis:=" , "Z"
],
["NAME:Attributes",
 "Name:=" , "Polygon1",
 "Flags:=" , "",
 "Color:=" , "(143 175 143)",
 "Transparency:=" , 0,
 "PartCoordinateSystem:=" , "Global",
 "UDMID:=" , "",
 "MaterialValue:=" , "\"copper\",
"SurfaceMaterialValue:=" , "\",
"SolveInside:=" , False,
"ShellElement:=" , False,
"ShellElementThickness:=" , "0mm",
"IsMaterialEditable:=" , True,
"UseMaterialAppearance:=" , False,
```

	"IsLightweight:=" , False])
--	---------------------------------

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- terialAppearance:=", _ false, "IsLightweight:=", false) </pre>

CreateRegularPolyhedron

Creates a regular polyhedron.

UI Access	Draw > Regular Polyhedron.
------------------	--------------------------------------

	Name	Type	Description
Parameters	<i><Parameters></i>	Array	<p>Structured array.</p> <pre>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">)</pre>
	<i><AttributesArray></i>	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",</pre>

```
"XStart:=" , "50000mm",
"YStart:=" , "-70000mm",
"ZStart:=" , "0mm",
"Height:=" , "50000mm",
"NumSides:=" , "8",
"WhichAxis:=" , "Z"
],
[ "NAME:Attributes",
  "Name:=" , "RegularPolyhedron1",
  "Flags:=" , "",
  "Color:=" , "(143 175 143)",
  "Transparency:=" , 0,
  "PartCoordinateSystem:=" , "Global",
  "UDMID:=" , "",
  "MaterialValue:=" , "\"copper\"",
  "SurfaceMaterialValue:=" , "\"\"",
  "SolveInside:=" , False,
  "ShellElement:=" , False,
  "ShellElementThickness:=" , "0mm",
  "IsMaterialEditable:=" , True,
```

```

    "UseMaterialAppearance:=", False,
    "IsLightweight:=", False
])

```

VB Syntax	CreateRegularPolyhedron <Parameters>, <Attributes>
VB Example	<pre> oEditor.CreateRegularPolyhedron Array("NAME:PolyhedronParameters", "XCenter:=", "-10000mm", "YCenter:=", "-50000mm", "ZCenter:=", "0mm", "XStart:=", "0mm", "YStart:=", "-50000mm", "ZStart:=", "0mm", "Height:=", "90000mm", "NumSides:=", "8", "WhichAxis:=", "Z"), Array("NAME:Attributes", "Name:=", "RegularPolyhedron2", "Flags:=", "", "Col- or:=", _ "(143 175 143)", "Transparency:=", 0, "PartCoordinateSystem:=", "Global", "UDMID:=", "", "MaterialValue:=", "" & Chr(34) & "copper" & Chr(34) & "", "Sur- faceMaterialValue:=", _ "" & Chr(34) & "" & Chr(34) & "", "SolveInside:=", false, "ShellElement:=", false, "ShellElementThickness:=", "0mm", "IsMaterialEditable:=", true, "UseMa- terialAppearance:=", _ false, "IsLightweight:=", false) </pre>

CreateSpiral

Creates a spiral by sweeping the specified object(s).

UI Access	Draw > Spiral.		
	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: SelectionsArray .
Parameters	<Parameters>	Array	<p>Structured array.</p> <pre>Array ("NAME:SpiralParameters", "XCenter:=" , <string>, "YCenter:=" , <string>, "ZCenter:=" , <string>, "YStartDir:=" , <string>, "ZStartDir:=" , <string>, "NumThread:=" , <string>, "RightHand:=" , <boolean>, "RadiusIncrement:=" , <string>)</pre>
Return Value	None.		

Python Syntax	CreateSpiral(<Parameters>, <Attributes>)
Python Example	<pre>oEditor.CreateSpiral(["NAME:Selections", "Selections:=" , "Polygon2", "NewPartsModelFlag:=" , "Model"</pre>

```
],
["NAME:SpiralParameters",
 "XCenter:=" , "-70000mm",
 "YCenter:=" , "50000mm",
 "ZCenter:=" , "0mm",
 "XStartDir:=" , "-60000mm",
 "YStartDir:=" , "-10000mm",
 "ZStartDir:=" , "0mm",
 "NumThread:=" , "1",
 "RightHand:=" , True,
 "RadiusIncrement:=" , "1mm"
])
```

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 <i><Parameters></i>	Type Array	Description Structured array. Array ("NAME:TorusParameters", "XCenter:=" , <string>, "YCenter:=" , <string>, "ZCenter:=" , <string>, "MajorRadius:=" , <string>, "MinorRadius:=" , <string>, "WhichAxis:=" , <string "X", "Y", or "Z">)
	<i><AttributesArray></i>	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:=" , "0.1mm", "MajorRadius:=" , "10mm", "MinorRadius:=" , "2mm", "WhichAxis:=" , "X")]) </pre>

```
"ZCenter:=" , "0mm",
"MajorRadius:=" , "0.365028153987289mm",
"MinorRadius:=" , "0.0821854415126694mm",
"WhichAxis:=" , "Z"
],
[ "NAME:Attributes",
  "Name:=" , "Torus1",
  "Flags:=" , "",
  "Color:=" , "(143 175 143)",
  "Transparency:=" , 0,
  "PartCoordinateSystem:=" , "Global",
  "UDMID:=" , "",
  "MaterialValue:=" , "\"copper\"",
  "SurfaceMaterialValue:=" , "\\"",
  "SolveInside:=" , False,
  "ShellElement:=" , False,
  "ShellElementThickness:=" , "0mm",
  "IsMaterialEditable:=" , True,
  "UseMaterialAppearance:=" , False,
  "IsLightweight:=" , False
])
```

VB Syntax	CreateTorus <Parameters>, <Attributes>
VB Example	<pre> oEditor.CreateTorus Array("NAME:TorusParameters", "XCenter:=", "0.6mm", "YCenter:=", _ "-2mm", "ZCenter:=", "0mm", "MajorRadius:=", "0.365028153987288mm", "MinorRadius:=", _ "0.0821854415126694mm", "WhichAxis:=", "Z"), Array("NAME:Attributes", "Name:=", _ "Torus2", "Flags:=", "", "Color:=", "(143 175 143)", "Transparency:=", 0, "PartCoordinateSystem:=", _ "Global", "UDMID:=", "", "MaterialValue:=", "" & Chr(34) & "copper" & Chr(34) & "", "SurfaceMaterialValue:=", _ "" & Chr(34) & "" & Chr(34) & "", "SolveInside:=", false, "ShellElement:=", _ false, "ShellElementThickness:=", "0mm", "IsMaterialEditable:=", true, "UseMaterialAppearance:=", _ false, "IsLightweight:=", false) </pre>

CreateUserDefinedModel

Creates a user-defined model.

UI Access	Draw > User-Defined Model > [Model].		
Parameters	Name <Parameters>	Type Array	Description Structured array. Array ("NAME:UserDefinedModelParameters", <definitionArray>,

			<pre> <optionsArray>, <geometryParamsArray>, "DllName:=", <string filepath>, "Library:=", <string>, "Version:=", <string> "ConnectionID:=", <string> </pre>
<definitionArray>	Array	Structured array containing string "NAME:Definition".	
<optionsArray>	Array	Structured array containing string "NAME:Options".	
<geometryParamsArray>	Array	<p>Structured array containing arrays for individual parameters:</p> <pre> Array("NAME:GeometryParameters", Array("NAME:UDMParam", "Name:=" , <string>, "Value:=" , <string>, "PropType2:=" , <integer>, "PropFlag2:=" , <integer>)) </pre> <p>Required UDM parameters depend on the UDM being created. To see which properties apply to a UDM, right-click the UDM in the Project Tree and select Properties. Then select the Parameters tab.</p> <p>PropType2 can be any of the following:</p> <ul style="list-style-type: none"> • 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). 	

		<ul style="list-style-type: none"> • 4 – Property is a file name. • 5 – Property corresponds to a check box. • 6 – Property specifies a 3D position. <p>PropFlag2 can be any of the following:</p> <ul style="list-style-type: none"> • 0 – No flags • 1 – Read-only • 2 – Must be integer • 4 – Must be real • 8 – Hidden <p>PropFlag2 values can be combined. For example, a read-only property that must be an integer would take the value 3. A hidden property that must be real would take the value 12.</p> <p>These values are further described in the <code>User-DefinedPrimitiveStructures.h</code> file included with the installation under <code>AnsysEM[Version]\Win64\UserDefinedPrimitives\Examples\Headers\</code></p>
Return Value	None.	

Python Syntax	CreateUserDefinedModel(<Parameters>)
Python Example	

VB Syntax	CreateUserDefinedModel <Parameters>
------------------	-------------------------------------

VB Example	
-------------------	--

CreateUserDefinedPart

Creates a user-defined part.

UI Access	Draw > User-Defined Primitive > [Part].		
Parameters	Name	Type	Description
	<Parameters>	Array	<p>Structured array.</p> <pre>Array("NAME:UserDefinedPrimitiveParameters", "DllName:=" , <string>, "Version:=" , <string>, "NoOfParameters:=" , <integer>, "Library:=" , <string>, <paramVectorArray>)</pre>
	<paramVectorArray>	Array	<p>Structured array containing arrays for each pair:</p> <pre>Array("NAME:ParamVector", <pair>, <pair>, <pair>,...)</pre>
	<pair>	Array	<p>Structured array:</p> <pre>Array("NAME:Pair", "Name:=" , <string>, "Value:=" , <string>)</pre>
	<Attributes>	Array	Structured array. See: AttributesArray .
Return Value	None.		

Python Syntax	CreateUserDefinedPart(<Parameters>, <Attributes>)
Python Example	<pre>oEditor.CreateUserDefinedPart(["NAME:UserDefinedPrimitiveParameters", "DllName:=" , "RMxprt/LapCoil.dll", "Version:=" , "16.0", "NoOfParameters:=" , 22, "Library:=" , "syslib", ["NAME:ParamVector", ["NAME:Pair", "Name:=" , "DiaGap", "Value:=" , "100mm"], ["NAME:Pair", "Name:=" , "DiaYoke", "Value:=" , "20mm"], ["NAME:Pair", "Name:=" , "Length", "Value:=" , "100mm"]</pre>

```
],  
    ["NAME:Pair",  
        "Name:=" , "Skew",  
        "Value:=" , "0deg"  
    ],  
    ["NAME:Pair",  
        "Name:=" , "Slots",  
        "Value:=" , "18"  
    ],  
    ["NAME:Pair",  
        "Name:=" , "SlotType",  
        "Value:=" , "1"  
    ],  
    ["NAME:Pair",  
        "Name:=" , "Hs0",  
        "Value:=" , "1mm"  
    ],  
    ["NAME:Pair",  
        "Name:=" , "Hs1",  
        "Value:=" , "1mm"  
    ],
```

```
[ "NAME:Pair",
  "Name:=" , "Hs2",
  "Value:=" , "10mm"
] ,
[ "NAME:Pair",
  "Name:=" , "Bs0",
  "Value:=" , "2.5mm"
] ,
[ "NAME:Pair",
  "Name:=" , "Bs1",
  "Value:=" , "8mm"
] ,
[ "NAME:Pair",
  "Name:=" , "Bs2",
  "Value:=" , "5mm"
] ,
[ "NAME:Pair",
  "Name:=" , "Rs",
  "Value:=" , "0mm"
] ,
```

```
[ "NAME:Pair",
  "Name:=" , "FilletType",
  "Value:=" , "0"
] ,
[ "NAME:Pair",
  "Name:=" , "Layers",
  "Value:=" , "2"
] ,
[ "NAME:Pair",
  "Name:=" , "CoilPitch",
  "Value:=" , "4"
] ,
[ "NAME:Pair",
  "Name:=" , "EndExt",
  "Value:=" , "5mm"
] ,
[ "NAME:Pair",
  "Name:=" , "SpanExt",
  "Value:=" , "25mm"
] ,
[ "NAME:Pair",
```

```
        "Name:=" , "BendAngle",
        "Value:=" , "0deg"
    ] ,
    [ "NAME:Pair",
        "Name:=" , "SegAngle",
        "Value:=" , "10deg"
    ] ,
    [ "NAME:Pair",
        "Name:=" , "LenRegion",
        "Value:=" , "200mm"
    ] ,
    [ "NAME:Pair",
        "Name:=" , "InfoCoil",
        "Value:=" , "0"
    ]
]
],
[ "NAME:Attributes",
    "Name:=" , "LapCoil1",
    "Flags:=" , "",
```

```

    "Color:="           , "(143 175 143)",
    "Transparency:="   , 0,
    "PartCoordinateSystem:=", "Global",
    "UDMID:="          , "", 
    "MaterialValue:="   , "\"copper\"",
    "SurfaceMaterialValue:=", "\\"",
    "SolveInside:="     , False,
    "ShellElement:="    , False,
    "ShellElementThickness:=", "0mm",
    "IsMaterialEditable:=" , True,
    "UseMaterialAppearance:=", False,
    "IsLightweight:="    , False
  ])
)

```

VB Syntax	CreateUserDefinedPart <Parameters>, <Attributes>
VB Example	<pre> oEditor.CreateUserDefinedPart Array("NAME:UserDefinedPrimitiveParameters", "DllName:=", - "SegmentedHelix/RectHelix.dll", "Version:=", "1.0", "NoOfParameters:=", 8, "Library:=", - "syslib", Array("NAME:ParamVector", Array("NAME:Pair", "Name:=", "RectHeight", "Value:=", _ "1mm"), Array("NAME:Pair", "Name:=", "RectWidth", "Value:=", "2mm"), Array("NAME:Pair", </pre>

```

"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.

UI Access	N/A		
Parameters	Name	Type	Description
	<compName>	String	Component name.
	<Parameters>	Array	Structured array.

			<pre>Array ("NAME>EditComponentParametersData", "NewComponentName:=", <string>, "GeometryParameters:=", <string>, "MaterialParameters:=", <string>, "DesignParameters:=", <string>, <ComponentMeshing>, <Excitations>)</pre>
	<i><ComponentMeshing></i>	Array	<p>Structured array.</p> <pre>Array ("NAME">Component Meshing", "MeshAssembly:=", <boolean>)</pre>
	<i><Excitations></i>	Array	<p>Structured array containing array of suppressed excitations.</p> <pre>Array ("NAME">Excitations", "Suppressed:=", <array>)</pre>
Return Value	None.		

Python Syntax	Edit3DComponent (<compName>, <Parameters>)
Python Example	<pre>oEditor.Edit3DComponent("Connector1", ["NAME>EditComponentParametersData", "NewComponentName:=", "Connector2", "GeometryParameters:=", "", "MaterialParameters:=", ""],</pre>

```
    "DesignParameters:=", "",  
    [ "NAME:Component Meshing",  
      "MeshAssembly:=", False],  
    [ "NAME:Excitations",  
      "Suppressed:=", []]  
  ]  
)
```

VB Syntax	Edit3DComponent <compName>, <Parameters>
VB Example	<pre>oEditor.Edit3DComponent "Connector1", Array("NAME:EditComponentParametersData", "NewComponentName:=", "Connector2", "GeometryParameters:=", "", "MaterialParameters:=", "", "DesignParameters:=", "", Array("NAME:Component Meshing", "MeshAssembly:=", false), Array("NAME:Excitations", "Suppressed:=", Array()))</pre>

)
--	---

Edit3DComponentDefinition

Edits definitions of a specified 3D component.

UI Access	N/A		
Parameters	Name <i><Parameters></i>	Type Array	Description 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", </pre>

```

    "GeometryParameters:=", "",  

    "MaterialParameters:=", "",  

    "DesignParameters:=", "",  

    Array("NAME:Component Meshing",  

          "MeshAssembly:=", false),  

    Array("NAME:Excitations",  

          "Suppressed:=", Array())  

)

```

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",

			<pre>"X:=", <value>, "Y:=", <value>, "Z:=", <value>)) <PolylineSegmentsArray></pre>
	<pre><OneSegmentArray></pre>	Array	<pre><PolylineSegmentsArray> Array("NAME:PolylineSegments", <OneSegmentArray>, <OneSegmentArray>, ...)</pre>
	<pre><OneSegmentArray></pre>	Array	<pre>Array("NAME:PLSegment", "SegmentType:=", <"Line", "Arc", "Spline", or "AngularArc">, "StartIndex:=", <value>, "NoOfPoints:=", <value>)</pre>
Return Value	None.		

Python Syntax	EditPolyline(<SelectionsArray>, <Parameters>)
Python Example	<pre>oEditor.EditPolyline(["NAME:Selections", "Selections:=", "Polyline1"] ["NAME:PolylineParameters", "IsPolylineCovered:=", True, "IsPolylineClosed:=", False, ["NAME:PolylinePoints",</pre>

```
[ "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-
tax**

EditPolyline <SelectionsArray>, <Parameters>

VB Example	<pre> oEditor>EditPolyline Array("NAME:Selections", "Selections:=", "Polyline1") Array ("NAME:PolylineParameters", "IsPolylineCovered:=", true, "IsPolylineClosed:=", _ false, Array("NAME:PolylinePoints", Array("NAME:PLPoint", "X:=", "40000mm", "Y:=", _ "50000mm", "Z:=", "0mm"), Array("NAME:PLPoint", "X:=", "-150000mm", "Y:=", "-50000mm", "Z:=", _ "0mm")), Array("NAME:PolylineSegments", Array("NAME:PLSegment", "SegmentType:=", "Line", "StartIndex:=", _ 0, "NoOfPoints:=", 2)), Array("NAME:PolylineXSection", "XSectionType:=", "None", "XSec- tionOrient:=", _ "Auto", "XSectionWidth:=", "0mm", "XSectionTopWidth:=", "0mm", "XSectionHeight:=", _ "0mm", "XSectionNumSegments:=", "0", "XSectionBendType:=", "Corner")), Array("NAME:At- tributes", "Name:=", _ "Polyline2", "Flags:=", "", "Color:=", "(143 175 143)", "Transparency:=", 0, "PartCoordi- nateSystem:=", _ "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	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td><i><DefinitionName></i></td><td>String</td><td>Definition name.</td></tr></tbody></table>		Name	Type	Description	<i><DefinitionName></i>	String	Definition name.
Name	Type	Description						
<i><DefinitionName></i>	String	Definition name.						
Return Value	Array containing instance names.							

Python Syntax	Get3DComponentInstanceNames(<DefinitionName>)
Python Example	<code>oEditor.Get3DComponentInstanceNames ("Connector")</code>

VB Syntax	Get3DComponentInstanceNames < <i>DefinitionName</i> >
VB Example	<code>oEditor.Get3DComponentInstanceNames "Connector"</code>

Get3DComponentMaterialNames

Returns material names for a specified *.a3dcomp format 3D component.

UI Access	N/A		
Parameters	Name < <i>InstanceName</i> >	Type String	Description Component instance name.
Return Value	Array containing material names.		

Python Syntax	Get3DComponentMaterialNames(< <i>InstanceName</i> >)
Python Example	<code>oEditor.Get3DComponentMaterialNames ("Connector1.a3dcomp")</code>

VB Syntax	Get3DComponentMaterialNames < <i>InstanceName</i> >		
VB Example	<code>oEditor.Get3DComponentMaterialNames "Connector1.a3dcomp"</code>		

Get3DComponentMaterialProperties

Returns material properties for a specified 3D component.

UI Access	N/A		
Parameters	Name < <i>MaterialName</i> >	Type String	Description 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	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td><compName></td><td>String</td><td>3D component name.</td></tr></tbody></table>	Name	Type	Description	<compName>	String	3D component name.
Name	Type	Description					
<compName>	String	3D component name.					
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	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><code><InstanceName></code></td> <td>String</td> <td>3D component instance.</td> </tr> </tbody> </table>	Name	Type	Description	<code><InstanceName></code>	String	3D component instance.
Name	Type	Description					
<code><InstanceName></code>	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	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><code><ComponentData></code></td> <td>Array</td> <td>Structured array. <code>Array ("NAME:InsertComponentData", "Parameters:=", <string>,</code></td> </tr> </tbody> </table>	Name	Type	Description	<code><ComponentData></code>	Array	Structured array. <code>Array ("NAME:InsertComponentData", "Parameters:=", <string>,</code>
Name	Type	Description					
<code><ComponentData></code>	Array	Structured array. <code>Array ("NAME:InsertComponentData", "Parameters:=", <string>,</code>					

			"TargetCS:=", <string>, "ComponentFile:=", <string filepath>)
Return Value	None.		

Python Syntax	Insert3DComponent(< <i>ComponentData</i> >)
Python Example	<pre>oEditor.Insert3DComponent(["NAME:InsertComponentData", "Parameters:=", "", "TargetCS:=", "Global", "ComponentFile:=", "C:\\tmp\\Connector.a3dcomp"])</pre>

VB Syntax	Insert3DComponent < <i>ComponentData</i> >
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	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><code><ComponentData></code></td> <td>Array</td> <td> <p>Structured array.</p> <pre>Array ("NAME:InsertComponentData", "Parameters:=", <string>, "TargetCS:=", <string>, "ComponentFile:=", <string filepath>)</pre> </td> </tr> </tbody> </table>	Name	Type	Description	<code><ComponentData></code>	Array	<p>Structured array.</p> <pre>Array ("NAME:InsertComponentData", "Parameters:=", <string>, "TargetCS:=", <string>, "ComponentFile:=", <string filepath>)</pre>
Name	Type	Description					
<code><ComponentData></code>	Array	<p>Structured array.</p> <pre>Array ("NAME:InsertComponentData", "Parameters:=", <string>, "TargetCS:=", <string>, "ComponentFile:=", <string filepath>)</pre>					
Return Value	None.						

Python Syntax	<code>InsertComponent(<ComponentData>)</code>
Python Example	<pre>oEditor.InsertComponent(["NAME:InsertComponentData", "Parameters:=", "", "TargetCS:=", "Global", "ComponentFile:=", "C:\tmp\Connector.a3dcomp"])</pre>

VB Syntax	<code>InsertComponent <ComponentData></code>
VB Example	<pre>oEditor.InsertComponent Array ("NAME:InsertComponentData", "Parameters:=", "", "TargetCS:=", "Global",</pre>

	"ComponentFile:=", "C:\tmp\Connector.a3dcomp")
--	--

InsertPolylineSegment

Insets a polyline segment before or after a specified existing segment.

UI Access	Draw > Line Segment > [Selection].		
Parameters	Name <i><Parameters></i>	Type Array	Description Structured array. Array("NAME:Insert Polyline Segment", "Selections:=" , <string>, "Segment Indices:=" , <array containing integers>, "At Start:=" , <boolean>, "SegmentType:=" , <string "Line", "Arc", "Spline", or "AngularArc">, <PolylinePointsArray>)
	<i><PolylinePointsArray></i>	Array	Structured array. See: CreatePolyline .
Return Value	None.		

Python Syntax	InsertPolylineSegment(<i><Parameters></i>)
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 <Parameters>
VB Example	<pre> oEditor.InsertPolylineSegment Array("NAME:Insert Polyline Segment", "Selections:=", "Polyline1>CreatePolyline:1", "Segment Indices:=", Array(1), "At Start:=", _</pre>

```

    false, "SegmentType:=", "Spline", Array("NAME:PolylinePoints", Array("NAME:PLPoint",
    "X:=", _  

    "-1.4mm", "Y:=", "0.4mm", "Z:=", "0mm"), Array("NAME:PLPoint", "X:=", "0.5mm", "Y:=",
    _  

    "1.1mm", "Z:=", "0mm"), Array("NAME:PLPoint", "X:=", "0.7mm", "Y:=", "-2.1mm", "Z:=",
    _  

    "0mm"), Array("NAME:PLPoint", "X:=", "0.4mm", "Y:=", "-1.1mm", "Z:=", "0mm")))
)

```

SweepAlongPath

Sweeps the specified 1D or 2D parts along a path. The last 1D object specified is the path for the sweep.

UI Access	Draw > Sweep > Along Path.		
Parameters	Name <i><SelectionsArray></i>	Type Array	Description Structured array. See: SelectionsArray .
	<i><PathSweepParametersArray></i>	Array	Array ("NAME:PathSweepParameters", "DraftAngle:=", <value>, "DraftType:=", <string>, "CheckFaceFaceIntersection:=", <bool>, "TwistAngle:=", <value>) Possible values for DraftType are "Extended", "Round", and "Natural".
Return Value	None.		

Python Syntax	SweepAlongPath(<SelectionsArray>, <PathSweepParametersArray>)
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>

```

    "DraftAngle:=", "0deg",
    "DraftType:=", "Round",
    "CheckFaceFaceIntersection:=", False,
    "TwistAngle:=", "30deg")

```

SweepAlongVector

Sweeps the specified 1D or 2D parts along a vector.

UI Access	Draw > Sweep > Along Vector.											
	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><SelectionsArray></td> <td>Array</td> <td>Structured array. See: SelectionsArray.</td> </tr> <tr> <td><VecSweepParametersArray></td> <td>Array</td> <td> Array ("NAME:VectorSweepParameters", "DraftAngle:=", <value>, "DraftType:=", <string>, "CheckFaceFaceIntersection:=", <bool>, "SweepVectorX:=", <value> "SweepVectorY:=", <value> "SweepVectorZ:=", <value>) </td> </tr> </tbody> </table> <p>Possible values for DraftType are "Extended", "Round", and "Natural".</p>	Name	Type	Description	<SelectionsArray>	Array	Structured array. See: SelectionsArray .	<VecSweepParametersArray>	Array	Array ("NAME:VectorSweepParameters", "DraftAngle:=", <value>, "DraftType:=", <string>, "CheckFaceFaceIntersection:=", <bool>, "SweepVectorX:=", <value> "SweepVectorY:=", <value> "SweepVectorZ:=", <value>)		
Name	Type	Description										
<SelectionsArray>	Array	Structured array. See: SelectionsArray .										
<VecSweepParametersArray>	Array	Array ("NAME:VectorSweepParameters", "DraftAngle:=", <value>, "DraftType:=", <string>, "CheckFaceFaceIntersection:=", <bool>, "SweepVectorX:=", <value> "SweepVectorY:=", <value> "SweepVectorZ:=", <value>)										
Parameters												
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>

```

"Selections:=", "Rectangle1",_
"NewPartsModelFlag:=", "Model")

Array ("NAME:VectorSweepParameters", _
"DraftAngle:=", "0deg",_
"DraftType:=", "Round",_
"CheckFaceFaceIntersection:=", False,_
"SweepVectorX:=", "0mm",_
"SweepVectorY:=", "0mm",_
"SweepVectorZ:=", "12mm")

```

SweepAroundAxis

Sweeps the specified 1D or 2D parts around an axis.

UI Access	Draw > Sweep > Around Axis.		
	Name	Type	Description
Parameters	<SelectionsArray> <AxisSweepParametersArray>	Array Array	Structured array. See: SelectionsArray . Array ("NAME:AxisSweepParameters", "DraftAngle:=", <value>, "DraftType:=", <string>, "CheckFaceFaceIntersection:=", <bool>, "SweepAxis:=", <value> "SweepAngle:=", <value>

		<p>"NumOfSegments:=" , <value>)</p> <p>Possible values for DraftType are "Extended", "Round", and "Natural".</p> <p>Possible values for SweepAxis are "X", "Y", and "Z".</p>
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>

])
--	----

VB Syntax	SweepAroundAxis <SelectionsArray>, <AxisSweepParametersArray>
VB Example	<pre> oEditor.SweepAroundAxis Array("NAME:Selections",_ "Selections:=", "Rectangle1",_ "NewPartsModelFlag:=", "Model") Array("NAME:AxisSweepParameters", _ "DraftAngle:=", "0deg",_ "DraftType:=", "Round",_ "CheckFaceFaceIntersection:=", False,_ "SweepAxis:=", "X",_ "SweepAngle:=", "360deg",_ "NumOfSegments:=", "12") </pre>

SweepFacesAlongNormal

Sweep the specified face(s) along normal.

UI Access	Modeler > Surface > Sweep Faces Along Normal		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: SelectionsArray .

	<code><parameters></code>	Array	Structured array. Array ("NAME:Parameters", "NAME:SweepFaceAlongNormalToParameters", "FacesToDetach:=", <faceIDarray>, "LengthOfSweep:=", "<value><units>")
Return Value	None		

Python Syntax	<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>

```

        Array ("NAME:Selections",
            "Selections:=", "Rectangle1",
            "NewPartsModelFlag:=", "Model"),
        Array ("NAME:Parameters",
            "NAME:SweepFaceAlongNormalToParameters",
            "FacesToDetach:=", Array(57),
            "LengthOfSweep:=", "0.5mm")
    )
)

```

SweepFacesAlongNormalWithAttributes

Sweep a face along normal, and specify attributes of the new object.

UI Access	Modeler > Surface > Sweep Faces Along Normal		
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>

```

    "IsMaterialEditable:=", True,
    "UseMaterialAppearance:=", False,
    "IsLightweight:=", False])

```

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>

```

    "UDMID:=" , "",  

    "MaterialValue:=" , "\"copper\"",  

    "SurfaceMaterialValue:=", "\"\"",  

    "SolveInside:=" , false,  

    "ShellElement:=" , false,  

    "ShellElementThickness:=" , "0mm",  

    "IsMaterialEditable:=" , true,  

    "UseMaterialAppearance:=" , false,  

    "IsLightweight:=" , false)
)

```

UpdateComponentDefinition

Updates a 3D component's definition.

UI Access	Draw > 3D Component Library > Definitions.		
Parameters	Name	Type	Description
	<data>	Array	<p>Structured array.</p> <pre>Array ("NAME:UpdateDefinitionData", "DefinitionNames:=" , <string>, "Passwords:=" , <array of strings>)</pre>
Return Value	None.		

Python Syntax	UpdateComponentDefinition(<data>)
---------------	-----------------------------------

Python Example

```
oEditor.UpdateComponentDefinition(  
    [ 'NAME:UpdateDefinitionData',  
        'DefinitionNames:=' , 'Connector, Magic_Tee',  
        'Passwords:=' , [ '' , '' ]  
    ]  
)
```

VB Syntax

UpdateComponentDefinition <data>

VB Example

```
oEditor.UpdateComponentDefinition  
Array("NAME:UpdateDefinitionData",  
    "DefinitionNames:=" , " Connector, Magic_Tee",  
    "Passwords:=" , Array("", ""))
```

Edit Menu Commands

[Copy](#)

[DeletePolylinePoint](#)

[DuplicateAlongLine](#)

[DuplicateAroundAxis](#)

[DuplicateMirror](#)

[Mirror](#)

[Move](#)

[OffsetFaces](#)[Paste](#)[Rotate](#)[Scale](#)

Copy

Copies specified part(s) to the clipboard.

UI Access	N/A		
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 <code><DeletePointArray></code>	Type Array	Description Structured array. <code>Array ("NAME:Delete Point", "Selections:=", <string "PolylineName>:<PolylineAction>:<int>>, "Segment Index:=", <integer>, "At Start:=", <bool True for start point; False for end point>)</code>
Return Value	None.		

Python Syntax	<code>DeletePolylinePoint (<DeletePointArray>)</code>
Python Example	<pre>oEditor.DeletePolylinePoint(["NAME:Delete Point", "Selections:=", "Polyline1>CreatePolyline:1", "Segment Index:=", 1, "At Start:=", True])</pre>

VB Syn-tax	<code>DeletePolylinePoint <DeletePointArray></code>
------------	---

VB Example	<pre><code>oEditor.DeletePolylinePoint Array("NAME:Delete Point", "Selections:=", "Polyline1>CreatePolyline:1", "Segment Index:=", 1, "At Start:=", True)</code></pre>
-------------------	---

DuplicateAlongLine

Duplicates specified parts along a line.

UI Access	Edit > Duplicate > Along Line.														
Parameters	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 20%;">Name</th> <th style="width: 20%;">Type</th> <th style="width: 60%;">Description</th> </tr> </thead> <tbody> <tr> <td style="height: 100px;"><code><SelectionsArray></code></td><td style="height: 100px;">Array</td><td>Structured array. <code>Array ("NAME:Selections", "Selections:=" , <string>, "NewPartsModelFlag:=" , <string>)</code></td></tr> <tr> <td style="height: 100px;"><code><ParametersArray></code></td><td style="height: 100px;">Array</td><td>Structured array. <code>Array ("NAME:DuplicateToAlongLineParameters", "CreateNewObjects:=" , <boolean>, "XComponent:=" , <string>, "YComponent:=" , <string>, "ZComponent:=" , <string>, "NumClones:=" , <string containing number greater than 1>)</code></td></tr> <tr> <td style="height: 100px;"><code><OptionsArray></code></td><td style="height: 100px;">Array</td><td>Structured array. <code>Array ("NAME:Options", "DuplicateAssignments:=" , <boolean>)</code></td></tr> </tbody> </table>	Name	Type	Description	<code><SelectionsArray></code>	Array	Structured array. <code>Array ("NAME:Selections", "Selections:=" , <string>, "NewPartsModelFlag:=" , <string>)</code>	<code><ParametersArray></code>	Array	Structured array. <code>Array ("NAME:DuplicateToAlongLineParameters", "CreateNewObjects:=" , <boolean>, "XComponent:=" , <string>, "YComponent:=" , <string>, "ZComponent:=" , <string>, "NumClones:=" , <string containing number greater than 1>)</code>	<code><OptionsArray></code>	Array	Structured array. <code>Array ("NAME:Options", "DuplicateAssignments:=" , <boolean>)</code>		
Name	Type	Description													
<code><SelectionsArray></code>	Array	Structured array. <code>Array ("NAME:Selections", "Selections:=" , <string>, "NewPartsModelFlag:=" , <string>)</code>													
<code><ParametersArray></code>	Array	Structured array. <code>Array ("NAME:DuplicateToAlongLineParameters", "CreateNewObjects:=" , <boolean>, "XComponent:=" , <string>, "YComponent:=" , <string>, "ZComponent:=" , <string>, "NumClones:=" , <string containing number greater than 1>)</code>													
<code><OptionsArray></code>	Array	Structured array. <code>Array ("NAME:Options", "DuplicateAssignments:=" , <boolean>)</code>													

	<code><CreateGroup></code>	Array	Optional. Structured array. <code>Array("CreateGroupsForNewObjects:=", <boolean>)</code>
Return Value	None.		

Python Syntax	<code>DuplicateAlongLine (<SelectionsArray>, <ParametersArray>, <OptionsArray>, <CreateGroup>)</code>
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>

```
],
["CreateGroupsForNewObjects:=", False
])
```

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 <code><SelectionsArray></code>	Type Array	Description Structured array. Array ("NAME:Selections", "Selections:=" , <string>, "NewPartsModelFlag:=" , <string>)
	<code><ParametersArray></code>	Array	Structured array. Array ("NAME:DuplicateAroundAxisParameters", "CreateNewObjects:=" , <boolean>, "WhichAxis:=" , <string>, "AngleStr:=" , <string>, "NumClones:=" , <string containing number greater than 1>)
	<code><OptionsArray></code>	Array	Structured array. Array ("NAME:Options", "DuplicateAssignments:=" , <boolean>)
	<code><CreateGroup></code>	Array	Optional. Structured array. Array ("CreateGroupsForNewObjects:=" , <boolean>)
Return Value	None.		

Python Syntax	DuplicateAroundAxis (<SelectionsArray>, <ParametersArray>, <OptionsArray>, <CreateGroup>)
Python Example	<pre> oEditor.DuplicateAroundAxis(["NAME:Selections", "Selections:=" , "Box1", "NewPartsModelFlag:=" , "Model"], ["NAME:DuplicateAroundAxisParameters", "CreateNewObjects:=" , True, "WhichAxis:=" , "Z", "AngleStr:=" , "90deg", "NumClones:=" , "2"], ["NAME:Options", "DuplicateAssignments:=", False], ["CreateGroupsForNewObjects:=", False]) </pre>

VB Syntax	DuplicateAroundAxis <SelectionsArray>, <ParametersArray>, <OptionsArray>, <CreateGroup>
VB Example	<pre> oEditor.DuplicateAroundAxis Array("NAME:Selections", </pre>

```

    "Selections:=", "Box1",
    "NewPartsModelFlag:=", "Model")

Array("NAME:DuplicateAroundAxisParameters",
      "CreateNewObjects:=", true,
      "WhichAxis:=", "Z",
      "AngleStr:=", "90deg",
      "NumClones:=", "2")

Array("NAME:Options",
      "DuplicateAssignments:=", false)

Array("CreateGroupsForNewObjects:=", false)

```

DuplicateMirror

Duplicates specified parts according to a mirror plane.

UI Access	Edit > Duplicate > Mirror.		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	<p>Structured array.</p> <pre>Array("NAME:Selections", "Selections:=" , <string>, "NewPartsModelFlag:=" , <string>)</pre>
	<ParametersArray>	Array	<p>Structured array.</p> <pre>Array("NAME:DuplicateToMirrorParameters",</pre>

			<pre>"DuplicateMirrorBaseX:=", <string>, "DuplicateMirrorBaseY:=", <string>, "DuplicateMirrorNormalX:=", <string>, "DuplicateMirrorNormalY:=", <string>, "DuplicateMirrorNormalZ:=", <string>)</pre>
	<i><OptionsArray></i>	Array	<p>Structured array.</p> <pre>Array("NAME:Options", "DuplicateAssignments:=", <boolean>)</pre>
	<i><CreateGroup></i>	Array	<p>Optional. Structured array.</p> <pre>Array("CreateGroupsForNewObjects:=", <boolean>)</pre>
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", "DuplicateMirrorNormalX:=", "0.0mm", "DuplicateMirrorNormalY:=", "0.0mm", "DuplicateMirrorNormalZ:=", "1.0mm"])</pre>

```
"DuplicateMirrorBaseZ:=", "0mm",
"DuplicateMirrorNormalX:=", "0.124034734589208mm",
"DuplicateMirrorNormalY:=", "0.992277876713668mm",
"DuplicateMirrorNormalZ:=", "0mm"
],
[ "NAME:Options",
  "DuplicateAssignments:=", False
],
[ "CreateGroupsForNewObjects:=", False
])
```

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",</pre>

```

    "DuplicateMirrorBaseZ:=", "0mm",
    "DuplicateMirrorNormalX:=", "0.124034734589208mm",
    "DuplicateMirrorNormalY:=", "0.992277876713668mm",
    "DuplicateMirrorNormalZ:=", "0mm")

Array("NAME:Options",
      "DuplicateAssignments:=", false)

Array("CreateGroupsForNewObjects:=", false)

```

Mirror

Mirrors specified part(s).

UI Access	Edit > Arrange > Mirror.											
	<table border="1"> <thead> <tr> <th>Name</th><th>Type</th><th>Description</th></tr> </thead> <tbody> <tr> <td><SelectionsArray></td><td>Array</td><td>Structured array. See: SelectionsArray.</td></tr> <tr> <td><MirrorParameters></td><td>Array</td><td> Structured array. Array ("NAME:MirrorParameters", "MirrorBaseX:=" , <string>, "MirrorBaseY:=" , <string>, "MirrorBaseZ:=" , <string>, "MirrorNormalX:=" , <string>, "MirrorNormalY:=" , <string>, "MirrorNormalZ:=" , <string>) </td></tr> </tbody> </table>	Name	Type	Description	<SelectionsArray>	Array	Structured array. See: SelectionsArray .	<MirrorParameters>	Array	Structured array. Array ("NAME:MirrorParameters", "MirrorBaseX:=" , <string>, "MirrorBaseY:=" , <string>, "MirrorBaseZ:=" , <string>, "MirrorNormalX:=" , <string>, "MirrorNormalY:=" , <string>, "MirrorNormalZ:=" , <string>)		
Name	Type	Description										
<SelectionsArray>	Array	Structured array. See: SelectionsArray .										
<MirrorParameters>	Array	Structured array. Array ("NAME:MirrorParameters", "MirrorBaseX:=" , <string>, "MirrorBaseY:=" , <string>, "MirrorBaseZ:=" , <string>, "MirrorNormalX:=" , <string>, "MirrorNormalY:=" , <string>, "MirrorNormalZ:=" , <string>)										
Parameters												
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",</pre>

	<pre> "NewPartsModelFlag:=", "Model") Array("NAME:MirrorParameters", "MirrorBaseX:=", "-0.2mm", "MirrorBaseY:=", "-1.2mm", "MirrorBaseZ:=", "0mm", "MirrorNormalX:=", "-0.316227766016838mm", "MirrorNormalY:=", "0.948683298050514mm", "MirrorNormalZ:=", "0mm") </pre>

Move

Moves specified part(s).

UI Access	Edit > Arrange > Move.											
Parameters	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><SelectionsArray></td> <td>Array</td> <td>Structured array. See: SelectionsArray.</td> </tr> <tr> <td><TranslateParameters></td> <td>Array</td> <td>Structured array. Array (["NAME:TranslateParameters", "TranslateVectorX:=" , <string>, "TranslateVectorY:=" , <string>, "TranslateVectorZ:=" , <string>) </td> </tr> </tbody> </table>	Name	Type	Description	<SelectionsArray>	Array	Structured array. See: SelectionsArray .	<TranslateParameters>	Array	Structured array. Array (["NAME:TranslateParameters", "TranslateVectorX:=" , <string>, "TranslateVectorY:=" , <string>, "TranslateVectorZ:=" , <string>)		
Name	Type	Description										
<SelectionsArray>	Array	Structured array. See: SelectionsArray .										
<TranslateParameters>	Array	Structured array. Array (["NAME:TranslateParameters", "TranslateVectorX:=" , <string>, "TranslateVectorY:=" , <string>, "TranslateVectorZ:=" , <string>)										
Return Value	None.											

Python Syntax	Move(<SelectionsArray>, <TranslateParameters>)
Python Example	<pre> oEditor.Move(["NAME:Selections", "Selections:=" , "Box1_1", "NewPartsModelFlag:=" , "Model"], ["NAME:TranslateParameters", "TranslateVectorX:=" , "-0.5mm", "TranslateVectorY:=" , "0.1mm", "TranslateVectorZ:=" , "0mm"]) </pre>

VB Syntax	Move <SelectionsArray> <TranslateParameters>
VB Example	<pre> oEditor.Move Array("NAME:Selections", "Selections:=", "Box1_1", "NewPartsModelFlag:=", "Model") Array("NAME:TranslateParameters", "TranslateVectorX:=", "-0.5mm", "TranslateVectorY:=", "0.1mm", "TranslateVectorZ:=" , "0mm")) </pre>

	"TranslateVectorZ:=", "0mm")
--	------------------------------

OffsetFaces

Offsets the faces of selected part(s).

UI Access	Edit > Arrange > Offset.		
Parameters	Name <i><SelectionsArray></i>	Type Array	Description Structured array. See: SelectionsArray .
	<i><OffsetParameters></i>	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	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><SelectionsArray></td> <td>Array</td> <td>Structured array. See: SelectionsArray.</td> </tr> <tr> <td><RotateParameters></td> <td>Array</td> <td>Structured array. Array ("NAME:RotateParameters", "RotateAxis:=", <string "X", "Y", or "Z">, "RotateAngle:=", <string>)</td> </tr> </tbody> </table>	Name	Type	Description	<SelectionsArray>	Array	Structured array. See: SelectionsArray .	<RotateParameters>	Array	Structured array. Array ("NAME:RotateParameters", "RotateAxis:=", <string "X", "Y", or "Z">, "RotateAngle:=", <string>)
Name	Type	Description								
<SelectionsArray>	Array	Structured array. See: SelectionsArray .								
<RotateParameters>	Array	Structured array. Array ("NAME:RotateParameters", "RotateAxis:=", <string "X", "Y", or "Z">, "RotateAngle:=", <string>)								
Return Value	None.									

Python Syntax	Rotate(<SelectionsArray>, <RotateParameters>)
Python Example	<pre>oEditor.Rotate(["NAME:Selections", "Selections:=", "Box1_1", "NewPartsModelFlag:=", "Model"],</pre>

```
[ "NAME:RotateParameters",
  "RotateAxis:=", "Z",
  "RotateAngle:=", "90deg"
])
```

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>, "ScaleY:=" , <string containing scale factor>, "ScaleZ:=" , <string containing scale factor>)
Return Value	None.		

Python Syntax	Scale (<SelectionsArray>, <ScaleParameters>)
Python Example	<pre> oEditor.Scale(["NAME:Selections", "Selections:=", "Box1", "NewPartsModelFlag:=", "Model"], ["NAME:ScaleParameters", "ScaleX:=", "2", "ScaleY:=", "2", "ScaleZ:=", "2"]) </pre>

VB Syntax	Scale <SelectionsArray> <ScaleParameters>
VB Example	<pre> oEditor.Scale Array("NAME:Selections", "Selections:=", "Box1",) </pre>

```
"NewPartsModelFlag:=", "Model")
Array("NAME:ScaleParameters",
      "ScaleX:=", "2",
      "ScaleY:=", "2",
      "ScaleZ:=", "2")
```

Modeler Menu Commands

[AssignMaterial](#)

[Chamfer](#)

[Connect](#)

[CoverLines](#)

[CoverSurfaces](#)

[CreateEntityList](#)

[CreateFaceCS](#)

[CreateGroup](#)

[CreateObjectCS](#)

[CreateObjectFromEdges](#)

[CreateObjectFromFaces](#)

[CreateRelativeCS](#)

[DeleteEmptyGroups](#)

[DeleteLastOperation](#)

[DetachFaces](#)
[EditEntityList](#)
[EditFaceCS](#)
[EditObjectCS](#)
[EditRelativeCS](#)
[Export](#)
[ExportModelImageToFile](#)
[ExportModelMeshToFile](#)
[Fillet](#)
[FlattenGroup](#)
[Generate History](#)
[GetActiveCoordinateSystem](#)
[GetCoordinateSystems](#)
[HealObject](#)
[Import](#)
[ImportDXF](#)
[ImportGDSII \[Modeler\]](#)
[Intersect](#)
[MoveCStoEnd](#)
[MoveEntityToGroup](#)
[MoveFaces](#)

[ProjectSheet](#)

[PurgeHistory](#)

[ReplaceWith3DComponent](#)

[Section](#)

[SeparateBody](#)

[SetModelUnits](#)

[SetWCS](#)

[ShowWindow](#)

[Split](#)

[Subtract](#)

[SweepFacesAlongNormal](#)

[ThickenSheet](#)

[UncoverFaces](#)

[Unite](#)

[Ungroup](#)

[WrapSheet](#)

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: <ul style="list-style-type: none">• MaterialValue• SolveInside• ShellElement• ShellElementThickness• IsMaterialEditable• UseMaterialAppearance• IsLightweight
Return Value	None.		

Python Syntax	AssignMaterial(<SelectionsArray>, <AttributesArray>)
---------------	--

```
oEditor.AssignMaterial(  
    ["NAME:Selections",  
        "AllowRegionDependentPartSelectionForPMLCreation:=", True,  
        "AllowRegionSelectionForPMLCreation:=", True,  
        "Selections:=", "Box1"  
    ],  
    ["NAME:Attributes",  
        "MaterialValue:=", "diamond",  
        "SolveInside:=", False,  
        "ShellElement:=", False,  
        "ShellElementThickness:=", "nan",  
        "IsMaterialEditable:=", True,  
        "UseMaterialAppearance:=", False,  
        "IsLightweight:=", False  
    ])
```

VB Syntax

AssignMaterial <SelectionsArray> <AttributesArray>

VB Example

```

oEditor.AssignMaterial

    Array("NAME:Selections",
          "AllowRegionDependentPartSelectionForPMLCreation:=", true,
          "AllowRegionSelectionForPMLCreation:=", true,
          "Selections:=" , "Box1")

    Array("NAME:Attributes",
          "MaterialValue:=" , "diamond",
          "SolveInside:=" , false,
          "ShellElement:=" , false,
          "ShellElementThickness:=" , "nan",
          "IsMaterialEditable:=" , true,
          "UseMaterialAppearance:=" , false,
          "IsLightweight:=" , false)

```

Chamfer

Creates a chamfer.

UI Access	Modeler > Chamfer.		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: 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>

```
    "LeftDistance:="           , "0.1mm",
    "RightDistance:="          , "0.1mm",
    "ChamferType:="            , "Symmetric"
]
])
```

VB Syntax	Chamfer <SelectionsArray> <Parameters>
VB Example	<pre>oEditor.Chamfer Array("NAME:Selections", "Selections:=" , "Box2", "NewPartsModelFlag:=" , "Model") Array("NAME:Parameters", Array("NAME:ChamferParameters", "Edges:=" , [42], "Vertices:=" , [], "LeftDistance:=" , "0.1mm", "RightDistance:=" , "0.1mm", "ChamferType:=" , "Symmetric"))</pre>

CleanUpModel

Cleans up history tree operations.

UI Access	Modeler > Cleanup Model History.
Parameters	None.
Return Value	None.

Python Syntax	CleanUpModel()
Python Example	<code>oEditor.CleanUpModel ()</code>

VB Syntax	CleanUpModel
VB Example	<code>oEditor.CleanUpModel</code>

Connect

Connects two or more 1D polyline objects or 2D sheet objects.

UI Access	Modeler > Surface > Connect.		
Parameters	Name <code><SelectionsArray></code>	Type Array	Description Structured array. See: SelectionsArray .
Return Value	None.		

Python Syntax	Connect(<code><SelectionsArray></code>)
Python Example	<code>oEditor.Connect(</code> <code>["NAME:Selections",</code> <code>"Selections:=", "Polyline2,Polyline1"])</code>

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	<table border="1"><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><SelectionsArray></td><td>Array</td><td>Structured array. See: SelectionsArray.</td></tr></table>	Name	Type	Description	<SelectionsArray>	Array	Structured array. See: SelectionsArray .
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	<table border="1" data-bbox="443 628 1896 726"> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> <tr> <td><SelectionsArray></td> <td>Array</td> <td>Structured array. See: SelectionsArray.</td> </tr> </table>	Name	Type	Description	<SelectionsArray>	Array	Structured array. See: SelectionsArray .
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(</pre>

	<pre>"NAME:Selections", "Selections:=", "Obj1_Face1,Obj2_Face2", "NewPartsModelFlag:=", "Model")</pre>
--	--

CreateEntityList

Creates a list of entities containing objects or faces (not both).

UI Access	Modeler > List > Create > [Object / Face] List.		
Parameters	Name <i><Parameters></i>	Type Array	Description Structured array. Array ("NAME:GeometryEntityListParameters", "EntityType:=" , <string "Object" or "Face">, "EntityList:=" , <string of object names or IDs>) See GetObjectIDByName for returning object IDs.
	<i><AttributesArray></i>	Array	Structured array. See: AttributesArray . CreateEntityList takes only the "Name" parameter.
Return Value	None.		

Python Syntax	CreateEntityList(<i><Parameters></i> , <i><AttributesArray></i>)
Python Example	<pre>oEditor.CreateEntityList(["NAME:GeometryEntityListParameters",</pre>

```

    "EntityType:=", "Object",
    "EntityList:=", "Bondwire1,Bondwire2"
],
[ "NAME:Attributes",
    "Name:=", "Objectlist1"
])

```

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	<table border="1"> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> <tr> <td><Parameters></td> <td>Array</td> <td>Structured array. Array ("NAME:FaceCSParameters", <OriginArray>,</td> </tr> </table>	Name	Type	Description	<Parameters>	Array	Structured array. Array ("NAME:FaceCSParameters", <OriginArray>,		
Name	Type	Description							
<Parameters>	Array	Structured array. Array ("NAME:FaceCSParameters", <OriginArray>,							

			<pre> "MoveToEnd:=" , <boolean>, "FaceID:=" , <integer>, <AxisPosnArray>, "WhichAxis:=" , <string "X", "Y", or "Z">, "ZRotationAngle:=" , <string>, "XOffset:=" , <string>, "YOffset:=" , <string>, "AutoAxis:=" , <boolean>) </pre>
<OriginArray>	Array	Structured array.	<p>Array("NAME:Origin",</p> <pre> "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>, </pre>

			<pre>"XPosition:=" , <string>, "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.</pre>
<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 .	
Return Value	None.		

Python Syntax	CreateFaceCS(<Parameters>,<AttributesArray>)
Python Example	<pre> oEditor.CreateFaceCS([["NAME:FaceCSPARAMETERS", [["NAME:Origin", ["IsAttachedToEntity:=" , True, "EntityID:=" , 46, "FacetedBodyTriangleIndex:=" , -1, "TriangleVertexIndex:=" , -1, "PositionType:=" , "FaceCenter", "UParam:=" , 0, "VParam:=" , 0, "XPosition:=" , "0", "YPosition:=" , "0", "ZPosition:=" , "0"], "MoveToEnd:=" , False, "FaceID:=" , 46, ["NAME:AxisPosn", ["IsAttachedToEntity:=" , True, "EntityID:=" , 46,]]]]]]) </pre>

```

    "FacetedBodyTriangleIndex:=", -1,
    "TriangleVertexIndex:=", -1,
    "PositionType:=" , "OnFace",
    "UParam:=" , 0.487129134674319,
    "VParam:=" , 0.308528523557527,
    "XPosition:=" , "1292.27748080459mm",
    "YPosition:=" , "-814.882885865484mm",
    "ZPosition:=" , "0mm"

] ,
"WhichAxis:=" , "X",
"ZRotationAngle:=" , "0deg",
"XOffset:=" , "0mm",
"YOffset:=" , "0mm",
"AutoAxis:=" , False
] ,
[ "NAME:Attributes",
  "Name:=" , "FaceCS1",
  "PartName:=" , "Rectangle1"
])

```

**VB Syn-
tax**

CreateFaceCS <Parameters> <AttributesArray>

VB Example

```

oEditor.CreateFaceCS Array("NAME:FaceCSParameters", Array("NAME:Origin", "IsAt-
tachedToEntity:=", _
true, "EntityID:=", 58, "FacetedBodyTriangleIndex:=", -1, "TriangleVertexIndex:=", -
1, "PositionType:=", "FaceCenter", "UParam:=", 0, "VParam:=", 0, "XPosition:=", -
"0", "YPosition:=", "0", "ZPosition:=", "0"), "MoveToEnd:=", false, "FaceID:=", -
58, Array("NAME:AxisPosn", "IsAttachedToEntity:=", true, "EntityID:=", 58,
"FacetedBodyTriangleIndex:=", -
1, "TriangleVertexIndex:=", -1, "PositionType:=", "OnFace", "UParam:=", -
0.0664066713146499, "VParam:=", 0.407014331135309, "XPosition:=", -
"1826.56266852586mm", "YPosition:=", "-1355.79140131881mm", "ZPosition:=", "0mm"),
"WhichAxis:=", -
"X", "ZRotationAngle:=", "0deg", "XOffset:=", "0mm", "YOffset:=", "0mm", "AutoAxis:=",
-
false), Array("NAME:Attributes", "Name:=", "FaceCS2", "PartName:=", "Rectangle2")

```

CreateGroup

Creates a group from objects specified in the history tree.

UI Access	Modeler > Group > Create.		
Parameters	Name <i><Parameters></i>	Type Array	Description Structured array. Array ("NAME:GroupParameter", "ParentGroupID:=" , <string>,

			"Parts:=" , <string>, "SubmodelInstances:=" , <string>, "Groups:=" , <string>)
Return Value	None.		

Python Syntax	CreateGroup(<Parameters>)
Python Example	<pre>oEditor.CreateGroup(["NAME:GroupParameter", "ParentGroupID:=" , "Model", "Parts:=" , "Box1,Box2,Box3", "SubmodelInstances:=" , "", "Groups:=" , ""])</pre>

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	<p>Structured array.</p> <pre>Array ("NAME:ObjectCSParameters", <OriginArray> "MoveToEnd:=" , <boolean>, "ReverseXAxis:=" , <boolean>, "ReverseYAxis:=" , <boolean>, <xAxisArray / xAxisPosArray> <yAxisArray / yAxisPosArray>)</pre> <p>Note: xAxisArray and xAxisPosArray differ. Use xAxisArray for absolute position and xAxisPosArray for relative position. Do the same for yAxisArray and yAxisPosArray.</p>
	<OriginArray>	Array	<p>Structured array.</p> <pre>Array ("NAME:Origin", "IsAttachedToEntity:=" , <boolean>, "EntityID:=" , <integer>, "FacetedBodyTriangleIndex:=" , <integer>, "TriangleVertexIndex:=" , <integer>, "PositionType:=" , <string "OnVertex", "EdgeCenter", "FaceCenter", "OnEdge", or "AbsolutePosition">,</pre>

		<pre> "UParam:=" , <float between 0 and 1 representing the relative position of the point on the edge or face>, "VParam:=" , <float between 0 and 1 representing the relative position of the point on the edge or face>, "XPosition:=" , <string>, "YPosition:=" , <string>, "ZPosition:=" , <string>) IsAttachedToEntity specifies whether the point is anchored. If True, provide UParam and VParam. If False, provide XPosition, YPosition, and ZPosition to provide fixed position. Pass "0" for unused parameters. </pre>
<xAxisArray>	Array	<p>Structured array for absolute position:</p> <pre> Array ("NAME:xAxis", "DirectionType:=" , "AbsoluteDirection", "EdgeID:=" , <integer>, "FaceID:=" , <integer>, "xDirection:=" , <string>, "yDirection:=" , <string>, "zDirection:=" , <string>, "UParam:=" , <float>, "VParam:=" , <float>) </pre>
<xAxisPosArray>	Array	<p>Structured array for relative position:</p> <pre> Array ("NAME:xAxisPos", "UParam:=" , <float between 0 and 1 representing the relative position of the point on the edge or face>, "VParam:=" , <float between 0 and 1 representing the relative position of the point on the edge or face>, "XPosition:=" , <string>, "YPosition:=" , <string>, "ZPosition:=" , <string>) </pre>

			<pre> "IsAttachedToEntity:=" , <boolean>, "EntityID:=" , <integer>, "FacetedBodyTriangleIndex:=" , <integer>, "TriangleVertexIndex:=" , <integer>, "PositionType:=" , <string "OnVertex", "EdgeCenter", "FaceCenter", or "OnEdge">, "UParam:=" , <float>, "VParam:=" , <float>, "XPosition:=" , <string>, "YPosition:=" , <string>, "ZPosition:=" , <string> </pre>
<yAxisArray>	Array	Structured array for absolute position:	<pre> Array ("NAME:yAxis", "DirectionType:=" , "AbsoluteDirection", "EdgeID:=" , <integer>, "FaceID:=" , <integer>, "xDirection:=" , <string>, "yDirection:=" , <string>, "zDirection:=" , <string>, "UParam:=" , <float>, "VParam:=" , <float>) </pre>

	<code><yAxisPosArray></code>	Array	Structured array for relative position: 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>)
	<code><AttributesArray></code>	Array	Structured array. See: AttributesArray .
Return Value	None.		

Python Syntax	<code>CreateObjectCS(<Parameters>,<AttributesArray>)</code>
Python Example	<pre> oEditor.CreateObjectCS(["NAME:ObjectCSParameters", ["NAME:Origin", "IsAttachedToEntity:=" , True, "EntityID:=" , 59,</pre>

```
"FacetedBodyTriangleIndex:=", -1,  
"TriangleVertexIndex:=", -1,  
"PositionType:=", "OnVertex",  
"UParam:=", 0,  
"VParam:=", 0,  
"XPosition:=", "0",  
"YPosition:=", "0",  
"ZPosition:=", "0"  
,  
"MoveToEnd:=", False,  
"ReverseXAxis:=", False,  
"ReverseYAxis:=", False,  
["NAME:xAxis",  
"DirectionType:=", "AbsoluteDirection",  
"EdgeID:=", -1,  
"FaceID:=", -1,  
"xDirection:=", "1",  
"yDirection:=", "0",  
"zDirection:=", "0",  
"UParam:=", 0,
```

```

        "VParam:=" , 0
    ] ,
    [ "NAME:yAxis",
        "DirectionType:=" , "AbsoluteDirection",
        "EdgeID:=" , -1,
        "FaceID:=" , -1,
        "xDirection:=" , "0",
        "yDirection:=" , "1",
        "zDirection:=" , "0",
        "UParam:=" , 0,
        "VParam:=" , 0
    ]
],
[ "NAME:Attributes",
    "Name:=" , "ObjectCS1",
    "PartName:=" , "Box2"
])

```

VB Syntax	CreateObjectCS <Parameters> <AttributesArray>
VB Example	oEditor.CreateObjectCS Array("NAME:ObjectCSParameters", Array("NAME:Origin", "IsAttachedToEntity:=", _

```

    false, "EntityID:=", -1, "FacetedBodyTriangleIndex:=", -1, "TriangleVertexIndex:=", _  

    -1, "PositionType:=", "AbsolutePosition", "UParam:=", 0, "VParam:=", 0, "XPosition:=", _  

    -  

    "0mm", "YPosition:=", "0mm", "ZPosition:=", "0mm"), "MoveToEnd:=", false, "ReverseXAxis:=", _  

    false, "ReverseYAxis:=", false, Array("NAME:xAxisPos", "IsAttachedToEntity:=", true,  

    "EntityID:=", _  

    80, "FacetedBodyTriangleIndex:=", -1, "TriangleVertexIndex:=", -1, "PositionType:=", _  

    "EdgeCenter", "UParam:=", 0, "VParam:=", 0, "XPosition:=", "0", "YPosition:=", _  

    "0", "ZPosition:=", "0"), Array("NAME:yAxisPos", "IsAttachedToEntity:=", true,  

    "EntityID:=", _  

    69, "FacetedBodyTriangleIndex:=", -1, "TriangleVertexIndex:=", -1, "PositionType:=", _  

    "EdgeCenter", "UParam:=", 0, "VParam:=", 0, "XPosition:=", "0", "YPosition:=", _  

    "0", "ZPosition:=", "0)), Array("NAME:Attributes", "Name:=", "ObjectCS2",  

    "PartName:=", _  

    "Box3")

```

CreateObjectFromEdges

Creates an object from the specified object edge.

UI Access	Modeler > Edge > Create Object From Edge		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: SelectionsArray .
	<ParametersArray>	Array	Structured array.

			Array("NAME:Parameters", Array("NAME:BodyFromEdgeToParameters", "Edges:=", <array containing integer edges>))
	<CreateGroupsForNewObjects>	Array	Structured array. Array("CreateGroupsForNewObjects:=", <boolean True to create groups for new objects; else False>)
Return Value	None.		

Python Syntax	CreateObjectFromEdges(<SelectionsArray>, <ParametersArray>, <CreateGroupsForNewObjects>)
Python Example	<pre> oEditor.CreateObjectFromEdges (["NAME:Selections", "Selections:=", "Box2", "NewPartsModelFlag:=", "Model"], ["NAME:Parameters", ["NAME:BodyFromEdgeToParameters", "Edges:=", [41]]]) </pre>

```
    ],
    ["CreateGroupsForNewObjects:=", False
  ])
```

VB Syntax	CreateObjectFromEdges <SelectionsArray> <ParametersArray> <CreateGroupsForNewObjects>
VB Example	<pre>oEditor.CreateObjectFromEdges Array("NAME:Selections", "Selections:=", "Box1", "NewPartsModelFlag:=", "Model"), Array("NAME:Parameters", Array("NAME:BodyFromEdgeToParameters", "Edges:=", Array(13))), Array("CreateGroupsForNewObjects:=", false)</pre>

CreateObjectFromFace

Creates 2D objects from specified face(s).

UI Access	N/A														
Parameters	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><SelectionsArray></td> <td>Array</td> <td>Structured array. See: SelectionsArray.</td> </tr> <tr> <td><Parameters></td> <td>Array</td> <td>Structured array. Array ("NAME:Parameters", <BodyFromFaceToParameters>)</td> </tr> <tr> <td><CreateGroupsForNewObjects></td> <td>Array</td> <td>Optional. Structured array. Array ("CreateGroupsForNewObjects:=", <boolean>)</td> </tr> </tbody> </table>	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>)		
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))) </pre>

	Array ("CreateGroupsForNewObjects:=", false)
--	--

CreateObjectFromFaces

Creates 2D objects from specified face(s).

UI Access	Modeler > Surface > Create Object from Face		
Parameters	Name <i><SelectionsArray></i>	Type Array	Description Structured array. See: SelectionsArray .
	<i><Parameters></i>	Array	Structured array. Array ("NAME:Parameters", <i><BodyFromFaceToParameters></i>)
	<i><CreateGroupsForNewObjects></i>	Array	Structured array. Array ("CreateGroupsForNewObjects:=", <i><boolean></i>)
Return Value	None.		

Python Syntax	CreateObjectFromFaces (<SelectionsArray>,<Parameters>,<CreateGroupsForNewObjects>)
Python Example	<pre> oEditor.CreateObjectFromFaces (["NAME:Selections", "Selections:=" , "Box3", "NewPartsModelFlag:=" , "Model"],) </pre>

```
[ "NAME:Parameters",
  [
    "NAME:BodyFromFaceToParameters",
    "FacesToDetach:=" , [68]
  ]
],
[ "CreateGroupsForNewObjects:=", False
])
```

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 <Parameters>	Type Array	Description Structured array. Array ("NAME:RelativeCSPARAMETERS", "Mode:=" , "Axis/Position",

			<pre>"OriginX:=" , <string>, "OriginY:=" , <string>, "OriginZ:=" , <string>, "XAxisXvec:=" , <string>, "XAxisYvec:=" , <string>, "XAxisZvec:=" , <string>, "YAxisXvec:=" , <string>, "YAxisYvec:=" , <string>, "YAxisZvec:=" , <string>)</pre>
	<p><AttributesArray></p>	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",</pre>

```

    "XAxisXvec:=", "1mm",
    "XAxisYvec:=", "0mm",
    "XAxisZvec:=", "0mm",
    "YAxisXvec:=", "0mm",
    "YAxisYvec:=", "1mm",
    "YAxisZvec:=", "0mm"
],
[ "NAME:Attributes",
  "Name:=", "RelativeCS1"
])

```

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 <i><Parameters></i>	Type Array	Description Structured array. Array("Groups:=", <array of string group IDs>)
Return Value	None.		

Python Syntax	DeleteEmptyGroups(<Parameters>)
Python Example	oEditor.DeleteEmptyGroups (["Groups:=", ["Group1", "Group2", "Group3"]])

VB Syntax	DeleteEmptyGroups <Parameters>
VB Example	oEditor.DeleteEmptyGroups Array("Groups:=", Array("Group1", "Group2", "Group3"))

DeleteLastOperation

Deletes the last operation performed on the specified object(s).

UI Access	Modeler > Delete Last Operation.		
Parameters	Name <i><SelectionsArray></i>	Type Array	Description 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	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><ParamsArray></td> <td>Array</td> <td> <p>Structured array.</p> <p>Array ("NAME:Parameters", Array ("NAME:PartOperations", Array ("NAME:<PartName>", "OperationIndices:=", <array of operation indices>)),</p> </td> </tr> </tbody> </table>	Name	Type	Description	<ParamsArray>	Array	<p>Structured array.</p> <p>Array ("NAME:Parameters", Array ("NAME:PartOperations", Array ("NAME:<PartName>", "OperationIndices:=", <array of operation indices>)),</p>
Name	Type	Description					
<ParamsArray>	Array	<p>Structured array.</p> <p>Array ("NAME:Parameters", Array ("NAME:PartOperations", Array ("NAME:<PartName>", "OperationIndices:=", <array of operation indices>)),</p>					

			Array ("NAME:UDMOoperations")
Return Value	None.		

Python Syntax	DeleteOperation(<ParamsArray>)
Python Example	<pre> oEditor.DeleteOperation(["NAME:Parameters", ["NAME:PartOperations", ["NAME:Coil_0", "OperationIndices:=", [1]]], ["NAME:UDMOoperations"]]) </pre>

VB Syntax	DeleteOperation <ParamsArray>
VB Example	<pre> oEditor.DeleteOperation _ </pre>

```

Array("NAME:Parameters", _
      Array("NAME:PartOperations", _
            Array("NAME:Coil_0", _
                  "OperationIndices:=", Array(1))), _
      Array("NAME:UDMOperations"))

```

DetachEdges

Detaches the specified edge(s) from an object.

UI Access	Modeler > Edge > Detach Edges.		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: SelectionsArray .
	<Parameters>	Array	Structured array. Array ("NAME:Parameters", <DetachEdgesArray>)
<DetachEdgesArray>	Array	Structured array. Array ("NAME:DetachEdgesToParameters", "EdgesToDelete:=" , <array containing integer edge IDs>)	
Return Value	None.		

Python Syntax	DetachEdges(<SelectionsArray>, <Parameters>)
Python Example	<pre> oEditor.DetachEdges (["NAME:Selections", </pre>

```

    "Selections:=", "Rectangle1",
    "NewPartsModelFlag:=", "Model"
],
[ "NAME:Parameters",
  [ "NAME:DetachEdgesToParameters",
    "EdgesToDetach:=", [18,17]
  ]
]
)

```

VB Syntax	DetachEdges <SelectionsArray> <Parameters>
VB Example	<pre> oEditor.DetachEdges Array("NAME:Selections", "Selections:=", "Rectangle1", "NewPartsModelFlag:=", "Model") Array("NAME:Parameters", Array("NAME:DetachEdgesToParameters", "EdgesToDetach:=", [18,17])) </pre>

DetachFaces

Detaches the specified face(s) from an object.

UI Access	Modeler > Surface > Detach Faces.		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: SelectionsArray .
	<Parameters>	Array	Structured array.

		Array ("NAME:Parameters", <DetachFacesArray>)
<DetachFacesArray>	Array	Structured array. Array ("NAME:DetachFacesToParameters", "FacesToDetach:=" , <array containing integer face IDs>)
Return Value	None.	

Python Syntax	DetachFaces(<SelectionsArray>, <Parameters>)
Python Example	<pre> oEditor.DetachFaces (["NAME:Selections", "Selections:=", "Box3", "NewPartsModelFlag:=", "Model"], ["NAME:Parameters", ["NAME:DetachFacesToParameters", "FacesToDetach:=", [68, 67]]]) </pre>

VB Syntax	DetachFaces <SelectionsArray> <Parameters>
------------------	--

VB Example	<pre> oEditor.DetachFaces Array("NAME:Selections", "Selections:=", "Box3", "NewPartsModelFlag:=", "Model") Array("NAME:Parameters", Array("NAME:DetachFacesToParameters", "FacesToDetach:=", [68,67])) </pre>
-------------------	---

EditEntityList

Modifies an entity list.

UI Access	Modeler > List > Reassign.											
Parameters	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><SelectionsArray></td> <td>Array</td> <td>Structured array. See: SelectionsArray.</td> </tr> <tr> <td><Parameters></td> <td>Array</td> <td>Structured array. Array ("NAME:GeometryEntityListParameters", "EntityType:=" , <string "Object" or "Face">, "EntityList:=" , <string list>) </td> </tr> </tbody> </table>	Name	Type	Description	<SelectionsArray>	Array	Structured array. See: SelectionsArray .	<Parameters>	Array	Structured array. Array ("NAME:GeometryEntityListParameters", "EntityType:=" , <string "Object" or "Face">, "EntityList:=" , <string list>)		
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	<pre>EditEntityList(<SelectionsArray>, <Parameters>)</pre>
Python Example	<pre> oEditor.EditEntityList(["NAME:Selections", "Selections:=" , "Objectlist1"]) </pre>

```

        ],
        [
        "NAME:GeometryEntityListParameters",
        "EntityType:=", "Object",
        "EntityList:=", "Box1, Box2, Box3"
    ]
)

```

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	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><Parameters></td> <td>Array</td> <td> <p>Structured array.</p> <pre> Array("NAME:FaceCSParameters", <OriginArray>, "MoveToEnd:=", <boolean>, ...) </pre> </td> </tr> </tbody> </table>	Name	Type	Description	<Parameters>	Array	<p>Structured array.</p> <pre> Array("NAME:FaceCSParameters", <OriginArray>, "MoveToEnd:=", <boolean>, ...) </pre>		
Name	Type	Description							
<Parameters>	Array	<p>Structured array.</p> <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	<p>Structured array.</p> <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>

			<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.</pre>
	<AxisPosnArray>	Array	<p>Structured array.</p> <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",
```

```

    "UParam:=" , 0.62951717774066,
    "VParam:=" , 0.226514925559344,
    "XPosition:=" , "1200mm",
    "YPosition:=" , "-354.697014888131mm",
    "ZPosition:=" , "125.903435548132mm"),
    "WhichAxis:=" , "X",
    "ZRotationAngle:=" , "0deg",
    "XOffset:=" , "0mm",
    "YOffset:=" , "0mm",
    "AutoAxis:=" , False)

Array("NAME:Attributes",
      "Name:=" , "FaceCS1",
      "PartName:=" , "Box1")

```

EditObjectCS

Edits an existing object coordinate system. See: [CreateObjectCS](#).

UI Access	Modeler > Coordinate System > Edit.								
Parameters	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><Parameters></td> <td>Array</td> <td>Structured array. Array ("NAME:ObjectCSParameters", <OriginArray> "MoveToEnd:=" , <boolean>,</td> </tr> </tbody> </table>	Name	Type	Description	<Parameters>	Array	Structured array. Array ("NAME:ObjectCSParameters", <OriginArray> "MoveToEnd:=" , <boolean>,		
Name	Type	Description							
<Parameters>	Array	Structured array. Array ("NAME:ObjectCSParameters", <OriginArray> "MoveToEnd:=" , <boolean>,							

			<pre> "ReverseXAxis:=" , <boolean>, "ReverseYAxis:=" , <boolean>, <xAxisArray / xAxisPosArray> <yAxisArray / yAxisPosArray> </pre> <p>Note: xAxisArray and xAxisPosArray differ. Use xAxisArray for absolute position and xAxisPosArray for relative position. Do the same for yAxisArray and yAxisPosArray.</p>
	<i><OriginArray></i>	Array	<p>Structured array.</p> <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>) IsAttachedToEntity specifies whether the point is anchored. If True, provide UParam and VParam. If False, provide XPosition, YPosition, and ZPosition to provide fixed position. Pass "0" for unused parameters.</pre>
<xAxisArray>	Array	<p>Structured array for absolute position:</p> <pre>Array ("NAME:xAxis", "DirectionType:=" , "AbsoluteDirection", "EdgeID:=" , <integer>, "FaceID:=" , <integer>, "xDirection:=" , <string>, "yDirection:=" , <string>, "zDirection:=" , <string>, "UParam:=" , <float>, "VParam:=" , <float>)</pre>
<xAxisPosArray>	Array	<p>Structured array for relative position:</p> <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	<p>Structured array for absolute position:</p> <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	<p>Structured array for relative position:</p> <pre>Array ("NAME:yAxisPos", "IsAttachedToEntity:=" , <boolean>, "EntityID:=" , <integer>, "FacetedBodyTriangleIndex:=" , <integer>, "TriangleVertexIndex:=" , <integer>,</pre>

			<pre>"PositionType:=" , <string "OnVertex", "EdgeCenter", "FaceCenter", or "OnEdge">, "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	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,
```

```

        "FaceID:="           , -1,
        "xDirection:="       , "0",
        "yDirection:="       , "1",
        "zDirection:="       , "0",
        "UParam:="           , 0,
        "VParam:="           , 0
    ]
],
[ "NAME:Attributes",
    "Name:="             , "ObjectCS1",
    "PartName:="          , "Box2"
])

```

VB Syntax	EditObjectCS <Parameters> <AttributesArray>
VB Example	<pre> oEditor>EditObjectCS Array("NAME:ObjectCSParameters", Array("NAME:Origin", "IsAttachedToEntity:=", false, "EntityID:=", -1, "FacetedBodyTriangleIndex:=", -1, "TriangleVertexIndex:=", -1, "PositionType:=", "AbsolutePosition", "UParam:=", 0, "VParam:=", 0, "XPosition:=", - "0mm", "YPosition:=", "0mm", "ZPosition:=", "0mm"), "MoveToEnd:=", false, "ReverseXAxis:=", - false, "ReverseYAxis:=", false, Array("NAME:xAxisPos", "IsAttachedToEntity:=", true, </pre>

```

"EntityID:=", _
80, "FacetedBodyTriangleIndex:=", -1, "TriangleVertexIndex:=", -1, "PositionType:=", _
"EdgeCenter", "UParam:=", 0, "VParam:=", 0, "XPosition:=", "0", "YPosition:=", _
"0", "ZPosition:=", "0"), Array("NAME:yAxisPos", "IsAttachedToEntity:=", true,
"EntityID:=", _
69, "FacetedBodyTriangleIndex:=", -1, "TriangleVertexIndex:=", -1, "PositionType:=", _
"EdgeCenter", "UParam:=", 0, "VParam:=", 0, "XPosition:=", "0", "YPosition:=", _
"0", "ZPosition:=", "0)), Array("NAME:Attributes", "Name:=", "ObjectCS2",
"PartName:=", _
"Box3")

```

EditRelativeCS

Edits an existing Relative Coordinate System. See: [CreateRelativeCS](#).

UI Access	Modeler > Coordinate System > Edit.								
Parameters	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><Parameters></td> <td>Array</td> <td> <p>Structured array.</p> <p>Array ("NAME:RelativeCSPARAMETERS",</p> <p>"Mode:=" , "Axis/Position",</p> <p>"OriginX:=" , <string>,</p> <p>"OriginY:=" , <string>,</p> <p>"OriginZ:=" , <string>,</p> </td></tr> </tbody> </table>	Name	Type	Description	<Parameters>	Array	<p>Structured array.</p> <p>Array ("NAME:RelativeCSPARAMETERS",</p> <p>"Mode:=" , "Axis/Position",</p> <p>"OriginX:=" , <string>,</p> <p>"OriginY:=" , <string>,</p> <p>"OriginZ:=" , <string>,</p>		
Name	Type	Description							
<Parameters>	Array	<p>Structured array.</p> <p>Array ("NAME:RelativeCSPARAMETERS",</p> <p>"Mode:=" , "Axis/Position",</p> <p>"OriginX:=" , <string>,</p> <p>"OriginY:=" , <string>,</p> <p>"OriginZ:=" , <string>,</p>							

			<pre>"XAxisXvec:=", <string>, "XAxisYvec:=", <string>, "XAxisZvec:=", <string>, "YAxisXvec:=", <string>, "YAxisYvec:=", <string>, "YAxisZvec:=", <string>)</pre>
	<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>

```
    "YAxisZvec:=" , "0mm"
],
[ "NAME:Attributes",
  "Name:=" , "RelativeCS1"
])
```

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", "YAx- isZvec:=", _ "0mm"), Array("NAME:Attributes", "Name:=", "RelativeCS1")</pre>

Export

Exports the model to a file.

Note:

This script does not export image file types or GDSII files. See: [ExportModelImageToFile](#) and [ExportGDSII](#).

UI Access	Modeler > Export.		
Parameters	Name	Type	Description
	<Parameters>	Array	<p>Structured array.</p> <pre>Array ("NAME:ExportParameters", "AllowRegionDependentPartSelectionForPMLCreation:=", <boolean>, "AllowRegionSelectionForPMLCreation:=", <boolean>, "Selections:=" , <string list>, "File Name:=" , <string filepath>, "Major Version:=" , <integer (-1 if not applicable)>, "Minor Version:=" , <integer (-1 if not applicable)>)</pre>
Return Value	None.		

Python Syntax	Export(<Parameters>)
Python Example	<pre>oEditor.Export(["NAME:ExportParameters", "AllowRegionDependentPartSelectionForPMLCreation:=", True, "AllowRegionSelectionForPMLCreation:=", True, "Selections:=" , "Box1,Box2,Box3",</pre>

```

    "File Name:="           , "C:/Users/jdoe/Desktop/export.sab",
    "Major Version:="      , 25,
    "Minor Version:="      , 0
]
)

```

VB Syntax	Export <Parameters>
VB Example	<pre> oEditor.Export Array("NAME:ExportParameters", "AllowRegionDependentPartSelectionForPMLCreation:=", True, "AllowRegionSelectionForPMLCreation:=", True, "Selections:=" , "Box1,Box2,Box3", "File Name:=" , "C:/Users/jdoe/Desktop/export.sab", "Major Version:=" , 25, "Minor Version:=" , 0) </pre>

ExportModelImageToFile

Exports the model as an image file (*.avz, *.bmp, *.gif, *.jpeg, *.tiff, *.wrl). In Release 23.1, this command is fully supports -ng (non-graphical) mode. To export to Ensight use *.avz. For export to Ensight in -ng mode, the corresponding version of Ensight must be installed. On Linux, it might need manual set environment variable AWP_ROOT212 to its installation path, e.g. AWP_ROOT212-2=/installations/ansys_inc/v212/ for AnsysEDT v21.2 and Ensight 21.2.

ExportModelImageToFile supports export overlay of polar plot 3D with existing transformation (scaling, rotation and translation) in -ng (non-graphical) mode.

UI Access	Modeler > Export.															
Parameters	<table border="1"> <thead> <tr> <th data-bbox="456 270 1015 303">Name</th> <th data-bbox="1015 270 1100 303">Type</th> <th data-bbox="1100 270 1886 303">Description</th> </tr> </thead> <tbody> <tr> <td data-bbox="456 303 1015 336"><path></td><td data-bbox="1015 303 1100 336">String</td><td data-bbox="1100 303 1886 336">Full file path including extension.</td></tr> <tr> <td data-bbox="456 336 1015 368"><width></td><td data-bbox="1015 336 1100 368">Integer</td><td data-bbox="1100 336 1886 368">Width in pixels (use 0 for default).</td></tr> <tr> <td data-bbox="456 368 1015 401"><height></td><td data-bbox="1015 368 1100 401">Integer</td><td data-bbox="1100 368 1886 401">Height in pixels (use 0 for default).</td></tr> <tr> <td data-bbox="456 401 1015 1127"><Parameters></td><td data-bbox="1015 401 1100 1127">Array</td><td data-bbox="1100 401 1886 1127"> <p>Structured array.</p> <pre data-bbox="1100 483 1854 1127"> 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. "Orientation:=" , <string> "ShowOrientationGadget:=" , <False> </pre> </td></tr> </tbody> </table>	Name	Type	Description	<path>	String	Full file path including extension.	<width>	Integer	Width in pixels (use 0 for default).	<height>	Integer	Height in pixels (use 0 for default).	<Parameters>	Array	<p>Structured array.</p> <pre data-bbox="1100 483 1854 1127"> 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. "Orientation:=" , <string> "ShowOrientationGadget:=" , <False> </pre>
Name	Type	Description														
<path>	String	Full file path including extension.														
<width>	Integer	Width in pixels (use 0 for default).														
<height>	Integer	Height in pixels (use 0 for default).														
<Parameters>	Array	<p>Structured array.</p> <pre data-bbox="1100 483 1854 1127"> 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. "Orientation:=" , <string> "ShowOrientationGadget:=" , <False> </pre>														
Return Value	None.															

Python Syntax	ExportModelImageToFile(<path> <width> <height> <Parameters>)
Python Example	oEditor.ExportModelImageToFile

```
("C:/Users/jdoe/Desktop/export.bmp",
1920,
1080,
[
    "NAME:SaveImageParams",
    "ShowAxis:=" , "True",
    "ShowGrid:=" , "True",
    "ShowRuler:=" , "True",
    "ShowRegion:=" , "Default",
    "Selections:=" , "",
    "FieldPlotSelections:=" , "",
    "FitToSelections:=" , "",
    "FitToFieldPlotSelections:=" , "",
    "Orientation:=" , ""
])
])
```

VB Syntax	ExportModelImageToFile <path> <width> <height> <Parameters>
VB Example	<pre>oEditor.ExportModelImageToFile "C:/Users/jdoe/Desktop/export.bmp", 1383,</pre>

```

512,
Array("NAME:SaveImageParams",
      "ShowAxis:=", "True",
      "ShowGrid:=", "True",
      "ShowRuler:=", "True",
      "ShowRegion:=", "Default",
      "Selections:=", "",
      "FitToFieldPlotSelections:=", "",
      "Orientation:=", ""
)

```

ExportModelMeshToFile

Exports geometry model to a 3D model file (e.g. *.obj, *.wrl, etc.).

UI Access	N/A		
Parameters	Name	Type	Description
	<filePath>	String	Full file path, including extension *.obj, *.wrl, etc
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>

Fillet

Performs a fillet on specified edge(s).

UI Access	Modeler > Fillet.														
Parameters	<table border="1"> <tr> <td>Name</td> <td>Type</td> <td>Description</td> </tr> <tr> <td><SelectionsArray></td> <td>Array</td> <td>Structured array. See: SelectionsArray.</td> </tr> <tr> <td><Parameters></td> <td>Array</td> <td>Structured array. Array ("NAME:Parameters", <FilletParametersArray>)</td> </tr> <tr> <td><FilletParametersArray></td> <td>Array</td> <td>Structured array. Array ("NAME:FilletParameters", "Edges:=" , <array containing integer edge IDs>, "Vertices:=" , <empty array>, "Radius:=" , <string>, "Setback:=" , <string>)</td> </tr> </table>	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>)		
Name	Type	Description													
<SelectionsArray>	Array	Structured array. See: SelectionsArray .													
<Parameters>	Array	Structured array. Array ("NAME:Parameters", <FilletParametersArray>)													
<FilletParametersArray>	Array	Structured array. Array ("NAME:FilletParameters", "Edges:=" , <array containing integer edge IDs>, "Vertices:=" , <empty array>, "Radius:=" , <string>, "Setback:=" , <string>)													
Return Value	None.														

Python Syntax	Fillet(<SelectionsArray>, <Parameters>)
----------------------	---

Python Example <pre> oEditor.Fillet(["NAME:Selections", "Selections:=" , "Box1", "NewPartsModelFlag:=" , "Model"], ["NAME:Parameters", ["NAME:FilletParameters", "Edges:=" , [13], "Vertices:=" , [], "Radius:=" , "1mm", "Setback:=" , "0mm"]]) </pre>

VB Syntax <pre>Fillet <SelectionsArray> <Parameters></pre>	
VB Example <pre> oEditor.Fillet Array("NAME:Selections", "Selections:=" , "Box1", "NewPartsModelFlag:=" , "Model") Array("NAME:Parameters", Array("NAME:FilletParameters", </pre>	

	"Edges:=" , Array(13), "Vertices:=" , Array(), "Radius:=" , "1mm", "Setback:=" , "0mm"))
--	--

FlattenGroup

Flattens a specified history tree group.

UI Access	Modeler > Group > Flatten.		
Parameters	Name <GroupID>	Type Array	Description Structured array. Array("Groups:=", Array(<string list of group IDs>))
Return Value	None.		

Python Syntax	FlattenGroup (<GroupID>)
Python Example	oEditor.FlattenGroup (["Groups:=", ["Group1"]])

VB Syntax	FlattenGroup <GroupID>
VB Example	oEditor.FlattenGroup Array("Groups:=", Array("Group1"))

GenerateHistory

Generates the history for specified 1D object(s).

UI Access	Modeler > Generate History.		
Parameters	Name <SelectionsArray>	Type Array	Description Structured array. See: SelectionsArray .
Return Value	None.		

Python Syntax	GenerateHistory (<SelectionsArray>)
Python Example	<pre> oEditor.GenerateHistory(["NAME:Selections", "Selections:=" , "Polyline1", "NewPartsModelFlag:=" , "Model", "UseCurrentCS:=" , True]) </pre>

VB Syntax	GenerateHistory <SelectionsArray>
------------------	-----------------------------------

VB Example	<pre>oEditor.GenerateHistory Array("NAME:Selections", "Selections:=" , "Polyline1", "NewPartsModelFlag:=" , "Model", "UseCurrentCS:=" , True)</pre>
-------------------	--

GetActiveCoordinateSystem

Returns the active coordinate system.

UI Access	None.
Parameters	None.
Return Value	String name of active coordinate system.

Python Syntax	GetActiveCoordinateSystem()
Python Example	<pre>oEditor.GetActiveCoordinateSystem()</pre>

VB Syntax	GetActiveCoordinateSystem
VB Example	<pre>dim csName csName = oEditor.GetActiveCoordinateSystem</pre>

GetActiveCoordinateSystemTransform

Returns the transformation information of active coordinate system.

UI Access	None.
Parameters	None.
Return Value	Array of strings containing transformation information.

Python Syntax	GetActiveCoordinateSystemTransform()
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	<code>dim csNames</code>

	csNames = oEditor.GetCoordinateSystems
--	--

HealObject

Heals an imported object.

UI Access	Modeler > Model Preparation > Heal.											
Parameters	<table border="1"> <tr> <td>Name</td> <td>Type</td> <td>Description</td> </tr> <tr> <td><SelectionsArray></td> <td>Array</td> <td>Structured array. See: SelectionsArray.</td> </tr> <tr> <td><Parameters></td> <td>Array</td> <td> <p>Structured array.</p> <pre>Array ("NAME:ObjectHealingParameters", "Version:=" , <integer>, "AutoHeal:=" , <boolean>, "TolerantStitch:=" , <boolean>, "SimplifyGeom:=" , <boolean>, "TightenGaps:=" , <boolean>, "HealToSolid:=" , <boolean>, "StopAfterFirstStitchError:=" , <boolean>, "MaxStitchTol:=" , <float>, "ExplodeAndStitch:=" , <boolean>, "GeomSimplificationTol:=" , <integer>, "Max- imumGeneratedRadiusForSimplification:=" , <integer>,</pre> </td> </tr> </table>	Name	Type	Description	<SelectionsArray>	Array	Structured array. See: SelectionsArray .	<Parameters>	Array	<p>Structured array.</p> <pre>Array ("NAME:ObjectHealingParameters", "Version:=" , <integer>, "AutoHeal:=" , <boolean>, "TolerantStitch:=" , <boolean>, "SimplifyGeom:=" , <boolean>, "TightenGaps:=" , <boolean>, "HealToSolid:=" , <boolean>, "StopAfterFirstStitchError:=" , <boolean>, "MaxStitchTol:=" , <float>, "ExplodeAndStitch:=" , <boolean>, "GeomSimplificationTol:=" , <integer>, "Max- imumGeneratedRadiusForSimplification:=" , <integer>,</pre>		
Name	Type	Description										
<SelectionsArray>	Array	Structured array. See: SelectionsArray .										
<Parameters>	Array	<p>Structured array.</p> <pre>Array ("NAME:ObjectHealingParameters", "Version:=" , <integer>, "AutoHeal:=" , <boolean>, "TolerantStitch:=" , <boolean>, "SimplifyGeom:=" , <boolean>, "TightenGaps:=" , <boolean>, "HealToSolid:=" , <boolean>, "StopAfterFirstStitchError:=" , <boolean>, "MaxStitchTol:=" , <float>, "ExplodeAndStitch:=" , <boolean>, "GeomSimplificationTol:=" , <integer>, "Max- imumGeneratedRadiusForSimplification:=" , <integer>,</pre>										

			<pre> "SimplifyType:=" , <integer>, "TightenGapsWidth:=" , <float>, "RemoveSliverFaces:=" , <boolean>, "RemoveSmallEdges:=" , <boolean>, "RemoveSmallFaces:=" , <boolean>, "SliverFaceTol:=" , <integer>, "SmallEdgeTol:=" , <integer>, "SmallFaceAreaTol:=" , <integer>, "BoundingBoxScaleFactor:=" , <integer>, "RemoveHoles:=" , <boolean>, "RemoveChamfers:=" , <boolean>, "RemoveBlends:=" , <boolean>, "HoleRadiusTol:=" , <integer>, "ChamferWidthTol:=" , <integer>, "BlendRadiusTol:=" , <integer>, "AllowableSurfaceAreaChange:=" , <integer>, "AllowableVolumeChange:=" , <integer>) </pre>
Return Value	None.		

Python Syntax	HealObject(<SelectionsArray>, <Parameters>)
----------------------	---

Python Example

```
oEditor.HealObject(  
    ["NAME:Selections",  
        "Selections:=", "Box1",  
        "NewPartsModelFlag:=", "Model"  
    ],  
    ["NAME:ObjectHealingParameters",  
        "Version:=", 1,  
        "AutoHeal:=", True,  
        "TolerantStitch:=", True,  
        "SimplifyGeom:=", True,  
        "TightenGaps:=", True,  
        "HealToSolid:=", False,  
        "StopAfterFirstStitchError:=", False,  
        "MaxStitchTol:=", 0.001,  
        "ExplodeAndStitch:=", True,  
        "GeomSimplificationTol:=", -1,  
        "MaximumGeneratedRadiusForSimplification:=", -1,  
        "SimplifyType:=", 2,  
        "TightenGapsWidth:=", 1E-06,  
        "RemoveSliverFaces:=", False,
```

```

    "RemoveSmallEdges:="      , False,
    "RemoveSmallFaces:="      , False,
    "SliverFaceTol:="        , 0,
    "SmallEdgeTol:="          , 0,
    "SmallFaceAreaTol:="      , 0,
    "BoundingBoxScaleFactor:=", 1250,
    "RemoveHoles:="           , False,
    "RemoveChamfers:="        , False,
    "RemoveBlends:="          , False,
    "HoleRadiusTol:="         , 0,
    "ChamferWidthTol:="       , 0,
    "BlendRadiusTol:="        , 0,
    "AllowableSurfaceAreaChange:=", 5,
    "AllowableVolumeChange:=", 5
  ])
)

```

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:=", _ </pre>

```

true, "SimplifyGeom:=", true, "TightenGaps:=", true, "HealToSolid:=", false, "StopAfter-
FirstStitchError:=", _
false, "MaxStitchTol:=", 0.001, "ExplodeAndStitch:=", true, "GeomSimplificationTol:=",
-
-1, "MaximumGeneratedRadiusForSimplification:=", -1, "SimplifyType:=", 2,
"TightenGapsWidth:=", _
1E-06, "RemoveSliverFaces:=", false, "RemoveSmallEdges:=", false, "RemoveSmallFaces:=",
-
false, "SliverFaceTol:=", 0, "SmallEdgeTol:=", 0, "SmallFaceAreaTol:=", 0, "Bound-
ingBoxScaleFactor:=", _
1250, "RemoveHoles:=", false, "RemoveChamfers:=", false, "RemoveBlends:=", _
false, "HoleRadiusTol:=", 0, "ChamferWidthTol:=", 0, "BlendRadiusTol:=", 0, "Allow-
ableSurfaceAreaChange:=", _
5, "AllowableVolumeChange:=", 5)

```

Import

Imports a 3D model file.

Note:

This script does not import DXF or GDSII models. See: [ImportDXF](#) and [ImportGDSII](#).

UI Access	Modeler > Import.		
Parameters	Name	Type	Description
	<Parameters>	Array	Structured array.

		Array ("NAME:NativeBodyParameters", "HealOption:=" , <integer>, "Options:=" , <string>, "FileType:=" , <string "UnRecognized">, "MaxStitchTol:=" , <integer>, "ImportFreeSurfaces:=" , <boolean>, "GroupByAssembly:=" , <boolean>, "CreateGroup:=" , <boolean>, "STLFileUnit:=" , <string>, "MergeFacesAngle:=" , <float>, "HealSTL:=" , <boolean>, "ReduceSTL:=" , <boolean>, "ReduceMaxError:=" , <integer>, "ReducePercentage:=" , <integer>, "PointCoincidenceTol:=" , <float>, >CreateLightweightPart:=" , <boolean>, "ImportMaterialNames:=" , <boolean>, "SeparateDisjointLumps:=" , <boolean>, "SourceFile:=" , <string>)
Return Value	None.	

Python Syntax	<code>Import(<Parameters>)</code>
Python Example	<pre> oEditor.Import(["NAME:NativeBodyParameters", "HealOption:=" , 0, "Options:=" , "-1", "FileType:=" , "UnRecognized", "MaxStitchTol:=" , -1, "ImportFreeSurfaces:=" , False, "GroupByAssembly:=" , False, "CreateGroup:=" , True, "STLFileUnit:=" , "Auto", "MergeFacesAngle:=" , 0.02, "HealSTL:=" , False, "ReduceSTL:=" , False, "ReduceMaxError:=" , 0, "ReducePercentage:=" , 100, "PointCoincidenceTol:=" , 1E-06, "CreateLightweightPart:=" , False, "ImportMaterialNames:=" , False, "SeparateDisjointLumps:=" , False,]) </pre>

	"SourceFile:=" , "C:\\Users\\jdoe\\Desktop\\export.model"])
--	---

VB Syntax	Import <Parameters>
VB Example	<pre> oEditor.Import Array ("NAME:NativeBodyParameters", "HealOption:=" , 0, "Options:=" , "-1", "FileType:=" , "UnRecognized", "MaxStitchTol:=" , -1, "ImportFreeSurfaces:=" , False, "GroupByAssembly:=" , False, "CreateGroup:=" , True, "STLFileUnit:=" , "Auto", "MergeFacesAngle:=" , 0.02, "HealSTL:=" , False, "ReduceSTL:=" , False, "ReduceMaxError:=" , 0, "ReducePercentage:=" , 100, "PointCoincidenceTol:=" , 1E-06, "CreateLightweightPart:=" , False, "ImportMaterialNames:=" , False,</pre>

	<pre>"SeparateDisjointLumps:=", False, "SourceFile:=" , "C:\\Users\\jdoe\\Desktop\\export.model")</pre>
--	---

ImportDXF [Modeler]

Imports a DXF model file.

UI Access	Modeler > Import.								
Parameters	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><Parameters></td> <td>Array</td> <td> <p>Structured array.</p> <pre>Array("NAME:options", "FileName:=" , <string>, "Scale:=" , <float>, "AutoDetectClosed:=" , <boolean>, "SelfStitch:=" , <boolean>, "DefeatureGeometry:=" , <boolean>, "DefeatureDistance:=" , <integer>, "RoundCoordinates:=" , <boolean>, "RoundNumDigits:=" , <integer>, "WritePolyWithWidthAsFilledPoly:=" , <boolean>, "ImportMethod:=" , <integer>, "2DSheetBodies:=" , <boolean>, <layersArray>)</pre> </td> </tr> </tbody> </table>	Name	Type	Description	<Parameters>	Array	<p>Structured array.</p> <pre>Array("NAME:options", "FileName:=" , <string>, "Scale:=" , <float>, "AutoDetectClosed:=" , <boolean>, "SelfStitch:=" , <boolean>, "DefeatureGeometry:=" , <boolean>, "DefeatureDistance:=" , <integer>, "RoundCoordinates:=" , <boolean>, "RoundNumDigits:=" , <integer>, "WritePolyWithWidthAsFilledPoly:=" , <boolean>, "ImportMethod:=" , <integer>, "2DSheetBodies:=" , <boolean>, <layersArray>)</pre>		
Name	Type	Description							
<Parameters>	Array	<p>Structured array.</p> <pre>Array("NAME:options", "FileName:=" , <string>, "Scale:=" , <float>, "AutoDetectClosed:=" , <boolean>, "SelfStitch:=" , <boolean>, "DefeatureGeometry:=" , <boolean>, "DefeatureDistance:=" , <integer>, "RoundCoordinates:=" , <boolean>, "RoundNumDigits:=" , <integer>, "WritePolyWithWidthAsFilledPoly:=" , <boolean>, "ImportMethod:=" , <integer>, "2DSheetBodies:=" , <boolean>, <layersArray>)</pre>							

	<code><layersArray></code>	Array	Structured array. Array ("NAME:LayerInfo", <layer>, <layer>, <layer>, ...)
	<code><layer></code>	Array	Structured array. Array ("NAME:<layerName>", "source:=" , <string>, "display_source:=" , <string>, "import:=" , <boolean>, "dest:=" , <string>, "dest_selected:=" , <boolean>, "layer_type:=" , <string>)
Return Value	None.		

Python Syntax	<code>ImportDXF(<Parameters>)</code>
Python Example	<pre>oEditor.ImportDXF(["NAME:options", "FileName:=", "C:/Users/jdoe/Desktop/export.dxf", "Scale:=" , 0.001, "AutoDetectClosed:=" , True, "SelfStitch:=" , True, "DefeatureGeometry:=" , False,</pre>

```
"DefeatureDistance:=" , 0,
"RoundCoordinates:=" , False,
"RoundNumDigits:=" , 4,
"WritePolyWithWidthAsFilledPoly:=", False,
"ImportMethod:=" , 1,
"2DSheetBodies:=" , False,
[ "NAME:LayerInfo",
  [ "NAME:0",
    "source:=" , "0",
    "display_source:=" , "0",
    "import:=" , True,
    "dest:=" , "0",
    "dest_selected:=" , False,
    "layer_type:=" , "signal"
  ],
  [ "NAME:LAYER_1",
    "source:=" , "LAYER_1",
    "display_source:=" , "LAYER_1",
    "import:=" , True,
    "dest:=" , "LAYER_1",
  ]
]
```

```
"dest_selected:=" , False,
"layer_type:=" , "signal"
] ,
["NAME: LAYER_2",
"source:=" , "LAYER_2",
"display_source:=" , "LAYER_2",
"import:=" , True,
"dest:=" , "LAYER_2",
"dest_selected:=" , False,
"layer_type:=" , "signal"
]
])
```

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 </pre>

```
( "NAME:0", "source:=", _  
    "0", "display_source:=", "0", "import:=", true, "dest:=", "0", "dest_selected:=", _  
    false, "layer_type:=", "signal"), Array("NAME:LAYER_1", "source:=", "LAYER_1", "dis-  
    play_source:=", _  
    "LAYER_1", "import:=", true, "dest:=", "LAYER_1", "dest_selected:=", false, "layer_  
    type:=", _  
    "signal"), Array("NAME:LAYER_2", "source:=", "LAYER_2", "display_source:=", "LAYER_2",  
    "import:=", _  
    true, "dest:=", "LAYER_2", "dest_selected:=", false, "layer_type:=", "signal")))
```

ImportFromClipboard

Imports a model from clipboard.

UI Access	Modeler > Import From Clipboard.
Parameters	None.
Return Value	None.

Python Syntax	ImportFromClipboard ()
Python Example	oEditor.ImportFromClipboard()

VB Syntax	ImportFromClipboard
VB Example	oEditor.ImportFromClipboard

ImportGDSII [Modeler]

Imports a GDSII model file.

UI Access	Modeler > Import.		
Parameters	Name <i><Parameters></i>	Type Array	Description <p>Structured array.</p> <pre>Array("NAME:options", "FileName:=" , <string>, "FlattenHierarchy:=" , <boolean>, "ImportMethod:=" , <integer>, <layerMapArray>, <layerMapArray>, <layerMapArray>,...) "OrderMap:=" , ["entry:=" , <entry>, <entry>, <entry>,...]])</pre>
	<i><layerMapArray></i>	Array	Structured array. <pre>Array("NAME:LayerMapInfo", "LayerNum:=" , <integer>, "DestLayer:=" , <string>, "layer_type:=" , <string>)</pre>
	<i><entry></i>	Array	Structured array. <pre>Array("order:=" , <integer LayerNum>, "layer:=" ,</pre>

		<string DestLayer>
Return Value	None.	

Python Syntax	ImportGDSII(<Parameters>)
Python Example	<pre> oEditor.ImportGDSII (["NAME:options", "FileName:=", "C:/Users/coil2.gds", "FlattenHierarchy:=", True, "ImportMethod:=", 1, ["NAME:LayerMap", ["NAME:LayerMapInfo", "LayerNum:=", 12, "DestLayer:=", "Signal12", "layer_type:=", "signal"], ["NAME:LayerMapInfo", "LayerNum:=", 13, "DestLayer:=", "Signal13", "layer_type:=", "signal"],],],) </pre>

```

        [ "NAME:LayerMapInfo",
          "LayerNum:=" , 14,
          "DestLayer:=" , "Signal14",
          "layer_type:=" , "signal"
        ]
      ],
      "OrderMap:=",
      [
        "entry:=",
        [ "order:=", 0, "layer:=", "Signal12"],
        "entry:=",
        [ "order:=", 1, "layer:=", "Signal13"],
        "entry:=",
        [ "order:=", 2, "layer:=", "Signal14"]
      ]
    ]
  )
)

```

VB Syntax	ImportGDSII <Parameters>
VB Example	<pre> oEditor.ImportGDSII Array("NAME:options", "FileName:=", "C:/export.gds", "FlattenHierarchy:=", True, </pre>

```

    "ImportMethod:=", 1,
    Array("NAME:LayerMap",
        Array ("NAME:LayerMapInfo",
            "LayerNum:=", 1,
            "DestLayer:=", "Signal1",
            "layer_type:=", "signal")))
    "OrderMap:=", Array(
        "entry:=", Array(
            "order:=", 0,
            "layer:=", "Signal1")))

```

Imprint

Imprints the geometry of one object upon another.

UI Access	Modeler > Boolean > Imprint....		
Parameters	Name <i><Selections></i>	Type Array	Description Structured array. Array ("NAME:Selections", "Blank Parts:=", <string, object name>, "Tool Parts:=", <string, object name>)
	<i><Parameters></i>	Array	Structured array. Array ("NAME:ImprintParameters",

		"KeepOriginals:=", <boolean>)
Return Value	None.	

Python Syntax	Imprint (<Selections>, <Parameters>)
Python Example	<pre> oEditor.Imprint(["NAME:Selections", "Blank Parts:=", "Cylinder1", "Tool Parts:=", "Box1"], ["NAME:ImprintParameters", "KeepOriginals:=", False]) </pre>

VB Syntax	Imprint <Selections>, <Parameters>
VB Example	<pre> oEditor.Imprint Array("NAME:Selections", "Blank Parts:=", "Cylinder1", "Tool Parts:=", "Box1"), Array("NAME:ImprintParameters", "KeepOriginals:=", false) </pre>

ImprintProjection

Projects the form of a sheet object onto the face or faces of another object (either solid or sheet).

UI Access	Modeler > Boolean > Imprint Projection.		
Parameters	Name	Type	Description
	<Selections>	Array	<p>Structured array.</p> <pre>Array("NAME:Selections", "Selections:=", <string, selected objects>)</pre>
Return Value			None.

Python Syntax	ImprintProjection (<Selections>, <Parameters>)
Python Example	<pre> oEditor.ImprintProjection(["NAME:Selections", "Selections:=", "Rectangle1,Cylinder1"], ["NAME:ImprintProjectionParameters",])</pre>

```

    "KeepOriginals:=", False,
    "NormalProjection:=", True,
    "Distance:=", "1.36014705087354mm",
    "DirectionX:=", "0.882257546512569",
    "DirectionY:=", "0.294085848837523",
    "DirectionZ:=", "0.367607311046904"])

```

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 <i><SelectionsArray></i>	Type Array	Description Structured array. See: SelectionsArray .
	<i><Parameters></i>	Array	Structured array. Array ("NAME:IntersectParameters", "KeepOriginals:=" , <boolean True to keep original objects; False to delete>)
Return Value	None.		

Python Syntax	Intersect(<SelectionsArray>, <Parameters>)
Python Example	<pre> oEditor.Intersect(["NAME:Selections", "Selections:=", "Rectangle1,Rectangle2"], ["NAME:IntersectParameters", "KeepOriginals:=", False]) </pre>

VB Syntax	Intersect <SelectionsArray> <Parameters>
VB Example	<pre> oEditor.Intersect Array ("NAME:Selections", </pre>

	<pre>"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 <i><SelectionsArray></i>	Type Array	Description 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 <code><Objects></code>	Type Array	Description Structured array. Selects the entity/entities to move. <code>Array("Objects:=", <array containing string object IDs>)</code>
	<code><MoveEntityToGroup></code>	Array	Structured array. <code>Array("ParentGroup:=", <string group name>)</code>
Return Value	None.		

Python Syntax	<code>MoveEntityToGroup (<Objects>, <MoveEntityToGroup>)</code>
Python Example	<pre>oEditor.MoveEntityToGroup (["Objects:=", ["Box3"]], ["ParentGroup:=", "Box_Group"])</pre>

VB Syntax	<code>MoveEntityToGroup <Objects> <MoveEntityToGroup></code>
VB Example	<pre>oEditor.MoveEntityToGroup Array("Objects:=", Array("Box3"))</pre>

	<code>Array("ParentGroup:=", "Box_Group")</code>
--	--

MoveFaces

Moves the specified faces along normal or along a vector.

UI Access	Modeler > Surface > Move Faces > Along [Normal/Vector].		
	Name	Type	Description
	<code><SelectionsArray></code>	Array	Structured array. See: SelectionsArray .
Parameters	<code><MoveFacesParametersArray></code>	Array	<p>Structured array.</p> <pre>Array ("NAME:Parameters", <FacesOfOneObjToMove>, <FacesOfOneObjToMove>, ...)</pre>
	<code><FacesOfOneObjToMove></code>	Array	<p>Structured array.</p> <pre>Array ("Name:MoveFacesParameters", "MoveAlongNormalFlag:=", <boolean>, "OffsetDistance:=", <string>, "MoveVectorX:=", <string>, "MoveVectorY:=", <string>, "MoveVectorZ:=", <string>, "FacesToMove:=", <array>)</pre> <p><code>MoveAlongNormalFlag</code> specifies whether to move along the face normal or along a vector. If false, provide the <code>MoveVectorX</code>, <code>MoveVectorY</code>, and <code>MoveVectorZ</code> parameters.</p> <p><code>FacesToMove</code> is an array of integers (the IDs of the faces to move).</p>

Return Value	None
---------------------	------

Python Syntax	MoveFaces (<SelectionsArray>, <MoveFacesParametersArray>)
Python Example	<pre> oEditor.MoveFaces(["NAME:Selections", "Selections:=" , "Rectangle1", "NewPartsModelFlag:=" , "Model"], ["NAME:Parameters", ["NAME:MoveFacesParameters", "MoveAlongNormalFlag:=" , True, "OffsetDistance:=" , "1mm", "MoveVectorX:=" , "0mm", "MoveVectorY:=" , "0mm", "MoveVectorZ:=" , "0mm", "FacesToMove:=" , [183]]]) </pre>

VB Syntax	MoveFaces <SelectionsArray>, <MoveFacesParametersArray>
VB Example	<pre> oEditor.MoveFaces _ Array("NAME:Selections", "Selections:=", _ </pre>

```

    "Box2,Box1"), _

    Array("NAME:Parameters", _

    Array("NAME:MoveFacesParameters", _

        "MoveAlongNormalFlag:=", true, _

        "OffsetDistance:=", "1mm", _

        "FacesToMove:=", Array(218)),_

        Array("NAME:MoveFacesParameters", _

            "MoveAlongNormalFlag:=", false,_

            "OffsetDistance:=", "1mm", _

            "MoveVectorX:=", "1mm", _

            "MoveVectorY:=", "0mm", _

            "MoveVectorZ:=", "0mm", _

            "FacesToMove:=", Array(185)))

```

ProjectSheet

Project a sheet object, typically for modeling thin conformal deposits. Typically followed by **Thicken Sheet**.

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 <SelectionsArray>	Type Array	Description Structured array. See: SelectionsArray .

Return Value	None.
---------------------	-------

Python Syntax	PurgeHistory (<SelectionsArray>)
Python Example	<pre> oEditor.PurgeHistory(["NAME:Selections", "Selections:=", "Box2", "NewPartsModelFlag:=", "Model"]) </pre>

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 <GeometryData> <DesignData>	Type Array Array	Description Structured array. Structured array.

	<ImageFile>	Array	Structured array.
Return Value	None.		

Python Syntax	ReplaceWith3DComponent(<GeometryData>, <DesignData>, <ImageFile>)
Python Example	<pre> oEditor.ReplaceWith3DComponent (["NAME:ReplaceData", "ComponentName:=", "CoaxBend", "Company:=", "", "Company URL:=", "", "Model Number:=", "", "Help URL:=", "", "Version:=", "1.0", "Notes:=", "", "IconType:=", "", "Owner:=", "Jane Doe", "Email:=", "jdoe@email.com", "Date:=", "3:46:31 PM Jul 26, 2020", "HasLabel:=", False, "IncludedParts:=", ["outer", "teflon", "inner", "teflon_1"],]) </pre>

```

    "HiddenParts:=", [],
    "IncludedCS:=", [],
    "ReferenceCS:=", "Global",
    "IncludedParameters:=", ["bend_angle"],
    "ParameterDescription:=", ["bend_angle:=", ""]
],
["NAME:DesignData",
    "Excitations:=", ["1","2"]
],
["NAME:ImageFile",
    "ImageFile:=", ""
]
)

```

VB Syntax	ReplaceWith3DComponent <GeometryData>, <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", _ </pre>

```

    "HasLabel:=", false, "IsEncrypted:=", false, "AllowEdit:=", false, _
    "SecurityMessage:=", "", "Password:=", "", "EditPassword:=", "", _
    "PasswordType:=", "UnknownPassword", "HideContents:=", true, "ReplaceNames:=", true, _
    "ComponentOutline:=", "None", "IncludedParts:=", _
    Array("pin", "solderl", "ConnectorDielectric", _
    "ConnectorOuter", "pcap", "solderr"), "HiddenParts:=", Array(), _
    "IncludedCS:=", Array(), "ReferenceCS:=", _
    "Global", "IncludedParameters:=", Array("lcon", "lpin", "rpin"), _
    "ParameterDescription:=", Array("lcon:=", "", "lpin:=", "", "rpin:=", ""), "IsLi-
    censed:=", false, _
    "LicensingDllName:=", "", "VendorComponentIdentifier:=", _
    "", "PublicKeyFile:=", ""), Array("NAME:DesignData", "Boundaries:=", Array( _
    "PinSurface", "Connect"), "Excitations:=", Array("1", "2"), "MeshOperations:=", Array(
    -
    "Length1", "SkinDepth1")), "", Array("NAME:ImageFile", "ImageFile:=", ""))

```

Section

Creates a 2D cross-section of the selection in the specified plane.

UI Access	Modeler > Surface > Section.		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: 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, "SectionPlane:=" , "XY", "SectionCrossObject:=" , False) </pre>
--

SeparateBody

Separates the bodies of specified multi-lump objects.

UI Access	Modeler > Boolean > Separate Bodies.						
Parameters	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 33%;">Name</th> <th style="width: 33%;">Type</th> <th style="width: 33%;">Description</th> </tr> </thead> <tbody> <tr> <td style="height: 20px;"><i><SelectionsArray></i></td> <td style="height: 20px;">Array</td> <td>Structured array. See: SelectionsArray.</td> </tr> </tbody> </table>	Name	Type	Description	<i><SelectionsArray></i>	Array	Structured array. See: SelectionsArray .
Name	Type	Description					
<i><SelectionsArray></i>	Array	Structured array. See: SelectionsArray .					
Return Value	None.						

Python Syntax	SeparateBody (<SelectionsArray>)
Python Example	<pre> oEditor.SeparateBody(["NAME:Selections", "Selections:=", "Rectangle1,Circle1",] </pre>

	<pre>"NewPartsModelFlag:=", "Model"])</pre>
--	--

VB Syntax	SeparateBody <SelectionsArray>
VB Example	<pre>oEditor.SeparateBody Array("NAME:Selections", "Selections:=", "Rectangle1,Circle1", "NewPartsModelFlag:=", "Model")</pre>

SetModelUnits

Sets the model units.

UI Access	Modeler > Units.						
Parameters	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><Parameters></td> <td>Array</td> <td> <p>Structured array.</p> <pre>Array("NAME:Units Parameter", "Units:=", <string>, "Rescale:=", <boolean True to rescale model; else False>)</pre> <p>To see valid unit strings, select Modeler > Units.</p> </td> </tr> </tbody> </table>	Name	Type	Description	<Parameters>	Array	<p>Structured array.</p> <pre>Array("NAME:Units Parameter", "Units:=", <string>, "Rescale:=", <boolean True to rescale model; else False>)</pre> <p>To see valid unit strings, select Modeler > Units.</p>
Name	Type	Description					
<Parameters>	Array	<p>Structured array.</p> <pre>Array("NAME:Units Parameter", "Units:=", <string>, "Rescale:=", <boolean True to rescale model; else False>)</pre> <p>To see valid unit strings, select Modeler > Units.</p>					
Return Value	None.						

Python Syntax	SetModelUnits (<Parameters>)
Python Example	<pre> oEditor.SetModelUnits(["NAME:Units Parameter", "Units:=", "km", "Rescale:=", False]) </pre>

VB Syntax	SetModelUnits <Parameters>
VB Example	<pre> oEditor.SetModelUnits Array("NAME:Units Parameter", "Units:=", "km", "Rescale:=", False) </pre>

SetWCS

Sets the working coordinate system.

UI Access	Modeler > Coordinate System > Set Working CS.		
Parameters	Name <Parameters>	Type Array	Description Structured array. Array ("NAME:SetWCS Parameter", "Working Coordinate System:=", <string CS name> ,

			"RegionDepCSOk:=" , <boolean True if region-dependent; else False)
Return Value	None.		

Python Syntax	SetWCS (<Parameters>)
Python Example	<pre> oEditor.SetWCS (["NAME:SetWCS Parameter", "Working Coordinate System:=", "Global", "RegionDepCSOk:=", False]) </pre>

VB Syntax	SetWCS <Parameters>
VB Example	<pre> oEditor.SetWCS Array("NAME:SetWCS Parameter", "Working Coordinate System:=", "Global", "RegionDepCSOk:=", False) </pre>

ShowWindow

Opens the active 3D Modeler window.

UI Access	None.
------------------	-------

Parameters	None.
Return Value	None.

Python Syntax	ShowWindow
Python Example	<code>oEditor.ShowWindow()</code>

VB Syntax	ShowWindow
VB Example	<code>oEditor.ShowWindow</code>

Simplify

Converts a complex MCAD object into simpler primitives which are easy to mesh and solve.

UI Access	Modeler > Model Preparation > Simplify.		
Parameters	Name <code><Selections></code>	Type Array	Description Structured array. See: SelectionsArray .

			<pre>"Splitting:=", <boolean>, "SeparateBodies:=", <boolean>, "CloneBody:=", <boolean>, "Generate Primitive History:=", <boolean>, "NumberPointsCurve:=", <integer>, "LengthThresholdCurve:=", <integer>)</pre>
	<p><CreateGroup></p>	Array	<p>Structured array.</p> <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,</pre>

```
        "CloneBody:=", False,
        "Generate Primitive History:=", True,
        "NumberPointsCurve:=", 3,
        "LengthThresholdCurve:=", 20],
    ["CreateGroupsForNewObjects:=", True]
)
```

VB Syntax	Simplify <Selections>, <Parameters>, <CreateGroup>
VB Example	<pre>oEditor.Simplify Array("NAME:Selections", "Selections:=", "Cylinder1,Box2", "NewPartsModelFlag:=", "Model"), Array("NAME:SimplifyParameters", "Type:=", "Polygon Fit", "ExtrusionAxis:=", "Auto", "Cleanup:=", true, "Splitting:=", true, "SeparateBodies:=", false, "CloneBody:=", false,</pre>

```

    "Generate Primitive History:=", true,
    "NumberPointsCurve:=", 3,
    "LengthThresholdCurve:=", 20),
Array("CreateGroupsForNewObjects:=", true)

```

Split

Splits the specified object(s) along a plane.

UI Access	Modeler > Boolean > Split.		
	Name <i><SelectionsArray></i>	Type Array	Description Structured array. See: SelectionsArray .
Parameters	<i><Parameters></i>	Array	<p>Structured array.</p> <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> <p>You can split an object either along an existing plane , or by creating a plane using an edge.</p>

			<p>For existing plane:</p> <ul style="list-style-type: none"> • SplitPlane is "XY", "YZ", or "ZX" • ToolType is "PlaneTool" • ToolEntityID is -1 <p>For an edge:</p> <ul style="list-style-type: none"> • 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",]) </pre>

```

    "ToolEntityID:="           , -1,
    "SplitCrossingObjectsOnly:=", False,
    "DeleteInvalidObjects:=", True
]
)

```

VB Syntax	Split <SelectionsArray> <Parameters>
VB Example	<pre> oEditor.Split Array("NAME:Selections", "Selections:=" , "Box1", "NewPartsModelFlag:=" , "Model") Array("NAME:SplitToParameters", "SplitPlane:=" , "XY", "WhichSide:=" , "PositiveOnly", "ToolType:=" , "PlaneTool", "ToolEntityID:=" , -1, "SplitCrossingObjectsOnly:=", False, "DeleteInvalidObjects:=", True) </pre>

Stitch

Stitches selected sheets.

UI Access

Modeler > Model Preparation > Stitch Sheets.

Parameters	Name	Type	Description
	<Selections>	Array	Structured array. See: SelectionsArray .
	<Parameters>	Array	Structured array. Array ("NAME:StitchParameters", "MaxTol:=", <integer maximum tolerance>)
Return Value	None.		

Python Syntax	Stitch (<Selections>, <Parameters>)
Python Example	<pre> oEditor.Stitch(["NAME:Selections", "Selections:=", "Rectangle3,Rectangle4", "NewPartsModelFlag:=", "Model"], ["NAME:StitchParameters", "MaxTol:=", -1]) </pre>

VB Syntax	Stitch <Selections>, <Parameters>
VB Example	<pre> oEditor.Stitch Array("NAME:Selections", </pre>

	<pre>"Selections:=", "Rectangle3,Rectangle4", "NewPartsModelFlag:=", "Model"), Array("NAME:StitchParameters", "MaxTol:=", -1)</pre>

Subtract

Subtracts the specified object(s).

UI Access	Modeler > Boolean > Subtract.		
Parameters	Name <i><Selections></i>	Type Array	Description <p>Structured array.</p> <pre>Array("NAME:Selections", "Blank Parts:=" , <string object name(s)>, "Tool Parts:=" , <string object name(s)>)</pre> <p>The Tool Parts object is the object being removed.</p> <p>The Blank Parts object has any Tool Parts overlap removed.</p> <p>Either string can contain more than one object.</p>
	<i><Parameters></i>	Array	<p>Structured array.</p> <pre>Array("NAME:SubtractParameters", "KeepOriginals:=" , <boolean>)</pre>
Return Value	None.		

Python Syntax	Subtract(<Selections>, <Parameters>)
Python Example	<pre>oEditor.Subtract(["NAME:Selections", "Blank Parts:=", "Rectangle1", "Tool Parts:=", "Rectangle2"], ["NAME:SubtractParameters", "KeepOriginals:=", False])</pre>

VB Syntax	Subtract <Selections> <Parameters>
VB Example	<pre>oEditor.Subtract Array("NAME:Selections", "Blank Parts:=", "Rectangle1", "Tool Parts:=", "Rectangle2") Array("NAME:SubtractParameters", "KeepOriginals:=", false)</pre>

SweepFacesAlongNormal

Sweep the specified face(s) along normal.

UI Access	Modeler > Surface > Sweep Faces Along Normal		
	Name	Type	Description
Parameters	<SelectionsArray>	Array	Structured array. See: SelectionsArray .
Return Value	None		

Python Syntax	SweepFacesAlongNormal(<SelectionsArray> <parameters>)
Python Example	<pre> oEditor.SweepFacesAlongNormal (["NAME:Selections", "Selections:=", "Rectangle1", "NewPartsModelFlag:=", "Model"], ["NAME:Parameters", "NAME:SweepFaceAlongNormalToParameters", "FacesToDetach:=", [183], "LengthOfSweep:=", "0.1mm"]) </pre>

VB Syntax	SweepFacesAlongNormal <SelectionsArray> <parameters>
VB Example	<pre> oEditor.SweepFacesAlongNormal Array ("NAME:Selections", "Selections:=", "Rectangle1", "NewPartsModelFlag:=", "Model"), Array ("NAME:Parameters", "NAME:SweepFaceAlongNormalToParameters", "FacesToDetach:=", Array(57), "LengthOfSweep:=", "0.5mm")) </pre>

ThickenSheet

Thickens a sheet object to convert it to a 3D object.

UI Access	Modeler > Surface > Thicken Sheet		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: SelectionsArray .
Return Value	None.		

Python Syntax	ThickenSheet (<SelectionsArray> <parameters>)
Python Example	<pre> oEditor.ThickenSheet(["NAME:Selections", "Selections:=", "NewPartsModelFlag:=",], ["NAME:SheetThickenParameters", "Thickness:=", "BothSides:=",])) </pre>

VB Syntax	ThickenSheet <SelectionsArray> <parameters>
VB Example	<pre> oEditor.ThickenSheet Array("NAME:Selections", "Selections:=", "NewPartsModelFlag:="), Array("NAME:SheetThickenParameters", "Thickness:=", "BothSides:=")) </pre>

UncoverFaces

Uncovers the specified face(s).

UI Access	Modeler > Surface > Uncover Faces		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: SelectionsArray .
Return Value		None.	

Python Syntax	UncoverFaces(<SelectionsArray> <parameters>)
Python Example	<pre>oEditor.UncoverFaces (["NAME:Selections", "Selections:=" , "Box1", "NewPartsModelFlag:=" , "Model"], ["NAME:Parameters", ["NAME:UncoverFacesParameters", "FacesToUncover:=" , <array of face IDs>)])</pre>

```

    "FacesToUncover:="      , [12,16,18]
]
])

```

VB Syntax	UncoverFaces <SelectionsArray> <parameters>
VB Example	<pre> oEditor.UncoverFaces Array("NAME:Selections", "Selections:=" , "Box1", "NewPartsModelFlag:=" , "Model") Array("NAME:Parameters", Array("NAME:UncoverFacesParameters", "FacesToUncover:=" , Array(12,16,18))) </pre>

Unite

Unites the specified objects.

UI Access	Modeler > Boolean > Unite		
	Name	Type	Description
Parameters	<SelectionsArray>	Array	Structured array. See: SelectionsArray .
	<parameters>	Array	Structured array. Array ("NAME:UniteParameters", "KeepOriginals:=" , <boolean>)

Return Value	None.
---------------------	-------

Python Syntax	Unite(<SelectionsArray>, <parameters>)
Python Example	<pre>oEditor.Unite(["NAME:Selections", "Selections:=", "Rectangle1,Rectangle2"], ["NAME:UniteParameters", "KeepOriginals:=", False])</pre>

VB Syntax	Unite <SelectionsArray> <parameters>
VB Example	<pre>oEditor.Unite Array("NAME:Selections", "Selections:=", "Rectangle1,Rectangle2") Array("NAME:UniteParameters", "KeepOriginals:=", false)</pre>

Ungroup

Ungroups a specified history tree group.

UI Access	Modeler > Group > Ungroup		
Parameters	Name <i><Groups></i>	Type Array	Description Structured array. Array("Groups:=", <array of group names to ungroup>)
Return Value	None		

Python Syntax	Ungroup(<i><Groups></i>)
Python Example	<code>oEditor.Ungroup(["Groups:=", ["Group1"]])</code>

VB Syntax	Ungroup <i><Groups></i>
VB Example	<code>oEditor.Ungroup Array("Groups:=", Array("Group1"))</code>

WrapSheet

Wraps a sheet object to another object.

UI Access	None.		
Parameters	Name <i><SelectionsArray></i>	Type Array	Description Structured array. See: SelectionsArray .
	<i><Parameters></i>	Array	Description Structured array. Array ("NAME:WrapSheetParameters", "Imprinted:=", <boolean>)
Return Value	None.		

Python Syntax	WrapSheet (<SelectionsArray>, <Parameters>)
Python Example	<pre>oEditor.WrapSheet(["NAME:Selections", "Selections:=", "Rectangle1,Box1"], ["NAME:WrapSheetParameters", "Imprinted:=", True])</pre>

VB Syntax	WrapSheet <SelectionsArray> <Parameters>
VB Example	<pre>oEditor.WrapSheet Array("NAME:Selections", "Selections:=", "Rectangle1,Box1"), Array("NAME:WrapSheetParameters", "Imprinted:=", true)</pre>

Other oEditor Commands

[AddDefinitionFromBlock](#)

[AddDefinitionFromLibFile](#)

[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](#)
[GetObject NameByFaceID](#)
[GetObjectsByMaterial](#)
[GetObjectsInGroup](#)
[GetObjectVolume](#)
[GetProperty Value](#)
[GetPropEvaluatedValue](#)
[GetPropNames](#)
[GetPropSIValue](#)
[GetPropValue](#)
[GetSelections](#)
 [GetUserPosition](#)
[GetVertexIDsFromEdge](#)
[GetVertexIDsFromFace](#)
[GetVertexIDsFromObject](#)
[GetVertexPosition](#)
[OpenExternalEditor](#)

[PageSetup](#)[RenamePart](#)[SetPropValue \[Modeler\]](#)

AddDefinitionFromBlock

Adds a material definition from block text (same definition format as would be contained in the material library file) by library type (using definition folder name). This scripting command directly supports the .AMAT (or .ASURF) definition formats.

UI Access	N/A		
Parameters	Name	Type	Description
	<defBlock>	String	Text of the new material definition in block form.
	<defFolderName>	String	Library type (by definition folder name)
	<newTimeStamp>	String	New timestamp (time_t as integer number of seconds since 1/1/1970 12:00am, as string), default is current time
	<replaceExisting>	Boolean	True to replace existing, False to choose a new unique name if an existing definition is found
Return Value	A property scripting object for the definition.		

P-y-t-h-o-n-S-y-n-t-a-x	AddDefinitionFromBlock(<defBlock>,<defFolderName>,<newTimeStamp>,<replaceExisting>)
-------------------------	---

```
oProject = oDesktop.NewProject()
oProject.InsertDesign("HFSS", "HFSSDesign1", "DrivenModal", "")
oDesign = oProject.SetActiveDesign("HFSSDesign1")
oEditor = oDesign.SetActiveEditor("3D Modeler")
oEditor.CreateBox(
    [
        P-
        yt-
        h-
        o-
        n-
        E-
        x-
        a-
        m-
        pl-
        e
        "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  
  
        ]  
  
    ]  
]
```

])
--	--	---	---

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") oEditor = oDesign.SetActiveEditor("3D Modeler") </pre>

```
oEditor.CreateBox(  
    [  
        "NAME:BoxParameters",  
  
        "XPosition:=" , "-0.4mm",  
        "YPosition:=" , "-1mm",  
        "ZPosition:=" , "0mm",  
        "XSize:=" , "1.4mm",  
        "YSize:=" , "1.6mm",  
        "ZSize:=" , "0.6mm"  
    ],  
    [  
        "NAME:Attributes",  
        "Name:=" , "Box1",  
        "Flags:=" , "",  
        "Color:=" , "(143 175 143)",  
        "Transparency:=" , 0,  
    ]
```

```
"PartCoordinateSystem:=", "Global",

"UDMId:=" , "",

"MaterialValue:=" , "\\"vacuum\\\"",

"SurfaceMaterialValue:=" , "\\"\\\"",

"SolveInside:=" , True,

"ShellElement:=" , False,

"ShellElementThickness:=" , "0mm",

"IsMaterialEditable:=" , True,

"UseMaterialAppearance:=" , False,

"IsLightweight:=" , False

])

oDefinitionManager = oProject.GetDefinitionManager()

scriptDir = os.path.dirname(os.path.realpath(__file__))

amatFilePath = scriptDir + '/material0.txt'
```

```
materialName = "material0"

materialNameInQuotes = "\"" + materialName + "\""

added = oDefinitionManager.AddDefinitionFromLibFile(amatFilePath, materialName, "Materials", "")

addedName = ''

    if isinstance(added, basestring):

addedName = added

    elif isinstance(added, list):

addedName = added[0]

else:

    addedName = added.GetName().replace("Materials:", "")

AddInfoMessage(os.path.basename(__file__) + " result: " + addedName)

oEditor.ChangeProperty(
    [
        "NAME:AllTabs",
        [
            "NAME:Geometry3DAttributeTab",
            [
                "NAME:PropServers",
                "Box1"
            ],
            [

```

```

        "NAME:ChangedProps",
        [
            "NAME:Material",
            "Value:=", materialNameInQuotes
        ]
    ]
)

```

AddViewOrientation

Creates a new orientation using a script. Input parameters can be *either* vector components *or* angles. Local view orientation specific to a design can be saved in project by calling `Save()` on the project object.

Note:

You cannot use the name of an existing view. Default views include: top, bottom, right, left, front, back, trimetric, dimetric, and isometric.

UI Access	View > Modify Attributes > Orientation List. Enter parameters and click Add.		
Parameters	Name	Type	Description
	<code><orientationName></code>	String	Name of the new orientation.
	<code><isGlobalOrientation></code>	Long	1 to save global; else 0.
	<code><OrientationVectorComponents></code>	Array	Structured array. Use <i>either</i> <code><OrientationVectorComponents></code> <i>or</i> <code><OrientationAnglesArray></code> .

		<pre>Array("NAME:OrientationVectorComponents", Array("NAME:Direction", <float>, <float>, <float>, Array("NAME:Up", <float>, <float>, <float>))</pre>
<i><OrientationAnglesArray></i>	Array	<p>Structured array. Use <u>either</u> <i><OrientationVectorComponents></i> <u>or</u> <i><OrientationAnglesArray></i>.</p> <pre>Array("NAME:OrientationAngles", "Psi:=" , <float>, "Theta:=" , <float>, "Phi:=" , <float>)</pre>
<i><ProjectionParams></i>	Array	<p><i>Optional.</i> Structured array containing arrays of four optional parameters: Scaling, Translation, Projection Type, and Projection Limits.</p> <pre>Array("NAME:ProjectionParams", Array("NAME:Scaling", <float>, <float>, <float>)</pre>

			<pre>) , Array("NAME:Translation", <float>, <float>, <float>), "ProjectionType:=" , <string "Orthographic" or "Perspective">, Array("NAME:ProjectionLimits", <float>, <float>, <float>, <float>, <float>, <float>))</pre>
Return Value	None.		

Python Syntax	AddViewOrientation(<orientationName>, <isGlobalOrientation>, <OrientationVectorComponents or Ori-
----------------------	---

	<i>entationAnglesArray>, <ProjectionParams>)</i>
Python Example	<p>Example Using Angles and Projection Parameters:</p> <pre>oEditor.AddViewOrientation("orientation3", 0, ["NAME:OrientationAngles", "Psi:=" , -161.565002441406, "Theta:=" , 32.3115005493164, "Phi:=" , 0], ["NAME:ProjectionParams", ["NAME:Scaling", 0.591607987880707, 0.591607987880707, 0.591607928276062], ["NAME:Translation", 0, 2.38418579101563E-07, -5.7486834526062]])</pre>

```
        ],
        "ProjectionType": "Orthographic",
        [
            "NAME:ProjectionLimits",
            -2.44362282752991,
            2.44362282752991,
            -1,
            1,
            0.625207424163818,
            10.8721580505371
        ]
    ]
)
```

Example Using Vector without Projection Parameters:

```
oEditor.AddViewOrientation("orientation6", 0,
    [
        "NAME:OrientationVectorComponents",
        [
            "NAME:Direction",
            -0.801783978939056,
            -0.267262011766434,
```

```

        -0.534521996974945
    ],
    [
        "NAME:Up",
        -0.507091999053955,
        -0.169030994176865,
        0.845155000686646
    ]
)

```

VB Syntax	AddViewOrientation <orientationName>, <isGlobalOrientation>, <OrientationVectorComponents or OrientationAnglesArray>, <ProjectionParams>
VB Example	<p>Example Using Angles and Projection Parameters:</p> <pre> oEditor.AddViewOrientation "orientation3", 0, Array("NAME:OrientationAngles", "Psi:=" , -161.565002441406, "Theta:=" , 32.3115005493164, "Phi:=" , 0), Array("NAME:ProjectionParams", Array("NAME:Scaling", 0.591607987880707, </pre>

```
    0.591607987880707,  
  
    0.591607928276062  
) ,  
  
    Array ("NAME:Translation",  
  
    0,  
    2.38418579101563E-07,  
    -5.7486834526062  
) ,  
    "ProjectionType:="      , "Orthographic",  
  
    Array ("NAME:ProjectionLimits",  
  
    -2.44362282752991,  
    2.44362282752991,  
  
    -1,  
  
    1,
```

```
    0.625207424163818,  
    10.8721580505371  
)  
)
```

Example Using Vector without Projection Parameters:

```
oEditor.AddViewOrientation "orientation6", 0,  
Array("NAME:OrientationVectorComponents",  
    Array("NAME:Direction",  
        -0.801783978939056,  
        -0.267262011766434,  
        -0.534521996974945  
,  
    Array("NAME:Up",  
        -0.507091999053955,  
        -0.169030994176865,  
        0.845155000686646  
)  
)
```

AssignSurfaceMaterial

Assigns a material to specified surfaces.

UI Access	N/A									
Parameters	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><code><SelectionsArray></code></td> <td>Array</td> <td>Structured array. See: SelectionsArray.</td> </tr> <tr> <td><code><AttributesArray></code></td> <td>Array</td> <td>Structured array. See: AttributesArray.</td> </tr> </tbody> </table> <p>This script supports the following attributes:</p> <ul style="list-style-type: none"> • MaterialValue • SolveInside • ShellElement • ShellElementThickness • IsMaterialEditable • UseMaterialAppearance • IsLightweight 	Name	Type	Description	<code><SelectionsArray></code>	Array	Structured array. See: SelectionsArray .	<code><AttributesArray></code>	Array	Structured array. See: AttributesArray .
Name	Type	Description								
<code><SelectionsArray></code>	Array	Structured array. See: SelectionsArray .								
<code><AttributesArray></code>	Array	Structured array. See: AttributesArray .								
Return Value	None.									

Python Syntax	<code>AssignSurfaceMaterial(<SelectionsArray>, <AttributesArray>)</code>
----------------------	--

```
oEditor.AssignSurfaceMaterial(  
    ["NAME:Selections",  
        "AllowRegionDependentPartSelectionForPMLCreation:=", True,  
        "AllowRegionSelectionForPMLCreation:=", True,  
        "Selections:=", "Rectangle1"  
    ],  
    ["NAME:Attributes",  
        "MaterialValue:=", "diamond",  
        "SolveInside:=", False,  
        "ShellElement:=", False,  
        "ShellElementThickness:=", "nan",  
        "IsMaterialEditable:=", True,  
        "UseMaterialAppearance:=", False,  
        "IsLightweight:=", False  
    ])
```

VB Syntax

AssignSurfaceMaterial <SelectionsArray> <AttributesArray>

VB Example

```
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)
```

BreakUDMConnection

Breaks a current UDM connection to SpaceClaim.

UI Access	Break Connection.		
Parameters	Name <i><SelectionsArray></i>	Type Array	Description Structured array. See: SelectionsArray .
Return Value	None.		

Python Syntax	BreakUDMConnection (<SelectionsArray>)
Python Example	<pre> oEditor.BreakUDMConnection (["NAME:Selections", "Selections:=" , "SpaceClaim1"]) </pre>

VB Syntax	BreakUDMConnection <SelectionsArray>
VB Example	<pre> oEditor.BreakUDMConnection Array("NAME:Selections", "Selections:=", "") </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	<pre> oEditor.CloseAllWindows() </pre>

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

Defeature

Removes irrelevant features from a primitive.

UI Access	N/A									
Parameters	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td><Selections></td><td>Array</td><td>Structured array. See: SelectionsArray.</td></tr><tr><td><Options></td><td>Array</td><td>Structured array. Array ("NAME:options", "tolerance:=", <double containing tolerance value>, "fix:=", <boolean, if true, then self-intersections are fixed.>)</td></tr></tbody></table>	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.>)
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 Array("NAME:Selections", "Selections:=", "Box1"), Array("NAME:Options", "tolerance:=", 1E-006, "fix:=", true) </pre>

Delete

Deletes the specified object(s).

UI Access	Edit > Delete.						
Parameters	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><SelectionsArray></td> <td>Array</td> <td>Structured array. See: SelectionsArray.</td> </tr> </tbody> </table>	Name	Type	Description	<SelectionsArray>	Array	Structured array. See: SelectionsArray .
Name	Type	Description					
<SelectionsArray>	Array	Structured array. See: SelectionsArray .					
Return Value	None.						

Python Syntax	Delete(<SelectionsArray>)
Python Example	<pre> oEditor.Delete(["NAME:Selections", "Selections:=", "Rectangle1,Rectangle2"]) </pre>

VB Syntax	Delete <SelectionsArray>
------------------	--------------------------

VB Example

```
oEditor.Delete Array("NAME:Selections", "Selections:=", "Rectangle1,Rectangle2")
```

FitAll

Fits the design to the modeling area.

UI Access	View > Fit All > All Views.
Parameters	None.
Return Value	None.

Python Syntax

```
FitAll()
```

Python Example

```
oEditor.FitAll()
```

VB Syntax	FitAll()
VB Example	<pre>Set oEditor = oDesign.SetActiveEditor("3D Modeler") oEditor.FitAll()</pre>

GenerateAllUserDefinedModels

Generates all user defined models.

UI Access	N/A
Parameters	None.

Return Value	Array of models generated.
---------------------	----------------------------

Python Syntax	GenerateAllUserDefinedModels ()
Python Example	<code>oEditor.GenerateAllUserDefinedModels ()</code>

VB Syntax	GenerateAllUserDefinedModels
VB Example	<code>oEditor.GenerateAllUserDefinedModels</code>

GenerateUserDefinedModel

Generates specified user defined model(s).

UI Access	N/A		
Parameters	Name <code><SelectionsArray></code>	Type Array	Description Structured array. See: SelectionsArray .
Return Value	Array of models generated.		

Python Syntax	GenerateUserDefinedModel (<SelectionsArray>)
Python Example	<code>oEditor.GenerateUserDefinedModel (["NAME:Selections", "Selections:=", "model1, model2"])</code>

VB Syntax	GenerateAllUserDefinedModels
VB Example	<pre>oEditor.GenerateUserDefinedModel Array("NAME:Selections", "Selections:=", "model1, model2")</pre>

GeometryCheckAndAutofix

Use: Runs Geometry Check and optionally applies autofixes.

Command: HFSS 3D Layout > Geometry Check

Syntax: GeometryCheckAndAutofix <ChecksArray>,

```
    minimum_area_meters_squared,  
    <FixesArray>
```

Return Value: None

Parameters: <ChecksArray> - Array("NAME:checks", <check 1>, <check 2>, ..., <check n>)

Specify the checks that should be included. Specifying fewer checks may speed up execution but may also result in less problems reported in the message manager (or the logfile) and consequently less problems that can be fixed by autofixes.

The following are valid checks that can be specified:

- "Self-Intersecting Polygons"
- "Disjoint Nets (Floating Nodes)"
- "DC-Short Errors"
- "Identical/Overlapping Vias"
- "Misalignments"

There may be no checks, all 5 of the checks, or anything in between. The order that checks are specified in is not relevant.

minimum_area_meters_squared

Specify a decimal value for the minimum area (e.g. .000015) optionally in scientific notation (e.g. 2E-006). Cutouts smaller than this minimum area may be ignored during validation check.

<FixesArray> - Array("NAME:fixes", <fix 1>, <fix 2>, ..., <fix n>)

Specify the autofixes that should be applied if they are found by a check.

The following are valid fixes that can be specified:

- "Self-Intersecting Polygons"
- "Disjoint Nets",
- "Identical/Overlapping Vias"
- "Traces-Inside-Traces Errors"
- "Misalignments (Planes/Traces/Vias)"

There may be no fixes specified, all 5 fixes specified, or anything in between. The order that fixes are specified in is not relevant.

Example:

```
oEditor.GeometryCheckAndAutofix _  
    Array ("NAME:checks", "Self-Intersecting Polygons", _  
        "Disjoint Nets (Floating Nodes)", "DC-Short Errors", _  
        "Identical/Overlapping Vias", "Misalignments"), _  
        "minimum_area_meters_squared:=", 2E-006, _
```

```
Array("NAME:fixes", "Self-Intersecting Polygons", _  
    "Disjoint Nets", "Identical/Overlapping Vias", _  
    "Traces-Inside-Traces Errors", _  
    "Misalignments (Planes/Traces/Vias)")
```

GetBodyNamesByPosition

Returns the names of objects that contact a specified point.

UI Access	N/A						
Parameters	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td><positionParameters></td><td>Array</td><td>Structured array containing position coordinates for active coordinate system. Array ("NAME:Parameters", "XPosition:=", <string>, "YPosition:=", <string>, "ZPosition:=", <string>)</td></tr></tbody></table>	Name	Type	Description	<positionParameters>	Array	Structured array containing position coordinates for active coordinate system. Array ("NAME:Parameters", "XPosition:=", <string>, "YPosition:=", <string>, "ZPosition:=", <string>)
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",</pre>

```

    "XPosition:=", "0mm",
    "YPosition:=", "15mm",
    "ZPosition:=", "0mm"
]
)

```

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 <category>	Type String	Description <i>Optional.</i> Passing no input returns the list of possible strings:

			<ul style="list-style-type: none"> • "AllParts" – Returns the names of all parts. • "CoordinateSystems" – Returns the names of all coordinate systems. • "Groups" – Returns the names of all groups. • "Lists" – Returns the names of all lists. • "ModelParts" – Returns names of model parts. • "NonModelParts" – Returns the names of non-model parts. • "Planes" – Returns the names of all planes. • "Points" – Returns the names of all points. • "SubmodelDefinitions" – Returns the names of sub-model definitions.
Return Value	Array containing object names in the selected category.		

Python Syntax	GetChildNames(<category>)
Python Example	<p>Standalone Example:</p> <pre>oEditor.GetChildNames ("ModelParts") oEditor.GetChildNames ("Points")</pre> <p>Example used in Conjunction with GetChildObject and GetPropNames:</p> <pre>oDesign = oProject.GetActiveDesign()</pre>

```

oModel = oDesign.GetChildObject("3D Modeler")
oModel.GetChildTypes()



- This returns an array containing strings: "ModelParts", "AllParts", "NonModelParts", "CoordinateSystems", "Points", "Planes", "SubmodelDefinitions", "Groups", and "Lists".



oModel.GetChildNames ("ModelParts")



- This returns an array containing string model parts. For example: "WG_Interior", "WG", "MT_Interior", "MT", "Box2", "Cylinder1".



oWGi = oModel.GetChildObject ("WG_Interior")



- This sets the object WG_Interior to variable oWGi.



oWGi.GetPropNames()



- This returns property names from child object assigned to oWGi. In this case: "Name", "Material", "Material/SIValue", "Material/EvaluatedValue", "Solve Inside", "Orientation", "Orientation/Choices", "Model", "Display Wireframe", "Material Appearance", "Color", "Color/Red", "Color/Green", "Color/Blue", and "Transparent".

```

VB Syntax	GetChildNames <category>
VB Example	<p>Standalone Example:</p> <pre> oEditor.GetChildNames "ModelParts" oEditor.GetChildNames "Points"</pre> <p>Example used in Conjunction with GetChildObject and GetPropNames:</p> <pre>Dim oAnsoftApp</pre>

```
Dim oDesktop
Dim oProject
Dim oDesign
Dim oEditor

Set oDesign = oProject.GetActiveDesign()

Set oModel = oDesign.GetChildObject "3D Modeler"

oModel.GetChildTypes()



- This returns an array containing strings: "ModelParts", "AllParts", "NonModelParts", "CoordinateSystems", "Points", "Planes", "SubmodelDefinitions", "Groups", and "Lists".



oModel.GetChildNames "ModelParts"



- This returns an array containing string model parts. For example: "WG_Interior", "WG", "MT_Interior", "MT", "Box2", "Cylinder1".



Set oWGi = oModel.GetChildObject "WG_Interior"



- This sets the object WG_Interior to variable oWGi.



oWGi.GetPropNames()



- This returns property names from child object assigned to oWGi. In this case: "Name", "Material", "Material/SIValue", "Material/EvaluatedValue", "Solve Inside", "Orientation", "Orientation/Choices", "Model", "Display Wireframe", "Material Appearance", "Color", "Color/Red", "Color/Green", "Color/Blue", and "Transparent".

```

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 <i><object></i>	Type String	Description 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	<p>Standalone Example:</p> <pre>oDesign = oProject.GetActiveDesign() oEditor = oDesign.GetChildObject("3D Modeler")</pre> <p>Example used in Conjunction with GetChildNames and GetPropNames:</p> <pre>oDesign = oProject.GetActiveDesign() oModel = oDesign.GetChildObject("3D Modeler") oModel.GetChildTypes() <ul style="list-style-type: none"> This returns an array containing strings: "ModelParts", "AllParts", "NonModelParts", "CoordinateSystems", "Points", "Planes", "SubmodelDefinitions", "Groups", and "Lists". oModel.GetChildNames("ModelParts")</pre>

- | | |
|--|--|
| | <ul style="list-style-type: none">• This returns an array containing string model parts. For example: "WG_Interior", "WG", "MT_Interior", "MT", "Box2", "Cylinder1". |
|--|--|

```
oWGi = oModel.GetChildObject ("WG_Interior")
```

- This sets the object WG_Interior to variable oWGi.

```
oWGi.GetPropNames ()
```

- This returns property names from child object assigned to oWGi. In this case: "Name", "Material", "Material/SIValue", "Material/EvaluatedValue", "Solve Inside", "Orientation", "Orientation/Choices", "Model", "Display Wireframe", "Material Appearance", "Color", "Color/Red", "Color/Green", "Color/Blue", and "Transparent".

VB Syntax	GetChildObject<object>
VB Example	<p>Standalone Example:</p> <pre>Dim oAnsoftApp Dim oDesktop Dim oProject Dim oDesign Dim oEditor Set oDesign = oProject.GetActiveDesign() Set oEditor = oDesign.GetChildObject "3D Modeler"</pre> <p>Example used in Conjunction with GetChildNames and GetPropNames:</p>

```
Dim oAnsoftApp
Dim oDesktop
Dim oProject
Dim oDesign
Dim oEditor
Set oDesign = oProject.GetActiveDesign()
Set oModel = oDesign.GetChildObject "3D Modeler"
oModel.GetChildTypes()



- This returns an array containing strings: "ModelParts", "AllParts", "NonModelParts", "CoordinateSystems", "Points", "Planes", "SubmodelDefinitions", "Groups", and "Lists".



oModel.GetChildNames "ModelParts"


- This returns an array containing string model parts. For example: "WG_Interior", "WG", "MT_Interior", "MT", "Box2", "Cylinder1".



Set oWGi = oModel.GetChildObject "WG_Interior"


- This sets the object WG_Interior to variable oWGi.



oWGi.GetPropNames()


- This returns property names from child object assigned to oWGi. In this case: "Name", "Material", "Material/SIVValue", "Material/EvaluatedValue", "Solve Inside", "Orientation", "Orientation/Choices", "Model", "Display Wireframe", "Material Appearance", "Color", "Color/Red", "Color/Green", "Color/Blue", and "Transparent".

```

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	<p>Standalone Example:</p> <pre>oDesign = oProject.GetActiveDesign() oEditor = oDesign.SetActiveEditor("3D Modeler") oEditor.GetChildTypes()</pre> <ul style="list-style-type: none">• This returns an array containing strings: "ModelParts", "AllParts", "NonModelParts", "CoordinateSystems", "Points", "Planes", "SubmodelDefinitions", "Groups", and "Lists". <p>Example Used in Conjunction with GetChildNames:</p> <pre>oProject = oDesktop.GetActiveProject()</pre>

	<pre> oDesign = oProject.GetActiveDesign() oDesign.GetChildNames() <ul style="list-style-type: none"> • This returns an array containing names of child objects for the design. For example: "Boundaries", "Nets", "Analysis", "Optimetrics", "Radiation", "Results", and "3D Modeler". oDesign.GetChildTypes() <ul style="list-style-type: none"> • This returns an array containing the child types. For example: "Module", "Editor", and "Variable". </pre>
--	---

VB Syntax	GetChildTypes()
VB Example	<p>Standalone Example:</p> <pre> Dim oDesign Dim oProject Dim oEditor Set oDesign = oProject.GetActiveDesign() Set oEditor = oDesign.SetActiveEditor "3D Modeler" oEditor.GetChildTypes() <ul style="list-style-type: none"> • This returns an array containing strings: "ModelParts", "AllParts", "NonModelParts", "CoordinateSystems", "Points", "Planes", "SubmodelDefinitions", "Groups", and "Lists". <p>Example Used in Conjunction with GetChildNames:</p> <pre> Dim oDesign Dim oProject </pre> </pre>

```

Dim oDesign

Set oDesign = oProject.GetActiveDesign()

Set oDesign.GetChildNames()

    • This returns an array containing names of child objects for the design. For example: "Boundaries", "Nets", "Analysis", "Optimetrics", "Radiation", "Results", and "3D Modeler".

Set oDesign.GetChildTypes()

    • This returns an array containing the child types. For example: "Module", "Editor", and "Variable".

```

GetEdgeByPosition

Returns the ID for edge(s) that contact a specified point.

UI Access	N/A								
Parameters	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><positionParameters></td> <td>Array</td> <td> <p>Structured array.</p> <p>Array ("NAME:EdgeParameters", "BodyName:=", <string object name>, "Xposition:=", <string>, "YPosition:=", <string>, "ZPosition:=", <string>)</p> <p>Note:</p> </td></tr> </tbody> </table>	Name	Type	Description	<positionParameters>	Array	<p>Structured array.</p> <p>Array ("NAME:EdgeParameters", "BodyName:=", <string object name>, "Xposition:=", <string>, "YPosition:=", <string>, "ZPosition:=", <string>)</p> <p>Note:</p>		
Name	Type	Description							
<positionParameters>	Array	<p>Structured array.</p> <p>Array ("NAME:EdgeParameters", "BodyName:=", <string object name>, "Xposition:=", <string>, "YPosition:=", <string>, "ZPosition:=", <string>)</p> <p>Note:</p>							

		For 2D XY Designs, ZPosition should be set to "0". For 2D RZ Designs, YPosition should be set to "0".
Return Value	Array containing string edge IDs.	

Python Syntax	GetEdgeByPosition (<positionParameters>)
Python Example	<pre> oEditor.GetEdgeByPosition(["NAME:EdgeParameters", "BodyName:=", "Box1", "Xposition:=", "10mm", "YPosition:=", "0mm", "ZPosition:=", "10mm"]) </pre>

VB Syntax	GetEdgeByPosition <positionParameters>
VB Example	<pre> oEditor.GetEdgeByPosition Array("NAME:EdgeParameters", "BodyName:=", "Box1", "Xposition:=", "10mm", "YPosition:=", "0mm", "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.
Return Value	Integer edge ID		

Python Syntax	GetEdgeIDFromNameForFirstOperation(<PartName>, <EdgeName>)
Python Example	oEditor.GetEdgeIDFromNameForFirstOperation("Coil_1", "Edge_4510")

VB Syntax	GetEdgeIDFromNameForFirstOperation <PartName>, <EdgeName>
VB Example	oEditor.GetEdgeIDFromNameForFirstOperation "Coil_1", "Edge_4510"

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 <object>	Type String	Description Object name.
Return Value	Array containing string edge IDs.		

Python Syntax	GetEdgeIDsFromObject (<object>)
Python Example	<code>oEditor.GetEdgeIDsFromObject ("Box1")</code>

VB Syntax	GetEdgeIDsFromObject <object>
------------------	-------------------------------

VB Example

```
oEditor.GetEdgeIDsFromObject "Box1"
```

GetEdgeLength

Returns the length of edges for a specified Face ID.

UI Access	N/A		
Parameters	Name <i><faceID></i>	Type Integer	Description ID of the specified face.
Return Value	Integer containing the length of found edges.		

Python Syntax

```
GetEdgeLength (<faceID>)
```

Python Example

```
oEditor.GetEdgeLength (20)
```

VB Syntax

```
GetEdgeLength <faceID>
```

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 <i><listName></i>	Type String	Description List name.

Return Value	Array containing string object IDs.
---------------------	-------------------------------------

Python Syntax	<code>GetEntityListIDByName (<listName>)</code>
Python Example	<code>oEditor.GetEntityListIDByName ("MyList")</code>

VB Syntax	<code>GetEntityListIDByName <listName></code>
VB Example	<code>oEditor.GetEntityListIDByName "MyList"</code>

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
	<code><defName></code>	String	Definition to retrieve.
	<code><defFolderName></code>	String	Library type (by definition folder name).
Return Value	An extended material wrapper script object (see material wrapper script object functions above).		

P- yt- h- o-	GetExtendedDefinitionObject(<defName>, <defFolderName>)
-----------------------	---

n s- y- nt- ax	
P- yt- h- o- n E- x- a- m- pl- e	<pre> oProject = oDesktop.GetActiveProject() oDefMgr = oProject.GetDefinitionManager() defBlock = "\$begin 'vacuum2' \$begin 'AttachedData' \$begin 'MatAppearanceData' property_data- ='appearance_data' Red=230 Green=230 Blue=230 Transparency=0.95 \$end 'MatAppearanceData' \$end 'AttachedData' simple('permittivity', 1) ModTime=1499970477 \$end 'vacuum2'" oDefMgr.AddDefinitionFromBlock(defBlock, 'Materials', '10101010', True) oMat = oDefMgr.GetExtendedDefinitionObject('vacuum2', 'Materials') # use extended definition object function(s) to manipulate the definition oMat.SetModTime("1630611487") </pre>

GetFaceArea

Returns the area of a specified face.

UI Access	N/A						
Parameters	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: left; padding: 2px;">Name</th> <th style="text-align: left; padding: 2px;">Type</th> <th style="text-align: left; padding: 2px;">Description</th> </tr> </thead> <tbody> <tr> <td style="text-align: left; padding: 2px;"><code><faceID></code></td> <td style="text-align: left; padding: 2px;">Integer</td> <td style="text-align: left; padding: 2px;">ID of specified face.</td> </tr> </tbody> </table>	Name	Type	Description	<code><faceID></code>	Integer	ID of specified face.
Name	Type	Description					
<code><faceID></code>	Integer	ID of specified face.					
Return Value	Long integer of face area.						

Python Syntax	GetFaceArea (<faceID>)
Python Example	<code>oEditor.GetFaceArea (19)</code>

VB Syntax	GetFaceArea <faceID>
VB Example	<code>oEditor.GetFaceArea 19</code>

GetFaceCenter

Returns the center position of a specified face.

UI Access	N/A		
Parameters	Name <planarFaceID>	Type Long	Description Face ID
Return Value	Array containing string X, Y, Z coordinates.		

Python Syntax	GetFaceCenter (<planarFaceID>)
Python Example	<code>oEditor.GetFaceCenter (12)</code>

VB Syntax	GetFaceCenter <planarFaceID>
VB Example	<code>oEditor.GetFaceCenter 12</code>

GetFaceByPosition

Returns the face ID located at a specified position.

Note:

The coordinates must point to exactly one face, not a vertex or edge where two or more faces join.

UI Access	N/A		
Parameters	Name <i><FaceParameters></i>	Type Array	Description Structured Array. Array ("NAME:FaceParameters", "BodyName:=", <string>, "XPosition:=", <string>, "YPosition:=", <string>, "ZPosition:=", <string>)
Return Value	Integer Face ID.		

Python Syntax	GetFaceByPosition(<i><FaceParameters></i>)
Python Example	<pre>oEditor.GetFaceByPosition(["NAME:FaceParameters", "BodyName:=", "Box1",</pre>

```

    "XPosition:=", "0.2mm",
    "YPosition:=", "-0.2mm",
    "ZPosition:=", "0.4mm"
]
)

```

VB Syntax	GetFaceByPosition <FaceParameters>
VB Example	<pre> dim FaceID FaceID = oEditor.GetFaceByPosition Array("NAME:FaceParameters", "BodyName:=", "Box1", "XPosition:=", "0.2mm", "YPosition:=", "-0.2mm", "ZPosition:=", "0.4mm") </pre>

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	<code>oEditor.GetFaceIDFromNameForFirstOperation("Rotor", "Face_14343")</code>

VB Syntax	GetFaceIDFromNameForFirstOperation <PartName>, <FaceName>
VB Example	<code>oEditor.GetFaceIDFromNameForFirstOperation "Rotor", "Face_14343"</code>

GetFaceIDs

Returns the face IDs associated with a specified object.

UI Access	N/A						
Parameters	<table border="1"><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><objectName></td><td>String</td><td>Object name.</td></tr></table>	Name	Type	Description	<objectName>	String	Object name.
Name	Type	Description					
<objectName>	String	Object name.					
Return Value	Array containing string face IDs.						

Python Syntax	GetFaceIDs (<objectName>)
Python Example	<code>oEditor.GetFaceIDs ('Box1')</code>

VB Syntax	GetFaceIDs <objectName>
VB Example	<code>oEditor.GetFaceIDs "Box1"</code>

GetFaceIDsOfSheet

Returns the face IDs associated with a specified sheet object.

UI Access	N/A		
Parameters	Name <code><sheetName></code>	Type String	Description Name of specified sheet object.
Return Value	Array containing string face IDs.		

Python Syntax	<code>GetFaceIDsOfSheet(<sheetName>)</code>
Python Example	<code>oEditor.GetFaceIDsOfSheet ('Sheet1')</code>

VB Syntax	<code>GetFaceIDsOfSheet <sheetName></code>
VB Example	<code>oEditor.GetFaceIDsOfSheet "Sheet1"</code>

GetGeometryModelerMode

Returns the modeler mode (3D or XY) for the current design. This lets you know whether the current model is 2D or 3D.

UI Access	N/A		
Parameters	None.		
Return Value	String containing the modeling mode ("3D" or "XY").		

Python Syntax	GetGeometryModelerMode()
Python Example	<code>oEditor.GetGeometryModelerMode ()</code>

VB Syntax	GetGeometryModelerMode
VB Example	<code>oEditor.GetGeometryModelerMode</code>

GetGroupSubmodelNames

Returns name of models belong to a specific group.

UI Access	N/A								
Parameters	<table border="1"><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><code><groupName></code></td><td>String</td><td>Group name.</td></tr></table>			Name	Type	Description	<code><groupName></code>	String	Group name.
Name	Type	Description							
<code><groupName></code>	String	Group name.							
Return Value	Array of strings containing the model names.								

Python Syntax	GetGroupSubmodelNames(<groupName>)
Python Example	<code>oEditor.GetGroupSubmodelNames ("Group 1")</code>

VB Syntax	GetGroupSubmodelNames <groupName>
VB Example	<code>oEditor.GetGroupSubmodelNames "Group 1"</code>

GetMatchedObjectName

Returns all object names containing the input text string.

UI Access	N/A								
Parameters	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><code><wildcardText></code></td> <td>String</td> <td>Text string present in object name(s).</td> </tr> </tbody> </table>			Name	Type	Description	<code><wildcardText></code>	String	Text string present in object name(s).
Name	Type	Description							
<code><wildcardText></code>	String	Text string present in object name(s).							
Return Value	Array containing string object names.								

Python Syntax	<code>GetMatchedObjectName (<wildcardText>)</code>
Python Example	<pre>oEditor.GetMatchedObjectName ('Box*') oEditor.GetMatchedObjectName ('?ox?')</pre>

VB Syntax	<code>GetMatchedObjectName <wildcardText></code>
VB Example	<code>oEditor.GetMatchedObjectName "Box*"</code>

GetModelBoundingBox

Returns the bounding box of the current model.

UI Access	N/A		
Parameters	None.		
Return Value	Array containing string Xmin, Ymin, Zmin, Xmax, Ymax, and Zmax values of the bounding box.		

Python Syntax	GetModelBoundingBox()
Python Example	<code>oEditor.GetModelBoundingBox()</code>

VB Syntax	GeModelBoundingBox
VB Example	<code>oEditor.GetModelBoundingBox</code>

GetModelUnits

Returns the model's unit of measure.

UI Access	N/A
Parameters	None.
Return Value	String containing unit of measure.

Python Syntax	GetModelUnits()
Python Example	<code>oEditor.GetModelUnits()</code>

VB Syntax	GetModelUnits
VB Example	<code>oEditor.GetModelUnits</code>

GetNumObjects

Returns the number of objects in a design.

UI Access	N/A
Parameters	None.
Return Value	Integer number of objects.

Python Syntax	GetNumObjects()
Python Example	<code>oEditor.GetNumObjects ()</code>

VB Syntax	GetNumObjects
VB Example	<code>oEditor.GetNumObjects ()</code>

GetObjectIDByName

Given an object's name, returns its ID. IDs are used with [CreateEntityList](#).

UI Access	N/A		
Parameters	Name <code><objectName></code>	Type String	Description Object name.
Return Value	Integer object ID.		

Python Syntax	GetObjectIDByName(<objectName>)
Python Example	<code>oEditor.GetObjectIDByName ('Box1')</code>

VB Syntax	GetObjectIDByName <objectName>
VB Example	<code>oEditor.GetObjectIDByName "Box1"</code>

GetObjectName

Returns an object's name from its specified base index (creation order).

UI Access	N/A		
Parameters	Name <index>	Type Integer	Description 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	GetObjectName <index>
VB Example	<code>oEditor.GetObjectName 3</code>

GetObjectByNameByEdgeID

Returns an object name given an edge ID.

UI Access	N/A		
Parameters	Name <i><EdgeID></i>	Type Integer	Description The edge ID.
Return Value	String containing object name.		

Python Syntax	GetObjectByNameByEdgeID(<EdgeID>)
Python Example	<code>oEditor.GetObjectByNameByEdgeID(88)</code>

VB Syntax	GetObjectByNameByEdgeID <EdgeID>
VB Example	<code>oEditor.GetObjectByNameByEdgeID 10</code>

GetObjectByNameByFaceID

Returns an object name given a face ID.

UI Access	N/A		
Parameters	Name <i><FaceID></i>	Type Integer	Description The face ID.
Return Value	String containing object name.		

Python Syntax	GetObjectByNameByFaceID (<FaceID>)
Python Example	<code>oEditor.GetObjectByNameByFaceID (88)</code>

VB Syntax	GetObjectByNameByFaceID <FaceID>
VB Example	<code>oEditor.GetObjectByNameByFaceID 10</code>

GetObjectByNameByID

Returns an object name given an ID.

UI Access	N/A						
Parameters	<table border="1"><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><ObjID></td><td>Integer</td><td>The object ID.</td></tr></table>	Name	Type	Description	<ObjID>	Integer	The object ID.
Name	Type	Description					
<ObjID>	Integer	The object ID.					
Return Value	String containing object name.						

Python Syntax	GetObjectByNameByID(<ObjID>)
Python Example	<code>oEditor.GetObjectByNameByID (88)</code>

VB Syntax	GetObjectByNameByID <ObjID>
VB Example	<code>oEditor.GetObjectByNameByID 10</code>

GetObjectByNameByVertexID

Returns an object name given a vertex ID.

UI Access	N/A								
Parameters	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><VertexID></td> <td>Integer</td> <td>The vertex ID.</td> </tr> </tbody> </table>			Name	Type	Description	<VertexID>	Integer	The vertex ID.
Name	Type	Description							
<VertexID>	Integer	The vertex ID.							
Return Value	String containing object name.								

Python Syntax	GetObjectByNameByVertexID(<VertexID>)
Python Example	<code>oEditor.GetObjectByNameByVertexID(88)</code>

VB Syntax	GetObjectByNameByVertexID <VertexID>
VB Example	<code>oEditor.GetObjectByNameByVertexID 10</code>

GetObjectsByMaterial

Returns a list of objects of a specified material.

UI Access	N/A								
Parameters	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><material></td> <td>String</td> <td>Material name.</td> </tr> </tbody> </table>			Name	Type	Description	<material>	String	Material name.
Name	Type	Description							
<material>	String	Material name.							
Return Value	Array containing string object names.								

Python Syntax	GetObjectsByMaterial (<material>)
Python Example	<code>oEditor.GetObjectsByMaterial ('copper')</code>

VB Syntax	GetObjectsByMaterial <material>
VB Example	<code>oEditor.GetObjectsByMaterial "copper"</code>

GetObjectShapeType

Returns the shape type of a specified object.

UI Access	N/A									
Parameters	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td><objName></td><td>String</td><td>Name of specified object.</td></tr><tr><td><csName></td><td>String</td><td>Name of coordinate system of the object.</td></tr></tbody></table>	Name	Type	Description	<objName>	String	Name of specified object.	<csName>	String	Name of coordinate system of the object.
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	GetObjectShapeType(<objName>, <csName>)
Python Example	<code>oEditor.GetObjectShapeType ("box1", "Global")</code>

VB Syntax	GetObjectShapeType <objName>, <csName>
VB Example	<code>oEditor.GetObjectShapeType "box1", "Global"</code>

GetObjectsInGroup

Returns a list of objects in a specified group.

UI Access	N/A								
Parameters	<table border="1"> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> <tr> <td><group></td> <td>String</td> <td> Group name. One of:<ul style="list-style-type: none"> • "<materialName>" • "<assignmentName>" • "Non Model" • "Solids" • "Unclassified" • "Sheets" • "Lines" </td> </tr> </table>	Name	Type	Description	<group>	String	Group name. One of: <ul style="list-style-type: none"> • "<materialName>" • "<assignmentName>" • "Non Model" • "Solids" • "Unclassified" • "Sheets" • "Lines" 		
Name	Type	Description							
<group>	String	Group name. One of: <ul style="list-style-type: none"> • "<materialName>" • "<assignmentName>" • "Non Model" • "Solids" • "Unclassified" • "Sheets" • "Lines" 							
Return Value	Array containing string object names.								

Python Syntax	GetObjectsInGroup (<group>)
Python Example	<code>oEditor.GetObjectsInGroup ('Sheets')</code>

VB Syntax	GetObjectsInGroup <group>
VB Example	<code>oEditor.GetObjectsInGroup "Sheets"</code>

GetObjectVolume

Returns an object's volume from its name).

UI Access	N/A		
Parameters	Name <i><name></i>	Type String	Description Object name
Return Value	.Real		

Python Syntax	GetObjectVolume(<name>)
Python Example	<code>oEditor.GetObjectVolume("Box1")</code>

VB Syntax	GetObjectVolume <name>
VB Example	<code>oEditor.GetObjectVolume "Box1"</code>

GetObjPath [Editor]

Obtains the path to the 3D modeler.

UI Access	N/A
Parameters	None.
Return Value	String containing the path.

Python Syntax	GetObjPath()
Python Example	<code>oEditor.GetObjPath()</code>

VB Syntax	GetObjPath
VB Example	<code>oEditor.GetObjPath</code>

GetPartsForUserDefinedModel

Obtains parts from a specified user defined model.

UI Access	N/A		
Parameters	Name <i><udmName></i>	Type String	Description Name of user defined model.
Return Value	Array of strings containing associated parts.		

Python Syntax	GetPartsForUserDefinedModel (<udmName>)
Python Example	<code>oEditor.GetPartsForUserDefinedModel ("OnDieSpiralInductor1")</code>

VB Syntax	GetPartsForUserDefinedModel <udmName>
VB Example	<code>oEditor.GetPartsForUserDefinedModel "OnDieSpiralInductor1"</code>

GetPoints [3D Modeler Editor]

Returns all the points defined in current 3D modeler.

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

Python Syntax	GetPoints ()
Python Example	<code>oEditor.GetPoints()</code>

VB Syntax	GetPoints
VB Example	<code>oEditor.GetPoints</code>

GetPropEvaluatedValue

Returns the Evaluated-Value for Value-Property and Variable. Returns the Property-value as text string for other property types

Note:

This command is not supported by the EMIT and Circuit design types.

UI Access	N/A
------------------	-----

Parameters	Name <i><PropName></i>	Type String	Description Name of the property.
Return Value	String value of the evaluated value.		

Python Syntax	GetPropEvaluatedValue (<i><PropName></i>)
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 <i><PropTab></i>	Type String	Description One of the following, where tab titles are shown in parentheses: <ul style="list-style-type: none">• PassedParameterTab ("Parameter Values")• DefinitionParameterTab (Parameter Defaults")• LocalVariableTab ("Variables" or "Local Variables")

		<ul style="list-style-type: none"> • ProjectVariableTab ("Project variables") • ConstantsTab ("Constants") • BaseElementTab ("Symbol" or "Footprint") • ComponentTab ("General") • Component("Component") • CustomTab ("Intrinsic Variables") • Quantities ("Quantities") • Signals ("Signals")
	<p><i><PropServer></i></p>	<p>String</p> <p>An object identifier, generally returned from another script method, such as CompInst@R;2;3</p>
	<p><i><PropName></i></p>	<p>String</p> <p>Name of the property.</p>
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.GetPropertyValues("PassedParameterTab", k, "R") ... </pre>

VB Syntax	GetPropertyValue (<PropTab>, <PropServer>, <PropName>)
------------------	--

VB Example	<pre>selectionArray = oEditor.GetSelections for k in selectionArray: val = oEditor.GetPropertyValues("PassedParameterTab", k, "R") ... </pre>
-------------------	---

GetPropNames [Modeler]

Returns the property names for the active model object, or specified property values.

Note:

This command is not supported by the EMIT and Circuit design types.

UI Access	N/A						
Parameters	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 30%;">Name</th> <th style="width: 20%;">Type</th> <th style="width: 50%;">Description</th> </tr> </thead> <tbody> <tr> <td style="height: 100px; vertical-align: top; padding: 5px;"><i><IncludeReadOnly></i></td> <td style="height: 100px; vertical-align: top; padding: 5px;">Boolean</td> <td style="height: 100px; vertical-align: top; padding: 5px;"> Optional. <ul style="list-style-type: none"> • True - includes all properties. • False - returns only property names that can be changed. True by default. </td></tr> </tbody> </table>	Name	Type	Description	<i><IncludeReadOnly></i>	Boolean	Optional. <ul style="list-style-type: none"> • True - includes all properties. • False - returns only property names that can be changed. True by default.
Name	Type	Description					
<i><IncludeReadOnly></i>	Boolean	Optional. <ul style="list-style-type: none"> • 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	GetPropNames(<i><IncludeReadOnly></i>)
Python Example	oEditor.GetPropNames (True)

VB Syntax	GetPropNames(<IncludeReadOnly>)
VB Example	<code>oEditor.GetPropertyName 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	<table border="1"><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><PropName></td><td>String</td><td>Name of the property.</td></tr></table>			Name	Type	Description	<PropName>	String	Name of the property.
Name	Type	Description							
<PropName>	String	Name of the property.							
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"</pre>

```

oCreateBox.GetPropEvaluatedValue("xSize")
return '0.4mm'

oCreateBox.GetPropSIValue("xSize")
return 0.0004

```

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 <i><propPath></i>	Type	Description Optional. Child object's property path. See: Object Property Function Summary .
Return Value	The property value.		

Python Syntax	GetPropValue(<i><propPath></i>)
Python Example	<pre>Rh1 = oDesign.GetChildObject('Box1') Rh1.GetPropValue('Orientation')</pre> <p>Returns the specified Property Value: 'Global'</p>

VB Syntax	GetPropValue(<propPath>)
VB Example	Rh1 = oDesign.GetChildObject("Box1") GetPropValue("Orientation")

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 <i><group></i>	Type String	Description Group name. One of: <ul style="list-style-type: none">• "<<i>materialName</i>>"• "<<i>assignmentName</i>>"• "Non Model"• "Solids"

			<ul style="list-style-type: none">• "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	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td><prompt></td><td>String</td><td>"Enter a point." Then click a point in the 3D Modeler window.</td></tr></tbody></table>			Name	Type	Description	<prompt>	String	"Enter a point." Then click a point in the 3D Modeler window.
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	<code>oEditor.GetUserPosition("Enter a point.")</code>

VB Syntax	GetUserPosition <prompt>
VB Example	<code>oEditor.GetUserPosition "Enter a point."</code>

GetVertexIDFromNameForFirstOperation

Returns vertex ID associated with a specified vertex belongs to the first operation of a part.

UI Access	N/A		
Parameters	Name	Type	Description
	<PartName>	String	Name of specified part.
Return Value	Integer representing vertex ID.		

Python Syntax	GetVertexIDFromNameForFirstOperation(<PartName>, <VertexName>)
Python Example	<code>oEditor.GetVertexIDFromNameForFirstOperation("Part1", "Vertex1")</code>

VB Syntax	GetVertexIDFromNameForFirstOperation <PartName>, <VertexName>
VB Example	<code>oEditor.GetVertexIDFromNameForFirstOperation "Part1", "Vertex1"</code>

GetVertexIDsFromEdge

Returns vertex IDs associated with a specified edge.

UI Access	N/A								
Parameters	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td><edgeID></td><td>Integer</td><td>Edge ID.</td></tr></tbody></table>			Name	Type	Description	<edgeID>	Integer	Edge ID.
Name	Type	Description							
<edgeID>	Integer	Edge ID.							
Return Value	Array containing string vertex IDs.								

Python Syntax	GetVertexIDsFromEdge(<edgeID>)
Python Example	<code>oEditor.GetVertexIDsFromEdge(10)</code>

VB Syntax	GetVertexIDsFromEdge <edgeID>
VB Example	<code>oEditor.GetVertexIDsFromEdge 10</code>

GetVertexIDsFromFace

Returns vertex IDs associated with a specified face.

UI Access	N/A								
Parameters	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td><faceID></td><td>Integer</td><td>Face ID.</td></tr></tbody></table>			Name	Type	Description	<faceID>	Integer	Face ID.
Name	Type	Description							
<faceID>	Integer	Face ID.							
Return Value	Array containing string vertex IDs.								

Python Syntax	GetVertexIDsFromFace(<faceID>)
Python Example	<code>oEditor.GetVertexIDsFromFace(10)</code>

VB Syntax	GetVertexIDsFromFace <faceID>
VB Example	<code>oEditor.GetVertexIDsFromFace 10</code>

GetVertexIDsFromObject

Returns vertex IDs associated with a specified object.

UI Access	N/A						
Parameters	<table border="1"> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> <tr> <td><objectName></td> <td>String</td> <td>Object name.</td> </tr> </table>	Name	Type	Description	<objectName>	String	Object name.
Name	Type	Description					
<objectName>	String	Object name.					
Return Value	Array containing string vertex IDs.						

Python Syntax	GetVertexIDsFromObject(<objectName>)
Python Example	<code>oEditor.GetVertexIDsFromObject('Box1')</code>

VB Syntax	GetVertexIDsFromObject <objectName>
VB Example	<code>oEditor.GetVertexIDsFromObject "Box1"</code>

GetVertexPosition

Returns an array of coordinates for a specified vertex.

UI Access	N/A						
Parameters	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td><vertexID></td><td>Integer</td><td>Vertex ID.</td></tr></tbody></table>	Name	Type	Description	<vertexID>	Integer	Vertex ID.
Name	Type	Description					
<vertexID>	Integer	Vertex ID.					
Return Value	Array containing X, Y, Z coordinates.						

Python Syntax	GetVertexPosition(<vertexID>)
Python Example	<code>oEditor.GetVertexPosition(1)</code>

VB Syntax	GetVertexPosition <vertexID>
VB Example	<code>oEditor.GetVertexPosition 1</code>

GetWireBodyNames

Returns the wire body names in current 3D modeler.

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

Python Syntax	GetWireBodyNames()
Python Example	<code>oEditor.GetWireBodyNames ()</code>

VB Syntax	GetWireBodyNames
VB Example	<code>oEditor.GetWireBodyNames</code>

OpenExternalEditor

Launches a SpaceClaim session.

UI Access	Modeler > SpaceClaim Link > Connect to Active Session.						
Parameters	<table border="1"> <tr> <td>Name</td> <td>Type</td> <td>Description</td> </tr> <tr> <td><code><SelectionsArray></code></td> <td>Array</td> <td>Structured array. See: SelectionsArray.</td> </tr> </table>	Name	Type	Description	<code><SelectionsArray></code>	Array	Structured array. See: SelectionsArray .
Name	Type	Description					
<code><SelectionsArray></code>	Array	Structured array. See: SelectionsArray .					
Return Value	None.						

Python Syntax	OpenExternalEditor (<SelectionsArray>)
Python Example	<pre>oEditor.OpenExternalEditor (["NAME:Selections", "Selections:=" , "SpaceClaim1"])</pre>

VB Syntax	OpenExternalEditor <SelectionsArray>
VB Example	<pre>oEditor.OpenExternalEditor Array ("NAME:Selections", "Selections:=", "")</pre>

PageSetup

Specifies Page Setup settings for printing.

UI Access	File > Page Setup.		
Parameters	Name <i><Parameters></i>	Type Array	Description Structured array. ["NAME:PageSetupData", "margins:=", <SetupArray>] <i><SetupArray></i>
		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>

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 <i><EdgeList></i>	Type Array	Description Array containing edge IDs.
Return Value	None.		

Python Syntax	RemoveBadEdges (<i><EdgeList></i>)
Python Example	<code>oEditor.RemoveBadEdges ([18, 30])</code>

VB Syntax	RemoveBadEdges <i><EdgeList></i>
VB Example	<code>oEditor.RemoveBadEdges Array(18, 30)</code>

RemoveBadFaces

Removes bad faces from specified list.

UI Access	N/A		
Parameters	Name <i><FaceList></i>	Type Array	Description 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	<table border="1"> <tr> <td>Name</td> <td>Type</td> <td>Description</td> </tr> <tr> <td><VertexList></td> <td>Array</td> <td>Array containing vertex IDs.</td> </tr> </table>			Name	Type	Description	<VertexList>	Array	Array containing vertex IDs.
Name	Type	Description							
<VertexList>	Array	Array containing vertex IDs.							
Return Value	None.								

Python Syntax	RemoveBadVertices (<VertexList>)
Python Example	<code>oEditor.RemoveBadVertices ([324, 325])</code>

VB Syntax	RemoveBadVertices <VertexList>
------------------	--------------------------------

VB Example

```
oEditor.RemoveBadVertices Array(324, 325)
```

RenamePart

Renames an object.

UI Access	Enter new name in Name field.		
Parameters	Name <i><renameParametersArray></i>	Type Array	Description Structured array. Array("NAME:Rename Data", "Old Name:=", <string>, "New Name:=", <string>)
Return Value	None.		

Python Syntax

```
oEditor.RenamePart(<renameParametersArray>)
```

Python Example

```
oEditor.RenamePart(
[
  'NAME:Rename Data',
  'Old Name:=' , 'partname',
  'New Name:=' , 'newpartname',
])
```

)
--	---

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
	<code><propPath></code>	String	Child object's property path. See: Object Property Function Summary .
Return Value	Boolean: <ul style="list-style-type: none"> • 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 <RelativeCS>	Type String	Description 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 <i><RelativeCS></i>	Type String	Description 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	<code>oEditor.SetTopDownViewDirectionForAllViews "Global"</code>

UpdatePriorityList

Updates specified priority lists.

UI Access	N/A		
Parameters	Name <i><Lists></i>	Type Array	Description Array of priority list names.
Return Value	None.		

Python Syntax	UpdatePriorityList (<Lists>)
Python Example	<code>oEditor.UpdatePriorityList(["PriorityList1", "PriorityList2"])</code>

VB Syntax	UpdatePriorityList <Lists>
VB Example	<code>oEditor.UpdatePriorityList Array("PriorityList1", "PriorityList2")</code>

UpgradeVersion

Upgrades legacy geometry to current version.

UI Access	Right-click on an operation icon in the history tree, select Upgrade Version .			
Parameters	<table border="1"><tr><td>Name <Selections></td><td>Type Array</td><td>Description Structured array. Array ("NAME:Parameters", Array ("NAME:PartOperations", Array ("NAME:<string>",), "OperationIndices:=", <array of integers>), Array ("NAME:UDMOperations"))</td></tr></table>	Name <Selections>	Type Array	Description Structured array. Array ("NAME:Parameters", Array ("NAME:PartOperations", Array ("NAME:<string>",), "OperationIndices:=", <array of integers>), Array ("NAME:UDMOperations"))
Name <Selections>	Type Array	Description 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><code>oEditor.UpgradeVersion(["NAME:Parameters", ["NAME:PartOperations", ["NAME:source", "OperationIndices:=", [0]]], ["NAME:UDMOperations"]))</code></pre>
-----------------------	--

VB Syntax	UpgradeVersion <Lists>
VB Example	<pre><code>oEditor.UpgradeVersion Array("NAME:Parameters", Array("NAME:PartOperations", Array("NAME:source", "OperationIndices:=", Array(0))), Array"NAME:UDMOperations"))</code></pre>

Validate3DComponent

Validates a 3D component.

UI Access	N/A									
Parameters	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><componentPath></td> <td>String</td> <td>Path to a 3D component.</td> </tr> <tr> <td><password></td> <td>String</td> <td>Optional. Password to access the component.</td> </tr> </tbody> </table>	Name	Type	Description	<componentPath>	String	Path to a 3D component.	<password>	String	Optional. Password to access the component.
Name	Type	Description								
<componentPath>	String	Path to a 3D component.								
<password>	String	Optional. Password to access the component.								
Return Value	<p>Boolean:</p> <ul style="list-style-type: none"> • 1 - component is valid. • 0 - component is not valid. 									

Python Syntax	Validate3DComponent (<componentPath>, <password>)
Python Example	<pre>oEditor.Validate3DComponent("C:/temp/component.3dcomp", "")</pre>

VB Syntax	Validate3DComponent <componentPath>, <password>
VB Example	<pre>oEditor.Validate3DComponent "C:/temp/component.3dcomp", ""</pre>

WriteHistoryTreeLayoutForTest

Writes history tree layout to a file.

UI Access	N/A						
Parameters	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><filePath></td> <td>String</td> <td>Path to the file.</td> </tr> </tbody> </table>	Name	Type	Description	<filePath>	String	Path to the file.
Name	Type	Description					
<filePath>	String	Path to the file.					

	<i><OrgByMaterial></i>	Integer	<ul style="list-style-type: none"> • 1 - organize objects by material. • 0 - do not organize by material.
	<i><OrgByAssignment></i>	Integer	<ul style="list-style-type: none"> • 1 - organize sheets by assignment. • 0 - do not organize by assignment.
	<i><OrgByCompDefinition></i>	Integer	<ul style="list-style-type: none"> • 1 - organize components by definition. • 0 - do not organize by definition.
	<i><DoNotOrgUnderGroup></i>	Integer	<ul style="list-style-type: none"> • 1 - do not organize within group. • 0 - organize within group.
	<i><ShowGroup></i>	Integer	<p>Optional.</p> <ul style="list-style-type: none"> • 1 - show group. • 0 - do not show.
Return Value	None.		

Python Syntax	WriteHistoryTreeLayoutForTest (<filePath>, <OrgByMaterial>, <OrgByAssignment>, <OrgByCompDefinition>, <DoNotOrgUnderGroup>, <>ShowGroup>)
Python Example	<pre>oEditor.WriteHistoryTreeLayoutForTest ('C:\temp', 1, 0, 0, 0, 1)</pre>

VB Syntax	WriteHistoryTreeLayoutForTest <filePath>, <OrgByMaterial>, <OrgByAssignment>, <OrgByCompDefinition>, <DoNotOrgUnderGroup>, <>ShowGroup>
VB Example	<pre>oEditor.WriteHistoryTreeLayoutForTest "C:\temp", 1, 0, 0, 0, 1</pre>

Cable Modeling Commands

Cable Modeling is a Beta Feature to 3D Component Modeling for Release 21.2. Cables are created and edited as design objects using the Cable Setup module, and then inserted as native components. For example, elements of a cable are created:

```
oModule = oDesign.GetModule("CableSetup")
oModule.CreateStraightWireCable(....)
```

And then, when a Harness has been created from various components, it is inserted to the Modeler:

```
oEditor = oDesign.SetActiveEditor("3D Modeler")
oEditor.InsertNativeComponent(
[
    "NAME:InsertNativeComponentData",
    "TargetCS:=", "Global",
    "SubmodelDefinitionName:=", "MyHarness",
...
]
```

[AddCableToBundle](#)

[CreateCableBundle](#)

[CreateCableHarness](#)

[CreateClockSource](#)

[CreatePWLSource](#)

[CreateStraightWireCable](#)

[CreateTwistedPairCable](#)[ExportCableLibrary](#)[ImportCableLibrary](#)[RemoveCable](#)[UpdateCableHarness](#)

AddCableToBundle

Add a cable to an existing bundle using the Beta Cable Modeling feature. Right click on the 3D Component icon and select **Cables>Edit Cables** to open the **Cable Editor**.

UI Access	3D Components>Cables > Edit Cables
Parameters	None
Return Value	None

Python Syntax	AddCableToBundle(<parameters>))
Python Example	<pre> oModule.AddCableToBundle("bundle1", "tpwire2", 1, ["NAME:CableInstParams", "XPos:=" , "0mm", "YPos:=" , "0mm", "RotX:=" , "0deg"]) </pre>

```
    ] ,  
    [  
        "NAME:CableInstAttribs",  
        "Name:=" , "tpwire2"  
    ] )
```

VB Syntax	AddCableToBundle <name> Array(<parameters>)
VB Example	<pre>oModule.AddCableToBundle "bundle2", "bundle1", 1, Array("NAME:CableInstParams", "XPos:=", _ "0mm", "YPos:=", "0mm", "RotX:=", "0deg"), Array("NAME:CableInstAttribs", "Name:=", _ "bundle1")</pre>

CreateCableBundle

Creates a cable bundle using the Beta Cable Modeling feature. Right click on the 3D Component icon and select **Cables>Edit Cables** to open the **Cable Editor**. Then **Add>Bundle....**

UI Access	3D Components>Cables > Edit Cables
Parameters	None
Return Value	None

Python Syntax	CreateCableBundle([<parameters>], [<name>])
Python Example	<pre> oModule.CreateCableBundle(["NAME:BundleParams", "AutoPack:=" , True, ["NAME:InsulationJacketParams", "InsThickness:=" , "0.25mm", "JacketMaterial:=" , "PVC plastic", "InnerDiameter:=" , "2.5mm"]], ["NAME:BundleAttribs", "Name:=" , "bundle1"]) </pre>

VB Syntax	CreateCableBundle Array(<parameters>, <name>)
VB Example	<pre> oModule.CreateCableBundle Array("NAME:BundleParams", "AutoPack:=", true, Array("NAME:InsulationJacketParams", "InsThickness:=", _ </pre>

	"0.25mm", "JacketMaterial:=", "PVC plastic", "InnerDiameter:=", "2.5mm")), Array("NAME:BundleAttribs", "Name:=", __ "bundle3")
--	--

CreateCableHarness

Creates a cable harness using the Beta Cable Modeling feature. Right click on the 3D Component icon and select **Cables>New Cable Harness** to open the **Insert Cable Harness Component Editor**.

UI Access	3D Components>Cables > New Cable Harness...
Parameters	None
Return Value	None

Python Syntax	CreateCableHarness(<parameters>))
Python Example	<pre>oModule.CreateCableHarness (["NAME:HarnessParams", "Bundle:=" , "bundle1", "TwistAlongPath:=" , "720deg", "Route:=" , "Polyline1", "AutoOrient:=" , False, "Origin:=" , ["0mm", "0mm", "0mm"],</pre>

```
"XAxisEnd:=" , [ "-6mm", "0mm", "0mm"] ,  
"PlaneFlip:=" , True,  
[  
    "NAME:RefConductors",  
    "tpwire2/w1"  
,  
[  
    "NAME:InputTerminations",  
    "tpwire2/w1:=" , [ "Source:="  
    "tpwire2/w2:=" , [ "Imped:="  
,  
[  
    "NAME:OutputTerminations",  
    "tpwire2/w1:=" , [ "Imped:="  
    "tpwire2/w2:=" , [ "Source:="  
,  
[  
    "NAME:HarnessAttribs",  
    "Name:=" , "MyHarness"  
] )
```

VB Syntax	CreateCableHarness Array(<parameters>)
VB Example	<pre>oModule.CreateCableHarness Array("NAME:HarnessParams", "Bundle:=", "bundle1", "TwistAlongPath:=", _ "360deg", "Route:=", "Polyline2", "AutoOrient:=", false, "Origin:=", Array(_ "-24mm", "24mm", "0mm"), "XAxisEnd:=", Array("-30mm", "24mm", "0mm"), "PlaneFlip:=", _ true, Array("NAME:RefConductors", "tpwire2/w1"), Array("NAME:InputTerminations", "tpwire2/w1:=", Array("Source:=", _ "5V", "Imped:=", "50ohm"), "tpwire2/w2:=", Array("Source:=", "5V", "Imped:=", _ "50ohm")), Array("NAME:OutputTerminations", "tpwire2/w1:=", Array("Source:=", "5V", "Imped:=", _ "50ohm"), "tpwire2/w2:=", Array("Imped:=", "50ohm))), Array("NAME:HarnessAttribs", "Name:=", _ "MyNewHarness")</pre>

CreateClockSource

Creates a clock source using the Beta Cable Modeling feature. Right click on the 3D Component icon and select **Cables>Edit Cable Sources...** to open the **Cable Time Domain Sources Editor**.

UI Access	3D Components>Cables > Edit Cables
Parameters	None
Return Value	None

Python Syntax	CreateClockSource(<parameters>))
Python Example	<pre> oModule.CreateClockSource(["NAME:ClockSignalParams", "Period:=" , "35us", "LowPulseVal:=" , "0V", "HighPulseVal:=" , "1V", "Risetime:=" , "5us", "Falltime:=" , "5us", "PulseWidth:=" , "20us"], ["NAME:TDSourceAttrs", "Name:=" , "clock1"]) </pre>

VB Syntax	CreateClockSourceArray(<parameters>)
VB Example	<pre> oModule.CreateClockSource Array("NAME:ClockSignalParams", "Period:=", "35us", "LowPulseVal:=", _ </pre>

```
"0V", "HighPulseVal:=", "1V", "Risetime:=", "5us", "Falltime:=", "5us",
"PulseWidth:=", _
"20us"), Array("NAME:TDSourceAttribs", "Name:=", "clock2")
```

CreatePWLSource

Creates a PWL source using the Beta Cable Modeling feature. Right click on the 3D Component icon and select **Cables > Edit Cable Sources...** to open the **Cable Time Domain Sources Editor**.

UI Access	3D Components>Cables > Edit Cable Sources...
Parameters	None
Return Value	None

Python Syntax	CreatePWLSource(<parameters>)
Python Example	<pre>oModule.CreatePWLSource(["NAME:PWLSignalParams", ["NAME:SignalValues", "0V", "0.5V", "0V"]])</pre>

```

        ],
        [
            "NAME:TimeValues",
            "0ns",
            "1ns",
            "2ns"
        ]
    ],
    [
        "NAME:TDSourceAttrs",
        "Name:="           , "pwl3"
    ]
)

```

VB Syntax	CreatePWLSourceArray(<parameters>)
VB Example	<pre> oModule.CreatePWLSource Array("NAME:PWLSignalParams", Array("NAME:SignalValues", "0V", _ "0.5V", "0V"), Array("NAME:TimeValues", "0ns", "1ns", "2ns")), Array ("NAME:TDSourceAttrs", "Name:=", _ "pwl1") </pre>

CreateStraightWireCable

Creates a straight Wire Cable using the Beta Cable Modeling feature. Right click on the 3D Component icon and select **Cables > Edit Cables** to open the **Cable Editor**. Then **Add Straight Wire Cable**.

UI Access	3D Components>Cables > Edit Cables
Parameters	None
Return Value	None

Python Syntax	CreateStraightWireCable([<parameters>]), [<name>])
Python Example	<pre>oModule = oDesign.GetModule("CableSetup") oModule.CreateStraightWireCable(["NAME:StWireParams", "WireStandard:=" , "ISO", "WireGauge:=" , "0.13", "CondDiameter:=" , "0.55mm", "CondMaterial:=" , "copper", "InsThickness:=" , "0.25mm", "InsMaterial:=" , "PVC plastic", "InsType:=" , "Thin Wall"])</pre>

```

        ],
        [
            "NAME:StWireAttribs",
            "Name:=" , "stwire1"
        ]
    )

```

VB Syntax	Paste
VB Example	<pre> oModule.CreateStraightWireCable Array("NAME:StWireParams", "WireStandard:=", "ISO", "WireGauge:=", _ "0.13", "CondDiameter:=", "0.55mm", "CondMaterial:=", "copper", "InsThickness:=", _ "0.25mm", "InsMaterial:=", "PVC plastic", "InsType:=", "Thin Wall"), Array ("NAME:StWireAttribs", _ "Name:=", "stwire3") </pre>

CreateTwistedPairCable

Creates a twisted pair cable using the Beta Cable Modeling feature. Right click on the 3D Component icon and select **Cables>Edit Cables** to open the **Cable Editor**. Then **Add Twisted Pair Cable**.

UI Access	3D Components>Cables > Edit Cables
Parameters	None
Return Value	None

Python Syntax	CreateTwisted Pair Cable([<parameters>]), [<name>])
Python Example	<pre>oModule.CreateTwistedPairCable("stwire1", ["NAME:TwistedPairParams", ["NAME:VirtualJacketParams", "JacketMaterial:=", "", "", "InnerDiameter:=", "0"], "IsLayLengthSpecified:=", False, "LayLength:=", "13.888888888889mm", "TurnsPerMeter:=", "72"], ["NAME:TwistedPairAttribs", "Name:=", "tpwire1"])</pre>

VB Syntax	CreateTwistedPairCable "<name>" , Array(<parameters>)
------------------	---

VB Example	<pre> oModule.CreateTwistedPairCable "stwire1", Array("NAME:TwistedPairParams", _ Array("NAME:VirtualJacketParams", "JacketMaterial:=", _ "", "InnerDiameter:=", "0"), "IsLayLengthSpecified:=", false, "LayLength:=", _ "13.888888888889mm", "TurnsPerMeter:=", "72"), Array("NAME:TwistedPairAttribs", "Name:=", _ "tpwire3") </pre>
-------------------	--

ExportCableLibrary

Exports a cable library created using the Beta Cable Modeling feature. Right click on the 3D Component icon and select **Cables>Export Cable Library**. to open a browser window.

UI Access	3D Components>Cables > Export Cable Library...
Parameters	None
Return Value	None

Python Syntax	ExportCableLibrary(<parameters>)
Python Example	<pre> oModule = oDesign.GetModule("CableSetup") oModule.ExportCableLibrary("D:\\\\Users\\\\JaneDoe\\\\PersonalLib\\\\MyCables") </pre>

VB Syntax	ExportCableLibrary(<parameters>)
------------------	-----------------------------------

VB Example

```
oModule = oDesign.GetModule("CableSetup")
oModule.ExportCableLibrary("D:\\Users\\JaneDoe\\PersonalLib\\MyCables")
```

ImportCableLibrary

Imports an existing cable library created using the Beta Cable Modeling feature. Right click on the 3D Component icon and select **Cables>Export Cable Library.** to open a browser window.

UI Access	3D Components>Cables > Import Cable Library...
Parameters	None
Return Value	None

Python Syntax

```
ImportCableLibrary(<parameters>)
```

Python Example

```
oModule = oDesign.GetModule("CableSetup")
oModule.ExportCableLibrary("D:\\Users\\JaneDoe\\PersonalLib\\MyCables")
```

VB Syntax

```
ImportCableLibrary(<parameters> )
```

VB Example

```
oModule = oDesign.GetModule("CableSetup")
oModule.ExportCableLibrary("D:\\Users\\JaneDoe\\PersonalLib\\MyCables")
```

RemoveCable

Removes an existing cable created using the Beta Cable Modeling feature. Right click on the 3D Component icon and select **Cables>Edit Cables....** to open a **Cable Editor** window.

UI Access	3D Components>Cables > Edit Cables...
Parameters	None
Return Value	None

Python Syntax	RemoveCable(<parameters>))
Python Example	<pre>oModule = oDesign.GetModule("CableSetup") oModule.RemoveCable ("bundle4")</pre>

VB Syntax	ImportCableLibrary(<parameters>)
VB Example	<pre>oModule = oDesign.GetModule("CableSetup") oModule.RemoveCable ("bundle4")</pre>

UpdateCableHarness

Creates a cable harness using the Beta Cable Modeling feature. Right click on the 3D Component icon and select **Cables>New Cable Harness** to open the **Insert Cable Harness Component Editor**.

UI Access	3D Components>Cables > New Cable Harness...
Parameters	None

Return Value	None
---------------------	------

Python Syntax	UpdateCableHarness(<parameters>))
Python Example	<pre>oModule.UpdateCableHarness (["NAME:HarnessParams", "Bundle:=" , "bundle1", "TwistAlongPath:=" , "720deg", "Route:=" , "Polyline1", "AutoOrient:=" , False, "Origin:=" , ["0mm", "0mm", "0mm"], "XAxisEnd:=" , ["-6mm", "0mm", "0mm"], "PlaneFlip:=" , True, ["NAME:RefConductors", "tpwire2/w1"], ["NAME:InputTerminations", </pre>

```

        "tpwire2/w1:=" , [ "Source:="
        "tpwire2/w2:=" , [ "Imped:=" ]
    ],
    [
        "NAME:OutputTerminations",
        "tpwire2/w1:=" , [ "Imped:="
        "tpwire2/w2:=" , [ "Source:=" ]
    ],
    [
        "NAME:HarnessAttrs",
        "Name:=" , "MyHarness"
    ])

```

VB Syntax	UpdateCableHarness Array(<parameters>)
VB Example	<pre> oModule.UpdateCableHarness Array("NAME:HarnessParams", "Bundle:=", "bundle1", "TwistAlongPath:=", _ "360deg", "Route:=", "Polyline2", "AutoOrient:=", false, "Origin:=", Array(_ "-24mm", "24mm", "0mm"), "XAxisEnd:=", Array("-30mm", "24mm", "0mm"), "PlaneFlip:=", _ true, Array("NAME:RefConductors", "tpwire2/w1"), Array("NAME:InputTerminations", </pre>

```
"tpwire2/w1:=", Array("Source:=", __
    "5V", "Imped:=", "50ohm"), "tpwire2/w2:=", Array("Source:=", "5V", "Imped:=", __
    "50ohm)), Array("NAME:OutputTerminations", "tpwire2/w1:=", Array("Source:=", __
    "5V", "Imped:=", __
    "50ohm), "tpwire2/w2:=", Array("Imped:=", "50ohm))), Array("NAME:Harn-
nessAttrs", "Name:=", __
"MyNewHarness")
```

11 - Output Variable Script Commands

Output variable commands should be executed by the "OutputVariable" module.

First, obtain the output variable module from oDesign and use it for output variable commands:

```
Set oModule = oDesign.GetModule("OutputVariable")
oModule.<CommandName><args>
```

The commands are:

[CreateOutputVariable](#)

[DeleteOutputVariable](#)

[DoesOutputVariableExist](#)

[EditOutputVariable](#)

[GetOutputVariableValue](#)

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	Icepak > Results > Output Variables.		
Parameters	Name	Type	Description

	<table border="1"><tr><td><OutputVarName></td><td>String</td><td>Name of the new output variable.</td></tr><tr><td><Expression></td><td>Value</td><td>Value to assign to the variable.</td></tr><tr><td><SolutionName></td><td>String</td><td>Name of the solution, as seen in the output variable UI.</td></tr><tr><td><reportType></td><td>String</td><td>The name of the report type as seen in the output variable UI.</td></tr><tr><td><ContextArray></td><td>Array</td><td>Structured array containing context for which the output variable expression is being evaluated. Array ("Context:=", <string>)</td></tr></table>	<OutputVarName>	String	Name of the new output variable.	<Expression>	Value	Value to assign to the variable.	<SolutionName>	String	Name of the solution, as seen in the output variable UI.	<reportType>	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. Array ("Context:=", <string>)
<OutputVarName>	String	Name of the new output variable.														
<Expression>	Value	Value to assign to the variable.														
<SolutionName>	String	Name of the solution, as seen in the output variable UI.														
<reportType>	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. Array ("Context:=", <string>)														
Return Value	None.															

Python Syntax	CreateOutputVariable (<OutputVarName>, <Expression>, <SolutionName>, <reportType solutionType>, <contextArray domainArray>)
Python Example	

VB Syntax	CreateOutputVariable <OutputVarName>, <Expression>, <SolutionName>, <reportType solutionType>, <contextArray domainArray>
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	Icepak > Results > Output Variables. In the Output Variables window, click Delete .						
Parameters	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><OutputVarName></td> <td>String</td> <td>Name of the output variable.</td> </tr> </tbody> </table>	Name	Type	Description	<OutputVarName>	String	Name of the output variable.
Name	Type	Description					
<OutputVarName>	String	Name of the output variable.					
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	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><outputVariableName></td> <td>String</td> <td>The output variable name.</td> </tr> </tbody> </table>	Name	Type	Description	<outputVariableName>	String	The output variable name.
Name	Type	Description					
<outputVariableName>	String	The output variable name.					
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-

			able UI.
<ContextArray>	Array	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

	<FileName>	String	Name of the file include path.
Return Value	Boolean: <ul style="list-style-type: none">• 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	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>, <ContextArray>
VB Example	Val = oDesign.GetOutputVariableValue "test", "Freq = '20Ghz' Theta='20deg' Phi='30deg'", "TR", "Standard", Array()

ImportOutputVariables

Imports output variables from a file.

UI Access	Click on Import in the Output Variables dialog.						
Parameters	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><FileName></td> <td>String</td> <td>Name of the file include path.</td> </tr> </tbody> </table>	Name	Type	Description	<FileName>	String	Name of the file include path.
Name	Type	Description					
<FileName>	String	Name of the file include path.					
Return Value	<p>Boolean:</p> <ul style="list-style-type: none"> • True - output variable successfully imported. • False - error when import output variables. 						

Python Syntax	ImportOutputVariables(<FileName>)
Python Example	oModule.ImportOutputVariables ("C:/output_var.aoutvar")

VB Syntax	ImportOutputVariables <FileName>
VB Example	oModule.ImportOutputVariables "C:/output_var.aoutvar"

SimValueContext

SimValueContext holds context information for a trace, and describes how data for a trace should be extracted from the simulation. SimValueContext contains a list of 14 required initial values:

```
SimValueContext (
    Domain ID, Calculation Type, Number of Cycles, Rise Time,
    Step, Impulse, Context ID, Window Width,
    Window Type, TDR Kaiser Parameter, Hold Time, DeviceName,
    TDR Step Time, DR Maximum Time )
```

For example, the following indicates a trace in the Time Domain, Standard Calculation with the number of cycles being 2:

```
"SimValueContext:=", Array(1, 0, 2, 0, false, false, -1, 1, 0, 1, 1, "", 0, 0)
```

Additional, context-specific values may follow the required values, as described in subsection 15 below.

1. Domain ID

No Domain	0
Time Domain	1

Spectrum Domain	2
Sweep Domain	3
Device Domain	4
SinglePt Domain	5
LoadPull Domain	6
Transient Domain	7
Budget Domain	8
NetworkFunction Domain	9
Oscillator Domain	55802
Noise Domain	55803
Transfer Function Domain	55807
Time Frequency Domain	55808
Transient Time Domain	55809
Periodic AC Domain	55818
UI Domain	55819
Eye Measurement Domain	55823
Initial Response Domain	55824
Peak Distortion Domain	55825

2. Calculation Type

Standard Calculation	0
Device2_DCIV	1

Device3_DCIV_Output	2
Device3_DCIV_Input	3
Device3_DCIV_Transfer	4
Device3_DCIV_Reverse	5
Device2_ACLoad	6
Device3_ACLoad_Output	7
Device3_ACLoad_Input	8
Device3_ACLoad_Transfer	9
Device3_ACLoad_Reverse	10
Constellation	11
EyeDiagram	12
FreeX (Statistic Report)	13

3. **Number of Cycles** — Used in Time Domain in HarmonicBalance analysis.
4. **Rise Time** — Not used by Designer/Nexxim.
5. **Step** — Not used by Designer/Nexxim.
6. **Impulse** — Not used by Designer/Nexxim.
7. **Context ID** — Not used by Designer/Nexxim.
8. **Window Width** — Not used by Designer/Nexxim.
9. **Window Type** — Not used by Designer/Nexxim.
10. **TDR Kaiser Parameter** — Not used by Designer/Nexxim.
11. **Hold Time** — Not used by Designer/Nexxim.
12. **DeviceName** — Not used by Designer/Nexxim.

13. **TDR Step Time** — Not used by Designer/Nexxim.
14. **TDR Maximum Time** — Not used by Designer/Nexxim.
15. **Context-specific values** — Used in Time Domain in HarmonicBalance analysis.

Context-specific values are entered in the format "key, true/false, keyvalue", where:

- **"key"** is the name of the key being set.
- **"true/false"** indicates whether the key is a string value.
- **"keyvalue"** is the value of the key.
- The order of the context keys is not significant.
- Context keys have software defaults that will be used if not provided in the script.

VB Example:

```
"SimValueContext:=", Array(1, 0, 2, 0, false, false, -1, 1, 0, 1, 1, "", 0, 0,  
"DE", false, "0",  
"DP", false, "20000000",  
"DT", false, "0.001",  
"WE", false, "10ns",  
"WM", false, "10ns",  
"WN", false, "0ns",  
"WS", false, "0ns"))
```

a. Plotting Range for Time domain in Transient and QuickEye analysis:

Description	Key Name	Is a string?	Key Value
Start Time	WS	False	0ns
Stop Time	WE	False	10ns
Minimum Time	WM	False	0ns
Maximum Time	WN	False	10ns
Is Thinning Enabled?	DE	False	0
Dy/dx Tolerance	DT	False	0.001
Number of points	DP	False	20000000

b.Transient report context for Spectral domain in Transient analysis:

Description	Key Name	Is a string?	Key Value
Start Time	TS	False	0ns
Stop Time	TE	False	10ns
Max Harmonics	MH	False	100
Max Frequency	MF	False	*
Window type	WT	False	0
Width Percentage	WW	False	100
Kaiser Parameter	KP	False	0
Adjust Coherent Gain	CG	False	0

* Script can specify either MH or MF. If neither is specified, Max Harmonics is set to 100. If both are specified, MF is used.

Window Type	ID
Rectangular	0
Bartlett	1
Blackman	2
Hamming	3
Hanning	4
Kaiser	5
Welch	6
Weber	7
Lanzcos	8

c. Eyeprobe index context for UI domain, Time domain, Eye Measurement domain in VerifEye and QuickEye analysis:

Description	Key Name	Is a string?	Key Value
Eyeprobe compinst ID	PCID	False	0

d. Eyesource index context for Initial Response domain and Peak Distortion domain in VerifEye and QuickEye analysis:

Description	Key Name	Is a string?	Key Value
Eyesource compinst ID	SCID	False	0

e. UI domain context in VerifEye and QuickEye analysis:

Description	Key Name	Is a string?	Key Value
Use midpoint?	MIDPOINT	False	0 - Don't use midpoint. 1 - Use midpoint of amplitude. 2 - Use midpoint of UI.
Minimum latch overdrive	MLO	False	0

f. Distribution Context for UI Domain in VerifEye and QuickEye analysis:

Description	Key Name	Is a string?	Key Value
Use distribution?	USE_DIST	False	0 - No 1 - Yes
Distribution type	DIST	False	0 - Receiver Jitter 1 - Receiver Noise 2 - User Defined

Receiver Jitter Parameters

Description	Key Name	Is a string?	Key Value
DLL standard deviation	DSD	False	0
Distribution type	DIST	False	0
DLL taps	DMN	False	0
Static Offset	SOFF	False	0
Number of Gaussian data sets	NUMG	False	0
Gaussian std deviation	GS0,GS1...	False	0
Offset mean	GM1,GM1...	False	0
Number of Uniform data sets	NUMU	False	0
Uniform width	UW0,UW1...	False	0
Uniform mean	UM1,UM1...	False	0

Receiver Noise Parameters

Description	Key Name	Is a string?	Key Value
Number of Gaussian data sets	NUMG	False	0
Gaussian std deviation	GS0,GS1...	False	0
Number of Uniform data sets	NUMU	False	0
Uniform width	UW0,UW1...	False	0

User Defined Parameters

Description	Key Name	Is a string?	Key Value
Number of XY data pairs	NUMG	False	0
X data	X0,X1,X2...	False	0
Y data	Y0,Y1,Y2...	False	0
Cutoff probability	CP	False	0

This page intentionally
left blank.

12 - 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](#)

[AddVerifEyeAnalysis](#)

[ClearAllMarkers](#)

[ClearAllTraceCharacteristics](#)

[CopyReportDefinitions](#)

[CopyReportData](#)

[CopyTraceDefinitions](#)

[CopyTracesData](#)

[CreateReport](#)

CreateReportFromFile

[CreateReportFromTemplate](#)

[CreateReportOfAllQuantities](#)

[DeleteAllReports](#)

[DeleteReports](#)

[DeleteTraces](#)

EditQuickEyeAnalysis

EditVerifEyeAnalysis

[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	<table border="1"><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><ReportName></td><td>String</td><td>Name of the report.</td></tr></table>			Name	Type	Description	<ReportName>	String	Name of the report.
Name	Type	Description							
<ReportName>	String	Name of the report.							
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	<table border="1"><tr><td>Name</td><td>Type</td><td>Description</td></tr></table>			Name	Type	Description
Name	Type	Description				

	<ReportName>	String	Name of the report.
	<Def>	Array	<p>Structured array:</p> <pre>Array("NAME:CartesianLimitLine", Array("NAME:XValues", <integer X values>), "XUnits:=", <string unit of measure for X>, Array("NAME:YValues", <integer Y values>), "YUnits:=", "<string unit of measure for Y>", "YAxis:=", <string name of associated Y axis>)</pre>
Return Value	None.		

	Python Syntax	AddCartesianLimitLine (<ReportName>, <Def>)
	Python Example	<pre>oModule.AddCartesianLimitLine("Project Outputs", ["NAME:CartesianLimitLine", ["NAME:XValues", 0, 2, 5, 7, 10, 15], "XUnits:=", "s", ["NAME:YValues", 0.05, 0.3, 0.65, 0.825, 0.95, 1], "YUnits:=", "mV", "YAxis:=", "Y1"])</pre>

VB Syntax	AddCartesianLimitLine < <i>ReportName</i> >, < <i>Def</i> >
VB Example	<pre> oModule.AddCartesianLimitLine "Project Outputs", Array("NAME:CartesianLimitLine", Array("NAME:XValues", 0, 2, 5, 7, 10, 15), "XUnits:=", "s", Array("NAME:YValues", 0.05, 0.3, 0.65, 0.825, 0.95, 1), "YUnits:=", "mV", "YAxis:=", "Y1")) </pre>

AddCartesianLimitLineFromCurve

Adds a limit line to a report from selected curve on the plot.

UI Access	Report2D > Add Limit Line > From Selected Curve...		
Parameters	Name < <i>ReportName</i> >	Type String	Description Name of the report.
	< <i>LimitLineParams</i> >	Type String	<p>Structured array.</p> <pre> Array("NAME:CartesianLimitLineFromCurve", "TraceName:=", <string name of selected trace>, "CurveName:=", <string name of selected curve>, "Start:=", "<value><unit>", "Stop:=", "<value><unit>", "YAxis:=", <integer Y-Axis number>,) </pre>

		<pre>"YOffset:=", <value offset from Y-Axis>, "CreateMode:=", "<AboveCurve BelowCurve Above and Below Curve>", "YShiftPercent:=", <value percentage to shift from Y>)</pre>
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< <i>ReportName</i> >, < <i>LimitLineParams</i> >
VB Example	<pre> oModule.AddCartesianLimitLineFromCurve _ "Variables Plot 2", _ Array("NAME:CartesianLimitLineFromCurve", _ "TraceName:=", "Phase", "CurveName:=", "", _ "Start:=", "0deg", "Stop:=", "375deg", _ "YAxis:=", 1, "YOffset:=", 0, _ "CreateMode:=", "AboveCurve", _ "YShiftPercent:=", 10) </pre>

AddCartesianLimitLineFromEquation

Adds a limit line to a report from a specified equation.

UI Access	Report2D > Add Limit Line > Specify Equation...											
Parameters	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><<i>ReportName</i>></td> <td>String</td> <td>Name of the report.</td> </tr> <tr> <td><<i>LimitLineParams</i>></td> <td>Array</td> <td> <p>Structured array.</p> <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>, </pre> </td> </tr> </tbody> </table>	Name	Type	Description	< <i>ReportName</i> >	String	Name of the report.	< <i>LimitLineParams</i> >	Array	<p>Structured array.</p> <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>, </pre>		
Name	Type	Description										
< <i>ReportName</i> >	String	Name of the report.										
< <i>LimitLineParams</i> >	Array	<p>Structured array.</p> <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>, </pre>										

		"Equation:=", <string specified equation>
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"]) </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")
--	----------------------

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.
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.
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	<table border="1"> <tr> <td>Name</td> <td>Type</td> <td>Description</td> </tr> <tr> <td><ReportName></td> <td>String</td> <td>Name of report.</td> </tr> </table>			Name	Type	Description	<ReportName>	String	Name of report.
Name	Type	Description							
<ReportName>	String	Name of report.							

	<code><MarkerName></code>	String	Marker name, including any trailing number.
	<code><CurveName></code>	String	Full trace name.
	<code><PrimarySweepValue></code>	String	Primary sweep value, including unit.
Return Value	None		

Python Syntax	<code>AddMarker(<ReportName>, <MarkerName>, <CurveName>, <PrimarySweepValue>)</code>
Python Example	<pre>oModule.AddMarker("XY Plot 1", "m5", "GS1.VAL : TR4 : Cartesian", "3.6159999999997s")</pre>

VB Syntax	<code>AddMarker <ReportName>, <MarkerName>, <CurveName>, <PrimarySweepValue></code>
VB Example	<pre>Set oModule = oDesign.GetModule("ReportSetup") oModule.AddMarker "XY Plot1", "m1", "mag(S(Port1 Port1)) : Setup1: LastAdaptive : Cartesian", "0.3in"</pre>

AddNote

Adds a note at a specified location to a given report.

UI Access	Right-click on the plot and select Add Note .
------------------	--

Parameters	Name	Type	Description
	<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>

```

    "String:=", "This is a note."
]
)

```

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

		<p>no arguments exist.</p> <p>To see which argument(s) a function takes, see the bottom of the Add Trace Characteristics dialog box.</p> <p>For function with one argument:</p> <pre>Array(<value>)</pre> <p>For function with multiple arguments:</p> <pre>Array(<value>, <value>, ...)</pre>
<RangeArgs>	Array	<p>Required. Array containing either string "Full" for a full sweep range, or "Specified" and strings containing the start and end values for the frequency range.</p> <p>For example:</p> <pre>Array("Specified", "19.5GHz", "24.4GHz")</pre>
Return Value	None.	

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	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><ReportName></td> <td>String</td> <td>Name of Report</td> </tr> <tr> <td><SolutionName></td> <td>String</td> <td>Name of the solution as listed in the Modify Report dialog box.</td> </tr> <tr> <td><ContextArray></td> <td>Array</td> <td> Context for which the expression is being evaluated. This can be an empty string if there is no context. Array("Domain:=", <DomainType>) <DomainType> ex. "Sweep" or "Time" Array("Context:=", <GeometryType>) <GeometryType> ex. "Infinite Spheren", "Spheren", "Polylinen" </td> </tr> <tr> <td><FamiliesArray></td> <td>Array</td> <td> Contains sweep definitions for the report. Array("<VariableName>:= ", <ValueArray>) <ValueArray> Array("All") or Array("Value1", "Value2", </td> </tr> </tbody> </table>	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. Array("Domain:=", <DomainType>) <DomainType> ex. "Sweep" or "Time" Array("Context:=", <GeometryType>) <GeometryType> ex. "Infinite Spheren", "Spheren", "Polylinen"	<FamiliesArray>	Array	Contains sweep definitions for the report. Array("<VariableName>:= ", <ValueArray>) <ValueArray> Array("All") or Array("Value1", "Value2",
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. Array("Domain:=", <DomainType>) <DomainType> ex. "Sweep" or "Time" Array("Context:=", <GeometryType>) <GeometryType> ex. "Infinite Spheren", "Spheren", "Polylinen"														
<FamiliesArray>	Array	Contains sweep definitions for the report. Array("<VariableName>:= ", <ValueArray>) <ValueArray> Array("All") or Array("Value1", "Value2",														

			<p>... "Valuen")</p> <p>examples of <VariableName> "Freq", "Theta", "Distance"</p>
	<ReportdataArray>	Array	<p>This array contains the report quantity and X, Y, and (Z) axis definitions.</p> <pre>Array("X Component:=", <VariableName>, "Y Component:=", <VariableName> <ReportQuantityArray>) <ReportQuantityArray> ex. Array("dB(S(Port1, Port1))")</pre>
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 < <i>ReportName</i> > < <i>SolutionName</i> > < <i>ContextArray</i> > < <i>FamiliesArray</i> > < <i>ReportdataArray</i> >
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	<table border="1"> <tr> <td>Name</td> <td>Type</td> <td>Description</td> </tr> <tr> <td><<i>ReportName</i>></td> <td>String</td> <td>Name of the report to apply settings to.</td> </tr> <tr> <td><<i>TemplateFile</i>></td> <td>String</td> <td>Template file name with path.</td> </tr> <tr> <td><<i>PropertyType</i>></td> <td>String</td> <td>Property types to apply. ("Graphical" "Data" "All")</td> </tr> </table>	Name	Type	Description	< <i>ReportName</i> >	String	Name of the report to apply settings to.	< <i>TemplateFile</i> >	String	Template file name with path.	< <i>PropertyType</i> >	String	Property types to apply. ("Graphical" "Data" "All")
Name	Type	Description											
< <i>ReportName</i> >	String	Name of the report to apply settings to.											
< <i>TemplateFile</i> >	String	Template file name with path.											
< <i>PropertyType</i> >	String	Property types to apply. ("Graphical" "Data" "All")											

Return Value	None.
---------------------	-------

Python Syntax	ApplyReportTemplate(<ReportName>, <TemplateFile>, <PropertyType>)
Python Example	oModule.ApplyReportTemplate("Variables Plot 1", "C:/MyTemplate.rpt", "Graphical")

VB Syntax	ApplyReportTemplate <ReportName>, <TemplateFile>, <PropertyType>
VB Example	oModule.ApplyReportTemplate "Variables Plot 1", "C:/MyTemplate.rpt", "Graphical"

ClearAllMarkers

Clears all markers from a report.

UI Access	Report2D > Markers > Clear All.						
Parameters	<table border="1"> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> <tr> <td><ReportName></td> <td>String</td> <td>Name of Report</td> </tr> </table>	Name	Type	Description	<ReportName>	String	Name of Report
Name	Type	Description					
<ReportName>	String	Name of Report					
Return Value	None						

Python Syntax	ClearAllMarkers(<ReportName>)
Python Example	oModule.ClearAllMarkers("XY Plot 1")

VB Syntax	ClearAllMarkers <ReportName>
VB Example	oModule.ClearAllMarkers "XY Plot 1"

ClearAllTraceCharacteristics

Clears all trace characteristics from the legend in a report.

UI Access	Report2D > Trace Characteristics > Clear All.						
Parameters	<table border="1"><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><PlotName></td><td>String</td><td>Name of the plot</td></tr></table>	Name	Type	Description	<PlotName>	String	Name of the plot
Name	Type	Description					
<PlotName>	String	Name of the plot					
Return Value	None						

Python Syntax	ClearAllTraceCharacteristics(<PlotName>)
Python Example	oModule.ClearAllTraceCharacteristics("XY Plot 1")

VB Syntax	ClearAllTraceCharacteristics <PlotName>
VB Example	oModule.ClearAllTraceCharacteristics "XY Plot 1"

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.
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	<table border="1"><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><ReportsArray></td><td>Array</td><td>Names of reports from which to copy the definitions</td></tr></table>			Name	Type	Description	<ReportsArray>	Array	Names of reports from which to copy the definitions
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	<table border="1"> <tr> <td>Name</td> <td>Type</td> <td>Description</td> </tr> <tr> <td><ReportsArray></td> <td>Array</td> <td>Names of reports from which to copy data</td> </tr> </table>			Name	Type	Description	<ReportsArray>	Array	Names of reports from which to copy data
Name	Type	Description							
<ReportsArray>	Array	Names of reports from which to copy data							
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	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><ReportName></td> <td>String</td> <td>Name of Report</td> </tr> <tr> <td><TracesArray></td> <td>Array</td> <td>Trace definitions to copy</td> </tr> </tbody> </table>	Name	Type	Description	<ReportName>	String	Name of Report	<TracesArray>	Array	Trace definitions to copy
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	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><ReportName></td> <td>String</td> <td>Name of Report.</td> </tr> <tr> <td><TraceArray></td> <td>Array</td> <td>Trace definitions from which to copy corresponding data.</td> </tr> </tbody> </table>	Name	Type	Description	<ReportName>	String	Name of Report.	<TraceArray>	Array	Trace definitions from which to copy corresponding data.
Name	Type	Description								
<ReportName>	String	Name of Report.								
<TraceArray>	Array	Trace definitions from which to copy corresponding data.								

Return Value	None.
---------------------	-------

Python Syntax	CopyTracesData(<ReportName>, <TracesArray>)
Python Example	<pre>oModule.CopyTracesData ("Transmission", ["mag(S(Port1,Port2))"])</pre>

VB Syntax	CopyTracesData <ReportName>, <TracesArray>
VB Example	<pre>oModule.CopyTracesData _ "C11", Array("mag(S(Port1,Port2))")</pre>

y, l

CreateReport

Creates a new report with a single trace and adds it to the **Results** branch in the project tree.

UI Access	Right-click on Results > Create [Type] Report									
Parameters	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><ReportName></td> <td>String</td> <td>Name of report.</td> </tr> <tr> <td><ReportType></td> <td>String</td> <td>Type of report. Possible values are: "Modal S Parameters" - Only for Driven Modal solution-type problems with ports. "Terminal S Parameters" - Only for Driven Terminal solution-type problems with ports.</td> </tr> </tbody> </table>	Name	Type	Description	<ReportName>	String	Name of report.	<ReportType>	String	Type of report. Possible values are: "Modal S Parameters" - Only for Driven Modal solution-type problems with ports. "Terminal S Parameters" - Only for Driven Terminal solution-type problems with ports.
Name	Type	Description								
<ReportName>	String	Name of report.								
<ReportType>	String	Type of report. Possible values are: "Modal S Parameters" - Only for Driven Modal solution-type problems with ports. "Terminal S Parameters" - Only for Driven Terminal solution-type problems with ports.								

		<p>"Eigenmode Parameters" - Only for Eigenmode solution-type problems.</p> <p>"Fields"</p> <p>"Far Fields" - Only for problems with radiation or PML boundaries.</p> <p>"Near Fields" - Only for problems with radiation or PML boundaries.</p> <p>"Emission Test"</p>
<DisplayType>	String	<p>Type of display.</p> <p>If ReportType is "Modal S Parameters", "Terminal S Parameters", or "Eigenmode Parameters", then set to one of the following:</p> <p>"Rectangular Plot", "Polar Plot", "Radiation Pattern", "Smith Chart", "Data Table", "3D Rectangular Plot", "3D Rectangular Bar Plot" or "3D Polar Plot".</p> <p>If <ReportType> is "Fields", then set to one of the following:</p> <p>"Rectangular Plot", "Polar Plot", "Radiation Pattern", "Rectangular Contour Plot", "Data Table", "3D Rectangular Plot", or "3D Rectangular Bar Plot".</p> <p>If <ReportType> is "Far Fields" or "Near Fields", then set to one of the following:</p> <p>"Rectangular Plot", "Radiation Pattern", "Rectangular Contour Plot" "Data Table", "3D Rectangular Plot", "3D Rectangular Bar Plot" or "3D Polar Plot"</p> <p>If <ReportType> is "Emission Test", then set to one of the following:</p>

		"Rectangular Plot" or "Data Table"
	<SolutionName>	String Name of the solution as listed in the Modify Report dialog box.
	<ContextArray>	Array Context for which the expression is being evaluated. This can be an empty string if there is no context. Array("Domain:=", <DomainType> <DomainType> ex. "Sweep" or "Time" Array("Context:=", <GeometryType> <GeometryType> ex. "Infinite Spheren", "Spheren", "Polylinen"
	<FamiliesArray>	Array Contains sweep definitions for the report. Array("<VariableName>:= ", <ValueArray> <ValueArray> Array("All") or Array("Value1", "Value2", ... "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 Syn-	CreateReport(<ReportName>, <ReportType>, <DisplayType>, <SolutionName>, <ContextArray>, <FamiliesArray>,
-------------	---

tax	<ReportdataArray>, <ExtendedInfo>)
	<p>Three examples are given below: (nominal, Optimetrics, spectral)</p> <p>Nominal Example:</p> <pre>oModule.CreateReport("XY Plot 2", "Standard",_ "Rectangular Plot", "TR", ["NAME:Context",_ "SimValueContext:=", [] 1, 0, 2, 0, false,_ false, -1, 1, 0, 1, 1, "", 0, 0]], ["Time:=",_ ["All"], "aaa:=", ["Nominal"]],_ ["X Component:=", "Time", "Y Component:=",_ ["E1.I"]]][])</pre> <p>Optimetrics Example:</p> <pre>oModule.CreateReport ("XY Plot 3", "Standard",_ "Rectangular Plot", "TR", ["NAME:Context",_ "OptiSetup:=", "ParametricSetup1", "SimValueCor_ [1, 0, 2, 0, false, false, 0, 1,_ 0, 1, 1, "", 0, 0]], ["Time:=", ["All"],_ "aaa:=", ["Nominal"]], ["X Component:=",_ "Time", "Y Component:=", ["E1.I"]], [])</pre> <p>Spectral Example:</p> <pre>oModule.CreateReport ("XY Plot 4", "Standard",_</pre>
Python Example	

```

    "Rectangular Plot", "TR", ["NAME:Context",_
    "SimValueContext:=", [ 2, 0, 2, 0, false, false,
    -1, 1, 0, 1, 1, "", 0, 0, "CG", false, "0", "K",
    false, "0", "MH", false, "100", "TH", false, "4",
    "TH", false, "40", "TS", false, "0ns", "UF", fa,
    "0", "WT", false, "0", "WW", false, "100"]],_
    ["Spectrum:=", ["All"], "aaa:=", _,
    ["Nominal"]]], ["X Component:=", "Spectrum", _,
    "Y Component:=", ["mag(E1.I)"]], [])

```

Partial Discharge Example

```

oModule.CreateReport("Partial Discharge Plot 2", "Partial Discharge", "Rectangular Plot",
"PartialDischarge1 : PartialDischarge", [],

[

    "GasPressure:="      , ["All"],
    "Freq:="             , ["All"],
    "bend_angle:="       , ["All"]

], [
    [
        "X Component:="   , "GasPressure",
        "Y Component:="   , ["Power"]
    ]
])

```

Example 3D Rectangular Bar Plot with Change Bar properties

```
oModule = oDesign.GetModule("ReportSetup")

oModule.CreateReport("S Parameter Plot 4", "Modal Solution Data", "3D Rectangular Bar
Plot", "Setup1 : Sweep", [],

[

    "Freq:="                  , ["All"],
    "UU:="                   , ["All"],
    "ZZZ:="                  , ["Nominal"]

], 

[

    "X Component:="          , "Freq",
    "Y Component:="          , "UU",
    "Z Component:="          , ["dB(S(wDipole1_1_p1,wDipole1_1_p1))"]

])

oModule.ChangeProperty(
[

    "NAME:AllTabs",
[

    "NAME:Bar",
[


```

```
        "NAME:PropServers",
        "S Parameter Plot 4:Traces-dB(S(wDipole1_1_p1,wDipole1_1_p1))"
    ],
    [
        "NAME:ChangedProps",
        [
            "NAME:Transparency",
            "Value:=" , "0.5"
        ]
    ]
)
oModule.ChangeProperty(
[
    "NAME:AllTabs",
    [
        "NAME:Bar",
        [
            "NAME:PropServers",
            "S Parameter Plot 4:Traces-dB(S(wDipole1_1_p1,wDipole1_1_p1))"
        ],
    ]
]
```

```
[  
    "NAME:ChangedProps",  
    [  
        "NAME:Filled",  
        "Value:=" , False  
    ]  
]  
]  
]  
)  
oModule.ChangeProperty(  
[  
    "NAME:AllTabs",  
    [  
        "NAME:Bar",  
        [  
            "NAME:PropServers",  
            "S Parameter Plot 4:Traces-dB(S(wDipole1_1_p1,wDipole1_1_p1))"  
        ],  
        [  
            "NAME:ChangedProps",
```

```
[  
    "NAME:Filled",  
    "Value:=" , True  
]  
]  
]  
]  
)  
oModule.ChangeProperty(  
[  
    "NAME:AllTabs",  
[  
    "NAME:Bar",  
[  
    "NAME:PropServers",  
    "S Parameter Plot 4:Traces-dB(S(wDipole1_1_p1,wDipole1_1_p1))"  
],  
[  
    "NAME:ChangedProps",  
[  
    "NAME>Show Outline",  
    "Value:=" , True
```

```
        ]
    ]
]
])

oModule.ChangeProperty(
[
    "NAME:AllTabs",
    [
        "NAME:Bar",
        [
            "NAME:PropServers",
            "S Parameter Plot 4:Traces-dB(S(wDipole1_1_p1,wDipole1_1_p1))"
        ],
        [
            "NAME:ChangedProps",
            [
                "NAME:Outline Color",
                "R:=" , 0,
                "G:=" , 0,
                "B:=" , 160
            ]
        ]
    ]
]
```

```
        ]
    ]
]
])

oModule.ChangeProperty(
[
    "NAME:AllTabs",
    [
        "NAME:Bar",
        [
            "NAME:PropServers",
            "S Parameter Plot 4:Traces-dB(S(wDipole1_1_p1,wDipole1_1_p1))"
        ],
        [
            "NAME:ChangedProps",
            [
                "NAME:Outline Width",
                "Value:=" , "2"
            ]
        ]
    ]
]
```

```
        ])
oModule.ChangeProperty(
    [
        "NAME:AllTabs",
        [
            "NAME:Bar",
            [
                "NAME:PropServers",
                "S Parameter Plot 4:Traces-dB(S(wDipole1_1_p1,wDipole1_1_p1))"
            ],
            [
                "NAME:ChangedProps",
                [
                    "NAME:Bar Width",
                    "Value:=" , "Narrow"
                ]
            ]
        ]
    ])
oModule.ChangeProperty(
```

```
[  
    "NAME:AllTabs",  
    [  
        "NAME:Bar",  
        [  
            "NAME:PropServers",  
            "S Parameter Plot 4:Traces-dB(S(wDipole1_1_p1,wDipole1_1_p1))"  
        ],  
        [  
            "NAME:ChangedProps",  
            [  
                "NAME:Bar Width",  
                "Value:=" , "Wide"  
            ]  
        ]  
    ]  
)  
oModule.ChangeProperty(  
    [  
        "NAME:AllTabs",  
        [  
    ]
```

```
"NAME:Bar",
[
    "NAME:PropServers",
    "S Parameter Plot 4:Traces-dB(S(wDipole1_1_p1,wDipole1_1_p1))"
],
[
    "NAME:ChangedProps",
    [
        "NAME:Bar Infinity",
        "Value:=" , True
    ]
]
])
```

VB Syntax	CreateReport <ReportName>, <ReportType>, <DisplayType>, <SolutionName>, <ContextArray>, <FamiliesArray>, <ReportdataArray>, <ExtendedInfo>
VB Example	oModule.CreateReport "3D Cartesian Plot1", "Far Fields", _

```
"3D Cartesian Plot", "Setup1 : LastAdaptive", _  
    Array("Context:=", "Infinite Sphere1", "Domain:=", "Sweep"), _  
    Array("Theta:=", Array("All"), "Phi:=", Array("All")), _  
    "Freq:=", Array("10GHz")), _  
    Array("X Component:=", "Theta", _  
        "Y Component:=", "Phi", _  
        "Z Component:=", Array("rETotal")), _  
    Array()  
  
oModule.CreateReport "ReptSmithFreq", _  
    "Modal Solution Data", "Smith Plot", "Setup1 : Sweep1", _  
    Array("Domain:=", "Sweep"), _  
    Array("Freq:=", Array("All")), _  
    Array("Polar Component:=", _  
        Array("ln(Y(LumpPort1,LumpPort1))")), _  
    Array()  
  
oModule.CreateReport "Rectangular Contour Plot 2", "Far Fields", _  
    "Rectangular Contour Plot", "Setup1 : LastAdaptive", Array("Context:=", _  
        "Infinite Sphere1"), Array("_u:=", Array("All"), "_v:=", Array("All"), "Freq:=", _  
    Array( _  
        "5GHz")), Array("X Component:=", "_u", "Y Component:=", "_v", "Z Component:=", Array( _  
            "rETotal")), Array()
```

```
oModule.CreateReport "Rept2DRectFreq",_
    "Modal Solution Data", "XY Plot", _
    "Setup1 : Sweep1", _
    Array("Domain:=", "Sweep"), _
    Array("Freq:=", Array("All")), _
    Array("X Component:=", "Freq", _
    "Y Component:=", _ Array("dB(S(LumpPort1,LumpPort1))")), _
    Array()
```

CreateReport [Designer]

*Use:*Creates a new report with a single trace and adds it to the **Results** branch in the project tree.

*Command:*Product Menu>Results>Create <type> Report

*Syntax:*CreateReport <ReportName> <ReportType> <DisplayType> <SolutionName> <ContextArray> <FamiliesArray> <ReportdataArray>

*Return Value:*None

Parameters:<ReportName>

Type: <string>

Name of Report.

<ReportType>

Type: <string>

Possible values are:

- "Standard" - For most plot types.
- "Load Pull" - For load pull plots.
- "Constellation" - For constellation plots.
- "Data table" - For data tables.
- "Eye Diagram" - For eye diagrams.
- "Statistical" - For statistical plots.

<DisplayType>

Type: <string>

Possible values are:

- "Rectangular Plot", "Polar Plot", "Radiation Pattern", "Smith Chart", "Data Table", "3D Rectangular Plot", "3D Polar Plot", or
- "Rectangular Stacked Plot".

<SolutionName>

Type: <string>

Name of the solution as listed in the **Modify Report** dialog box.

For example: "Setup1 : Last Adaptive"

<ContextArray>

Type: Array of strings

Context for which the expression is being evaluated. This can be an empty string if there is no context.

Array("Domain:=", <DomainType>)

<DomainType>

ex. "Sweep" or "Time"

Array ("Context:=", <SimValueContext>)

Context for the trace. For more information see [SimValueContext](#).

<FamiliesArray>

Type: Array of strings

Contains sweep definitions for the report.

Array("<VariableName>:= ", <ValueArray>)

<ValueArray>

Array("All") or Array("Value1", "Value2", ... "Valuen")

examples of <VariableName>

"Freq", "Theta", "Distance"

<ReportDataArray>

Type: Array of strings

This array contains the report quantity and X, Y, and (Z) axis definitions.

```
Array("X Component:=", <VariableName>, "Y Component:=", <VariableName> | <ReportQuantityArray>)
```

```
<ReportQuantityArray>
```

```
ex. Array("dB(S(Port1, Port1))")
```

VB Example:

```
oModule.CreateReport "XY Stacked Plot 1", "Standard", "Rectangular Stacked Plot", _  
"LinearFrequency", Array("NAME:Context", "SimValueContext:=", Array(3, 0, 2, 0, _  
false, false, -1, 1, 0, 1, 1, "", 0, 0)), Array("F:=", Array("All")), Array("X Component:=", _  
"F", "Y Component:=", Array("dB(S(Port1,Port1))", "dB(S(Port1,Port2))", _  
"dB(S(Port2,Port1))", "dB(S(Port2,Port2))))), Array()
```

VB Example:

```
oModule.CreateReport "Data Table 1", "Standard", "Data Table", "LinearFrequency", _  
Array("NAME:Context", "SimValueContext:=", Array( _  
3, 0, 2, 0, false, false, -1, 1, 0, 1, 1, "", 0, 0)), Array("F:=", Array("All")), Array("X Com-  
ponent:=", _  
"F", "Y Component:=", Array("dB(S(Port1,Port1))")), Array()
```

VB Example:

```
oModule.CreateReport "3D Rectangular Plot 1", "Standard", "3D Rectangular Plot", _
```

```
"LinearFrequency", Array("NAME:Context", "SimValueContext:=", Array(3, 0, 2, 0, _  
false, false, -1, 1, 0, 1, 1, "", 0, 0)), Array("F:=", Array("All")), Array("X Component:=", _  
"F", "Y Component:=", "F", "Z Component:=", Array("dB(S(Port1,Port1))")), Array()
```

VB Example:

```
oModule.CreateReport "3D Rectangular Plot 2", "Standard", "3D Rectangular Plot", _  
"LinearFrequency", Array("NAME:Context", "SimValueContext:=", Array(3, 0, 2, 0, _  
false, false, -1, 1, 0, 1, 1, "", 0, 0)), Array("F:=", Array("All")), Array("X Component:=", _  
"F", "Y Component:=", "F", "Z Component:=", Array("dB(S(Port1,Port1))")), Array()
```

Python Syntax	CreateReport(<data_block>)
Python Example	<pre>oModule = oDesign.GetModule("ReportSetup") oModule.CreateReport("QuickEye Voltage Plot 1", "Eye Diagram", "Rectangular Plot", "QuickEyeAnalysis", ["NAME:Context", "SimValueContext:=" , [1,0,2,0,False,False, -1,1,0,1,1,"",0,0,"DE",False,"0","DP",False, "20000000","DT",False,"0.001","NUMLEVELS",False,"1","PCID", False,"3","PID",False,"0", "WE",False,"49.99995us","WM",False,"49.99995us","WN",False,</pre>

```
    "0ps", "WS", False, "0ps"]
],
[
    "Time:=" , ["All"]
],
[
    "Component:=" , ["aeyeprobe3"]
],
[
    "Unit Interval:=" , "1e-009s",
    "Offset:=" , "0",
    "Auto Delay:=" , True,
    "Manual Delay:=" , "0ps",
    "AutoCompCrossAmplitude:=" , True,
    "CrossingAmplitude:=" , "0mV",
    "AutoCompEyeMeasurementPoint:=" , True,
    "EyeMeasurementPoint:=" , "5e-010s"
])
)
```

CreateReportFromTemplate

Creates a report from a saved template.

UI Access	[product] > Results > Report Templates > PersonalLib > [TemplateName]								
Parameters	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><i><TemplatePath></i></td> <td>String</td> <td>Path to report template</td> </tr> </tbody> </table>			Name	Type	Description	<i><TemplatePath></i>	String	Path to report template
Name	Type	Description							
<i><TemplatePath></i>	String	Path to report template							
Return Value	None								

Python Syntax	CreateReportFromTemplate(<i><TemplatePath></i>)
Python Example	<pre>oModule.CreateReportFromTemplate ("C:/MyHFSSProjects/PersonalLib/ReportTemplates/TestTemplate.rpt")</pre>

VB Syntax	CreateReportFromTemplate <i><TemplatePath></i>
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	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><i><ReportType></i></td> <td>String</td> <td>Report type name as input parameter</td> </tr> <tr> <td><i><DisplayType></i></td> <td>String</td> <td>Display type name as input parameter</td> </tr> <tr> <td><i><SolutionName></i></td> <td>String</td> <td>Solution name as input parameter</td> </tr> </tbody> </table>			Name	Type	Description	<i><ReportType></i>	String	Report type name as input parameter	<i><DisplayType></i>	String	Display type name as input parameter	<i><SolutionName></i>	String	Solution name as input parameter
Name	Type	Description													
<i><ReportType></i>	String	Report type name as input parameter													
<i><DisplayType></i>	String	Display type name as input parameter													
<i><SolutionName></i>	String	Solution name as input parameter													

	<table border="1"> <tr><td><SimValueCtxt></td><td>String</td><td>A context name, or array of string that encoded the context(I).</td></tr> <tr><td><CategoryName></td><td>String</td><td>Category name as input parameter</td></tr> <tr><td><PointSet></td><td>Array</td><td>Array of strings(II)</td></tr> <tr><td><CommonComponentsOfTraces></td><td>Array</td><td>Array of strings(III)</td></tr> <tr><td><ExtTraceInfo></td><td>Array</td><td>Array of strings(IV)</td></tr> </table>	<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)
<SimValueCtxt>	String	A context name, or array of string that encoded the context(I).														
<CategoryName>	String	Category name as input parameter														
<PointSet>	Array	Array of strings(II)														
<CommonComponentsOfTraces>	Array	Array of strings(III)														
<ExtTraceInfo>	Array	Array of strings(IV)														
Return Value	None.															

Python Syntax	CreateReportOfAllQuantities(<ReportNameArg>, <ReportType>, <DisplayType>, <SolutionName>, <SimValueCtxt>, <CategoryName>, <PointSet>, <CommonComponentsOfTraces>, <ExtTraceInfo>)
Python Example	<pre>oModule.CreateReportOfAllQuantities("Smith Chart all", "Modal Solution Data", "Smith Chart", "Setup1 : LastAdaptive", [], "S Parameter", ["Freq:=", ["All"]], "offset:=", ["All"], "a:=", ["Nominal"], "b:=", ["Nominal"]], [], [])</pre>

VB Syntax	CreateReportOfAllQuantities <ReportNameArg>, <ReportType>, <DisplayType>, <SolutionName>, <SimValueCtxt>, <CategoryName>, <PointSet>, <CommonComponentsOfTraces>, <ExtTraceInfo>
VB Example	<pre>oModule.CreateReportOfAllQuantities "Smith Chart all", _ "Modal Solution Data", "Smith Chart", "Setup1 : LastAdaptive", _</pre>

```
Array(),"S Parameter", _  
Array("Freq:=", Array("All"), "offset:=", Array("All"), _  
"a:=", Array("Nominal"), "b:=", Array("Nominal")), _  
Array(), Array()
```

DeleteMarker

Use: Deletes the specified marker.

UI Access	[product]> Fields > Fields > Marker > Delete Marker .						
Parameters	<table border="1"><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><MarkerName></td><td>String</td><td>Name of the marker.</td></tr></table>	Name	Type	Description	<MarkerName>	String	Name of the marker.
Name	Type	Description					
<MarkerName>	String	Name of the marker.					
Return Value	None.						

Python Syntax	DeleteMarker(<MarkerName>)
Python Example	oModule.DeleteMarker ("m1")

VB Syntax	DeleteMarker <MarkerName>
VB Example	oModule.DeleteMarker "m1"

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	<code>DeleteAllReports()</code>
Python Example	<code>oModule.DeleteAllReports ()</code>

VB Syntax	<code>DeleteAllReports</code>
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	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><code><ReportNameArray></code></td> <td>Array</td> <td>Array of report names to be deleted</td> </tr> </tbody> </table>	Name	Type	Description	<code><ReportNameArray></code>	Array	Array of report names to be deleted
Name	Type	Description					
<code><ReportNameArray></code>	Array	Array of report names to be deleted					
Return Value	None.						

Python Syntax	DeleteReports(<ReportNameArray>)
Python Example	<code>oModule.DeleteReports (["Rept2DRectFreq"])</code>

VB Syntax	DeleteReports <ReportNameArray>
VB Example	<code>oModule.DeleteReports Array("Rept2DRectFreq")</code>

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.
Return Value	None.		

Python Syntax	DeleteTraceCharacteristics(<ReportName>, <TraceCharsNames>)
Python Example	<code>oModule.DeleteTraceCharacteristics("Variables Plot 1", ["lowercutoff", "gain-crossover"])</code>

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	<table border="1"> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> <tr> <td><TraceSelection></td> <td>Array</td> <td>Structured array define selections. Array ("<ReportName>:=", <TracesArray>, <TracesAr- ray>, ...)</td> </tr> <tr> <td><ReportName></td> <td>String</td> <td>Name of Report</td> </tr> <tr> <td><TracesArray></td> <td>Array</td> <td>Contains the traces to delete within a report Array (<Trace>, <Trace>, ...)</td> </tr> <tr> <td><Trace></td> <td>String</td> <td>A specific trace that the user wishes to delete</td> </tr> </table>	Name	Type	Description	<TraceSelection>	Array	Structured array define selections. Array ("<ReportName>:=", <TracesArray>, <TracesAr- ray>, ...)	<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
Name	Type	Description														
<TraceSelection>	Array	Structured array define selections. Array ("<ReportName>:=", <TracesArray>, <TracesAr- ray>, ...)														
<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 < <i>TraceSelection</i> >
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	<table border="1"><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><<i>DisplayType</i>></td><td>String</td><td>Specified display type to check.</td></tr></table>	Name	Type	Description	< <i>DisplayType</i> >	String	Specified display type to check.
Name	Type	Description					
< <i>DisplayType</i> >	String	Specified display type to check.					
Return Value	<p>Integer</p> <ul style="list-style-type: none">• 1 - trace characteristics is supported.• 0 - trace characteristics not supported.						

Python Syntax	DoesSupportTraceCharacteristics(< <i>DisplayType</i> >)
Python Example	<pre>oModule.DoesSupportTraceCharacteristics("Rectangular Plot")</pre>

VB Syntax	DoesSupportTraceCharacteristics < <i>DisplayType</i> >
VB Example	<pre>oModule.DoesSupportTraceCharacteristics "Rectangular Plot"</pre>

DumpAllReportsData

Dumps all reports data to an Ansoft report data file.

UI Access	N/A		
Parameters	Name <i><FileName></i>	Type String	Description File name with path.
Return Value	None.		

Python Syntax	DumpAllReportsData(<i><FileName></i>)
Python Example	oModule.DumpAllReportsData ("C:/ReportsData.rdat")

VB Syntax	DumpAllReportsData <i><FileName></i>
VB Example	oModule.DumpAllReportsData "C:/ReportsData.rdat"

EditCartesianXMarker

Edits an XMarker value.

UI Access	N/A		
Parameters	Name <i><ReportName></i>	Type String	Description Name of report.

Return Value	None.
---------------------	-------

Python Syntax	EditCartesianXMarker (<ReportName>, <MarkerName>, <XValue>)
Python Example	<code>oModule.EditCartesianXMarker ("XY Plot 1", "MX1", 0)</code>

VB Syntax	EditCartesianXMarker <ReportName>, <MarkerName>, <XValue>
VB Example	<code>oModule.EditCartesianXMarker "XY Plot1", "MX1", 0</code>

EditCartesianYMarker

Edits a YMarker value.

UI Access	N/A		
Parameters	Name	Type	Description
	<ReportName>	String	Name of report.
	<MarkerName>	String	Marker name, including any trailing number.
	<AxisName>	String	Name of axis.
	<YValue>	Double	Y coordinate.
	<CurveName>	String	Name of curve.
Return Value	None		

Python Syntax	EditCartesianYMarker (<ReportName>, <MarkerName>, <AxisName>, <YValue>, <CurveName>)
Python Example	<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	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><ReportName></td> <td>String</td> <td>Name of report.</td> </tr> <tr> <td><MarkerName></td> <td>String</td> <td>Marker name, including any trailing number.</td> </tr> <tr> <td><CurveName></td> <td>String</td> <td>Full trace name.</td> </tr> <tr> <td><PrimarySweepValue></td> <td>String</td> <td>Primary sweep value, including unit.</td> </tr> </tbody> </table>	Name	Type	Description	<ReportName>	String	Name of report.	<MarkerName>	String	Marker name, including any trailing number.	<CurveName>	String	Full trace name.	<PrimarySweepValue>	String	Primary sweep value, including unit.
Name	Type	Description														
<ReportName>	String	Name of report.														
<MarkerName>	String	Marker name, including any trailing number.														
<CurveName>	String	Full trace name.														
<PrimarySweepValue>	String	Primary sweep value, including unit.														
Return Value	None.															

Python Syntax	EditMarker(<ReportName>, <MarkerName>, <CurveName>, <PrimarySweepValue>)
Python Example	<pre>oModule.EditMarker("XY Plot 1", "m5", "GS1.VAL : TR4 : Cartesian", "3.6159999999997s")</pre>

--	--

VB Syntax	EditMarker <ReportName>, <MarkerName>, <CurveName>, <PrimarySweepValue>
VB Example	<pre>Set oModule = oDesign.GetModule("ReportSetup") oModule.EditMarker "XY Plot1", "m1", "mag(S(Port1 Port1)) : Setup1: LastAdaptive : Cartesian", "0.3in"</pre>

ExportEyeMaskViolation

Exports eye mask violations to a file.

UI Access	N/A									
Parameters	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><ReportName></td> <td>String</td> <td>Name of specified report.</td> </tr> <tr> <td><ExportFileName></td> <td>String</td> <td>File name to export to.</td> </tr> </tbody> </table>	Name	Type	Description	<ReportName>	String	Name of specified report.	<ExportFileName>	String	File name to export to.
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	<pre>oModule.ExportEyeMaskViolation("Variables Plot 1", "C:/eyemask1.csv")</pre>

VB Syntax	ExportEyeMaskViolation < <i>ReportName</i> >, < <i>ExportFileName</i> >
VB Example	oModule.ExportEyeMaskViolation "Variables Plot 1", "C:/eyemask1.csv"

ExportImageToFile [Reporter]

Exports a report image in a specified format. In Release 23.1, this command is fully supports -ng (non-graphical) mode.

UI Access	N/A		
Parameters	Name	Type	Description
	< <i>ReportName</i> >	String	Name of report to be exported.
	< <i>FileName</i> >	String	Full path of the exported image file name; with extension of jpg, gif, tiff, tif, bmp, or wrl.
	< <i>Width</i> >	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.
	< <i>Height</i> >	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(< <i>ReportName</i> >, < <i>FileName</i> >, < <i>Width</i> >, < <i>Height</i> >)
Python Example	<pre>oModule.ExportImageToFile("Rectangular Contour Plot 1", "D:/work/2015/UV-Export/ppl.gif", 0, 0)</pre>

VB Syntax	ExportImageToFile <ReportName>, <FileName>, <Width>, <Height>
VB Example	<pre> oModule.ExportImageToFile _ "Rectangular Contour Plot 1", _ "D:/work/2015/UV-Export/ppl.gif", 0, 0 </pre>

ExportModelImageToFile

Exports the model as an image file (*.avz, *.bmp, *.gif, *.jpeg, *.tiff, *.wrl). In Release 23.1, this command is fully supports -ng (non-graphical) mode. To export to Ensight use *.avz. For export to Ensight in -ng mode, the corresponding version of Ensight must be installed. On Linux, it might need manual set environment variable AWP_ROOT212 to its installation path, e.g. AWP_ROOT212-2=/installations/ansys_inc/v212/ for AnsysEDT v21.2 and Ensight 21.2.

ExportModelImageToFile supports export overlay of polar plot 3D with existing transformation (scaling, rotation and translation) in -ng (non-graphical) mode.

UI Access	Modeler > Export.																	
Parameters	<table border="1"> <tr> <td>Name</td> <td>Type</td> <td>Description</td> </tr> <tr> <td><path></td> <td>String</td> <td>Full file path including extension.</td> </tr> <tr> <td><width></td> <td>Integer</td> <td>Width in pixels (use 0 for default).</td> </tr> <tr> <td><height></td> <td>Integer</td> <td>Height in pixels (use 0 for default).</td> </tr> <tr> <td><Parameters></td> <td>Array</td> <td> <p>Structured array.</p> <pre> Array("NAME:SaveImageParams", "ShowAxis:=" , <string containing boolean>, "ShowGrid:=" , <string containing boolean>, ...) </pre> </td> </tr> </table>	Name	Type	Description	<path>	String	Full file path including extension.	<width>	Integer	Width in pixels (use 0 for default).	<height>	Integer	Height in pixels (use 0 for default).	<Parameters>	Array	<p>Structured array.</p> <pre> Array("NAME:SaveImageParams", "ShowAxis:=" , <string containing boolean>, "ShowGrid:=" , <string containing boolean>, ...) </pre>		
Name	Type	Description																
<path>	String	Full file path including extension.																
<width>	Integer	Width in pixels (use 0 for default).																
<height>	Integer	Height in pixels (use 0 for default).																
<Parameters>	Array	<p>Structured array.</p> <pre> Array("NAME:SaveImageParams", "ShowAxis:=" , <string containing boolean>, "ShowGrid:=" , <string containing boolean>, ...) </pre>																

			<pre> >ShowRuler:=" , <string containing boolean>, >ShowRegion:=" , <string>, >Selections:=" , <string>, >FieldPlotSelections:=" , <string>' # Comma delimited string. #Use to set which field plot to show. >Orientation:=" , <string> >ShowOrientationGadget:=" , <False> </pre>
Return Value	None.		

Python Syntax	ExportModelImageToFile(<path> <width> <height> <Parameters>)
Python Example	<pre> oEditor.ExportModelImageToFile ("C:/Users/jdoe/Desktop/export.bmp", 1920, 1080, ["NAME:SaveImageParams", "ShowAxis:=" , "True", "ShowGrid:=" , "True", "ShowRuler:=" , "True", "ShowRegion:=" , "Default", </pre>

```
    "Selections:=" , "",  
    "FieldPlotSelections:=" , "",  
    "FitToSelections:=" , "",  
    "FitToFieldPlotSelections:=" , "",  
    "Orientation:=" , ""  
])
```

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:=" , "", "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
Return Value	None.		

Python Syntax	ExportModelMeshToFile <filePath>, <selections>
Python Example	<code>oEditor.ExportModelMeshToFile("E:/MyDir/scriptRun/2Selected-ng.obj", ["BotCover-", "AveragingVolumeAtPeakRMSEfieldLocation"])</code>

VB Syntax	ExportModelMeshToFile <filePath>, <selections>
VB Example	<code>oEditor.ExportModelMeshToFile("E:/MyDir/scriptRun/2Selected-ng.obj", ["BotCover-", "AveragingVolumeAtPeakRMSEfieldLocation"])</code>

ExportPlot3DToFile [Reporter]

Use: Exports 3D polar, spherical and rectangular plot to a case file. It works in both graphical and NG mode..

Command: None.

Syntax: ExportPlot3DToFile(<plotName>, <path>)

Return Value: A 3D plot file.

Parameters: <plotName>

Type: <string>

Plot name.

<Path>

Type: <string>

Path to file.

Python Syntax	ExportPlot3DToFile(<name> <Path>)
Python Example	<pre>oModule = oDesign.GetModule("ReportSetup") oModule.UpdateReports(["rE Plot 1"]) oModule.UpdateReports(["Directivity Plot 1"]) oModule.UpdateReports(["Gain Plot 1"]) oModule.ExportPlot3DToFile("rE Plot 1", "D:/projects/test2-output/rEPlot1.case") oModule.ExportPlot3DToFile("Directivity Plot 1", "D:/projects/test2-output/DirectivityPlot1.case") oModule.ExportPlot3DToFile("Gain Plot 1", "D:/projects/test2-output/GainPlot1.case")</pre>

VB Syntax	ExportPlot3DToFile <plotName> <Path>
VB Example	<pre>oModule.ExportPlot3DToFile("rE Plot 1", "D:/projects/test2-output/rEPlot1.case")</pre>

ExportReport

Note:

The ExportReport script command has been replaced by the script command [ExportToFile](#). ExportReport remains in order to retain backward compatibility for existing scripts, but it is strongly recommended that you now use [ExportToFile](#).

Export a report to a data file.

Command: None

Syntax: ExportReport <ReportName>, <FileName>, <FileExtension>

Return Value: None

Parameters: <ReportName>

Type: string

<Filename>

Type: string

<FileExtension>

Type: string

VB Example:

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

```
Dim oModule

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

Set oProject = oDesktop.SetActiveProject("BJTinverter")
Set oDesign = oProject.SetActiveDesign("Nexxim1")
oDesign.ExportReport "Data Table 1", "table_test", "csv"
```

Python Syntax	ExportReport(<ReportName>, <FileName>)
Python Example	<pre>oDesign.ExportReport("Plot1", "c:\report1.dat")</pre>

ExportReportDataToFile

Exports report data to a file.

UI Access	N/A									
Parameters	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td><ReportName></td><td>String</td><td>Name of specified report.</td></tr><tr><td><ExportFileName></td><td>String</td><td>Name of export file. File extenstion "rdat" is expected.</td></tr></tbody></table>	Name	Type	Description	<ReportName>	String	Name of specified report.	<ExportFileName>	String	Name of export file. File extenstion "rdat" is expected.
Name	Type	Description								
<ReportName>	String	Name of specified report.								
<ExportFileName>	String	Name of export file. File extenstion "rdat" is expected.								

Return Value	None.
---------------------	-------

Python Syntax	ExportReportDataToFile(<ReportName>, <ExportFileName>)
Python Example	oModule.ExportReportDataToFile("Plot 1", "C:/Plot1data.rdat")

VB Syntax	ExportReportDataToFile <ReportName>, <ExportFileName>
VB Example	oModule.ExportReportDataToFile "Plot 1", "C:/Plot1data.rdat"

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	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><ReportName></td> <td>String</td> <td>Name of specified report.</td> </tr> <tr> <td><ExportFileName></td> <td>String</td> <td>Name of export file.</td> </tr> <tr> <td><TableType></td> <td>String</td> <td>Type of marker table to export. "Marker" or "DeltaMarker".</td> </tr> </tbody> </table>	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".
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

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/cftt.csv", False, " kV, MHz, yd ", True)</pre>

VB Syntax	ExportToFile <ReportName>, <FileName>
VB Example	<pre>oModule.ExportToFile "rETotal", "C:/Documents/rETotal.csv"</pre>

ExportToFile [Reporter]

From a data table or plot, generates text format, comma delimited, tab delimited, .dat, or .rdat type output files.

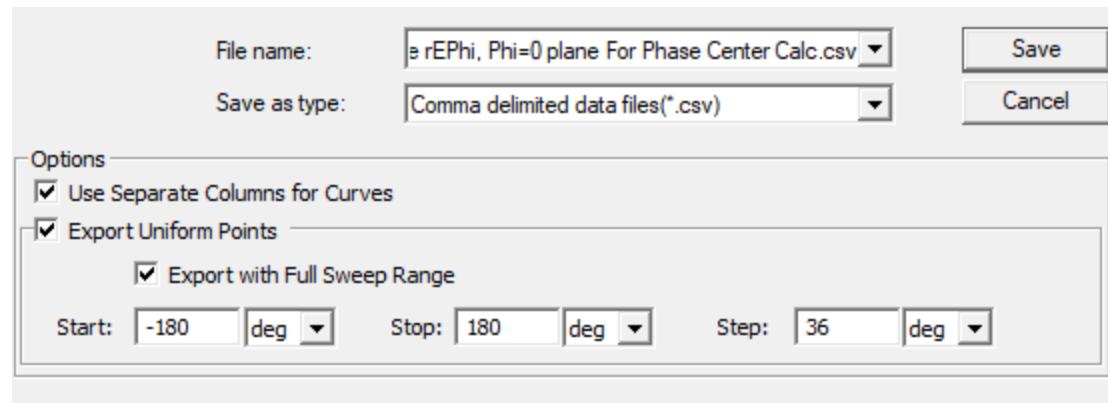
UI Access	Right-click on report name in the Project tree and select Export .																								
Parameters	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><ReportName></td> <td>String</td> <td>Name of the report</td> </tr> <tr> <td><FileName></td> <td>String</td> <td> Path and File Name Supported formats </td> </tr> <tr> <td></td> <td>.txt</td> <td>Post processor format file</td> </tr> <tr> <td></td> <td>.csv</td> <td>Comma-delimited data file</td> </tr> <tr> <td></td> <td>.tab</td> <td>Tab-separated file</td> </tr> <tr> <td></td> <td>.dat</td> <td>Ansys plot data file</td> </tr> <tr> <td></td> <td>.rdat</td> <td>Ansys report data file</td> </tr> </tbody> </table>	Name	Type	Description	<ReportName>	String	Name of the report	<FileName>	String	Path and File Name Supported formats		.txt	Post processor format file		.csv	Comma-delimited data file		.tab	Tab-separated file		.dat	Ansys plot data file		.rdat	Ansys report data file
Name	Type	Description																							
<ReportName>	String	Name of the report																							
<FileName>	String	Path and File Name Supported formats																							
	.txt	Post processor format file																							
	.csv	Comma-delimited data file																							
	.tab	Tab-separated file																							
	.dat	Ansys plot data file																							
	.rdat	Ansys report data file																							
Return Value	None																								

Python Syntax	ExportToFile (<ReportName>, <FileName>)
Python Example	<pre>oModule.ExportToFile("Plot1", "c:\report1.dat")</pre>

VB Syntax	ExportToFile <ReportName>, <FileName>
VB Example	<pre>oModule.ExportToFile "Plot1", "c:\report1.dat"</pre>

ExportUniformPointsToFile

Exports uniform points from a data table or plot report that includes the Export Uniform Points to File option enabled to text format, comma delimited, tab delimited, or .dat type output files.



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	ExportUniformPointsToFile(<ReportName>, <FileName>, <SweepRange>)
Python Example	<pre>oModule.ExportUniformPointsToFile("S Parameter Table 2", "D:/MyFiles/cftt.csv", "4GHz", "5GHz", "1GHz", False, " kV, MHz, yd ", False)</pre>

VB Syntax	ExportUniformPointsToFile <ReportName>, <FileName>
VB Example	<pre>oModule.ExportUniformPointsToFile "rETheta and rEPhi [dB]", "C:/MyFiles/rETheta and rEPhi [dB].csv", "-180deg", "180deg", "36deg"</pre>

GetAllCategories

Get all available category names (not including variable and output-variables) in a solution for a report type and display type, returned as an array of text strings.

UI Access	NA															
Parameters	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td><ReportType></td><td>String</td><td>Report type name.</td></tr><tr><td><DisplayType></td><td>String</td><td>Name of display type.</td></tr><tr><td><SolutionName></td><td>String</td><td>Name of solution.</td></tr><tr><td><SimValueCtx></td><td>Array</td><td>A context name, or array of strings that encode the contexts.</td></tr></tbody></table>	Name	Type	Description	<ReportType>	String	Report type name.	<DisplayType>	String	Name of display type.	<SolutionName>	String	Name of solution.	<SimValueCtx>	Array	A context name, or array of strings that encode the contexts.
Name	Type	Description														
<ReportType>	String	Report type name.														
<DisplayType>	String	Name of display type.														
<SolutionName>	String	Name of solution.														
<SimValueCtx>	Array	A context name, or array of strings that encode the contexts.														
Return Value	Array of text strings															

Python Syntax	GetAllCategories(<ReportType>, <DisplayType>, <SolutionName>, <SimValueCtxt>)
Python Example	<pre>oModule.GetAllCategories("Far Fields", "Rectangular Plot", "Setup1 : LastAdaptive", "Infinite Sphere1")</pre>

VB Syntax	GetAllCategories <ReportType>, <DisplayType>, <SolutionName>, <SimValueCtxt>
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	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><ReportType></td> <td>String</td> <td>Report type name.</td> </tr> <tr> <td><DisplayType></td> <td>String</td> <td>Display type name.</td> </tr> <tr> <td><SolutionName></td> <td>String</td> <td>Name of solution.</td> </tr> <tr> <td><SimValueCtxt></td> <td>Array</td> <td>A context name, or array of string that encoded the contexts(l).</td> </tr> <tr> <td><CategoryName></td> <td>String</td> <td>A category name as input parameter. a category name returned in GetAllCategories() or "Variables", or "Output Variables"</td> </tr> </tbody> </table>	Name	Type	Description	<ReportType>	String	Report type name.	<DisplayType>	String	Display type name.	<SolutionName>	String	Name of solution.	<SimValueCtxt>	Array	A context name, or array of string that encoded the contexts(l).	<CategoryName>	String	A category name as input parameter. a category name returned in GetAllCategories() or "Variables", or "Output Variables"
Name	Type	Description																	
<ReportType>	String	Report type name.																	
<DisplayType>	String	Display type name.																	
<SolutionName>	String	Name of solution.																	
<SimValueCtxt>	Array	A context name, or array of string that encoded the contexts(l).																	
<CategoryName>	String	A category name as input parameter. a category name returned in GetAllCategories() or "Variables", or "Output Variables"																	
Return Value	Array of text strings																		

Python Syntax	GetAllQuantities(<ReportType>,<DisplayType>, <SolutionName>, <SimValueCtxt>, <CategoryName>)
Python Example	<pre>oModule.GetAllQuantities("Far Fields", "Rectangular Plot", "Setup1 : LastAdaptive", "Infinite Spherel", "Gain")</pre>

VB Syntax	GetAllQuantities <ReportType>,<DisplayType>, <SolutionName>, <SimValueCtxt>, <CategoryName>
VB Example	<pre>oModule.GetAllQuantities _ "Far Fields", "Rectangular Plot", _ "Setup1 : LastAdaptive",) "Infinite Spherel", "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	<table border="1"> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> <tr> <td><code><ReportType></code></td> <td>String</td> <td>Report type name.</td> </tr> </table>	Name	Type	Description	<code><ReportType></code>	String	Report type name.
Name	Type	Description					
<code><ReportType></code>	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 <i><ReportType></i>	Type String	Description 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 <i><ObjectPath></i>	Type String	Description 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(<i><ObjectPath></i>)
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
	<code><ReportName></code>	String	Name of specified report.
Return Value	Array of string containing the PropServer name.		

Python Syntax	GetCurvePropServerName(<ReportName>, <TraceName>)
Python Example	<code>oModule.GetCurvePropServerName ("Plot 1", "Phase")</code>

VB Syntax	GetCurvePropServerName <ReportName>, <TraceName>
VB Example	<code>oModule.GetCurvePropServerName "Plot 1", "Phase"</code>

GetDisplayType

Gets the display type of a specified report.

UI Access	NA						
Parameters	<table border="1"><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><ReportName></td><td>String</td><td>Report name</td></tr></table>	Name	Type	Description	<ReportName>	String	Report name
Name	Type	Description					
<ReportName>	String	Report name					
Return Value	String containing display type.						

Python Syntax	GetDisplayType(<ReportName>)
Python Example	<code>oModule.GetDisplayType ("Design Plot 1")</code>

VB Syntax	GetDisplayType <ReportName>
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 <i><TraceName></i>	Type String	Description 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 <i><TraceName></i>	Type String	Description Name of specified source trace.
Return Value	Array of strings containing states.		

Python Syntax	GetDynLinkQtyValueState(<TraceName>, <QtyName>)
Python Example	<code>oModule.GetDynLinkQtyValueState("Plot 1:Trace1", "")</code>

VB Syntax	GetDynLinkQtyValueState <TraceName>, <QtyName>
VB Example	<code>oModule.GetDynLinkQtyValueState "Plot 1:Trace1", ""</code>

GetDynLinkTraces

Gets the names of the dynamic linked traces.

UI Access	N/A		
Parameters	Name	Type	Description
	<SoluName>	String	Name of the source solution. If empty, refer to current solution.
Return Value	Array of strings containing trace names.		

Python Syntax	GetDynLinkTraces(<SoluName>)
Python Example	<code>oModule.GetDynLinkTraces("")</code>

VB Syntax	GetDynLinkTraces <SoluName>
------------------	-----------------------------

VB Example	<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.
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 name of the active design.

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>

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	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><i><PropTab></i></td> <td>String</td> <td> One of the following, where tab titles are shown in parentheses: <ul style="list-style-type: none"> • PassedParameterTab ("Parameter Values") • DefinitionParameterTab (Parameter Defaults") • LocalVariableTab ("Variables" or "Local Variables") • ProjectVariableTab ("Project variables") • ConstantsTab ("Constants") • BaseElementTab ("Symbol" or "Footprint") • ComponentTab ("General") • Component("Component") • CustomTab ("Intrinsic Variables") • Quantities ("Quantities") • Signals ("Signals") </td> </tr> </tbody> </table>	Name	Type	Description	<i><PropTab></i>	String	One of the following, where tab titles are shown in parentheses: <ul style="list-style-type: none"> • 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") 		
Name	Type	Description							
<i><PropTab></i>	String	One of the following, where tab titles are shown in parentheses: <ul style="list-style-type: none"> • 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 CompInst@R;2;3
	<i><PropName></i>	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.GetPropertyValues("PassedParameterTab", k, "R") </pre>

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

GetPropNames [Reporter]

Report setup module does not have its own property, this function always returns empty array.

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

Python Syntax	GetPropNames()
Python Example	<code>oModule.GetPropNames ()</code>

VB Syntax	GetPropNames
VB Example	<code>oModule.GetPropNames</code>

GetPropValue [Report Setup]

Gets the property value for a Report, or reports' child object.

UI Access	N/A		
Parameters	Name	Type	Description
	<code><PropPath></code>	String	A child object's property path. See property path discussion here .
Return Value	String containing the property value.		

Python Syntax	GetPropValue(<PropPath>)
Python Example	<code>oModule.GetPropValue("S Parameter Plot 1/Display Type")</code>

VB Syntax	GetPropValue <PropPath>
VB Example	<code>oModule.GetPropValue "S Parameter Plot 1/Display Type"</code>

GetQtyExpressionsForSourceTrace

Gets the quantity expressions from a specified source trace.

UI Access	N/A						
Parameters	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td><SourceTraceName></td><td>String</td><td>Name of specified source trace.</td></tr></tbody></table>	Name	Type	Description	<SourceTraceName>	String	Name of specified source trace.
Name	Type	Description					
<SourceTraceName>	String	Name of specified source trace.					
Return Value	Array of strings containing quantity expressions.						

Python Syntax	GetQtyExpressionsForSourceTrace(<SourceTraceName>)
Python Example	<code>oModule.GetQtyExpressionsForSourceTrace("Plot 1:Trace1")</code>

VB Syntax	GetQtyExpressionsForSourceTrace <SourceTraceName>
VB Example	<code>oModule.GetQtyExpressionsForSourceTrace "Plot 1:Trace1"</code>

GetReportTraceNames

Gets the names of existing trace names in a plot.

UI Access	N/A		
Parameters	Name <i><PlotName></i>	Type String	Description 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 <i><ReportName></i>	Type String	Description 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	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><code><ReportType></code></td> <td>String</td> <td>Report type name.</td> </tr> <tr> <td><code><DisplayType></code></td> <td>String</td> <td>Display type name.</td> </tr> <tr> <td><code><SolutionName></code></td> <td>String</td> <td>Name of solution.</td> </tr> </tbody> </table>	Name	Type	Description	<code><ReportType></code>	String	Report type name.	<code><DisplayType></code>	String	Display type name.	<code><SolutionName></code>	String	Name of solution.
Name	Type	Description											
<code><ReportType></code>	String	Report type name.											
<code><DisplayType></code>	String	Display type name.											
<code><SolutionName></code>	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	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><reportTypeArg></td> <td>String</td> <td>Report type name as input parameter.</td> </tr> <tr> <td><solutionNameArg></td> <td>String</td> <td>Solution name as input parameter.</td> </tr> <tr> <td><simValueCtxtArg></td> <td>Structured Array</td> <td>Same as ContextArray values created in the relevant CreateReport script.</td> </tr> <tr> <td><familiesArg></td> <td>Array of Strings</td> <td>Same as FamiliesArray values created in the relevant CreateReport script.</td> </tr> <tr> <td><expressionArg></td> <td>String or Array of Strings</td> <td>Text string or array of text strings; valid expression, may validate it as the data-table Y-component.</td> </tr> </tbody> </table>	Name	Type	Description	<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.
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	<p>ARRAY of ISolutionDataResultComInterface objects, containing:</p> <ul style="list-style-type: none"> • GetSweepNames() • GetSweepUnits() • GetSweepValues() • IsPerQuantityPrimarySweep() • GetDataExpressions() – should be the same as <expressionArg> • GetPerQuantityPrimarySweepValues() • IsDataComplex() 																		

- | | |
|--|--|
| | <ul style="list-style-type: none"> • GetDataUnits() • GetRealDataValues() • GetImagDataValues() • ReleaseData() • GetDesignVariableNames() • GetDesignVariableUnits() • GetDesignVariableValue() • GetDesignVariationKey() |
|--|--|

Note:

- This command is *not* recordable from the UI, but its parameters are similar to [CreateReport](#), so you may record a CreateReport script to get the parameter values.
- For the returned ISolutionDataResultComInterface object, some of its functions have an optional boolean parameter: SIValue. SIValue defaults to True. When the pass in value is True, return data values will be in Standard International values; when False, return data values will be in the current units.

Example: Freq Sweep with [1GHz, 2GHz, 3GHz], GetSweepUnits("Freq") return "GHz"; GetSweepValues("Freq", True) return [1000000, 2000000, 3000000]; GetSweepValues("Freq", False) return [1, 2, 3].

Python Syntax	GetSolutionDataPerVariation(reportTypeArg, solutionNameArg, simValueCtxArg, familiesArg, expressionArg)
Python Example	<pre>oModule = oDesign.GetModule("ReportSetup") arr = oModule.GetSolutionDataPerVariation('Modal Solution Data', 'Setup1 :</pre>

	<code>Sweep', ['Domain:=' , 'Sweep'], ['Freq:=' , ['All']] , 'offset:=' , ['All']] , ['S(Port1,Port1)', 'dB(S(Port1,Port3))'])</code>
--	---

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 script does not function on its own, but as part of [GetSolutionDataPerVariation](#).

UI Access	N/A
Parameters	N/A
Return Value	Array of text strings

Python Syntax	GetDataExpressions()
Python Example	<pre>expressions = oModule.GetDataExpressions()</pre>

VB Syntax	GetDataExpressions()
VB Example	<pre>dim expressions expressions = oModule.GetDataExpressions()</pre>

GetDataUnits

Returns text string containing units.

Important:

This script does not function on its own, but as part of [GetSolutionDataPerVariation](#).

UI Access	N/A		
Parameters	Name	Type	Description
	<expressionString>	String	Can be returned from GetDataExpressions()
Return Value	Text string of units; empty if no units		

Python Syntax	GetDataUnits(<expressionString>)
Python Example	<code>oModule.GetDataUnits(expressions)</code>

VB Syntax	GetDataUnits(<expressionString>)
VB Example	<code>oModule.GetDataUnits(expressions)</code>

GetDesignVariableNames

Returns array of strings containing design variable names.

Important:

This script does not function on its own, but as part of [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 script does not function on its own, but as part of [GetSolutionDataPerVariation](#).

UI Access	N/A		
Parameters	Name <varName>	Type String	Description Can be returned from GetDesignVariableNames()
Return Value	Text string of units; empty if no units.		

Python Syntax	GetDesignVariableUnits(<varName>)
Python Example	<pre>units = oModule.GetDesignVariableUnits('Variable Name')</pre>

VB Syntax	GetDesignVariableUnits(<varName>)
------------------	-----------------------------------

VB Example	<pre>dim units units = oModule.GetDesignVariableUnits("Variable Name")</pre>
-------------------	--

GetDesignVariableValue

Returns a design variable's value.

Important:

This script does not function on its own, but as part of [GetSolutionDataPerVariation](#).

UI Access	N/A									
Parameters	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><varName></td> <td>String</td> <td>Can be returned from GetDesignVariableNames()</td> </tr> <tr> <td><siValue></td> <td>Boolean</td> <td>True to return SI-value; False to return by GetDesignVariableUnits()</td> </tr> </tbody> </table>	Name	Type	Description	<varName>	String	Can be returned from GetDesignVariableNames()	<siValue>	Boolean	True to return SI-value; False to return by GetDesignVariableUnits()
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	<code>GetDesignVariableValue(<varName>, <siValue>)</code>
Python Example	<code>value = oModule.GetDesignVariableValue('varName', 1)</code>

VB Syntax	<code>GetDesignVariableValue(<varName>, <siValue>)</code>
VB Example	<pre>dim value value = oModule.GetDesignVariableValue("varName", True)</pre>

GetDesignVariationKey

Returns a design's Variation Key.

Important:

This script does not function on its own, but as part of [GetSolutionDataPerVariation](#).

UI Access	N/A
Parameters	N/A
Return Value	String containing variation key.

Python Syntax	GetDesignVariationKey()
Python Example	<code>oModule.GetDesignVariationKey()</code>

VB Syntax	GetDesignVariationKey()
VB Example	<code>dim key set key = oModule.GetDesignVariationKey()</code>

GetImagDataValues

Returns array of imaginary data values.

Important:

This script does not function on its own, but as part of [GetSolutionDataPerVariation](#).

UI Access	N/A		
Parameters	Name	Type	Description
	<expressionString>	String	Can be returned from GetDataExpressions()
Return Value	Array of doubles		

Python Syntax	GetImagDataValues(<expressionString>,<siValue>)
Python Example	imaginaryvalues = oModule.GetImagDataValues('expression',1)

VB Syntax	GetImagDataValues(<expressionString>,<siValue>)
VB Example	dim imaginaryvalues imaginaryvalues = oModule.GetImagDataValues("expression",True)

GetPerQuantityPrimarySweepValues

Returns per quantity primary sweep values.

Important:

This script does not function on its own, but as part of [GetSolutionDataPerVariation](#).

UI Access	N/A		
Parameters	Name	Type	Description
	<expressionString>	String	Can be returned from GetDataExpressions() .
Return Value	Array of doubles if IsPerQuantityPrimarySweep() returned True; error if returned False		

Python Syntax	GetPerQuantitySweepValues(<expressionString>, <siValue>)
Python Example	sweepvalues = oModule.GetPerQuantitySweepValues('0.111,0.201,0.345,0.231', 1)

VB Syntax	GetPerQuantitySweepValues(<expressionString>, <siValue>)
VB Example	dim sweepvalues sweepvalues = oModule.GetPerQuantitySweepValues("0.111,0.201,0.345,0.231", True)

GetRealDataValues

Returns array of real data values.

Important:

This script does not function on its own, but as part of [GetSolutionDataPerVariation](#).

UI Access	N/A		
Parameters	Name	Type	Description
	<expressionString>	String	Can be returned from GetDataExpressions()
Return Value	Array of doubles		

Python Syntax	GetRealDataValues(<expressionString>,<siValue>)
Python Example	realvalues = oModule.GetRealDataValues('expression',1)

VB Syntax	GetRealDataValues(<expressionString>,<siValue>)
VB Example	dim realvalues realvalues = oModule.GetRealDataValues("expression",True)

GetSweepNames

Returns array of text strings containing primary sweep name(s).

Important:

This script does not function on its own, but as part of [GetSolutionDataPerVariation](#).

UI Access	N/A
Parameters	N/A
Return Value	Array of text strings

Python Syntax	GetSweepNames()
Python Example	<pre>sweepnames = oModule.GetSweepNames ()</pre>

VB Syntax	GetSweepNames()
VB Example	<pre>dim sweepnames sweepnames = oModule.GetSweepNames ()</pre>

GetSweepUnits

Returns text string containing units.

Important:

This script does not function on its own, but as part of [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	sweepunits = oModule.GetSweepUnits('Sweep 1')

VB Syntax	GetSweepUnits(<sweepName>)
VB Example	<pre>dim sweepunits sweepunits = oModule.GetSweepUnits("Sweep 1")</pre>

GetSweepValues

Returns sweep values.

Important:

This script does not function on its own, but as part of [GetSolutionDataPerVariation](#).

UI Access	N/A		
Parameters	Name	Type	Description
	<sweepName>	String	Primary sweep name
Return Value	Array of doubles		

Python Syntax	GetSweepValues(<sweepName>, <siValue>)
Python Example	sweepvalues = oModule.GetSweepValues('Sweep 1', True)

VB Syntax	GetSweepValues(<sweepName>, <siValue>)
VB Example	dim sweepvalues sweepvalues = oModule.GetSweepValues("Sweep 1", True)

IsDataComplex

Returns whether an expression is complex.

Important:

This script does not function on its own, but as part of [GetSolutionDataPerVariation](#).

UI Access	NA								
Parameters	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><expressionString></td> <td>String</td> <td>Can be returned from GetDataExpressions().</td> </tr> </tbody> </table>			Name	Type	Description	<expressionString>	String	Can be returned from GetDataExpressions() .
Name	Type	Description							
<expressionString>	String	Can be returned from GetDataExpressions() .							
Return Value	Boolean (True if expression is Complex data; False if not)								

Python Syntax	<code>IsDataComplex(<expressionString>)</code>
Python Example	<code>oModule.IsDataComplex('.001,.234,.455,.434')</code>

VB Syntax	<code>IsDataComplex(<expressionString>)</code>
VB Example	<code>oModule.IsDataComplex('.001,.234,.455,.434')</code>

IsPerQuantityPrimarySweep

Returns whether data expressions have different primary sweep values.

Important:

This script does not function on its own, but as part of [GetSolutionDataPerVariation](#).

UI Access	N/A
Parameters	N/A
Return Value	Boolean (True if data expressions have different primary sweep values)

Python Syntax	<code>IsPerQuantityPrimarySweep()</code>
Python Example	<code>var = oModule.IsPerQuantityPrimarySweep()</code>

VB Syntax	<code>IsPerQuantityPrimarySweep()</code>
VB Example	<code>dim var var = oModule.IsPerQuantityPrimarySweep()</code>

Release Data

Releases all cached data. After this function is called, all subsequent function calls to the object will fail.

Important:

This script does not function on its own, but as part of [GetSolutionDataPerVariation](#).

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	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><code><ReportName></code></td> <td>String</td> <td>Name of report.</td> </tr> <tr> <td><code><GroupStrategy></code></td> <td>String</td> <td>Strategy for grouping, "Single", "By Trace" or "By Units".</td> </tr> </tbody> </table>			Name	Type	Description	<code><ReportName></code>	String	Name of report.	<code><GroupStrategy></code>	String	Strategy for grouping, "Single", "By Trace" or "By Units".
Name	Type	Description										
<code><ReportName></code>	String	Name of report.										
<code><GroupStrategy></code>	String	Strategy for grouping, "Single", "By Trace" or "By Units".										
Return Value	None.											

Python Syntax	GroupPlotCurvesByGroupingStrategy(<ReportName>, <GroupStrategy>)
Python Example	<code>oModule.GroupPlotCurvesByGroupingStrategy ("Transient Plot 1", "By Trace")</code>

VB Syntax	GroupPlotCurvesByGroupingStrategy <ReportName>, <GroupStrategy>
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	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td><ReportName></td><td>String</td><td>Name of the Report</td></tr><tr><td><FileName></td><td>String</td><td>Path and File Name</td></tr><tr><td></td><td>.csv</td><td>Comma-delimited data file</td></tr><tr><td></td><td>.tab</td><td>Tab-separated file</td></tr><tr><td></td><td>.dat</td><td>Ansys plot data file</td></tr></tbody></table>	Name	Type	Description	<ReportName>	String	Name of the Report	<FileName>	String	Path and File Name		.csv	Comma-delimited data file		.tab	Tab-separated file		.dat	Ansys plot data file
Name	Type	Description																	
<ReportName>	String	Name of the Report																	
<FileName>	String	Path and File Name																	
	.csv	Comma-delimited data file																	
	.tab	Tab-separated file																	
	.dat	Ansys plot data file																	
Return Value	None																		

Python Syntax	<code>ImportIntoReport (<ReportName>, <FileName>)</code>
Python Example	<code>oDesign.ImportIntoReport ("Plot1", "c:\report1.dat")</code>

VB Syntax	<code>ImportIntoReport <ReportName>, <FileName></code>
VB Example	<code>oDesign.ImportIntoReport "Plot1", "c:\report1.dat"</code>

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.
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.

	<code><StackName></code>	String	Name of stack to move to.
Return Value	None.		

Python Syntax	<code>MovePlotCurvesToGroup(<ReportName>, <CurveArray>, <StackName>)</code>
Python Example	<code>oModule.MovePlotCurvesToGroup("XY Stacked Plot 1", ["R2.V : TR", "R2.I : TR"], "Stack 2")</code>

VB Syntax	<code>MovePlotCurvesToGroup <ReportName>, <CurveArray>, <StackName></code>
VB Example	<code>oModule.MovePlotCurvesToGroup _ "XY Stacked Plot 1", _ Array("R2.V : TR", "R2.I : TR"), _ "Stack 2"</code>

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	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><code><ReportName></code></td> <td>String</td> <td>Name of report.</td> </tr> <tr> <td><code><CurveArray></code></td> <td>Array</td> <td>Array of curve names to move.</td> </tr> </tbody> </table>			Name	Type	Description	<code><ReportName></code>	String	Name of report.	<code><CurveArray></code>	Array	Array of curve names to move.
Name	Type	Description										
<code><ReportName></code>	String	Name of report.										
<code><CurveArray></code>	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

```
oModule.OpenWindowForAllReports()
```

VB Syntax

```
OpenWindowForAllReports
```

VB Example

```
oModule.OpenWindowForAllReports
```

OpenWindowForReports

Opens windows for specified reports.

UI Access	N/A		
Parameters	Name <i><ReportNames></i>	Type Array	Description Array of strings containing report names.
Return Value	None.		

Python Syntax

```
OpenWindowForReports(<ReportNames>)
```

Python Example

```
oModule.OpenWindowForReports(["Report1", "Report2"])
```

VB Syntax

```
OpenWindowForReports <ReportNames>
```

VB Example

```
oModule.OpenWindowForReports Array("Report1", "Report2")
```

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.
Return Value	None.		

Python Syntax	PastePlotSettings(<ReportName>, <PropTypeToApply>)
Python Example	oModule.PastePlotSettings("Plot 1", "Graphical")

VB Syntax	PastePlotSettings <ReportName>, <PropTypeToApply>
VB Example	oModule.PastePlotSettings "Plot 1", "Graphical"

PasteReports

Paste copied reports to results in the current project.

UI Access	Edit > Paste
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 <i><ReportName></i>	Type String	Description Name of plot
Return Value	None		

Python Syntax	PasteTraces (<i><ReportName></i>)
Python Example	<code>oModule.PasteTraces ("XY Plot1")</code>

VB Syntax	PasteTraces <i><ReportName></i>
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 <i><ReportName></i>	Type String	Description Name of plot
Return Value	None		

Python Syntax	PasteTracesWithLegacyNames (<ReportName>)
Python Example	<code>oModule.PasteTracesWithLegacyNames ("XY Plot1")</code>

VB Syntax	PasteTracesWithLegacyNames <ReportName>
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	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td><OldReportName></td><td>String</td><td>Old Report Name</td></tr><tr><td><NewReportName></td><td>String</td><td>New Report Name</td></tr></tbody></table>	Name	Type	Description	<OldReportName>	String	Old Report Name	<NewReportName>	String	New Report Name
Name	Type	Description								
<OldReportName>	String	Old Report Name								
<NewReportName>	String	New Report Name								
Return Value	None.									

Python Syntax	RenameReport (<OldReportName>, <NewReportName>)
Python Example	<code>oModule.RenameReport ("XY Plot1", "Reflection")</code>

VB Syntax	RenameReport <OldReportName>, <NewReportName>
VB Example	oModule.RenameReport "XY Plot1", "Reflection"

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
Return Value	None.		

Python Syntax	RenameTrace(<ReportName>, <TraceName>, <NewName>)
Python Example	oModule.RenameTrace ("XY Plot1", "dB(S(WavePort1, WavePort1))1", "Port1dbS")

VB Syntax	RenameTrace <ReportName>, <TraceName>, <NewName>
VB Example	oModule.RenameTrace "XY Plot1", "dB(S(WavePort1, WavePort1))1", "Port1dbS"

ResetPlotSettings

Resets plot settings to defaults.

UI Access	Right-click on a plot, select Edit > Reset Plot Settings		
Parameters	Name <i><PlotName></i>	Type String	Description Name of specified plot.
Return Value	None.		

Python Syntax	ResetPlotSettings(<i><PlotName></i>)
Python Example	oModule.ResetPlotSettings("Differential S-parameters")

VB Syntax	ResetPlotSettings <i><PlotName></i>
VB Example	oModule.ResetPlotSettings("Differential S-parameters")

SavePlotSettingsAsDefault

Saves report plot settings as default.

UI Access	Report Templates > Save Settings as Default		
Parameters	Name <i><PlotName></i>	Type String	Description Name of plot to use for plot defaults.

Return Value	None.
---------------------	-------

Python Syntax	SavePlotSettingsAsDefault (" <i><PlotName></i> ")
Python Example	oModule.SavePlotSettingsAsDefault ("XY Plot1")

VB Syntax	SavePlotSettingsAsDefault " <i><PlotName></i> "
VB Example	oModule.SavePlotSettingsAsDefault "XY Plot 1"

SetLinkOutputTraces

Specifies dynamic link output traces from the current design.

UI Access	Right-click the Results , select Link Output...								
Parameters	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><i><TraceArray></i></td> <td>Array</td> <td>Array of traces to set. Array ("<i><ReportName>:=</i>", <array of trace names>, "<i><ReportName>:=</i>", <array of trace names>, ...)</td> </tr> </tbody> </table>			Name	Type	Description	<i><TraceArray></i>	Array	Array of traces to set. Array (" <i><ReportName>:=</i> ", <array of trace names>, " <i><ReportName>:=</i> ", <array of trace names>, ...)
Name	Type	Description							
<i><TraceArray></i>	Array	Array of traces to set. Array (" <i><ReportName>:=</i> ", <array of trace names>, " <i><ReportName>:=</i> ", <array of trace names>, ...)							
Return Value	None.								

Python Syntax	SetLinkOutputTraces(<i><TraceArray></i>)
Python Example	oModule.SetLinkOutputTraces (

```
[  
    "Plot 1:=", ["Trace1"],  
    "Plot 2:=", ["Trace1"]  
])
```

VB Syntax	SetLinkOutputTraces <TraceArray>
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 .
Return Value	True if the property is found and the new value is valid. Otherwise return False.		

Python	SetPropValue(<PropPath>, <NewValue>)
---------------	--------------------------------------

Syntax	
Python Example	<pre>oRptModule.SetPropValue("S Parameter Plot 1/Display Type", "DataTable") oRptModule.SetPropValue("S Parameter Plot 1/db(S(port1,port2)/Primary sweep", "Freq")</pre>

VB Syntax	SetPropValue <PropPath>, <NewValue>
VB Example	<pre>oRptModule.SetPropValue "S Parameter Plot 1/Display Type", "DataTable" oRptModule.SetPropValue "S Parameter Plot 1/db(S(port1,port2)/Primary sweep", "Freq"</pre>

UnGroupPlotCurvesInGroup

From a Stacked Plot, ungroups curves in a stack.

UI Access	N/A									
Parameters	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><ReportName></td> <td>String</td> <td>Name of report.</td> </tr> <tr> <td><GroupName></td> <td>String</td> <td>Stack group name.</td> </tr> </tbody> </table>	Name	Type	Description	<ReportName>	String	Name of report.	<GroupName>	String	Stack group name.
Name	Type	Description								
<ReportName>	String	Name of report.								
<GroupName>	String	Stack group name.								
Return Value	None.									

Python Syntax	UnGroupPlotCurvesInGroup(<ReportName>, <GroupName>)
Python Example	<pre>oModule.UngroupByPlotCurvesInGroup("S Parameter Plot 3", "Stack 1")</pre>

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	oModule.UpdateAllReports

UpdateReports

Updates specified reports.

UI Access	N/A
------------------	-----

Parameters	<table border="1"> <tr> <th>Name</th><th>Type</th><th>Description</th></tr> <tr> <td><ReportNames></td><td>Array</td><td>Array of strings containing report names.</td></tr> </table>	Name	Type	Description	<ReportNames>	Array	Array of strings containing report names.
Name	Type	Description					
<ReportNames>	Array	Array of strings containing report names.					
Return Value	None.						

Python Syntax	UpdateReports(<ReportNames>)
Python Example	<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	<table border="1"> <tr> <td>Name</td> <td>Type</td> <td>Description</td> </tr> <tr> <td><ReportName></td> <td>String</td> <td>Name of Report.</td> </tr> <tr> <td><TraceNames></td> <td>Array</td> <td>Array of strings containing trace names.</td> </tr> <tr> <td><SolutionName></td> <td>String</td> <td>Name of the solution.</td> </tr> <tr> <td><ContextArray></td> <td>Array</td> <td>Context for which the expression is being evaluated. This can be an empty string if there is no context. Array("Domain:=", <DomainType>) <DomainType> ex. "Sweep" or "Time"</td> </tr> </table>	Name	Type	Description	<ReportName>	String	Name of Report.	<TraceNames>	Array	Array of strings containing trace names.	<SolutionName>	String	Name of the solution.	<ContextArray>	Array	Context for which the expression is being evaluated. This can be an empty string if there is no context. Array("Domain:=", <DomainType>) <DomainType> ex. "Sweep" or "Time"		
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. Array("Domain:=", <DomainType>) <DomainType> ex. "Sweep" or "Time"																

		<pre>Array("Context:=", <GeometryType> <GeometryType> ex. "Infinite Spheren", "Spheren", "Polylinen"</pre>
<FamiliesArray>	Array	<p>Contains sweep definitions for the report.</p> <pre>Array("<VariableName>:= ", <ValueArray> <ValueArray> Array("All") or Array("Value1", "Value2", ..."ValueN") examples of <VariableName> "Freq", "Theta", "Distance"</pre>
<ReportdataArray>	Array	<p>This array contains the report quantity and X, Y, and (Z) axis definitions.</p> <pre>Array("X Component:=", <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",</pre>

```

"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)"]
], [])
)

```

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")), _ </pre>

```
Array("X Component:=", "Time",_
      "Y Component:=", Array("dB(S(WavePort1,WavePort1))")), _
      Array())
```

UpdateTracesContextAndSweeps

Edits sweeps and context of multiple traces without affecting their component expressions.

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"
	<PointSet>	Array	Point set for the selected traces, for example, X and Y values for the plot.
Return Value	None		

Python Syntax	UpdateTracesContextAndSweeps(<ReportName>, <TraceNames>, <SolutionName>, <ContextArray>, <PointSet>)
Python Example	<pre>oModule.UpdateTracesContextAndSweeps_ ("Active S Parameter Quick Report",</pre>

```

["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"]])

```

VB Syntax	UpdateTracesContextAndSweeps <ReportName>, <TraceNames>, <SolutionName>, <ContextArray>, <PointSet>
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", _</pre>

```
"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"))
```

13 - Boundary and Excitation Module Script Commands

Boundary condition 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 Conditions Module

[AddAssignmentToBoundary](#)

[DeleteAllBoundaries](#)

[DeleteBoundaries](#)

[GetBoundaryAssignment](#)

[GetBoundaries](#)

[GetBoundariesOfType](#)
[GetDefaultBaseName](#)
[GetNumBoundaries](#)
[GetNumBoundariesOfType](#)
[ReassignBoundary](#)
[RemoveAssignmentFromBoundary](#)
[RenameBoundary](#)
[ReprioritizeBoundaries](#)
[SetDefaultBaseName](#)

AddAssignmentToBoundary

Adds a new geometry assignment to a boundary.

UI Access	N/A		
Parameters	Name <i><Assignment></i>	Type Array	Description Structured array. Array ("Name:<BoundName>", "Objects:=", <AssignmentObjects>, "Faces:=", <AssignmentFaces>)
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"), _</pre>

	"Faces:=", Array(12, 11))
--	---------------------------

DeleteAllBoundaries

Deletes all boundaries.

UI Access	[product] > Boundaries > Delete All
Parameters	None.
Return Value	None.

Python Syntax	DeleteAllBoundaries()
Python Example	<code>oModule.DeleteAllBoundaries ()</code>

VB Syntax	DeleteAllBoundaries
VB Example	<code>oModule.DeleteAllBoundaries</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	<table border="1"> <tr> <td>Name <code><NameArray></code></td> <td>Type Array</td> <td>Description Array of boundary condition names.</td> </tr> </table>	Name <code><NameArray></code>	Type Array	Description Array of boundary condition names.
Name <code><NameArray></code>	Type Array	Description 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>
-------------------	--

GetBoundaries

Gets boundary names in the current design.

UI Access	N/A
Parameters	None.
Return Value	Array of boundary names.

Python Syntax	<code>GetBoundaries()</code>
Python Example	<code>oModule.GetBoundaries()</code>

VB Syntax	<code>GetBoundaries</code>
VB Example	<code>oModule.GetBoundaries</code>

GetBoundariesOfType

Gets boundary names of the given type.

UI Access	N/A						
Parameters	<table border="1"> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> <tr> <td><code><BoundaryType></code></td> <td>String</td> <td>Name of boundary type.</td> </tr> </table>	Name	Type	Description	<code><BoundaryType></code>	String	Name of boundary type.
Name	Type	Description					
<code><BoundaryType></code>	String	Name of boundary type.					
Return Value	Array of boundary names of the given type.						

Python Syntax	<code>GetBoundariesOfType(<BoundaryType>)</code>
Python Example	<code>oModule.GetBoundariesOfType("PerfectE")</code>

VB Syntax	<code>GetBoundariesOfType <BoundaryType></code>
------------------	---

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 <i><BoundaryName></i>	Type String	Description Name of the specified boundary or excitation.
Return Value	Array of face IDs or object IDs.		

Python Syntax	<code>GetBoundaryAssignment(<BoundaryName>)</code>
Python Example	<code>oModule.GetBoundaryAssignment ("Rad1")</code>

VB Syntax	<code>GetBoundaryAssignment <BoundaryName></code>
VB Example	<code>oModule.GetBoundaryAssignment "Rad1"</code>

GetDefaultBaseName

Gets the default base name for boundaries for a project.

UI Access	N/A		
Parameters	Name <i><BoundaryType></i>	Type String	Description Name of legal boundary type.
Return Value	String of boundary default base name.		

Python Syntax	<code>GetDefaultBaseName(<BoundaryType>)</code>
Python Example	<code>oModule.GetDefaultBaseName ("Radiation")</code>

VB Syntax	GetDefaultBaseName <BoundaryType>
VB Example	<code>oModule.GetDefaultBaseName "Radiation"</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	<table border="1"><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><BoundaryType></td><td>String</td><td>Specified boundary type.</td></tr></table>	Name	Type	Description	<BoundaryType>	String	Specified boundary type.
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 <BoundaryType>
------------------	---------------------------------------

VB Example	<code>oModule.GetNumBoundariesOfType "PerfectE"</code>
-------------------	--

ReassignBoundary

Specifies a new geometry assignment for a boundary.

UI Access	Right-click Boundaries > Reassign or Excitations > Reassign		
Parameters	Name <code><AssignmentArray></code>	Type Array	Description Structured array. <code>Array ("Name:<BoundName>", "Objects:=", <AssignmentObjects>, "Faces:=", <AssignmentFaces>)</code>
Return Value	None.		

Python Syntax	<code>ReassignBoundary(<AssignmentArray>)</code>
Python Example	<code>oModule.ReassignBoundary(["NAME:PerfH12", "Faces:=", [17]])</code>

VB Syntax	<code>ReassignBoundary <AssignmentArray></code>
VB Example	<code>oModule.ReassignBoundary _ Array ("NAME:PerfH12", _ "Faces:=", Array(17))</code>

RemoveAssignmentFromBoundary

Removes a geometry assignment from a boundary.

UI Access	Right-click on a boundary or an excitation, select Remove from Assignment .
------------------	--

Parameters	Name <i><AssignmentArray></i>	Type Array	Description Structured array. Array ("Name :<BoundName>", "Objects :=", <AssignmentObjects>, "Faces :=", <AssignmentFaces>)
Return Value	None.		

Python Syntax	RemoveAssignmentFromBoundary(<AssignmentArray>)
Python Example	<pre>oModule.RemoveAssignmentFromBoundary(["NAME:Rad1", "Faces :=", [11]])</pre>

VB Syntax	RemoveAssignmentFromBoundary <AssignmentArray>
VB Example	<pre>oModule.RemoveAssignmentFromBoundary _ Array("NAME:Rad1", _ "Faces :=", Array(11))</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 <i><OldName></i>	Type String	Description Name of the boundary to be renamed.
	<i><NewName></i>	String	New name for the boundary.
Return Value	None.		

Python Syntax	RenameBoundary(<OldName>, <NewName>)
Python Example	<code>oModule.RenameBoundary ("Rad2", "Rad3")</code>

VB Syntax	RenameBoundary <OldName>, <NewName>
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	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><NewOrderArray></td> <td>Array</td> <td>Structured array. Array ("NAME:NewOrder", <BoundName>, <BoundName>, . . .)</td> </tr> </tbody> </table>	Name	Type	Description	<NewOrderArray>	Array	Structured array. Array ("NAME:NewOrder", <BoundName>, <BoundName>, . . .)
Name	Type	Description					
<NewOrderArray>	Array	Structured array. Array ("NAME:NewOrder", <BoundName>, <BoundName>, . . .)					
Return Value	None.						

Python Syntax	ReprioritizeBoundaries(<NewOrderArray>)
Python Example	<code>oModule.ReprioritizeBoundaries (["NAME:NewOrder", "Imped1", "PerfE1", "PerfH1"])</code>

VB Syntax	ReprioritizeBoundaries <NewOrderArray>
VB	<code>oModule.ReprioritizeBoundaries Array ("NAME:NewOrder", _</code>

Example	"Imped1", "PerfE1", "PerfH1")
----------------	-------------------------------

SetDefaultBaseName

Sets the default base name for boundaries for a project.

UI Access	N/A		
Parameters	Name	Type	Description
	<BoundaryType>	String	Name of legal boundary type.
Return Value	None.		

Python Syntax	SetDefaultBaseName(<BoundaryType>, <DefaultName>)
Python Example	oModule.SetDefaultBaseName ("Radiation", "RadBnd")

VB Syntax	SetDefaultBaseName <BoundaryType>, <DefaultName>
VB Example	oModule.SetDefaultBaseName "Radiation", "RadBnd"

Icepak Boundary Condition Commands

Icepak thermal boundary condition commands should be executed by the `oModule` object. For example, use `oModule.AssignBlockBoundary` to assign a block thermal boundary condition.

See the following sections to view the parameters for each boundary condition.

- [AssignBlockBoundary](#)
- [AssignGrilleBoundary](#)
- [AssignNetworkBoundary](#)
- [AssignOpeningBoundary](#)
- [AssignRecircBoundary](#)
- [AssignAdiabaticPlateBoundary](#)
- [AssignConductingPlateBoundary](#)
- [AssignResistanceBoundary](#)
- [AssignSourceBoundary](#)
- [AssignStationaryWallBoundary](#)
- [AssignSymmetryWallBoundary](#)
- [AssignEMLoss](#)
- [AssignBlowerBoundary](#)

AssignBlockBoundary

Creates a block thermal boundary condition in an Icepak design.

UI Access	Icepak > Thermal > Assign > Block		
Parameters	Name <i><NAME></i>	Type string	Description Boundary condition name
	<i><Objects></i>	list	Objects included in the boundary condition assignment
	<i><Block Type></i>	string	Solid, Hollow, or Fluid
	<i><Use External Conditions></i>	bool	"True" to enable external thermal conditions or "False" to disable them
	<i><Heat Transfer Coefficient></i>	int	Heat transfer coefficient value and unit
	<i><Temperature></i>	int	Block temperature value and unit

	<i><Use Total Power></i>	bool	True or False
	<i><Total Power></i>	int	Block power value and unit
	<i><Power Density></i>	string	Block power density value and unit
	<i><NAME></i>	string	Total Power Variation Data
	<i><Variation Type></i>	string	Temp Dep, Transient, or Joule Heating
	<i><Variation Function></i>	string	<i>Temp Dep:</i> Piecewise Linear; <i>Transient:</i> Linear, Power Law, Exponential Sinusoidal, or Piecewise Linear; <i>Joule Heating:</i> None
	<i><Variation Value></i>	string	Temperature dependant or Transient variation function values.
	<i><Thermal Condition></i>	string	Hollow blocks only: Fixed Heat, Fixed Temperature, or Internal Conditions
	<i><Use Total Power></i>	bool	Hollow blocks only: "True" to enable total power or "False" to disable it
	<i><Use Heat Flux></i>	bool	Hollow blocks only: "True" to enable heat flux or "False" to disable it
	<i><Use Laminar Flow></i>	bool	Fluid blocks only: "True" to enabled laminar flow or 'False' to disable it
	<i><Use MRF></i>	bool	Fluid cylinders only: "True" to enabled moving reference frame or "False" to disable it
	<i><MRF></i>	int	Fluid cylinders only: Moving reference frame value and unit
	<i><X Fixed Velocity></i>	string	X direction fixed velocity value and unit
	<i><Y Fixed Velocity></i>	string	Y direction fixed velocity value and unit
	<i><Z Fixed Velocity></i>	string	Z direction fixed velocity value and unit
Return Value	None		

Python Syn-	AssignBlockBoundary (<NAME>, <Objects>, <Block Type>, <Use External
--------------------	---

tax	<i>Conditions>, <Heat Transfer Coefficient>, <Temperature>, <Total Power></i>
Python Example	<pre> oModule.AssignBlockBoundary(["NAME:Block1", "Objects:=" , ["Box1"], "Block Type:=" , "Solid", "Use External Conditions:=", True, "Heat Transfer Coefficient:=", "1w_per_m2kel", "Temperature:=" , "AmbientTemp", "Total Power:=" , "0W" ["NAME:Total Power Variation Data", "Variation Type:=" , "Temp Dep", "Variation Function:=" , "Piecewise Linear", "Variation Value:=" , "[\"1W\", \"1\"]"])]) </pre>

VB Syntax	AssignBlockBoundary (<NAME>, <Objects>, <Block Type>, <Use External Conditions>, <Heat Transfer Coefficient>, <Temperature>, <Total Power>)
VB Example	<pre> oModule.AssignBlockBoundary Array("NAME:Block6", "Objects:=", Array("Box3"), "Block Type:=" , "Solid", "Use External Conditions:=", true, "Heat Transfer Coefficient:=" , "1w_per_m2kel", "Tem- perature:=" , "AmbientTemp", "Total Power:=" , "5W") </pre>

AssignGrilleBoundary

Creates a grille thermal boundary condition in an Icepak design.

UI Access	Icepak > Thermal > Assign > Grille		
Parameters	Name <NAME> <Faces>	Type string list	Description Boundary con- dition name Faces included in the boundary con- dition assignment

	<i><Pressure Loss Type></i>	string	Coeff or Curve
	<i><Free Area Ratio></i>	string	Coeff only: greater than 0 and less than or equal to 1
	<i><Resistance Type></i>	string	Coeff only: Per- forated Thin Vent, Circular Metal Wire Screen, or Two-Plane Screen Cyl. Bars
	<i><External Rad. Temperature></i>	int	External radiation temperature value and unit
	<i><External Total Pressure></i>	int	External total pres- sure value and unit
	<i><NAME></i>	string	Curve only: dataset name
	<i><X></i>	list	Curve only: loss curve dataset x values
	<i><Y></i>	list	Curve only: loss curve dataset y values
Return Value	None		

Python Syntax	AssignGrilleBoundary (<NAME>, <Faces>, <Pressure Loss Type>, <Free Area Ratio>, <Resistance Type>, <External Rad. Temperature>, <External Total Pressure>, <NAME>, <X>, <Y>)
Python Example	<pre>oModule.AssignGrilleBoundary (["NAME:Grille1", "Faces:=" , [1], "Pressure Loss Type:=" , "Coeff", "Free Area Ratio:=" , "0.8", "Resistance Type:=" , "Perforated Thin Vent", "External Rad. Temperature:=", "AmbientTemp", "External Total Pressure:=", "AmbientPressure", ["NAME:DimUnits", """", """]])</pre>

	<pre>], "X:=" , ["0", "1", "2"] , "Y:=" , ["0", "1", "2"]])</pre>
--	---

VB Syntax	AssignGrilleBoundary (<NAME>, <Faces>, <Pressure Loss Type>, <Free Area Ratio>, <Resistance Type>, <External Rad. Temperature>, <External Total Pressure>, <NAME>, <X>, <Y>)
VB Example	<pre>oModule.AssignGrilleBoundary Array("NAME:Grille3", "Faces:=", Array(35), "Pressure Loss Type:=", _"Curve", "External Rad. Temperature:=", AmbientTemp", "External Total Pressure:=", _"Ambi- entPressure", Array ("NAME:DimUnits", "", ""), "X:=", Array("0", "1", "2"), "Y:=", Array(_"0", "1", "2"))</pre>

AssignNetworkBoundary

Creates a network thermal boundary condition in an Icepak design. See the Icepak Online Help for example scripts.

UI Access	Icepak > Thermal > Assign > Network		
Parameters	Name	Type	Description
	<NAME>	string	Boundary condition/component name
	<Faces>	list	Faces included in the boundary condition assignment
	<Face>	string	Face node (Face number and resistance choice)
	<Internal>	string	Internal node (power and unit, mass and unit, specific heat and unit)
	<Link1>	string	Link (linked nodes, Rlink type, and thermal resistance or heat transfer coefficient)
	<SchComplInst>	list	Schematic component instance (schematic coordinates)
	<NetName>	string	Wiredata net name
	<ID>	int	Node ID
	<WireSeg>	list	Coordinates of wire segment
	<Color>	int	Numerical color value

Return Value	None
---------------------	------

AssignOpeningBoundary

Creates a free opening thermal boundary condition in an Icepak design.

UI Access	Icepak > Thermal > Assign > Opening > Free		
Parameters	Name	Type	Description
	<NAME>	string	Boundary condition name
	<Faces>	list	Faces included in the boundary condition assignment
	<Temperature>	string	Opening temperature value and unit
	<External Rad. Temperature>	string	External radiation temperature value and unit
	<Inlet Type>	string	Pressure, Velocity, or Mass Flow
	<Total Pressure>	string	Pressure only: Total pressure value and unit
	<Static Pressure>	string	Velocity and Mass Flow only: Static pressure value and unit
	<Mass Flow Rate>	string	Mass Flow only: Mass flow rate and unit
	<Mass Flow Direction>	string	Normal to Boundary or Specified
	<Mass Flow Condition>	string	In Flow or Out Flow
	<X Velocity>	string	X direction velocity and unit
	<Y Velocity>	string	Y direction velocity and unit
	<Z Velocity>	string	Z direction velocity and unit
	<No Reverse Flow>	string	True or False
Return Value	None		

Python Syntax	AssignOpeningBoundary (<NAME>, <Faces>, <Temperature>, <External Rad. Temperature>, <Inlet Type>, <Total Pressure>)
Python Example	<pre> oModule.AssignOpeningBoundary (["NAME:Opening1", "Faces:=" , [12], "Temperature:=" , "AmbientTemp", "External Rad. Temperature:=" , "AmbientRadTemp", "Inlet Type:=" , "Pressure", "Total Pressure:=" , "AmbientPressure"]) </pre>

])
--	----

VB Syntax	AssignOpeningBoundary (<NAME>, <Faces>, <Temperature>, <External Rad. Temperature>, <Inlet Type>, <Total Pressure>)
VB Example	<pre> oModule.AssignBlockBoundary Array("NAME:Block1", "Objects:=", Array("Box1"), "Block Type:=", _ "Solid", "Use External Conditions:=", true, "Heat Transfer Coefficient:=", "1w_per_m2kel", "Temperature:=", Ambi- entTemp", "Total Power:=", "5W") </pre>

AssignRecircBoundary

Creates a recirculating opening thermal boundary condition in an Icepak design.

UI Access	Icepak > Thermal > Assign > Opening > Recirculating		
Parameters	Name	Type	Description
	<NAME>	string	Boundary condition name
	<Faces>	list	Faces included in the boundary condition assignment
	<Thermal Condition>	string	Temperature, Heat Input, or Conductance
	<Temperature Change (Temperature only)>	string	Temperature value and unit
	<Heat Flow (Heat Input only)>	string	Heat flow value and unit
	<Conductance (Conductance only)>	string	Conductance value and unit
	<Inlet Type>	string	Mass Flow, Mass Flux, or Volume Flow
	<Mass Flow Rate (Mass Flow only)>	string	Mass flow rate value and unit
	<Mass Flux Rate (Mass Flux only)>	string	Mass flux value and unit
	<Volume Flow Rate (Volume Flow only)>	string	Volume flow rate value and unit
	<Supply Flow Direction>	string	Please type in description manually
	<X, Y, and Z (Supply Flow Direction: Specified only)>	int	Direction vector values
	<ExtractFace>	list	Extract opening face

Return Value	None
---------------------	------

Python Syntax	AssignRecircBoundary (<NAME>, <Faces>, <Thermal Condition>, <Temperature Change>, <Inlet Type>, <Mass Flow Rate>, <Supply Flow Direction>, <ExtractFace>)
Python Example	<pre> oModule.AssignRecircBoundary(["NAME:RecircOpening1", "Faces:=" , [40,35], "Thermal Condition:=" , "Temperature", "Temperature Change:=" , "0cel", "Inlet Type:=" , "Mass Flow", "Mass Flow Rate:=" , "0kg_per_s", "Supply Flow Direction:=" , "Normal", "ExtractFace:=" , [40]]) </pre>

VB Syntax	AssignRecircBoundary (<NAME>, <Faces>, <Thermal Condition>, <Temperature Change>, <Inlet Type>, <Mass Flow Rate>, <Supply Flow Direction>, <ExtractFace>)
VB Example	<pre> oModule.AssignRecircBoundary Array("NAME:RecircOpening1", "Faces:=", Array(40, 38), "Thermal Condition:=" , _"Temperature", "Tem- perature Change:=" , "0cel", "Inlet Type:=" , "Mass Flow", "Mass Flow Rate:=" , _ "0kg_per_s", "Supply Flow Direction:=" , "Normal", "ExtractFace:=" , Array (40)) </pre>

AssignAdiabaticPlateBoundary

Creates an adiabatic plate thermal boundary condition in an Icepak design.

UI Access	Icepak > Thermal > Assign > Plate > Adiabatic		
Parameters	Name <NAME>	Type string	Description Boundary condition name
	<Faces>	list	Faces included in the boundary condition assignment

	<i><NAME></i>	string	LowSide
	<i><Radiate></i>	string	LowSide: "True" to enable low side for radiation or "False" to disable it
	<i><RadiateTo></i>	string	LowSide: AllObjects or RefTemperature
	<i><Ref. Temperature></i>	string	LowSide: Reference temperature and unit
	<i><View Factor></i>	string	LowSide: View factor
	<i><Surface Material></i>	string	LowSide: Surface material
	<i><NAME></i>	string	HighSide
	<i><Radiate></i>	string	HighSide: "True" to enable high side for radiation or "False" to disable it
	<i><RadiateTo - High></i>	string	HighSide: AllObjects or RefTemperature - High
	<i><Ref. Temperature - High></i>	string	HighSide: Reference temperature and unit
	<i><View Factor - High></i>	string	HighSide: View factor
	<i><Surface Material - High></i>	string	HighSide: Surface material
Return Value	None		

Python Syntax	AssignAdiabaticPlateBoundary (<NAME>, <Faces>, <NAME>, <Radiate>, <NAME>, <Radiate>)
Python Example	<pre> oModule.AssignAdiabaticPlateBoundary(["NAME:AdiabaticPlate1", "Faces:=" , [7], ["NAME:LowSide", "Radiate:=" , True, "RadiateTo:=" , "AllObjects", "Surface Material:=" , "Steel-oxidised-surface"], ["NAME:HighSide", "Radiate:=" , True, "RadiateTo - High:=" , "RefTemperature - High", "Ref. Temperature - High:=" , "0",]]) </pre>

	<pre> "View Factor - High:=" , "0", "Surface Material - High:=", "Steel-oxidised-surface"]) </pre>
--	--

VB Syntax	AssignAdiabaticPlateBoundary (<NAME>, <Faces>, <NAME>, <Radiate>, <NAME>, <Radiate>)
VB Example	<pre> oModule.AssignAdiabaticPlateBoundary Array("NAME:A- diabaticPlate3", "Faces:=", Array(_ 11), Array("NAME:LowSide", "Radiate:=", true, "RadiateTo:=", "AllObjects", "Surface Material:=", _ "Steel-oxidised-sur- face"), Array("NAME:HighSide", "Radiate:=", true, "RadiateTo - High:=", _ "RefTemperature - High", "Ref. Temperature - High:=", "0", "View Factor - High:=", _ "0", "Surface Material - High:=", "Steel-oxidised-surface")) </pre>

AssignConductingPlateBoundary

Creates a conducting plate thermal boundary condition in an Icepak design.

UI Access	Icepak > Thermal > Assign > Plate > Conducting		
Parameters	Name	Type	Description
	<NAME>	string	Boundary condition name
	<Faces>	list	Faces included in the boundary condition assignment
	<NAME>	string	LowSide
	<Radiate>	string	LowSide: "True" to enable low side for radiation or "False" to disable it
	<RadiateTo>	string	LowSide: AllObjects or RefTemperature
	<Ref. Temperature>	string	LowSide: Reference temperature and unit
	<View Factor>	string	LowSide: View factor
	<Surface Material>	string	LowSide: Surface material
	<NAME>	string	HighSide

	<i><Radiate></i>	string	HighSide: "True" to enable high side for radiation or "False" to disable it
	<i><RadiateTo - High></i>	string	HighSide: AllObjects or RefTemperature - High
	<i><Ref. Temperature - High></i>	string	HighSide: Reference temperature and unit
	<i><View Factor - High></i>	string	HighSide: View factor
	<i><Surface Material - High></i>	string	HighSide: Surface material
	<i><Thermal Specification></i>	string	Thickness, Conductance, Thermal Resistance, or Thermal Impedance
	<i><Thickness></i>	string	Thickness and unit
	<i><Solid Material></i>	string	Solid material
	<i><Conductance></i>	string	Conductance and unit
	<i><Thermal Resistance></i>	string	Thermal resistance and unit
	<i><Thermal Impedance></i>	string	Thermal impedance and unit
	<i><Total Power></i>	string	Total power and unit
	<i><Shell Conduction></i>	bool	"True" to enable shell conduction or "False" to disable it
Return Value	None		

Python Syntax	AssignConductingPlateBoundary (<NAME>, <Faces>, <NAME>, <Radiate>, <RadiateTo>, <Ref. Temperature>, <View Factor>, <Surface Material>, <NAME>, <Radiate>, <RadiateTo - High>, <Ref. Temperature - High>, <View Factor - High>, <Surface Material - High>, <Thermal Specification>, <Thickness>, <Solid Material>, <Conductance>, <Thermal Resistance>, <Thermal Impedance>, <Total Power>, <Shell Conduction>)
Python Example	<pre> oModule.AssignConductingPlateBoundary(["NAME:ConductingPlate1", "Faces:=" , [7], ["NAME:LowSide", "Radiate:=" , True, "RadiateTo:=" , "RefTemperature", "Ref. Temperature:=" , "0", "View Factor:=" , "0", "Surface Material:=" , "Steel-oxidised-surface"]]) </pre>

	<pre>], ["NAME:HighSide", "Radiate:=" , True, "RadiateTo - High:=" , "RefTemperature - High", "Ref. Temperature - High:=" , "0", "View Factor - High:=" , "0", "Surface Material - High:=" , "Steel-oxidised-surface"], "Thermal Specification:=" , "Thickness", "Thickness:=" , "1mm", "Solid Material:=" , "Al-Extruded", "Conductance:=" , "0W_per_Cel", "Thermal Resistance:=" , "0Kel_per_W", "Thermal Impedance:=" , "0celm2_per_w", "Total Power:=" , "0W", "Shell Conduction:=" , False]) </pre>
--	---

VB Syntax	AssignConductingPlateBoundary (<NAME>, <Faces>, <NAME>, <Radiate>, <RadiateTo>, <Ref. Temperature>, <View Factor>, <Surface Material>, <NAME>, <Radiate>, <RadiateTo - High>, <Ref. Temperature - High>, <View Factor - High>, <Surface Material - High>, <Thermal Specification>, <Thickness>, <Solid Material>, <Conductance>, <Thermal Resistance>, <Thermal Impedance>, <Total Power>, <Shell Conduction>)
VB Example	<pre> oModule.AssignConductingPlateBoundary Array("NAME:Con- ductingPlate2", "Faces:=", Array(_ 12), Array("NAME:LowSide", "Radiate:=", true, "RadiateTo:=", "RefTemperature", "Ref. Temperature:=", _ "0", "View Factor:=", "0", "Surface Material:=", "Steel-oxidised- surface"), Array("NAME:HighSide", "Radiate:=" , true, "RadiateTo - High:=", "RefTemperature - High", "Ref. Temperature - High:=", _ "0", "View Factor - High:=", "0", "Surface Material - High:=", _ </pre>

```

    "Steel-oxidised-surface"), "Thermal Specification:=", "Thickness",
    "Thickness:=", _ "1mm", "Solid Material:=", "Al-Extruded",
    "Conductance:=", "0W_per_Cel", "Thermal Resistance:=", _ 
    "0Kel_per_W", "Thermal Impedance:=", "0celm2_per_w", "Total
    Power:=",
    "0W", "Shell Conduction:=", _ true)

```

AssignResistanceBoundary

Creates a resistance thermal boundary condition in an Icepak design.

UI Access	Icepak > Thermal > Assign > Resistance		
Parameters	Name	Type	Description
	<NAME>	string	Boundary condition name
	<Objects>	list	Objects included in the boundary condition assignment
	<Pressure Loss Model>	string	Device/Approach, Power Law, or Loss Curve
	<Laminar Flow>	bool	"True" to enable laminar flow or 'False' to disable it
	<Thermal Power>	string	Resistance power value and unit
	<Thermal Power>	string	Resistance power value and unit
	<Fluid Material>	string	Fluid material
	<Linear X Coef-ficient>	string	Linear X direction loss coefficient and unit
	<Linear Y Coef-ficient>	string	Linear Y direction loss coefficient and unit
	<Linear Z Coef-ficient>	string	Linear Z direction loss coefficient and unit
	<Quadratic X Coef-ficient>	string	Quadratic X direction loss coefficient
	<Quadratic Y Coef-ficient>	string	Quadratic Y direction loss coefficient
	<Quadratic Z Coef-ficient>	string	Quadratic Z direction loss coefficient
	<Linear X Free Area Ratio>	string	Linear X direction free area ratio
	<Linear Y Free Area Ratio>	string	Linear Y direction free area ratio
	<Linear Z Free Area Ratio>	string	Linear Z direction free area ratio
	<Quadratic X Free	string	Quadratic X direction free area ratio

	<i>Area Ratio></i>		
	<i><Quadratic Y Free Area Ratio></i>	string	Quadratic Y direction free area ratio
	<i><Quadratic Z Free Area Ratio></i>	string	Quadratic Z direction free area ratio
	<i><Power Law Coefficient></i>	string	Power law coefficient
	<i><Power Law Exponent></i>	string	Power law exponent
	<i><NAME></i>	string	Pressure Loss Curve X
	<i><NAME></i>	string	Dataset units
	<i><X></i>	list	List of X data set values
	<i><Y></i>	list	List of Y data set values
	<i><NAME></i>	string	Pressure Loss Curve Y
	<i><NAME></i>	string	Dataset units
	<i><X></i>	list	List of X data set values
	<i><Y></i>	list	List of Y data set values
	<i><NAME></i>	string	Pressure Loss Curve Z
	<i><NAME></i>	string	Dataset units
	<i><X></i>	list	List of X data set values
	<i><Y></i>	list	List of Y data set values
Return Value	None		

Python Syntax	AssignResistanceBoundary (<NAME>, <Objects>, <Pressure Loss Model>, <Laminar Flow>, <Thermal Power>, <Thermal Power>, <Fluid Material>, <Linear X Coefficient>, <Linear Y Coefficient>, <Linear Z Coefficient>, <Quadratic X Coefficient>, <Quadratic Y Coefficient>, <Quadratic Z Coefficient>, <Linear X Free Area Ratio>, <Linear Y Free Area Ratio>, <Linear Z Free Area Ratio>, <Quadratic X Free Area Ratio>, <Quadratic Y Free Area Ratio>, <Quadratic Z Free Area Ratio>, <Power Law Coefficient>, <Power Law Exponent>, <NAME>, <NAME>, <X>, <Y>, <NAME>, <NAME>, <X>, <Y>, <NAME>, <NAME>, <X>, <Y>)
Python Example	<pre> oModule.AssignResistanceBoundary(["NAME:Resistance1", "Objects:=" , ["Region"], "Pressure Loss Model:=" , "Device/Approach", "Laminar Flow:=" , False, "Thermal Power:=" , "5W", "Thermal Power:=" , "5W",]) </pre>

```

    "Fluid Material:=" , "air",
    "Linear X Coefficient:=", "0m_per_sec",
    "Linear Y Coefficient:=", "0m_per_sec",
    "Linear Z Coefficient:=", "0m_per_sec",
    "Quadratic X Coefficient:=", "0",
    "Quadratic Y Coefficient:=", "0",
    "Quadratic Z Coefficient:=", "0",
    "Linear X Free Area Ratio:=", "0.8",
    "Linear Y Free Area Ratio:=", "0.8",
    "Linear Z Free Area Ratio:=", "0.8",
    "Quadratic X Free Area Ratio:=", "1",
    "Quadratic Y Free Area Ratio:=", "1",
    "Quadratic Z Free Area Ratio:=", "1",
    "Power Law Coefficient:=", "1",
    "Power Law Exponent:=" , "1",
    [
        "NAME:Pressure Loss Curve X",
        [
            "NAME:DimUnits",
            """",
            """
        ],
        "X:=" , ["0","1","2"],
        "Y:=" , ["0","1","2"]
    ],
    [
        "NAME:Pressure Loss Curve Y",
        [
            "NAME:DimUnits",
            """",
            """
        ],
        "X:=" , ["0","1","2"],
        "Y:=" , ["0","1","2"]
    ],
    [
        "NAME:Pressure Loss Curve Z",
        [
            "NAME:DimUnits",
            """",
            """
        ],
        "X:=" , ["0","1","2"],
        "Y:=" , ["0","1","2"]
    ]
]
)

```

VB Syntax <pre>AssignResistanceBoundary (<NAME>, <Objects>, <Pressure Loss Model>, <Laminar Flow>, <Thermal Power>, <Thermal Power>, <Fluid Material>, <Linear X Coefficient>, <Linear Y Coefficient>, <Linear Z Coefficient>, <Quadratic X Coefficient>, <Quadratic Y Coefficient>, <Quadratic Z Coefficient>, <Linear X Free Area Ratio>, <Linear Y Free Area Ratio>, <Linear Z Free Area Ratio>, <Quadratic X Free Area Ratio>, <Quadratic Y Free Area Ratio>, <Quadratic Z Free Area Ratio>, <Power Law Coefficient>, <Power Law Exponent>, <NAME>, <NAME>, <X>, <Y>, <NAME>, <NAME>, <X>, <Y>)</pre>
VB Example <pre>oModule.AssignResistanceBoundary Array("NAME:Resistance1", "Objects:=", Array(_ "Region"), "Pressure Loss Model:=", "Device/Approach", "Laminar Flow:=", false, "Thermal Power:=", _"5W", "Thermal Power:=", "5W", "Fluid Material:=", "air", "Linear X Coefficient:=", _"0m_per_sec", "Linear Y Coefficient:=", "0m_per_sec", "Linear Z Coefficient:=", _"0m_per_sec", "Quadratic X Coefficient:=", "0", "Quadratic Y Coefficient:=", "0", "Quadratic Z Coefficient:=", _"0", "Linear X Free Area Ratio:=", "0.8", "Linear Y Free Area Ratio:=", "0.8", "Linear Z Free Area Ratio:=", _"0.8", "Quadratic X Free Area Ratio:=", "1", "Quadratic Y Free Area Ratio:=", _"1", "Quadratic Z Free Area Ratio:=", "1", "Power Law Coefficient:=", "1", "Power Law Exponent:=", _"1", Array("NAME:Pressure Loss Curve X", Array("NAME:DimUnits", "", ""), "X:=", Array("0", _"1", "2"), "Y:=", Array("0", "1", "2")), Array("NAME:Pressure Loss Curve Y", Array("NAME:DimUnits", _ "", ""), "X:=", Array("0", "1", "2"), "Y:=", Array("0", "1", "2")), Array("NAME:Pressure Loss Curve Z", Array("NAME:DimUnits", _, ""), "X:=", Array("0", "1", "2"), "Y:=", Array("0", _"1", "2")))</pre>

AssignSourceBoundary

Creates a source thermal boundary condition in an Icepak design.

UI Access	Icepak > Thermal > Assign > Source		
Parameters	Name	Type	Description
	<NAME>	string	Boundary condition name
	<Faces>	list	Faces included in the boundary condition assignment
	<Thermal Condition>	string	Total Power, Surface flux, or Fixed Temperature
	<Total Power>	string	Source power value and unit
	<Surface Heat>	string	Source surface heat value and unit
	<Temperature>	string	Source temperature value and unit
	<NAME>	string	Radiation
	<Radiate>	string	"True" to enable radiation or "False" to disable it
	<Voltage/Current - Enabled>	bool	"True" to enable voltage or current or "False" to disable it
	<Voltage/Current Option>	string	Current or Voltage
	<Current>	string	Current value and unit
	<Voltage>	string	Voltage vale and unit
	<RadiateTo>	string	AllObjects or RefTemperature
	<Ref. Temperature>	string	Reference temperature and unit
	<View Factor>	string	View factor
	<Surface Material>	string	Surface material
Return Value	None		

Python Syntax	AssignSourceBoundary (<NAME>, <Faces>, <Thermal Condition>, <Total Power>, <Surface Heat>, <Temperature>, <NAME>, <Radiate>, <RadiateTo>, <Ref. Temperature>, <View Factor>, <Surface Material>)
Python Example	<pre> oModule.AssignSourceBoundary (["NAME:Source1", "Faces:=" , [7], "Thermal Condition:=" , "Total Power", "Total Power:=" , "5W", "Surface Heat:=" , "0irrad_W_per_m2", "Temperature:=" , "AmbientTemp",]) </pre>

	<pre>["NAME:Radiation", "Radiate:=" , True, "RadiateTo:=" , "RefTemperature", "Ref. Temperature:=" , "0", "View Factor:=" , "1", "Surface Material:=" , "Steel-oxidised-surface"] "Voltage/Current - Enabled:=", True, "Voltage/Current Option:=", "Current", "Current:=" , "0A", "Voltage:=" , "0V"])</pre>
--	---

VB Syntax	AssignSourceBoundary (<NAME>, <Faces>, <Thermal Condition>, <Total Power>, <Surface Heat>, <Temperature>, <NAME>, <Radiate>, <RadiateTo>, <Ref. Temperature>, <View Factor>, <Surface Material>)
VB Example	<pre>oModule.AssignSourceBoundary Array("NAME:Source2", "Faces:=", Array(12), "Thermal Condition:= ", _"Total Power", "Total Power:=", "5W", "Surface Heat:=", "0irrad_W_per_m2", "Temperature:=",_ "AmbientTemp", Array("NAME:Radiation", "Radiate:=", true, "RadiateTo:=", _"RefTemperature", "Ref. Temperature:=", "0", "View Factor:=", "1", "Surface Material:=", _Steel-oxidised-surface"))</pre>

AssignStationaryWallBoundary

Creates a stationary wall thermal boundary condition in an Icepak design.

UI Access	Icepak > Thermal > Assign > Wall > Stationary		
Parameters	Name	Type	Description
	<NAME>	string	Boundary condition name
	<Faces>	list	Faces included in the boundary condition assignment
	<Thickness>	string	Wall thickness and unit
	<Solid Material>	string	Solid material
	<External Material>	string	External material
	<Internal Material>	string	Internal material
	<External Condition>	string	Heat flux, Temperature, or Heat Transfer Coefficient

	<i><Heat Flux></i>	string	Wall heat flux and unit
	<i><Temperature></i>	string	Wall temperature and unit
	<i><Heat Transfer Coefficient></i>	string	Wall heat transfer coefficient and unit
	<i><Reference Temperature></i>	string	Reference temperature and unit
	<i><NAME></i>	string	Heat Transfer Data
	<i><Heat Transfer Correlation></i>	bool	"True" to enable heat transfer correlation or "False" to disable it
	<i><Heat Transfer Convection Type></i>	string	Forced or Natural
	<i><NAME></i>	string	Radiation
	<i><Radiate></i>	bool	"True" to enable radiation or "False" to disable it
	<i><Shell Conduction></i>	bool	"True" to enable shell conduction or "False" to disable it
	<i><Slip Flow Boundary></i>	bool	"True" to enable skip flow boundary or "False" to disable it
Return Value	None		

Python Syntax	AssignStationaryWallBoundary (<NAME>, <Faces>, <Thickness>, <Solid Material>, <External Material>, <Internal Material>, <External Condition>, <Heat Flux>, <Temperature>, <Heat Transfer Coefficient>, <Reference Temperature>, <NAME>, <Heat Transfer Correlation>, <Heat Transfer Convection Type>, <NAME>, <Radiate>, <Shell Conduction>)
Python Example	<pre> oModule.AssignStationaryWallBoundary (["NAME:StationaryWall1", "Faces:=" , [12], "Thickness:=" , "2mm", "Solid Material:=" , "Al-Extruded", "External Material:=" , "Steel-oxidised-surface", "Internal Material:=" , "Steel-oxidised-surface", "External Condition:=" , "Heat Flux", "Heat Flux:=" , "5uW_per_m2", "Temperature:=" , "AmbientTemp", "Heat Transfer Coefficient:=", "0w_per_m2kel", "Reference Temperature:=", "AmbientTemp", ["NAME:Heat Transfer Data", "Heat Transfer Correlation:=", False]]) </pre>

	<pre> "Heat Transfer Convection Type:=", "Forced Conve-], ["NAME:Radiation", "Radiate:=" , False], "Shell Conduction:=" , False]) </pre>
--	--

VB Syntax	AssignStationaryWallBoundary (<NAME>, <Faces>, <Thickness>, <Solid Material>, <External Material>, <Internal Material>, <External Condition>, <Heat Flux>, <Temperature>, <Heat Transfer Coefficient>, <Reference Temperature>, <NAME>, <Heat Transfer Correlation>, <Heat Transfer Convection Type>, <NAME>, <Radiate>, <Shell Conduction>)
VB Example	<pre> oModule.AssignStationaryWallBoundary Array("NAME:Sta- tionaryWall1", "Faces:=", Array(_ 12), "Thickness:=", "2mm", "Solid Material:=", "Al-Extruded", "External Material:=", _ "Steel-oxidised-surface", "Internal Material:=", "Steel-oxid- ised-surface", "External Condition:=", "Heat Flux", "Heat Flux:=", "5uW_per_m2", "Temperature:=", "AmbientTemp", "Heat Transfer Coefficient:=", "0w_per_m2kel", "Reference Temperature:=", "AmbientTemp", Array("NAME:Heat Transfer Data", "Heat Transfer Cor- relation:=", false, "Heat Transfer Convection Type:=", "Forced Convection"), Array("NAME:Ra- diation", "Radiate:=", false), "Shell Conduction:=", false) </pre>

AssignSymmetryWallBoundary

Creates a symmetry wall thermal boundary condition in an Icepak design.

UI Access	Icepak > Thermal > Assign > Wall > Symmetry		
Parameters	Name	Type	Description
	<NAME>	string	Boundary condition name
	<Faces>	list	Faces included in the boundary condition assignment

Return Value	None
---------------------	------

Python Syntax	AssignSymmetryWallBoundary (<NAME>, <Faces>)
Python Example	<pre>oModule.AssignSymmetryWallBoundary(["NAME:SymmetryWall1", "Faces:=", , [12]])</pre>

VB Syntax	AssignSymmetryWallBoundary (<NAME>, <Faces>)
VB Example	<pre>oModule.AssignSymmetryWallBoundary Array("NAME:SymmetryWall1", "Faces:=", Array(12))</pre>

AssignEMLoss

Creates an EM loss thermal boundary condition in an Icepak design.

UI Access	Icepak > Thermal > 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>	string	Params
	<ForceSourceToSolve>	bool	"True" to solve the source project or "False" to not solve it
	<PreservePartnerSoln>	bool	"True" to save the source design results or "False" to discard them
	<PathRelativeTo>	string	TargetProject or SourceProduct
Return Value	None		

Python Syntax	AssignEMLoss (<NAME>, <Objects>, <Project>, <Product>, <Design>, <Soln>, <NAME>, <ForceSourceToSolve>, <PreservePartnerSoln>, <PathRelativeTo>, <Intrinsics>, <SurfaceOnly>)
Python Example	<pre> oModule.AssignEMLoss(["NAME:EMLoss1", "Objects:=" , ["Stock_0","Stock_1","Stock_2"], "Project:=" , "This Project*", "Product:=" , "ElectronicsDesktop", "Design:=" , "Maxwell3DDesign1", "Soln:=" , "Setup1 : LastAdaptive", ["NAME:Params"], "ForceSourceToSolve:=" , False, "PreservePartnerSoln:=" , False, "PathRelativeTo:=" , "TargetProject", "Intrinsics:=" , ["200Hz"], "SurfaceOnly:=" , []]) </pre>

VB Syntax	AssignEMLoss (<NAME>, <Objects>, <Project>, <Product>, <Design>, <Soln>, <NAME>, <ForceSourceToSolve>, <PreservePartnerSoln>, <PathRelativeTo>, <Intrinsics>, <SurfaceOnly>)
VB Example	<pre> oModule.AssignEMLoss Array("NAME:EMLoss1", "Objects:=", Array ("Stock_0", "Stock_1", "Stock_2"), "Project:=", "This Project*", "Product:=", "Elec- tronicsDesktop", "Design:=", _"Maxwell3DDesign1", "Soln:=", "Setup1 : LastAdaptive", Array("NAME:Params"), "ForceSourceToSolve:=",_false, "PreservePartnerSoln:=", false, "PathRelativeTo:", "TargetProject", "Intrinsics:=", Array(_"200Hz"), "Sur- faceOnly:=", Array()) </pre>

AssignBlowerBoundary

Creates a blower thermal boundary condition in an Icepak design.

UI Access	Icepak > Thermal > Assign > Blower		
Parameters	Name <NAME>	Type string	Description Boundary condition name

	<i><Faces></i>	list	Faces included in the boundary condition assignment
	<i><NAME></i>	string	DimUnits
	<i><X></i>	list	Flow curve dataset x values
	<i><Y></i>	list	Flow curve dataset y values
	<i><Blower Type></i>	int	Type 1 or Type 2
	<i><Fan Blade Angle></i>	string	Type 1 only: angle of the blower fan blade
	<i><Blade RPM></i>	string	Type 1 only: rotations per minute value
	<i><Exhaust Exit Angle></i>	string	Type 2 only: blower exhaust exit angle value and unit
	<i><Blower Power></i>	string	Blower power value and unit
	<i><InletFace></i>	list	Inlet face
Return Value	None		

Python Syntax	AssignBlowerBoundary (<NAME>, <Faces>, <NAME>, <X>, <Y>, <Blower Type>, <Fan Blade Angle>, <Blade RPM>, <Blower Power>, <InletFace>)
Python Example	<pre> oModule.AssignBlowerBoundary(["NAME:Blower1", "Faces:=" , [95, 94, 93], ["NAME:DimUnits", "m_per_sec", "n_per_meter_sq"], "X:=" , ["0", "1", "2"], "Y:=" , ["0", "1", "2"], "Blower Type:=" , "Type 1", "Fan Blade Angle:=" , "0rad", "Blade RPM:=" , "0", "Blower Power:=" , "0W", "InletFace:=" , [95, 94]]) </pre>

VB Syntax	AssignBlowerBoundary (<NAME>, <Faces>, <NAME>, <X>, <Y>, <Blower Type>, <Fan Blade Angle>, <Blade RPM>, <Blower Power>, <InletFace>)
VB Example	<pre> oModule.AssignBlowerBoundary Array("NAME:Blower3", "Faces:=", Array(95, 94, 93), </pre>

```

Array("NAME:DimUnits", _ "m_per_sec", "n_per_meter_sq"),
"X:=", Array("0", "1",
"2"), "Y:=", Array("0", _ "1", "2"), "Blower Type:=", "Type
1", "Fan Blade Angle:=",
"0rad", "Blade RPM:=", _ "0", "Blower Power:=", "0W",
"InletFace:=", Array(95, 94))

```

f

14 - Mesh Region Module Script Commands

Commands for mesh setup and operations should be executed by the "MeshSetup" module.

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

The topics for this section include:

[Script Commands for Creating and Modifying Mesh Regions](#)

[General Commands Recognized by the Mesh Operations Module](#)

Script Commands for Creating and Modifying Mesh Regions

Script commands for creating and modifying mesh operations are as follows:

[EditGlobalMeshRegion](#)

[AssignVirtualMeshRegion](#)

[EditMeshRegion](#)

EditGlobalMeshRegion

Defines Icepak global mesh region settings.

UI Access	Icepak > Mesh > Edit Global Region		
Parameters	Name	Type	Description
	<NAME>	string	Settings
	<MeshMethod>	string	MesherHD
	<UserSpecifiedSettings>	bool	"True" to enable advanced mesh settings or "False" to disable them
	<ComputeGap>	bool	"True" to enable minimum gap

		override settings or "False" to disable it
Auto Mesh Setting		
<MeshRegionResolution>	int	1, 2, 3, 4, or 5
<MinGapX>	string	Minimum gap X value and unit
<MinGapY>	string	Minimum gap Y value and unit
<MinGapZ>	string	Minimum gap Z value and unit
<Objects>	list	Region
Advanced Mesh Settings		
<MaxElementSizeX>	string	Maximum Element Size X value and unit
<MaxElementSizeY>	string	Maximum Element Size Y value and unit
<MaxElementSizeZ>	string	Maximum Element Size Z value and unit
<MinElementsInGap>	string	Minimum
<MinElementsOnEdge>	string	Minimum
<MaxSizeRatio>	string	Maximum
<NoOGrids>	bool	"True" to enable no O grid meshing or "False" to disable it
<EnableMLM>	bool	"True" to enable multi-level meshing or "False" to disable it
<EnforeMLMType>	string	3D or 2D
<MaxLevels>	string	Maximum levels value
<BufferLayers>	string	Buffer layers value
<UniformMeshParametersType>	string	Average or XYZ Max Sizes
<StairStepMeshing>	bool	"True" to enable stairstep meshing or "False" to disable it
<MinGapX>	string	Minimum gap X value and unit
<MinGapY>	string	Minimum gap Y value and unit
<MinGapZ>	string	Minimum gap Z value and unit
<Objects>	list	Region
Return Value	None	

Python Syntax	EditGlobalMeshRegion (<NAME>, <MeshMethod>, <UserSpecifiedSettings>, <ComputeGap>, <MeshRegionResolution>, <MinGapX>, <MinGapY>, <MinGapZ>, <Objects>)
Python Example	<pre>oModule.EditGlobalMeshRegion (["NAME:Settings",</pre>

	<pre> "MeshMethod:=", "MesherHD", "UserSpecifiedSettings:=", True, "ComputeGap:=", True, "MaxElementSizeX:=", "0.02667mm", "MaxElementSizeY:=", "0.02667mm", "MaxElementSizeZ:=", "0.02667mm", "MinElementsInGap:=", "3", "MinElementsOnEdge:=", "2", "MaxSizeRatio:=", "2", "NoOGrids:=", False, "EnableMLM:=", True, "EnforeMLMTYPE:=", "3D", "MaxLevels:=", "0", "BufferLayers:=", "0", "UniformMeshParametersType:=", "Average", "StairStepMeshing:=", False, "MinGapX:=", "1mm", "MinGapY:=", "1mm", "MinGapZ:=", "1mm", "Objects:=", ["Region"]]) </pre>
--	--

VB Syntax	EditGlobalMeshRegion (<NAME>, <MeshMethod>, <UserSpecifiedSettings>, <ComputeGap>, <MeshRegionResolution>, <MinGapX>, <MinGapY>, <MinGapZ>, <Objects>)
VB Example	<pre> oModule.EditGlobalMeshRegion Array("NAME:Settings", "MeshMethod:=", "MesherHD", "UserSpecifiedSettings:=", _true, "ComputeGap:=", true, "MaxElementSizeX:=", "0.02667mm", "MaxElementSizeY:=", _"0.02667mm", "MaxElementSizeZ:=", "0.02667mm", "MinElementsInGap:=", "3", "MinElementsOnEdge:=", _"2", "MaxSizeRatio:=", "2", "NoOGrids:=", false, "EnableMLM:=", true, "EnforeMLMTYPE:=", _ "3D", "MaxLevels:=", "0", "BufferLayers:=", "0", "UniformMeshParametersType:=", _"Average", "StairStepMeshing:=", false, "MinGapX:=", "1mm", "MinGapY:=", "1mm", "MinGapZ:=", _ "1mm", "Objects:=", Array("Region")) </pre>

AssignVirtualMeshRegion

AssignVirtualMeshRegion allows you to assign a mesh region in Icepak.

UI Access	Right-click geometry > Assign Mesh Region		
Parameters	Name	Type	Description
	<NAME>	string	Mesh region name
	<Enable>	bool	True or False
	<MeshMethod>	string	MesherHD
	<UserSpecifiedSettings>	bool	True or False
	<MeshRegionResolution>	int	Auto mesh setting value (1-5)
	<MinGapX>	string	Minimum gap X value and unit
	<MinGapY>	string	Minimum gap Y value and unit
	<MinGapZ>	string	Minimum gap Z value and unit
	<Objects>	list	Selected objects
	<ProximitySizeFunction>	bool	True or False
	<CurvatureSizeFunction>	bool	True or False
	<EnableTransition>	bool	True or False
	<OptimizePCBMesh>	bool	True or False
	<Enable2DCutCell>	bool	True or False
	<EnforceCutCellMeshing>	bool	True or False
	<Enforce2dot5DCutCell>	bool	True or False
	<SlackMinX>	string	Slack minimum X direction value and unit
	<SlackMinY>	string	Slack minimum Y direction value and unit
	<SlackMinZ>	string	Slack minimum Z direction value and unit
	<SlackMaxX>	string	Slack maximum X direction value and unit
	<SlackMaxY>	string	Slack maximum Y direction value and unit
	<SlackMaxZ>	string	Slack maximum Z direction value and unit
Return Value	None		

Python Syntax	AssignVirtualMeshRegion (<NAME>, <Enable>, <MeshMethod>, <User-SpecifiedSettings>, <MeshRegionResolution>, <MinGapX>, <MinGapY>,
---------------	--

	<MinGapZ>, <Objects>, <ProximitySizeFunction>, <CurvatureSizeFunction>, <EnableTransition>, <OptimizePCBMesh>, <Enable2DCutCell>, <EnforceCutCellMeshing>, <Enforce2dot5DCutCell>, <SlackMinX>, <SlackMinY>, <SlackMinZ>, <SlackMaxX>, <SlackMaxY>, <SlackMaxZ>, <NAME>, <MinSlackX>, <MaxSlackX>, <MinSlackY>, <MaxSlackY>, <MinSlackZ>, <MaxSlackZ>, <MinBboxX>, <MaxBboxX>, <MinBboxY>, <MaxBboxY>, <MinBboxZ>, <MaxBboxZ>)
Python Example	<pre> oModule.AssignVirtualMeshRegion(["NAME:MeshRegion1", "Enable:=" , True, "MeshMethod:=" , "MesherHD", "UserSpecifiedSettings:=", False, "MeshRegionResolution:=", 3, "MinGapX:=" , "1mm", "MinGapY:=" , "1mm", "MinGapZ:=" , "1mm", "Objects:=" , ["HEAT_SINK"], "ProximitySizeFunction:=", True, "CurvatureSizeFunction:=", True, "EnableTransition:=" , False, "OptimizePCBMesh:=" , True, "Enable2DCutCell:=" , False, "EnforceCutCellMeshing:=", False, "Enforce2dot5DCutCell:=" , False, "SlackMinX:=" , "5mm", "SlackMinY:=" , "5mm", "SlackMinZ:=" , "5mm", "SlackMaxX:=" , "5mm", "SlackMaxY:=" , "5mm", "SlackMaxZ:=" , "5mm"],) </pre>

VB Syntax	AssignVirtualMeshRegion (<NAME>, <Enable>, <MeshMethod>, <UserSpecifiedSettings>, <MeshRegionResolution>, <MinGapX>, <MinGapY>, <MinGapZ>, <Objects>, <ProximitySizeFunction>, <CurvatureSizeFunction>, <EnableTransition>, <OptimizePCBMesh>, <Enable2DCutCell>, <EnforceCutCellMeshing>, <Enforce2dot5DCutCell>, <SlackMinX>, <SlackMinY>, <SlackMinZ>, <SlackMaxX>, <SlackMaxY>, <SlackMaxZ>, <NAME>, <MinSlackX>, <MaxSlackX>, <MinSlackY>, <MaxSlackY>, <MinSlackZ>, <MaxSlackZ>, <MinBboxX>, <MaxBboxX>, <MinBboxY>, <MaxBboxY>, <MinBboxZ>, <MaxBboxZ>)
VB Exa-	<pre> oModule.AssignVirtualMeshRegion Array("NAME:MeshRegion1", "Enable:=" , true"MeshMethod:=" , _MesherHD", </pre>

mple	<pre>"UserSpecifiedSettings:=", false, "MeshRegionResolution:=", 3, "MinGapX:=", "1mm", "MinGapY:=", "1mm", "MinGapZ:=", "1mm", "Objects:=", Array("HEAT_SINK"), "ProximitySizeFunction:=", _true, "CurvatureSizeFunction:=", true, "EnableTransition:=", false, "OptimizePCBMesh:=", _true, "Enable2DCutCell:=", false, "EnforceCutCellMeshing:=", false, "Enforce2dot5DCutCell:=", _false, "SlackMinX:=", "5mm", "SlackMinY:=", "5mm", "SlackMinZ:=", "5mm", "SlackMaxX:=", _"5mm", "SlackMaxY:=", "5mm", "SlackMaxZ:=", "5mm")</pre>
------	---

EditMeshRegion

Modifies an existing mesh region settings.

UI Access	Right-click on a mesh region in the project tree, then select Properties...									
Parameters	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><MeshRegionName></td> <td>String</td> <td>Name of specified mesh region.</td> </tr> <tr> <td><MeshRegionParams></td> <td>Array</td> <td>Structured array. Array ("NAME:<MeshRegionName>","Enabled:=", <boolean>)</td> </tr> </tbody> </table>	Name	Type	Description	<MeshRegionName>	String	Name of specified mesh region.	<MeshRegionParams>	Array	Structured array. Array ("NAME:<MeshRegionName>","Enabled:=", <boolean>)
Name	Type	Description								
<MeshRegionName>	String	Name of specified mesh region.								
<MeshRegionParams>	Array	Structured array. Array ("NAME:<MeshRegionName>","Enabled:=", <boolean>)								
Return Value	None.									

Python Syntax	EditMeshRegion (<MeshRegionName>, <MeshRegionParams>)
Python Example	<pre>oModule.EditMeshRegion ("MeshRegion1", ["NAME:MeshRegion2", "Enabled:=", True])</pre>

VB Syntax	EditMeshRegion <MeshRegionName>, <MeshRegionParams>
------------------	---

VB Example

```
oModule.EditMeshRegion "MeshRegion1"_
Array("NAME:MeshRegion2", _
"Enabled:=", true)
```

General Commands Recognized by the Mesh Operations Module

General commands recognized by the Mesh Operations Module:

[DeleteOp](#)

[GetMeshOpAssignment](#)

[GetOperationName](#)

[RenameOp](#)

DeleteOp

Deletes the specified mesh operations.

UI Access	Delete command in the List dialog box.		
Parameters	Name <i><NameArray></i>	Type Array	Description Array of mesh operation names.
Return Value	None.		

Python Syntax

`DeleteOp(<NameArray>)`

Python Example

```
oModule.DeleteOp(["Length1", "SkinDepth1", "Length2"])
```

VB Syntax

`DeleteOp <NameArray>`

VB Example

```
oModule.DeleteOp Array("Length1", "SkinDepth1",_
"Length2")
```

GetOperationNames

Gets the names of mesh operations defined in a design.

UI Access

N/A

Parameters	Name <i><OperationType></i>	Type String	Description Specified operation type.
Return Value	Array of strings containing mesh operation names.		

Python Syntax	GetOperationNames(<OperationType>)
Python Example	<code>oModule.GetOperationNames ("Length Based")</code>

VB Syntax	GetOperationNames <OperationType>
VB Example	<code>oModule.GetOperationNames "Length Based"</code>

ReassignOp

Reassigns a mesh operation

UI Access	Right-click on a mesh operation, then select Reassign		
Parameters	Name <i><OpName></i>	Type String	Description Operation name to reassign.
	<i><AssignParams></i>	Array	Structured array. Array ("Objects:=", <array of objects> "Faces:=", <array of faces>)
Return Value	None.		

Python Syntax	ReassignOp(<OpName>, <AssignParams>)
Python Example	<pre>oModule.ReassignOp ("ApplyCurvilinear1", ["Faces:=", [16]]) oModule.ReassignOp ("Length1", [</pre>

	"Objects:=", ["Box1"]])
--	-------------------------------

VB Syntax	ReassignOp <OpName>, <AssignParams>
VB Example	<pre> oModule.ReassignOp "ApplyCurvilinear1", _ Array("Faces:=", Array(16)) oModule.ReassignOp("Length1", _ Array("Objects:=", Array("Box1"))) </pre>

RenameOp

Renames a mesh operation.

UI Access	Right-click the mesh operation in the project tree, and then click Rename on the shortcut menu.									
Parameters	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><OldName></td> <td>String</td> <td>Old name for the mesh operation.</td> </tr> <tr> <td><NewName></td> <td>String</td> <td>New name for the mesh operation.</td> </tr> </tbody> </table>	Name	Type	Description	<OldName>	String	Old name for the mesh operation.	<NewName>	String	New name for the mesh operation.
Name	Type	Description								
<OldName>	String	Old name for the mesh operation.								
<NewName>	String	New name for the mesh operation.								
Return Value	None.									

Python Syntax	RenameOp(<OldName>, <NewName>)
Python Example	<pre> oModule.RenameOp("SkinDepth1", "NewName") </pre>

VB Syntax	RenameOp <OldName>, <NewName>
VB Example	<pre> oModule.RenameOp "SkinDepth1", "NewName" </pre>

15 - Analysis Setup Module Script Commands

Icepak analysis setup commands should be executed by the Analysis module, referred to in Icepak scripts as the "AnalysisSetup" module.

Set oModule = oDesign.GetModule("AnalysisSetup")

AddTwoWayCoupling

[ClearLinkedData](#)

[CopySetup](#)

CopySweep

[DeleteSetups](#)

DeleteTwoWayCoupling

[EditSetup](#)

EditTwoWayCoupling

[GetSetupCount](#)

[GetSetups](#)

[InsertSetup \(Icepak Steady State\)](#)

[InsertSetup \(Icepak Transient\)](#)

[PasteSetup](#)

[RenameSetup](#)

[RevertAllToInitial](#)

[RevertSetupToInitial](#)

ClearLinkedData (Module)

Clear the linked data of all the solution setups. Similar to the ClearLinkedData command for the design level.

Right-click menu of individual setups under the Analysis item.

UI Access	Project Manager > Design name > Analysis > right-click Setup name > Clear Linked Data		
Parameters	Name <i><SetupNameArray></i>	Type Array	Description 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

	<code><SetupName></code>	String	Name of the setup.
Return Value	None.		

Python Syntax	<code>CopySetup (<SetupName>)</code>
Python Example	<code>oModule.CopySetup ("OptimizationSetup1")</code>

VB Syntax	<code>CopySetup <SetupName></code>
VB Example	<code>oModule.CopySetup "OptimizationSetup1"</code>

DeleteSetups

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	<table border="1"> <tr> <td>Name</td> <td>Type</td> <td>Description</td> </tr> <tr> <td><code><SetupArray></code></td> <td>Array</td> <td>Array of solution setup names.</td> </tr> </table>			Name	Type	Description	<code><SetupArray></code>	Array	Array of solution setup names.
Name	Type	Description							
<code><SetupArray></code>	Array	Array of solution setup names.							
Return Value	None.								

Python Syntax	<code>DeleteSetups (<SetupArray>)</code>
Python Example	<code>oModule.DeleteSetups (["Setup1", "Setup2"])</code>

VB Syntax	DeleteSetups <SetupArray>
VB Example	<code>oModule.DeleteSetups Array("Setup1", "Setup2")</code>

EditSetup

Modifies an existing solution setup.

UI Access	Double-click a solution setup in the project tree to modify its settings.									
Parameters	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td><code><SetupName></code></td><td>String</td><td>Name of the solve setup being edited.</td></tr><tr><td><code><Attributes></code></td><td>Array</td><td>Structured array. <code>Array ("NAME:<NewSetupName>", <NamedParameters>)</code> See the <code>InsertSetup</code> command for details and examples.</td></tr></tbody></table>	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.
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.InsertSetup("HfssDrivenAuto",
[ "NAME:Setup1",
    "IsEnabled:=", True,
    "AutoSolverSetting:=", "Balanced",
    [ "NAME:Sweeps",
        [ "NAME:Sweep",
            "RangeType:=", "LinearStep",
            "RangeStart:=", "1GHz",
            "RangeEnd:=", "10GHz",
            "RangeStep:=", "1GHz"
        ]
    ]
]
```

```
        ],
        "SaveRadFieldsOnly:=", False,
        "SaveAnyFields:=", True,
        "Type:=", "Discrete"
    ])

oModule.InsertSetup("HfssDrivenAuto",
[
    "NAME:Setup2",
    "SolveType:=", "Auto",
    "IsEnabled:=", True,
    [
        "NAME:MeshLink",
        "ImportMesh:=", False
    ],
    "AutoSolverSetting:=", "Balanced",
    [
        "NAME:Sweeps",
        [
            "NAME:Sweep",
            "RangeType:=", "LinearCount",
```

```
        "RangeStart:=" , "0GHz",
        "RangeEnd:=" , "10GHz",
        "RangeCount:=" , 501
    ],
    "SaveRadFieldsOnly:=" , False,
    "SaveAnyFields:=" , True,
    "InfiniteSphereSetup:=" , "Infinite Sphere1",
    "ListsForFields:=" , ["Objectlist2"],
    "Type:=" , "Interpolating"
)

oModule.InsertSetup("HfssDriven",
["NAME:Setup3",
 "AdaptMultipleFreqs:=", False,
 "Frequency:=", "5GHz",
 "MaxDeltaS:=", 0.02,
 "PortsOnly:=", False,
 "UseMatrixConv:=", False,
 "MaximumPasses:=", 6,
```

```
"MinimumPasses:=", 1,  
"MinimumConvergedPasses:=", 1,  
"PercentRefinement:=", 30,  
"IsEnabled:=", True,  
"BasisOrder:=", 1,  
"DoLambdaRefine:=", True,  
"DoMaterialLambda:=", True,  
"SetLambdaTarget:=", False,  
"Target:=", 0.3333,  
"UseMaxTetIncrease:=", False,  
"PortAccuracy:=", 2,  
"UseABCOnPort:=", False,  
"SetPortMinMaxTri:=", False,  
"UseDomains:=", True,  
"UseIterativeSolver:=", False,  
"IterativeResidual:=", 1E-06,  
"DDMSolverResidual:=", 0.0001,  
"EnhancedLowFreqAccuracy:=", True,  
"SaveRadFieldsOnly:=", False,  
"SaveAnyFields:=", True,  
"IESolverType:=", "Auto",
```

```
    "LambdaTargetForIESolver:=", 0.15,
    "UseDefaultLambdaTgtForIESolver:=", True,
    "SkipPIERegionSolveDuringAdaptivePasses:=", True
    "RayDensityPerWavelength:=", 4,
    "MaxNumberOfBounces:=" , 5,
    "InfiniteSphereSetup:=", "Infinite Sphere1",
    "SkipSBRSSolveDuringAdaptivePasses:=", True,
    "PTDUTDSimulationSettings:=", "PTD Correction + UTD Rays",
    "PTDEdgeDensity:=" , 20
  ])
### Edit an SBR+ Setup with Fast Frequency Looping
oModule.EditSetup("HfssDriven",
  [
    "NAME:Setup1",
    "IsEnabled:=" , True,
    [
      "NAME:MeshLink",
      "ImportMesh:=" , False
    ],
    "IsSbrRangeDoppler:=" , False,
```

```
"RayDensityPerWavelength:=", 4,
"MaxNumberOfBounces:=", 5,
"IsMonostaticRCS:=", True,
"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",  
    "Postproc Sampling:="  , 500,  
    "Temperature:="        , "0cel",  
    "BuiltInGas:="         , "Helium"  
)
```

VB Syntax	EditSetup <SetupName>, <Attributes>
VB Example	<pre>oModule.EditSetup "Setup1", Array("NAME>NewSetup", "AdaptiveFreq:=", "1GHz", "EnableDistribProbTypeOption:=", false, "SaveFields:=", "true",</pre>

```
"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",  
        "MaxPass:=", 1,  
        "MinPass:=", 1,
```

```
        "MinConvPass:=", 1,
        "PerError:=", 1,
        "PerRefine:=", 30)),
Array("NAME:AC",
      "MaxPass:=", 10,
      "MinPass:=", 1,
      "MinConvPass:=", 2,
      "PerError:=", 1,
      "PerRefine:=", 30))
```

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	oModule.GetSetupCount ()

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>

InsertSetup (Icepak - Steady State)

Creates a solution setup in an Icepak design with a steady state solution type.

UI Access	Icepak > Analysis > Add Solution Setup		
Parameters	Name <code><NAME></code>	Type string	Description Solution setup name
	<code><Enabled></code>	bool	"True" to include setup in the analysis or "False" to exclude it

	<i><Flow Regime></i>	string	Laminar or Turbulent
	<i><Turbulent Model Eqn></i>	string	ZeroEquation, TwoEquation, RNG, EnhancedTwoEquation, SpalartAllmaras, EnhancedRNG, RealizableTwoEquation, EnhancedRealizableTwoEquation, or kOmegaSST
	<i><Include Temperature></i>	bool	"True" to include temperature calculation or "False" to exclude it
	<i><Include Flow></i>	bool	"True" to include the flow calculation or "False" to exclude it
	<i><Include Gravity></i>	bool	"True" to include the effect of gravity or "False" to exclude it
	<i><Solution Initialization - X Velocity></i>	string	X direction velocity initialization value and unit
	<i><Solution Initialization - Y Velocity></i>	string	Y direction velocity initialization value and unit
	<i><Solution Initialization - Z Velocity></i>	string	Z direction velocity initialization value and unit
	<i><Solution Initialization - Temperature></i>	string	AmbientTemp or temperature value and unit
	<i><Solution Initialization - Turbulent Kinetic Energy></i>	string	Turbulent kinetic energy initialization value and unit
	<i><Solution Initialization - Turbulent Dissipation Rate></i>	string	Turbulent dissipation rate initialization value and unit
	<i><Solution Initialization - Specific Dissipation Rate></i>	string	Specific dissipation rate initialization value and unit
	<i><Convergence Criteria - Flow></i>	string	Flow convergence value
	<i><Convergence Criteria - Energy></i>	string	Energy convergence value
	<i><Convergence Criteria - Turbulent Kinetic Energy></i>	string	Turbulent kinetic energy convergence value
	<i><Convergence Criteria - Turbulent Dissipation Rate></i>	string	Turbulent dissipation rate convergence value
	<i><Convergence Criteria - Specific Dissipation Rate></i>	string	Specific dissipation rate convergence value
	<i><Convergence Criteria - Discrete Ordinates></i>	string	Discrete ordinates convergence value
	<i><Convergence Criteria - Joule Heating></i>	string	Joule heating convergence value
	<i><IsEnabled></i>	bool	Not applicable to Icepak Solve Setup

	<i><Radiation Model></i>	string	Off, Discrete Ordinates Model, or Ray Tracing Model
	<i><Flow Iteration Per Radiation Iteration></i>	string	Number of flow iterations per radiation iteration
	<i><ThetaDivision></i>	string	Discrete Ordinates only: theta divisions value
	<i><PhiDivision></i>	string	Discrete Ordinates only: Phi divisions value
	<i><ThetaPixels></i>	string	Discrete Ordinates only: theta pixels value
	<i><PhiPixels></i>	string	Discrete Ordinates only: phi pixels value
	<i><Maximum Radiation Iteration></i>	string	Ray Tracing only: Maximum number of radiation iterations
	<i><Faces Per Surface Cluster></i>	string	Ray Tracing only: faces per surface cluster value
	<i><Resolution></i>	string	Ray Tracing only: resolution value
	<i><Under-relaxation - Pressure></i>	string	Pressure under-relaxation value
	<i><Under-relaxation - Momentum></i>	string	Momentum under-relaxation value
	<i><Under-relaxation - Temperature></i>	string	Temperature under-relaxation value
	<i><Under-relaxation - Turbulent Kinetic Energy></i>	string	Turbulent kinetic energy under-relaxation value
	<i><Under-relaxation - Turbulent Dissipation Rate></i>	string	Turbulent dissipation rate under-relaxation value
	<i><Under-relaxation - Specific Dissipation Rate></i>	string	Specific dissipation rate under-relaxation value
	<i><Under-relaxation - Joule Heating></i>	string	Joule heating under-relaxation value
	<i><Discretization Scheme - Pressure></i>	string	Standard, Second, Body Force, or PRESTO
	<i><Discretization Scheme - Momentum></i>	string	First or Second
	<i><Discretization Scheme - Temperature></i>	string	First or Second
	<i><Secondary Gradient></i>	bool	"True" to enable secondary gradient or "False" to disable it
	<i><Discretization Scheme - Turbulent Kinetic Energy></i>	string	First or Second
	<i><Discretization Scheme - Turbulent Dissipation Rate></i>	string	First or Second
	<i><Discretization Scheme - Specific Dissipation Rate></i>	string	First or Second
	<i><Discretization Scheme - Discrete Ordinates></i>	string	First or Second
	<i><Linear Solver Type - Pressure></i>	string	flex, V, W, or F

	<i><Linear Solver Type - Momentum></i>	string	flex, V, W, or F
	<i><Linear Solver Type - Temperature></i>	string	flex, V, W, or F
	<i><Linear Solver Type - Turbulent Kinetic Energy></i>	string	flex, V, W, or F
	<i><Linear Solver Type - Turbulent Dissipation Rate></i>	string	flex, V, W, or F
	<i><Linear Solver Type - Specific Dissipation Rate></i>	string	flex, V, W, or F
	<i><Linear Solver Type - Joule Heating></i>	string	flex, V, W, or F
	<i><Linear Solver Termination Criterion - Pressure></i>	string	Pressure termination criterion value
	<i><Linear Solver Termination Criterion - Momentum></i>	string	Momentum termination criterion value
	<i><Linear Solver Termination Criterion - Temperature></i>	string	Temperature termination criterion value
	<i><Linear Solver Termination Criterion - Turbulent Kinetic Energy></i>	string	Turbulent kinetic energy termination criterion value
	<i><Linear Solver Termination Criterion - Turbulent Dissipation Rate></i>	string	Turbulent dissipation rate termination criterion value
	<i><Linear Solver Termination Criterion - Specific Dissipation Rate></i>	string	Specific dissipation rate termination criterion value
	<i><Linear Solver Termination Criterion - Joule Heating></i>	string	Joule heating termination criterion value
	<i><Linear Solver Residual Reduction Tolerance - Pressure></i>	string	Pressure residual reduction tolerance value
	<i><Linear Solver Residual Reduction Tolerance - Momentum></i>	string	Momentum residual reduction tolerance value
	<i><Linear Solver Residual Reduction Tolerance - Temperature></i>	string	Temperature residual reduction tolerance value
	<i><Linear Solver Residual Reduction Tolerance - Turbulent Kinetic Energy></i>	string	Turbulent kinetic energy residual reduction tolerance value

	<i><Linear Solver Residual Reduction Tolerance - Turbulent Dissipation Rate></i>	string	Turbulent dissipation residual reduction tolerance value
	<i><Linear Solver Residual Reduction Tolerance - Specific Dissipation Rate></i>	string	Specific dissipation rate residual reduction tolerance value
	<i><Linear Solver Residual Reduction Tolerance - Joule Heating></i>	string	Joule heating residual reduction tolerance value
	<i><Linear Solver Stabilization - Pressure></i>	string	None or BCGSTAB
	<i><Linear Solver Stabilization - Temperature></i>	string	None or BCGSTAB
	<i><Linear Solver Stabilization - Joule Heating></i>	string	None or BCGSTAB
	<i><Coupled pressure-velocity formulation></i>	bool	True or False
	<i><2D Profile Interpolation Method></i>	string	Constant, Inverse Distance Weighted, or Least Squares
	<i><Frozen Flow Simulation></i>	bool	"False" to disable frozen flow, which is not available for steady-state simulations
	<i><Sequential Solve of Flow and Energy Equations></i>	bool	"True" to solve equations sequentially or "False" to solve simultaneously
	<i><Convergence Criteria - Max Iterations></i>	int	Maximum number of iterations
Return Value	None		

Python Syntax	InsertSetup (<NAME>, <Enabled>, <Flow Regime>, <Turbulent Model Eqn>, <Include Temperature>, <Include Flow>, <Include Gravity>, <Solution Initialization - X Velocity>, <Solution Initialization - Y Velocity>, <Solution Initialization - Z Velocity>, <Solution Initialization - Temperature>, <Solution Initialization - Turbulent Kinetic Energy>, <Solution Initialization - Turbulent Dissipation Rate>, <Solution Initialization - Specific Dissipation Rate>, <Convergence Criteria - Flow>, <Convergence Criteria - Energy>, <Convergence Criteria - Turbulent Kinetic Energy>, <Convergence Criteria - Turbulent Dissipation Rate>, <Convergence Criteria - Specific Dissipation Rate>, <Convergence Criteria - Discrete Ordinates>, <IsEnabled>, <Radiation Model>, <Flow Iteration Per Radiation Iteration>, <ThetaDivision>, <PhiDivision>, <ThetaPixels>, <PhiPixels>, <Under-relaxation - Pressure>, <Under-relaxation - Momentum>, <Under-relaxation - Temperature>, <Under-relaxation - Turbulent
----------------------	--

	<p><i>Kinetic Energy>, <Under-relaxation - Turbulent Dissipation Rate>, <Under-relaxation - Specific Dissipation Rate>, <Discretization Scheme - Pressure>, <Discretization Scheme - Momentum>, <Discretization Scheme - Temperature>, <Secondary Gradient>, <Discretization Scheme - Turbulent Kinetic Energy>, <Discretization Scheme - Turbulent Dissipation Rate>, <Discretization Scheme - Specific Dissipation Rate>, <Discretization Scheme - Discrete Ordinates>, <Linear Solver Type - Pressure>, <Linear Solver Type - Momentum>, <Linear Solver Type - Temperature>, <Linear Solver Type - Turbulent Kinetic Energy>, <Linear Solver Type - Turbulent Dissipation Rate>, <Linear Solver Type - Specific Dissipation Rate>, <Linear Solver Termination Criterion - Pressure>, <Linear Solver Termination Criterion - Momentum>, <Linear Solver Termination Criterion - Temperature>, <Linear Solver Termination Criterion - Turbulent Kinetic Energy>, <Linear Solver Termination Criterion - Turbulent Dissipation Rate>, <Linear Solver Termination Criterion - Specific Dissipation Rate>, <Linear Solver Residual Reduction Tolerance - Pressure>, <Linear Solver Residual Reduction Tolerance - Momentum>, <Linear Solver Residual Reduction Tolerance - Temperature>, <Linear Solver Residual Reduction Tolerance - Turbulent Kinetic Energy>, <Linear Solver Residual Reduction Tolerance - Turbulent Dissipation Rate>, <Linear Solver Residual Reduction Tolerance - Specific Dissipation Rate>, <Linear Solver Stabilization - Pressure>, <Linear Solver Stabilization - Temperature>, <Frozen Flow Simulation>, <Sequential Solve of Flow and Energy Equations>, <Convergence Criteria - Max Iterations>)</i></p>
Python Example	<pre> oModule.InsertSetup("IcepakSteadyState", ["NAME:Setup1", "Enabled:=" , True, "Flow Regime:=" , "Turbulent", "Turbulent Model Eqn:=" , "ZeroEquation", "Include Temperature:=" , True, "Include Flow:=" , True, "Include Gravity:=" , False, "Solution Initialization - X Velocity:=" , "-0.01m_per_sec", "Solution Initialization - Y Velocity:=" , "0m_per_sec", "Solution Initialization - Z Velocity:=" , "0m_per_sec",]) </pre>

```
"Solution Initialization - Temperature:=", "AmbientTemp",
"Solution Initialization - Turbulent Kinetic Energy:=", "1m2_per_s2",
"Solution Initialization - Turbulent Dissipation Rate:=", "1m2_per_s3",
"Solution Initialization - Specific Dissipation Rate:=", "1diss_per_s",
"Convergence Criteria - Flow:=", "0.001",
"Convergence Criteria - Energy:=", "1e-07",
"Convergence Criteria - Turbulent Kinetic Energy:=", "0.001",
"Convergence Criteria - Turbulent Dissipation Rate:=", "0.001",
"Convergence Criteria - Specific Dissipation Rate:=", "0.001",
"Convergence Criteria - Discrete Ordinates:=", "1e-06",
"Convergence Criteria - Joule Heating:=", "1e-07",
".IsEnabled:=" , False,
"Radiation Model:=" , "Discrete Ordinates Model",
"Flow Iteration Per Radiation Iteration:=", "3",
"ThetaDivision:=" , "1",
"PhiDivision:=" , "1",
"ThetaPixels:=" , "1",
"PhiPixels:=" , "1",
"Under-relaxation - Pressure:=", "0.3",
"Under-relaxation - Momentum:=", "0.7",
"Under-relaxation - Temperature:=", "1",
"Under-relaxation - Turbulent Kinetic Energy:=", "0.8",
"Under-relaxation - Turbulent Dissipation Rate:=", "0.8",
"Under-relaxation - Specific Dissipation Rate:=", "0.8",
"Under-relaxation - Joule Heating:=", "1",
"Discretization Scheme - Pressure:=", "Standard",
"Discretization Scheme - Momentum:=", "First",
"Discretization Scheme - Temperature:=", "First",
"Secondary Gradient:=" , False,
"Discretization Scheme - Turbulent Kinetic Energy:=", "First",
"Discretization Scheme - Turbulent Dissipation Rate:=", "First",
"Discretization Scheme - Specific Dissipation Rate:=", "First",
"Discretization Scheme - Discrete Ordinates:=", "First",
```

```
"Linear Solver Termination Criterion - Momentum:=", "0.1",
"Linear Solver Termination Criterion - Temperature:=", "0.1",
"Linear Solver Termination Criterion -
Turbulent Kinetic Energy:=", "0.1",
"Linear Solver Termination Criterion -
Turbulent Dissipation Rate:=", "0.1",
"Linear Solver Termination Criterion -
Specific Dissipation Rate:=", "0.1",
"Linear Solver Termination Criterion - Joule Heating:=", "0.1",
"Linear Solver Residual Reduction Tolerance - Pressure:=", "0.1",
"Linear Solver Residual Reduction Tolerance - Momentum:=", "0.1",
"Linear Solver Residual Reduction Tolerance - Temperature:=", "0.1",
"Linear Solver Residual Reduction Tolerance -
Turbulent Kinetic Energy:=", "0.1",
"Linear Solver Residual Reduction Tolerance -
Turbulent Dissipation Rate:=", "0.1",
"Linear Solver Residual Reduction Tolerance -
Specific Dissipation Rate:=", "0.1",
"Linear Solver Residual Reduction Tolerance - Joule Heating:=", "0.1",
"Linear Solver Stabilization - Pressure:=", "None",
"Linear Solver Stabilization - Temperature:=", "None",
"Linear Solver Stabilization - Joule Heating:=", "None",
"Coupled pressure-velocity formulation:=", False,
"2D Profile Interpolation Method:=", "Inverse Distance Weighted",
"Frozen Flow Simulation:=", False,
"Sequential Solve of Flow and Energy Equations:=", True,
"Convergence Criteria - Max Iterations:=", 100
])
```

VB Syntax	<pre>InsertSetup (<NAME>, <Enabled>, <Flow Regime>, <Turbulent Model Eqn>, <Include Temperature>, <Include Flow>, <Include Gravity>, <Solution Initialization - X Velocity>, <Solution Initialization - Y Velocity>, <Solution Initialization - Z Velocity>, <Solution Initialization - Temperature>, <Solution Initialization - Turbulent Kinetic Energy>, <Solution Initialization - Turbulent Dissipation Rate>, <Solution Initialization - Specific Dissipation Rate>, <Convergence Criteria - Flow>, <Convergence Criteria - Energy>, <Convergence Criteria - Turbulent Kinetic Energy>, <Convergence Criteria - Turbulent Dissipation Rate>, <Convergence Criteria - Specific Dissipation Rate>, <Convergence Criteria - Discrete Ordinates>, <IsEnabled>, <Radiation Model>, <Flow Iteration Per Radiation Iteration>, <ThetaDivision>, <PhiDivision>, <ThetaPixels>, <PhiPixels>, <Under-relaxation - Pressure>, <Under-relaxation - Momentum>, <Under-relaxation - Temperature>, <Under-relaxation - Turbulent Kinetic Energy>, <Under-relaxation - Turbulent Dissipation Rate>, <Under-relaxation - Specific Dissipation Rate>, <Discretization Scheme - Pressure>, <Discretization Scheme - Momentum>, <Discretization Scheme - Temperature>, <Secondary Gradient>, <Discretization Scheme - Turbulent Kinetic Energy>, <Discretization Scheme - Turbulent Dissipation Rate>, <Discretization Scheme - Specific Dissipation Rate>, <Discretization Scheme - Discrete Ordinates>, <Linear Solver Type - Pressure>, <Linear Solver Type - Momentum>, <Linear Solver Type - Temperature>, <Linear Solver Type - Turbulent Kinetic Energy>, <Linear Solver Type - Turbulent Dissipation Rate>, <Linear Solver Type - Specific Dissipation Rate>, <Linear Solver Termination Criterion - Pressure>, <Linear Solver Termination Criterion - Momentum>, <Linear Solver Termination Criterion - Temperature>, <Linear Solver Termination Criterion - Turbulent Kinetic Energy>, <Linear Solver Termination Criterion - Turbulent Dissipation Rate>, <Linear Solver Termination Criterion - Specific Dissipation Rate>, <Linear Solver Residual Reduction Tolerance - Pressure>, <Linear Solver Residual Reduction Tolerance - Momentum>, <Linear Solver Residual Reduction Tolerance - Temperature>, <Linear Solver Residual Reduction Tolerance - Turbulent Kinetic Energy>, <Linear Solver Residual Reduction Tolerance - Turbulent Dissipation Rate>, <Linear Solver Residual Reduction Tolerance - Specific Dissipation Rate>, <Linear Solver Stabilization - Pressure>, <Linear Solver Stabilization - Temperature>, <Frozen Flow Simulation>, <Sequential Solve of Flow and Energy Equations>, <Convergence Criteria - Max Iterations>)</pre>
VB Example	<pre>oModule.InsertSetup "IcepakSteadyState", Array("NAME:Setup1", "Enabled:=", true, "Flow Regime:=", "Turbulent", "Turbulent Model Eqn:=", "ZeroEquation", "Include Temperature:=", _true, "Include Flow:=", true, "Include Gravity:=", false, "Solution Initialization - X Velocity:=", "0m_per_sec", "Solution Initialization - Y Velocity:=", "0m_per_sec", "Solution Initialization - Z Velocity:=",</pre>

```
_ "0m_per_sec", "Solution Initialization - Temperature:=", "AmbientTemp", "Solution Initialization -  
Turbulent Kinetic Energy:=", _ "1m2_per_s2", "Solution Initialization - Turbulent Dissipation Rate:=", _  
"1m2_per_s3", "Solution Initialization - Specific Dissipation Rate:=", _ "1diss_per_s",  
"Convergence  
Criteria - Flow:=", "0.001", "Convergence Criteria - Energy:=", _ "1e-07", "Convergence Criteria -  
Turbulent Kinetic Energy:=", "0.001", "Convergence Criteria - Turbulent Dissipation Rate:=", _ "0.001",  
"Convergence Criteria - Specific Dissipation Rate:=", "0.001", "Convergence Criteria - Discrete Ordinates:=",  
_ "1e-06", "IsEnabled:=", false, "Radiation Model:=", "Discrete Ordinates Model", "Flow Iteration Per  
Radiation Iteration:=", _ "3", "ThetaDivision:=", "1", "PhiDivision:=", "1", "ThetaPixels:=", "1",  
"PhiPixels:=", _ "1", "Under-relaxation - Pressure:=", "0.3", "Under-relaxation - Momentum:=", _ "0.7",  
"Under-relaxation - Temperature:=", "1", "Under-relaxation - Turbulent Kinetic Energy:=", _ "0.8",  
"Under-relaxation - Turbulent Dissipation Rate:=", "0.8", "Under-relaxation - Specific Dissipation Rate:=",  
_ "0.8", "Discretization Scheme - Pressure:=", "Standard", "Discretization Scheme - Momentum:=", _ "First",  
"Discretization Scheme - Temperature:=", "First", "Secondary Gradient:=", _ false, "Dis-
```

```
cretization Scheme - 

Turbulent Kinetic Energy:=", "First", "Discretization Scheme - Turbulent Dissipation
Rate:=", _"First",

"Discretization Scheme - Specific Dissipation Rate:=", "First", "Discretization Scheme -
Discrete Ordinates:=", _
"First", "Linear Solver Type - Pressure:=", "V", "Linear Solver Type - Momentum:=", _
"flex", "Linear Solver Type -

Temperature:=", "F", "Linear Solver Type - Turbulent Kinetic Energy:=", _"flex", "Lin-
ear Solver Type - Turbulent

Dissipation Rate:=", "flex", "Linear Solver Type - Specific Dissipation Rate:=", _
"flex", "Linear Solver

Termination Criterion - Pressure:=", "0.1", "Linear Solver Termination Criterion -
Momentum:=", _"0.1", "Linear

Solver Termination Criterion - Temperature:=", "0.1", "Linear Solver Termination Cri-
terion - Turbulent Kinetic

Energy:=", _"0.1", "Linear Solver Termination Criterion - Turbulent Dissipation
Rate:=", _"0.1", "Linear Solver

Termination Criterion - Specific Dissipation Rate:=", _"0.1", "Linear Solver Residual
Reduction Tolerance -

Pressure:=", "0.1", "Linear Solver Residual Reduction Tolerance - Momentum:=", _"0.1",
"Linear Solver Residual

Reduction Tolerance - Temperature:=", "0.1", "Linear Solver Residual Reduction Tolerance
- Turbulent Kinetic

Energy:=", _"0.1", "Linear Solver Residual Reduction Tolerance - Turbulent Dissipation
Rate:=", _"0.1",

"Linear Solver Residual Reduction Tolerance - Specific Dissipation Rate:=", _"0.1",
```

```

    "Linear Solver Stabilization - Pressure:=", "None", "Linear Solver Stabilization - Temperature:=", _"None", "Frozen Flow Simulation:=", false,
    "Sequential Solve of Flow and Energy Equations:=", _true, "Convergence Criteria - Max Iterations:=", 100)

```

InsertSetup (Icepak - Transient)

Creates a solution setup in an Icepak design with a transient solution type.

UI Access	Icepak > Analysis > Add Solution Setup		
Parameters	Name <NAME>	Type string	Description Solution setup name
	<Enabled>	bool	"True" to include setup in the analysis or "False" to exclude it
	<Flow Regime>	string	Laminar or Turbulent
	<Turbulent Model Eqn>	string	ZeroEquation, TwoEquation, RNG, EnhancedTwoEquation, SpalartAllmaras, EnhancedRNG, RealizableTwoEquation, EnhancedRealizableTwoEquation, or kOmegaSST
	<Include Temperature>	bool	"True" to include temperature calculation or "False" to exclude it
	<Include Flow>	bool	"True" to include the flow calculation or "False" to exclude it
	<Include Gravity>	bool	"True" to include the effect of gravity or "False" to exclude it
	<Solution Initialization - X Velocity>	string	X direction velocity initialization value and unit
	<Solution Initialization - Y Velocity>	string	Y direction velocity initialization value and unit
	<Solution Initialization - Z Velocity>	string	Z direction velocity initialization value and unit

	<i><Solution Initialization - Temperature></i>	string	AmbientTemp or temperature value and unit
	<i><Solution Initialization - Turbulent Kinetic Energy></i>	string	Turbulent kinetic energy initialization value and unit
	<i><Solution Initialization - Turbulent Dissipation Rate></i>	string	Turbulent dissipation rate initialization value and unit
	<i><Solution Initialization - Specific Dissipation Rate></i>	string	Specific dissipation rate initialization value and unit
	<i><Convergence Criteria - Flow></i>	string	Flow convergence value
	<i><Convergence Criteria - Energy></i>	string	Energy convergence value
	<i><Convergence Criteria - Turbulent Kinetic Energy></i>	string	Turbulent kinetic energy convergence value
	<i><Convergence Criteria - Turbulent Dissipation Rate></i>	string	Turbulent dissipation rate convergence value
	<i><Convergence Criteria - Specific Dissipation Rate></i>	string	Specific dissipation rate convergence value
	<i><Convergence Criteria - Discrete Ordinates></i>	string	Discrete ordinates convergence value
	<i><Convergence Criteria - Joule Heating></i>	string	Joule heating convergence value
	<i><IsEnabled></i>	bool	Not applicable to Icepak Solve Setup
	<i><Radiation Model></i>	string	Off, Discrete Ordinates Model, or Ray Tracing Model
	<i><Flow Iteration Per Radiation Iteration></i>	string	Number of flow iterations per radiation iteration
	<i><ThetaDivision></i>	string	Discrete Ordinates only: theta divisions value
	<i><PhiDivision></i>	string	Discrete Ordinates only: Phi divisions value
	<i><ThetaPixels></i>	string	Discrete Ordinates only: theta pixels value
	<i><PhiPixels></i>	string	Discrete Ordinates only: phi pixels value
	<i><Maximum Radiation Iteration></i>	string	Ray Tracing only: Maximum number of radiation iterations
	<i><Faces Per Surface Cluster></i>	string	Ray Tracing only: faces per surface cluster value
	<i><Resolution></i>	string	Ray Tracing only: resolution value
	<i><Under-relaxation - Pressure></i>	string	Pressure under-relaxation value
	<i><Under-relaxation - Momentum></i>	string	Momentum under-relaxation value
	<i><Under-relaxation - Temperature></i>	string	Temperature under-relaxation value
	<i><Under-relaxation - Turbulent Kinetic Energy></i>	string	Turbulent kinetic energy under-relaxation value
	<i><Under-relaxation - Turbulent Dissipation Rate></i>	string	Turbulent dissipation rate under-relaxation value

	<i><Under-relaxation - Specific Dissipation Rate></i>	string	Specific dissipation rate under-relaxation value
	<i><Under-relaxation - Joule Heating></i>	string	Joule heating under-relaxation value
	<i><Discretization Scheme - Pressure></i>	string	Standard, Second, Body Force, or PRESTO
	<i><Discretization Scheme - Momentum></i>	string	First or Second
	<i><Discretization Scheme - Temperature></i>	string	First or Second
	<i><Secondary Gradient></i>	bool	"True" to enable secondary gradient or "False" to disable it
	<i><Discretization Scheme - Turbulent Kinetic Energy></i>	string	First or Second
	<i><Discretization Scheme - Turbulent Dissipation Rate></i>	string	First or Second
	<i><Discretization Scheme - Specific Dissipation Rate></i>	string	First or Second
	<i><Discretization Scheme - Discrete Ordinates></i>	string	First or Second
	<i><Linear Solver Type - Pressure></i>	string	flex, V, W, or F
	<i><Linear Solver Type - Momentum></i>	string	flex, V, W, or F
	<i><Linear Solver Type - Temperature></i>	string	flex, V, W, or F
	<i><Linear Solver Type - Turbulent Kinetic Energy></i>	string	flex, V, W, or F
	<i><Linear Solver Type - Turbulent Dissipation Rate></i>	string	flex, V, W, or F
	<i><Linear Solver Type - Specific Dissipation Rate></i>	string	flex, V, W, or F
	<i><Linear Solver Type - Joule Heating></i>	string	flex, V, W, or F
	<i><Linear Solver Termination Criterion - Pressure></i>	string	Pressure termination criterion value
	<i><Linear Solver Termination Criterion - Momentum></i>	string	Momentum termination criterion value
	<i><Linear Solver Termination Criterion - Temperature></i>	string	Temperature termination criterion value
	<i><Linear Solver Termination Criterion - Turbulent Kinetic Energy></i>	string	Turbulent kinetic energy termination criterion value
	<i><Linear Solver Termination Criterion - Turbulent Dissipation Rate></i>	string	Turbulent dissipation rate termination criterion value
	<i><Linear Solver Termination Criterion - Specific Dis-</i>	string	Specific dissipation rate termination criterion value

	<i>sipation Rate></i>		
	<i><Linear Solver Termination Criterion - Joule Heating></i>	string	Joule heating termination criterion value
	<i><Linear Solver Residual Reduction Tolerance - Pressure></i>	string	Pressure residual reduction tolerance value
	<i><Linear Solver Residual Reduction Tolerance - Momentum></i>	string	Momentum residual reduction tolerance value
	<i><Linear Solver Residual Reduction Tolerance - Temperature></i>	string	Temperature residual reduction tolerance value
	<i><Linear Solver Residual Reduction Tolerance - Turbulent Kinetic Energy></i>	string	Turbulent kinetic energy residual reduction tolerance value
	<i><Linear Solver Residual Reduction Tolerance - Turbulent Dissipation Rate></i>	string	Turbulent dissipation residual reduction tolerance value
	<i><Linear Solver Residual Reduction Tolerance - Specific Dissipation Rate></i>	string	Specific dissipation rate residual reduction tolerance value
	<i><Linear Solver Residual Reduction Tolerance - Joule Heating></i>	string	Joule heating residual reduction tolerance value
	<i><Linear Solver Stabilization - Pressure></i>	string	None or BCGSTAB
	<i><Linear Solver Stabilization - Temperature></i>	string	None or BCGSTAB
	<i><Linear Solver Stabilization - Joule Heating></i>	string	None or BCGSTAB
	<i><Coupled pressure-velocity formulation></i>	bool	True or False
	<i><2D Profile Interpolation Method></i>	string	Constant, Inverse Distance Weighted, or Least Squares
	<i><Frozen Flow Simulation></i>	bool	"True" to enable frozen flow or "False" to disable it
	<i><Start Time></i>	bool	Start time value and unit
	<i><Stop Time></i>	string	Stop time value and unit
	<i><Time Step></i>	string	Time step duration value and unit
	<i><Iterations per Time Step></i>	int	Number of iterations per time step
	<i><Import Start Time></i>	bool	"True" to enable import start time or "False" to disable it
	<i><SaveFieldsType></i>	string	None or Every N Steps
	<i><N Steps></i>	string	Number of time steps

<code><NAME></code>	string	SweepRanges
<code><NAME></code>	string	Subrange
<code><RangeType></code>	string	LinearStep
<code><RangeStart></code>	string	Range start time and unit
<code><RangeEnd></code>	string	Range end time and unit
<code><RangeStep></code>	string	Time step duration and unit
<code><NAME></code>	string	Time Step Variation Data
<code><Variation Type></code>	string	Transient
Time Step Variation = Linear		
<code><Variation Function></code>	string	Linear
<code><Variation Value></code>	list	Comma-separated values and units: (S0, a)
Time Step Variation = Square Wave		
<code><Variation Function></code>	string	Square Wave
<code><Variation Value></code>	list	Comma-separated values and units: (S0, Phase, On Time, Off Time, Off Value)
Time Step Variation = Piecewise Constant		
<code><Variation Function></code>	string	Piecewise Constant
<code><Variation Value></code>	list	S0 value and unit and dataset name (e.g., ["1s\\"", \"pwl(ds1,Time)\"])
Time Step Variation = Piecewise Linear		
<code><Variation Function></code>	string	Piecewise Linear
<code><Variation Value></code>	list	S0 value and unit and dataset name (e.g., ["1s\\"", \"pwl(ds1,Time)\"])
Time Step Variation =Automatic		
<code><Variation Function></code>		Automatic
<code><Variation Value></code>		Comma-separated values and units: (No of Fixed Time Steps, Min Time step size, Max Time step size, Min Step Change Factor, Max Step Change Factor, Error Tolerance)

Return Value	None
---------------------	------

Python Syntax	<code>InsertSetup (<NAME>, <Enabled>, <Flow Regime>, <Turbulent Model Eqn>, <Include Temperature>, <Include Flow>, <Include Gravity>, <Solution Initialization - X Velocity>, <Solution Initialization - Y Velocity>, <Solution Initialization - Z Velocity>, <Solution Initialization - Temperature>, <Solution Initialization - Turbulent Kinetic Energy>, <Solution Initialization - Turbulent Dissipation Rate>, <Solution Initialization - Specific Dissipation Rate>, <Convergence Criteria - Flow>, <Convergence Criteria - Energy>, <Convergence Criteria - Turbulent Kinetic Energy>, <Convergence Criteria - Turbulent Dissipation Rate>, <Convergence Criteria - Specific Dissipation Rate>, <Convergence Criteria - Discrete Ordinates>, <IsEnabled>, <Radiation Model>, <Flow Iteration Per Radiation Iteration>, <ThetaDivision>, <PhiDivision>, <ThetaPixels>, <PhiPixels>, <Under-relaxation - Pressure>, <Under-relaxation - Momentum>, <Under-relaxation - Temperature>, <Under-relaxation - Turbulent Kinetic Energy>, <Under-relaxation - Turbulent Dissipation Rate>, <Under-relaxation - Specific Dissipation Rate>, <Discretization Scheme - Pressure>, <Discretization Scheme - Momentum>, <Discretization Scheme - Temperature>, <Secondary Gradient>, <Discretization Scheme - Turbulent Kinetic Energy>, <Discretization Scheme - Turbulent Dissipation Rate>, <Discretization Scheme - Specific Dissipation Rate>, <Discretization Scheme - Discrete Ordinates>, <Linear Solver Type - Pressure>, <Linear Solver Type - Momentum>, <Linear Solver Type - Temperature>, <Linear Solver Type - Turbulent Kinetic Energy>, <Linear Solver Type - Turbulent Dissipation Rate>, <Linear Solver Type - Specific Dissipation Rate>, <Linear Solver Termination Criterion - Pressure>, <Linear Solver Termination Criterion - Momentum>, <Linear Solver Termination Criterion - Temperature>, <Linear Solver Termination Criterion - Turbulent Kinetic Energy>, <Linear Solver Termination Criterion - Turbulent Dissipation Rate>, <Linear Solver Termination Criterion - Specific Dissipation Rate>, <Linear Solver Residual Reduction Tolerance - Pressure>, <Linear Solver Residual Reduction Tolerance - Momentum>, <Linear Solver Residual Reduction Tolerance - Temperature>, <Linear Solver Residual Reduction Tolerance - Turbulent Kinetic Energy>, <Linear Solver Residual Reduction Tolerance - Turbulent Dissipation Rate>, <Linear Solver Residual Reduction Tolerance - Specific Dissipation Rate>, <Linear Solver Stabilization - Pressure>, <Linear Solver Stabilization - Temperature>, <Frozen Flow Simulation>, <Sequential Solve of Flow and Energy Equations>, <Convergence Criteria - Max Iterations>)</code>
----------------------	--

Python Example

```
oModule.InsertSetup("IcepakTransient",
[
    "NAME:Setup1",
    "Enabled:=" , True,
    "Flow Regime:=" , "Laminar",
    "Include Temperature:=" , True,
    "Include Flow:=" , True,
    "Include Gravity:=" , False,
    "Solution Initialization - X Velocity:=", "0m_per_sec",
    "Solution Initialization - Y Velocity:=", "0m_per_sec",
    "Solution Initialization - Z Velocity:=", "0m_per_sec",
    "Solution Initialization - Temperature:=", "AmbientTemp",
    "Solution Initialization - Turbulent Kinetic Energy:=", "1m2_per_s2",
    "Solution Initialization - Turbulent Dissipation Rate:=", "1m2_per_s3",
    "Solution Initialization - Specific Dissipation Rate:=", "1diss_per_s",
    "Convergence Criteria - Flow:=", "0.001",
    "Convergence Criteria - Energy:=", "1e-07",
    "Convergence Criteria - Turbulent Kinetic Energy:=", "0.001",
    "Convergence Criteria - Turbulent Dissipation Rate:=", "0.001",
    "Convergence Criteria - Specific Dissipation Rate:=", "0.001",
    "Convergence Criteria - Discrete Ordinates:=", "1e-06",
    "Convergence Criteria - Joule Heating:=", "1e-07",
    "IsEnabled:=" , False,
    "Radiation Model:=" , "Off",
    "Under-relaxation - Pressure:=", "0.3",
    "Under-relaxation - Momentum:=", "0.7",
    "Under-relaxation - Temperature:=", "1",
    "Under-relaxation - Turbulent Kinetic Energy:=", "0.8",
    "Under-relaxation - Turbulent Dissipation Rate:=", "0.8",
```

```
"Under-relaxation - Specific Dissipation Rate:=", "0.8",
"Under-relaxation - Joule Heating:=", "1",
"Discretization Scheme - Pressure:=", "Standard",
"Discretization Scheme - Momentum:=", "First",
"Discretization Scheme - Temperature:=", "First",
"Secondary Gradient:=" , False,
"Discretization Scheme - Turbulent Kinetic Energy:=", "First",
"Discretization Scheme - Turbulent Dissipation Rate:=", "First",
"Discretization Scheme - Specific Dissipation Rate:=", "First",
"Discretization Scheme - Discrete Ordinates:=", "First",
"Linear Solver Type - Pressure:=", "V",
"Linear Solver Type - Momentum:=", "flex",
"Linear Solver Type - Temperature:=", "F",
"Linear Solver Type - Turbulent Kinetic Energy:=", "flex",
"Linear Solver Type - Turbulent Dissipation Rate:=", "flex",
"Linear Solver Type - Specific Dissipation Rate:=", "flex",
"Linear Solver Type - Joule Heating:=", "flex",
"Linear Solver Termination Criterion - Pressure:=", "0.1",
"Linear Solver Termination Criterion - Momentum:=", "0.1",
"Linear Solver Termination Criterion - Temperature:=", "0.1",
"Linear Solver Termination Criterion -
Turbulent Kinetic Energy:=", "0.1",
"Linear Solver Termination Criterion -
Turbulent Dissipation Rate:=", "0.1",
"Linear Solver Termination Criterion -
Specific Dissipation Rate:=", "0.1",
"Linear Solver Termination Criterion - Joule Heating:=", "0.1",
"Linear Solver Residual Reduction Tolerance - Pressure:=", "0.1",
"Linear Solver Residual Reduction Tolerance - Momentum:=", "0.1",
"Linear Solver Residual Reduction Tolerance - Temperature:=", "0.1",
"Linear Solver Residual Reduction Tolerance -
Turbulent Kinetic Energy:=", "0.1",
"Linear Solver Residual Reduction Tolerance -
```

VB Syntax	InsertSetup (<NAME>, <Enabled>, <Flow Regime>, <Turbulent Model Eqn>, <Include Temperature>, <Include Flow>, <Include Gravity>, <Solution Initialization - X Velocity>, <Solution Initialization - Y Velocity>, <Solution Initialization - Z Velocity>, <Solution Initialization - Temperature>, <Solution Initialization - Turbulent Kinetic Energy>, <Solution Initialization - Turbulent Dissipation Rate>, <Solution Initialization - Specific Dissipation Rate>, <Convergence Criteria - Flow>, <Convergence Criteria - Energy>, <Convergence Criteria - Turbulent Kinetic Energy>, <Convergence Criteria - Turbulent Dissipation Rate>, <Convergence Criteria - Specific Dissipation Rate>, <Convergence Criteria - Discrete Ordinates>, <IsEnabled>, <Radiation Model>, <Flow Iteration Per Radiation Iteration>, <ThetaDivision>, <PhiDivision>, <ThetaPixels>, <PhiPixels>, <Under-relaxation - Pressure>, <Under-relaxation - Momentum>, <Under-relaxation - Temperature>, <Under-relaxation - Turbulent Kinetic Energy>, <Under-relaxation - Turbulent Dissipation Rate>, <Under-relaxation - Specific Dissipation Rate>, <Discretization Scheme - Pressure>, <Discretization Scheme - Momentum>, <Discretization Scheme - Temperature>, <Secondary Gradient>, <Discretization Scheme - Turbulent Kinetic Energy>, <Discretization Scheme - Turbulent Dissipation Rate>, <Discretization Scheme - Specific Dissipation Rate>, <Discretization Scheme - Discrete Ordinates>, <Linear Solver Type - Pressure>, <Linear Solver Type - Momentum>, <Linear Solver Type - Temperature>, <Linear Solver Type - Turbulent Kinetic Energy>, <Linear Solver Type - Turbulent Dissipation Rate>, <Linear Solver Type - Specific Dissipation Rate>, <Linear Solver Termination Criterion - Pressure>, <Linear Solver Termination Criterion - Momentum>, <Linear Solver Termination Criterion - Temperature>, <Linear Solver Termination Criterion - Turbulent Kinetic Energy>, <Linear Solver Termination Criterion - Turbulent Dissipation Rate>, <Linear Solver Termination Criterion - Specific Dissipation Rate>, <Linear Solver Residual Reduction Tolerance - Pressure>, <Linear Solver Residual Reduction Tolerance - Momentum>, <Linear Solver Residual Reduction Tolerance - Temperature>, <Linear Solver Residual Reduction Tolerance - Turbulent Kinetic Energy>, <Linear Solver Residual Reduction Tolerance - Turbulent Dissipation Rate>, <Linear Solver Residual Reduction Tolerance - Specific Dissipation Rate>, <Linear Solver Stabilization - Pressure>, <Linear Solver Stabilization - Temperature>, <Frozen Flow Simulation>, <Sequential Solve of Flow and Energy Equations>, <Convergence Criteria - Max Iterations>)
VB Example	<pre> oModule.InsertSetup "IcepakTransient", Array("NAME:Setup1", "Enabled:=", true, "Flow Regime:=", "Laminar", "Include Temperature:=", true, "Include Flow:=", true, "Include Gravity:=", false, "Solution Initialization - X Velocity:=",</pre>

```
"0m_per_sec", "Solution Initialization - Y Velocity:=", _"0m_per_sec", "Solution  
Initialization - Z Velocity:=", "0m_per_sec", "Solution Initialization - Tem-  
perature:=",  
    "AmbientTemp", "Solution Initialization - Turbulent Kinetic Energy:=", _"1m2_per_  
s2",  
    "Solution Initialization - Turbulent Dissipation Rate:=", _"1m2_per_s3", "Solution  
Initialization - Specific Dissipation Rate:=", _"1diss_per_s", "Convergence Criteria  
- Flow:=", "0.001", "Convergence Criteria - Energy:=", _"1e-07", "Convergence Cri-  
teria  
    - Turbulent Kinetic Energy:=", "0.001", "Convergence Criteria - Turbulent Dissipation  
Rate:=",  
        _"0.001", "Convergence Criteria - Specific Dissipation Rate:=", "0.001", "Convergence  
Criteria  
    - Discrete Ordinates:=", _"1e-06", "IsEnabled:=", false, "Radiation Model:=", "Off",  
"Under-relaxation - Pressure:=", _"0.3", "Under-relaxation - Momentum:=", "0.7",  
"Under-relaxation  
    - Temperature:=", _"1", "Under-relaxation - Turbulent Kinetic Energy:=", "0.8",  
"Under-relaxation -  
    Turbulent Dissipation Rate:=", _"0.8", "Under-relaxation - Specific Dissipation  
Rate:=", "0.8",  
"Discretization Scheme - Pressure:=", _"Standard", "Discretization Scheme -  
Momentum:=", "First",  
"Discretization Scheme - Temperature:=", _"First", "Secondary Gradient:=", false, "Dis-  
cretization  
Scheme - Turbulent Kinetic Energy:=", _"First", "Discretization Scheme - Turbulent Dis-  
sipation Rate:=",
```

```
"First", "Discretization Scheme - Specific Dissipation Rate:=", _"First", "Dis-
cretization Scheme - 

Discrete Ordinates:=", "First", "Linear Solver Type - Pressure:=", _"V", "Linear
Solver Type - 

Momentum:=", "flex", "Linear Solver Type - Temperature:=", _"F", "Linear Solver Type - 
Turbulent

Kinetic Energy:=", "flex", "Linear Solver Type - Turbulent Dissipation Rate:=", _ 
"flex", "Linear Solver

Type - Specific Dissipation Rate:=", "flex", "Linear Solver Termination Criterion - 
Pressure:=", _

"0.1", "Linear Solver Termination Criterion - Momentum:=", "0.1", "Linear Solver Ter-
mination Criterion

- Temperature:=", _"0.1", "Linear Solver Termination Criterion - Turbulent Kinetic
Energy:=", "0.1",

"Linear Solver Termination Criterion - Turbulent Dissipation Rate:=", _"0.1", "Linear
Solver Termination

Criterion - Specific Dissipation Rate:=", _"0.1", "Linear Solver Residual Reduction
Tolerance - 

Pressure:=", "0.1", "Linear Solver Residual Reduction Tolerance - Momentum:=", _ 
"0.1", "Linear Solver

Residual Reduction Tolerance - Temperature:=", "0.1", "Linear Solver Residual Reduc-
tion Tolerance - 

Turbulent Kinetic Energy:=", _"0.1", "Linear Solver Residual Reduction Tolerance - 
Turbulent Dissipation

Rate:=", _"0.1", "Linear Solver Residual Reduction Tolerance - Specific Dissipation
```

```

Rate:=", _"0.1",
      "Linear Solver Stabilization - Pressure:=", "None", "Linear Solver Stabilization - Temperature:=", _
      "None", "Frozen Flow Simulation:=", false, "Start Time:=", "0s", "Stop Time:=", _"20s", "Time Step:=",
      "1s", "Iterations per Time Step:=", 20, "Import Start Time:=", _false, "SaveField-
      sType:=", "Every N
      Steps", "N Steps:=", "10s", Array("NAME:SweepRanges", Array("NAME:Subrange",
      "RangeType:=", _"LinearStep",
      "RangeStart:=", "0s", "RangeEnd:=", "20s", "RangeStep:=", "10s")), Array("NAME:Time
      Step Variation Data",
      "Variation Type:=", "Transient", "Variation Function:=", "Linear", "Variation
      Value:=", "[0.01s, 1]"))

```

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	oModule.RenameSetup ("Setup1", "MySetup")

VB Syntax	RenameSetup <OldSetupName>, <NewSetupName>
VB Example	oModule.RenameSetup "Setup1", "MySetup"

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	<code>oModule.RevertAllToInitial()</code>
-----------------------	---

VB Syntax	<code>RevertAllToInitial</code>
VB Example	<code>oModule.RevertAllToInitial</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 <code><SetupName></code>	Type String	Description 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>

This page intentionally
left blank.

16 - Optimetrics Module Script Commands

Optimetrics script commands should be executed by the "Optimetrics" module.

```
Set oModule = oDesign.GetModule("Optimetrics")
oModule.CommandName <args>
```

Conventions Used in this Chapter

<VarName>

Type: <string>

Name of a variable.

<VarValue>

Type: <string>

Value with unit (i.e., <value>, but cannot be an expression).

<StartV>

Type: <VarValue>

The starting value of a variable.

<StopV>

Type: <VarValue>

The stopping value of a variable.

<MinV>

Type: <VarValue>

The minimum value of a variable.

<MaxV>

Type: <VarValue>

The maximum value of a variable.

<IncludeVar>

Type: <bool>

Specifies whether the variable is included in the analysis.

<StartingPoint>

```
Array("NAME:StartingPoint", "<VarName>:=",
      <VarValue>, .... "<VarName>:=", <VarValue>)
```

<SaveField>

Type: <bool>

Specifies whether HFSS will remove the non-nominal field solution.

<MaxIter>

Type: <int>

Maximum iteration allowed in an analysis.

<PriorSetup>

Type: <string>

The name of the embedded parametric setup.

<Precede>

Type: <bool>

If true, the embedded parametric setup will be solved before the analysis begins.

If false, the embedded parametric setup will be solved during each iteration of the analysis.

<Constraint>

```
    Array("NAME:LCS",
          "lc:=", Array("<VarName>:=",
                        <Coeff>, ...<VarName>:=", <Coeff>, "rel:=", <Cond>, "rhs:=", <Rhs>), ...
          "lc:=", Array("<VarName>:=", <Coeff>, ..."
```

```
<VarName>:=", <Coeff>, "rel:=", <Cond>, "rhs:=",
<Rhs>))
```

<Coeff>

Type: <double>

Coefficient for a variable in the linear constraint.

<Cond>

Type: <string>

Inequality condition.

<Rhs>

Type: <double>

Inequality value.

```
<OptiGoalSpec>

    "Solution:=", <Soln>, "Calculation:=", <Calc>,
    "Context:=", <Geometry>

    Array("NAME:Ranges",
        "Range:", Array("Var:=",
            <VarName>, "Type:=", <RangeType>, "Start:=",
            <StartV>, "Stop:=", <StopV>), ...
        "Range:", Array("Var:=", <VarName>, "Type:=",
            <RangeType>, "Start:=", <StartV>, "Stop:=",
            <StopV>))
    <Soln>

        Type: <string>
        Name of the solution.

<Calc>

        Type: <string>
        An expression that is composed of a basic solution quantity and an
        output variable.

<ContextName>

        Type: <string>
        Name of context needed in the evaluation of <Calc>.
```

<Geometry>

Type: <string>

Name of geometry needed in the evaluation of <Calc>.

<RangeType>

Type: <string>

if "r", start and stop values specify a range for the variable.

if "s", start values specify the single value for the variable.

[EditSetup](#)

[EditSetup \[Optimization\]](#)

[EditSetup \[Sensitivity\]](#)

[EditSetup \[Statistical\]](#)

[GetPropNames \[Optimetrics\]](#)

[GetPropValue \[Optimetrics\]](#)

[GetSetupNames \[Optimetrics\]](#)

[GetSetupNamesByType \[Optimetrics\]](#)

[InsertSetup \[Parametric\]](#)

[InsertSetup \[Optimization\]](#)

[InsertSetup \[Sensitivity\]](#)

[InsertSetup \[Statistical\]](#)

[PasteSetup \[Optimetrics\]](#)

[RenameSetup \[Optimetrics\]](#)

[SetPropValue \[Optimetrics\]](#)

[SolveSetup \[Optimetrics\]](#)

The topics for this section include:

[General Commands Recognized by the Optimetrics Module](#)

[Parametric Script Commands](#)

[Optimization Script Commands](#)

[Sensitivity Script Commands](#)

[Statistical Script Commands](#)

CopySetup

Copy the specified Optimetrics setup.

UI Access	NA						
Parameters	<table border="1"><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><SetupName></td><td>String</td><td>Name of the setup.</td></tr></table>	Name	Type	Description	<SetupName>	String	Name of the setup.
Name	Type	Description					
<SetupName>	String	Name of the setup.					
Return Value	None.						

Python Syntax	CopySetup (<SetupName>)
Python Example	<pre>oModule.CopySetup ("OptimizationSetup1")</pre>

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	<table border="1"> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> <tr> <td><NameArray></td> <td>Array of Strings</td> <td>An Array of Setup Names</td> </tr> </table>	Name	Type	Description	<NameArray>	Array of Strings	An Array of Setup Names
Name	Type	Description					
<NameArray>	Array of Strings	An Array of Setup Names					
Return Value	None						

Python Syntax	DeleteSetups (<NameArray>)
Python Example	<code>oModule.DeleteSetups (["OptimizationSetup1"])</code>

VB Syntax	DeleteSetups <NameArray>
VB Example	<code>oModule.DeleteSetups Array("OptimizationSetup1")</code>

DistributedAnalyzeSetup

Distributes all variable value instances within a parametric sweep to different machines already specified from within the user interface

UI Access	Right-click the parametric setup name in the project tree and select Distribute Analysis.
------------------	---

Parameters	Name <code><ParametricSetupName></code>	Type String	Description Name of the Setup
Return Value	None		

Python Syntax	<code>DistributedAnalyzeSetup (<ParametricSetupName>)</code>
Python Example	<code>oModule.DistributedAnalyzeSetup ("ParametricSetup1")</code>

VB Syntax	<code>DistributedAnalyzeSetup <ParametricSetupName></code>
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 <code><SetupName></code>	Type String	Description 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	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, ["NAME:Cond", "MaxPass:=", 10, </pre>

```
        "MinPass:=", 1,
        "MinConvPass:=", 1,
        "PerError:=", 1,
        "PerRefine:=", 30),
    [ "NAME:Mult",
        "MaxPass:=", 1,
        "MinPass:=", 1,
        "MinConvPass:=", 1,
        "PerError:=", 1,
        "PerRefine:=", 30],
    [ "NAME:AC",
        "MaxPass:=", 10,
        "MinPass:=", 1,
        "MinConvPass:=", 2,
        "PerError:=", 1,
        "PerRefine:=", 30]
)
oModule.InsertSetup("HfssDrivenAuto",
[ "NAME:Setup1",
    "IsEnabled:=", True,
```

```
"AutoSolverSetting:=", "Balanced",
[ "NAME:Sweeps",
    [ "NAME:Sweep",
        "RangeType:=", "LinearStep",
        "RangeStart:=", "1GHz",
        "RangeEnd:=", "10GHz",
        "RangeStep:=", "1GHz"
    ]
],
"SaveRadFieldsOnly:=", False,
"SaveAnyFields:=", True,
"Type:=", "Discrete"
])

oModule.InsertSetup("HfssDrivenAuto",
[
    "NAME:Setup2",
    "SolveType:=" , "Auto",
    "IsEnabled:=" , True,
[
    "NAME:MeshLink",
```

```
        "ImportMesh:="           , False
    ],
    "AutoSolverSetting:="   , "Balanced",
    [
        "NAME:Sweeps",
        [
            "NAME:Sweep",
            "RangeType:="          , "LinearCount",
            "RangeStart:="          , "0GHz",
            "RangeEnd:="             , "10GHz",
            "RangeCount:="           , 501
        ]
    ],
    "SaveRadFieldsOnly:="   , False,
    "SaveAnyFields:="       , True,
    "InfiniteSphereSetup:=" , "Infinite Sphere1",
    "ListsForFields:="      , ["Objectlist2"],
    "Type:="                 , "Interpolating"
])
```

```
oModule.InsertSetup("HfssDriven",
["NAME:Setup3",
 "AdaptMultipleFreqs:=", False,
 "Frequency:=", "5GHz",
 "MaxDeltaS:=", 0.02,
 "PortsOnly:=", False,
 "UseMatrixConv:=", False,
 "MaximumPasses:=", 6,
 "MinimumPasses:=", 1,
 "MinimumConvergedPasses:=", 1,
 "PercentRefinement:=", 30,
 "IsEnabled:=", True,
 "BasisOrder:=", 1,
 "DoLambdaRefine:=", True,
 "DoMaterialLambda:=", True,
 "SetLambdaTarget:=", False,
 "Target:=", 0.3333,
 "UseMaxTetIncrease:=", False,
 "PortAccuracy:=", 2,
 "UseABCOnPort:=", False,
 "SetPortMinMaxTri:=", False,
```

```
"UseDomains:=", True,  
"UseIterativeSolver:=", False,  
"IterativeResidual:=", 1E-06,  
"DDMSolverResidual:=", 0.0001,  
"EnhancedLowFreqAccuracy:=", True,  
"SaveRadFieldsOnly:=", False,  
"SaveAnyFields:=", True,  
"IESolverType:=", "Auto",  
"LambdaTargetForIESolver:=", 0.15,  
"UseDefaultLambdaTgtForIESolver:=", True,  
"SkipPIERegionSolveDuringAdaptivePasses:=", True  
"RayDensityPerWavelength:=", 4,  
"MaxNumberOfBounces:=" , 5,  
"InfiniteSphereSetup:=" , "Infinite Sphere1",  
"SkipSBRSSolveDuringAdaptivePasses:=", True,  
"PTDUTDSimulationSettings:=", "PTD Correction + UTD Rays",  
"PTDEdgeDensity:=" , 20  
])  
### Edit an SBR+ Setup with Fast Frequency Looping  
oModule.EditSetup("HfssDriven",
```

```
[  
    "NAME:Setup1",  
    "IsEnabled:="           , True,  
    [  
        "NAME:MeshLink",  
        "ImportMesh:="          , False  
    ],  
    "IsSbrRangeDoppler:="   , False,  
    "RayDensityPerWavelength:=" , 4,  
    "MaxNumberOfBounces:="   , 5,  
    "IsMonostaticRCS:="     , True,  
    "EnableCW Rays:="       , False,  
    "RadiationSetup:="      , "",  
    "PTDUTDSimulationSettings:=" , "None",  
    "FastFrequencyLooping:=" , True,  
    [  
        "NAME:Sweeps",  
        [  
            "NAME:Sweep",  
            "RangeType:="          , "LinearStep",  
            "RangeStart:="          , "1GHz",  
        ]  
    ]  
]
```

```
        "RangeEnd:=" , "10GHz",
        "RangeStep:=" , "1GHz"
    ],
],
"ComputeFarFields:=" , True
"UseSBREnhancedRadiatedPowerCalculation:=", True,
"IsGOBlockageEnabled:=" , False,
"GOBlockageSurfaceSelfBlock:=", False
])

##### Edit and RF Discharge Setup for HFSS
import ScriptEnv
ScriptEnv.Initialize("Ansoft.ElectronicsDesktop")
oDesktop.RestoreWindow()
oProject = oDesktop.SetActiveProject("coaxbend_discharge_r212")
oDesign = oProject.SetActiveDesign("HFSSDesign60degBendTeflon")
oModule = oDesign.GetModule("AnalysisSetup")
oModule.EditSetup("RFDischarge1",
[
    "NAME:RFDischarge1",
    "Enabled:=" , True,
```

```
[  
    "NAME:MeshLink",  
    "ImportMesh:="           , True,  
    "Project:="              , "This Project*",  
    "Product:="              , "HFSS",  
    "Design:="               , "This Design*",  
    "Soln:="                 , "Setup1 : Sweep",  
    [  
        "NAME:Params",  
        "bend_angle:="          , "bend_angle"  
    ],  
    "ForceSourceToSolve:="   , True,  
    "PreservePartnerSoln:=" , False,  
    "PathRelativeTo:="       , "SourceProduct",  
    "ApplyMeshOp:="          , True  
],  
[  
    "NAME:Excitations",  
    [  
        "NAME:1:1",  
        "Magnitude:="          , "1",  
    ]]
```

```
        "Phase:="           , "0deg"
    ] ,
    [
        "NAME:2:1",
        "Magnitude:="      , "0",
        "Phase:="           , "0deg"
    ]
],
[
    "NAME:Frequencies",
    "10GHz"
],
[
    "Minimum Power:="   , "0.01",
    "Maximum Power:="   , "1000000",
    "Minimum Pressure:=" , "100pascal",
    "Maximum Pressure:=" , "101325pascal",
    "Postproc Sampling:=" , 500,
    "Temperature:="      , "0cel",
    "BuiltInGas:="        , "Helium"
]
)
```

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", "Residual:=", 1E-005, "SolveResOnly:=", false, Array("NAME:Cond", "MaxPass:=", 10, </pre>

```
        "MinPass:=", 1,
        "MinConvPass:=", 1,
        "PerError:=", 1,
        "PerRefine:=", 30),
    Array("NAME:Mult",
          "MaxPass:=", 1,
          "MinPass:=", 1,
          "MinConvPass:=", 1,
          "PerError:=", 1,
          "PerRefine:=", 30)),
    Array("NAME:AC",
          "MaxPass:=", 10,
          "MinPass:=", 1,
          "MinConvPass:=", 2,
          "PerError:=", 1,
          "PerRefine:=", 30))
```

EnableSetup

Enables and disables a defined optimetrics analysis setup.

UI Access	Right-click on a setup in the project tree, select Enable Setup or Disable Setup
-----------	--

Parameters	Name	Type	Description
	<SetupName>	String	Name of specified setup.
	<Enable>	Boolean	Determines whether enable or disable a setup. <ul style="list-style-type: none">• True - enable setup.• False - disable setup.
Return Value	None.		

Python Syntax	EnableSetup(<SetupName>, <Enable>)
Python Example	oModule.EnableSetup ("OptimizationSetup1", True)

VB Syntax	EnableSetup <SetupName>, <Enable>
VB Example	oModule.EnableSetup "OptimizationSetup1", true

ExportDXConfigFile

Create an xml file with the setup information for Design 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	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td><SetupName></td><td>String</td><td>Must be one of the existing Parametric, Optimization, Sensitivity, Statistical or DesignXplorer setup names</td></tr><tr><td><FileName></td><td>String</td><td>Must be a valid file path and name with extension of csv, tab, dat, or txt</td></tr><tr><td>[profileNum]</td><td>String</td><td>Must be a numeric string. Optional: defaulted to last profile number. It should be a zero indexed profile number.</td></tr></tbody></table>			Name	Type	Description	<SetupName>	String	Must be one of the existing Parametric, Optimization, Sensitivity, Statistical or DesignXplorer setup names	<FileName>	String	Must be a valid file path and name with extension of csv, tab, dat, or txt	[profileNum]	String	Must be a numeric string. Optional: defaulted to last profile number. It should be a zero indexed profile number.
Name	Type	Description													
<SetupName>	String	Must be one of the existing Parametric, Optimization, Sensitivity, Statistical or DesignXplorer setup names													
<FileName>	String	Must be a valid file path and name with extension of csv, tab, dat, or txt													
[profileNum]	String	Must be a numeric string. Optional: defaulted to last profile number. It should be a zero indexed profile number.													
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	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><SetupName></td> <td>String</td> <td>Must be one of the existing Optimization, Sensitivity, Statistical, or DesignXplorer setup names</td> </tr> <tr> <td><FileName></td> <td>String</td> <td>Must be a valid file path and name with extension of csv, tab, dat, or txt..</td> </tr> <tr> <td>[useFullOutputName]</td> <td>Boolean</td> <td>Optional: defaulted to false. If set to true values will be printed with units. This parameter is ignored for Optimization and Statistical results.</td> </tr> </tbody> </table>	Name	Type	Description	<SetupName>	String	Must be one of the existing Optimization, Sensitivity, Statistical, or DesignXplorer setup names	<FileName>	String	Must be a valid file path and name with extension of csv, tab, dat, or txt..	[useFullOutputName]	Boolean	Optional: defaulted to false. If set to true values will be printed with units. This parameter is ignored for Optimization and Statistical results.
Name	Type	Description											
<SetupName>	String	Must be one of the existing Optimization, Sensitivity, Statistical, or DesignXplorer setup names											
<FileName>	String	Must be a valid file path and name with extension of csv, tab, dat, or txt..											
[useFullOutputName]	Boolean	Optional: defaulted to false. If set to true values will be printed with units. This parameter is ignored for Optimization and Statistical results.											
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	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td><SetupName></td><td>String</td><td>Must be one of the existing Parametric setup names</td></tr><tr><td><FileName></td><td>String</td><td>Must be a valid file path and name with extension of csv, tab, dat or txt</td></tr><tr><td><bOutputUnits></td><td>Boolean</td><td>If set to true, values will be printed with units</td></tr></tbody></table>	Name	Type	Description	<SetupName>	String	Must be one of the existing Parametric setup names	<FileName>	String	Must be a valid file path and name with extension of csv, tab, dat or txt	<bOutputUnits>	Boolean	If set to true, values will be printed with units
Name	Type	Description											
<SetupName>	String	Must be one of the existing Parametric setup names											
<FileName>	String	Must be a valid file path and name with extension of csv, tab, dat or txt											
<bOutputUnits>	Boolean	If set to true, values will be printed with units											
Return Value	None												

Python Syntax	ExportParametricResults (<SetupName>, <FileName>, <bOutputUnits>)
Python Example	<pre>oModule.ExportParametricResults (</pre>

	"ParametricSetup1", "c:/exportdir/test.csv", False)
--	---

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	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><SetupName></td> <td>String</td> <td>Name of the setup.</td> </tr> <tr> <td><filePath></td> <td>String</td> <td>Full path for file export.</td> </tr> </tbody> </table>	Name	Type	Description	<SetupName>	String	Name of the setup.	<filePath>	String	Full path for file export.
Name	Type	Description								
<SetupName>	String	Name of the setup.								
<filePath>	String	Full path for file export.								
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.
Return Value	None.		

Python Syntax

```
ExportRespSurfaceMinMaxTable(<DOEName>, <FileName>)
```

Python Example

```
oModule.ExportRespSurfaceMinMaxTable ("DesignOfExperimentsSetup1",
"C:/temp/DesignOfExperimentsSetup1_Min-Max_Search.csv")
```

VB Syntax

```
ExportRespSurfaceMinMaxTable <DOEName>, <FileName>
```

VB Example

```
oModule.ExportRespSurfaceMinMaxTable _
"DesignOfExperimentsSetup1", _
"C:/temp/DesignOfExperimentsSetup1_Min-Max_Search.csv"
```

ExportRespSurfaceRefinePoints

Exports refinement points table to a file

UI Access	From the Response Surface tab of the Design of Experiments Post Analysis Display dialog box, select Refinement Points option under View , Click on Export....											
Parameters	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><DOEName></td> <td>String</td> <td>Name of the Design of Experiments (DOE) setup.</td> </tr> <tr> <td><FileName></td> <td>String</td> <td>Output file name with path.</td> </tr> </tbody> </table>			Name	Type	Description	<DOEName>	String	Name of the Design of Experiments (DOE) setup.	<FileName>	String	Output file name with path.
Name	Type	Description										
<DOEName>	String	Name of the Design of Experiments (DOE) setup.										
<FileName>	String	Output file name with path.										
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	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><DOEName></td> <td>String</td> <td>Name of the Design of Experiments (DOE) setup.</td> </tr> <tr> <td><FileName></td> <td>String</td> <td>Output file name with path.</td> </tr> </tbody> </table>	Name	Type	Description	<DOEName>	String	Name of the Design of Experiments (DOE) setup.	<FileName>	String	Output file name with path.
Name	Type	Description								
<DOEName>	String	Name of the Design of Experiments (DOE) setup.								
<FileName>	String	Output file name with path.								
Return Value	None.									

Python Syntax	ExportRespSurfaceResponsePoints (<DOEName>, <FileName>)
Python Example	<pre>oModule.ExportRespSurfaceResponsePoints ("DesignOfExperimentsSetup1", "C:/temp/DesignOfExperimentsSetup1_Response_Points.csv")</pre>

VB Syntax	ExportRespSurfaceResponsePoints <DOEName>, <FileName>
VB Example	<pre>oModule.ExportRespSurfaceResponsePoints_ "DesignOfExperimentsSetup1", _ "C:/temp/DesignOfExperimentsSetup1_Response_Points.csv"</pre>

ExportRespSurfaceVerificationPoints

Exports verification points table to a file

UI Access	From the Response Surface tab of the Design of Experiments Post Analysis Display dialog box, select Verification Points option under View , Click on Export....
------------------	--

Parameters	Name	Type	Description
	<DOEName>	String	Name of the Design of Experiments (DOE) setup.
	<FileName>	String	Output file name with path.
Return Value	None.		

Python Syntax	ExportRespSurfaceVerificationPoints (<DOEName>, <FileName>)
Python Example	<pre>oModule.ExportRespSurfaceVerificationPoints("DesignOfExperimentsSetup1", "C:/temp/DesignOfExperimentsSetup1_Veri_Points.csv")</pre>

VB Syntax	ExportRespSurfaceVerificationPoints <DOEName>, <FileName>
VB Example	<pre>oModule.ExportRespSurfaceVerificationPoints_ "DesignOfExperimentsSetup1", _ "C:/temp/DesignOfExperimentsSetup1_Veri_Points.csv"</pre>

GenerateVariationData [Parametric]

Generate variation data before parametric solve for CAD integrated project

Command: Right click on the parametric setup in the project tree and choose "Generate Variation Data"

Syntax: GenerateVariationData <SetupName>

Return Value: None

Parameters: <SetupName>

Name of the setup.

VB Example:

```
oModule.GenerateVariationData "ParametricSetup1"
```

GetChildNames [Optimetrics]

If used without a specific optimization setup name, gets a list of all setups for all types. If a with a specific setup name, returns names for that optimization setup.

UI Access	NA		
Parameters	Name	Type	Description
	typeName	text string	Optional, default to get all types of setup names. Or one of type name return in GetChildTypes(). Also, the type name can be used without the prefix "Opti".
Return Value	Array of setup names.		

Python Syntax	GetChildNames()
Python Example	<pre>oOptimModule = oDesign.GetChildObject("Optimetrics") arrAllSetup = oOptimModule.GetChildNames() arrParmSetup = oOptimModule.GetChildNames("'OptiParametric'") arrOptimizeSetup = oOptimModule.GetChildNames("'Optimization'")</pre>

GetChildObject [Optimetrics]

Gets a Setup Object of the Optimetrics module

UI Access	NA						
Parameters	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>Setup Name</td> <td>text string</td> <td>A optimetrics setup name, names returned by the GetChildNames().</td> </tr> </tbody> </table>	Name	Type	Description	Setup Name	text string	A optimetrics setup name, names returned by the GetChildNames().
Name	Type	Description					
Setup Name	text string	A optimetrics setup name, names returned by the GetChildNames().					
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 name of the active design.

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
	<code><SetupName></code>	String	Optimetrics setup name.
	<code><vars></code>	Array	Array containing string variable names. Use the Sweep Definitions tab in the UI or the <code><SweepDefs></code> parameter in the InsertSetup script to determine appropriate inputs.
	<code><values></code>	Array	<i>Optional.</i> Array containing string values. When multiple variables and values are provided, the order must be the same in both the <code><vars></code> and <code><values></code> 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	<pre>oModule.GetOptimetricResult('ParametricSetup1', ['AR', 'Re'], ['4.64', '6e+04'])</pre>

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	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><SetupName></td> <td>String</td> <td>Must be one of the existing Optimization, Sensitivity, Statistical, or DesignXplorer setup names</td> </tr> <tr> <td><FileName></td> <td>String</td> <td>Must be a valid file path and name with extension of csv, tab, dat, or txt..</td> </tr> <tr> <td>[useFullOutputName]</td> <td>Boolean</td> <td>Optional: defaulted to false. If set to true values will be printed with units. This</td> </tr> </tbody> </table>			Name	Type	Description	<SetupName>	String	Must be one of the existing Optimization, Sensitivity, Statistical, or DesignXplorer setup names	<FileName>	String	Must be a valid file path and name with extension of csv, tab, dat, or txt..	[useFullOutputName]	Boolean	Optional: defaulted to false. If set to true values will be printed with units. This
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	<table border="1"><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>property-path</td><td></td><td>a child object's property path. See property path discussion here.</td></tr></table>			Name	Type	Description	property-path		a child object's property path. See property path discussion here .
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	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>None</td> <td></td> <td></td> </tr> </tbody> </table>	Name	Type	Description	None		
Name	Type	Description					
None							
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	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><Optimetrics type></td> <td>String</td> <td>Examples: parametric, optimization, statistical, sensitivity</td> </tr> </tbody> </table>	Name	Type	Description	<Optimetrics type>	String	Examples: parametric, optimization, statistical, sensitivity
Name	Type	Description					
<Optimetrics type>	String	Examples: parametric, optimization, statistical, sensitivity					
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 <SetupTypeName>	Type String	Description Must be one of "OptiParametric" , "OptiOptimization", "OptiSensitivity", "OptiStatistical", or "OptiDesignExplorer".

UI Access	NA		
Parameters	<SetupInfo>	Array <SetupName> Type: <string>	Array("NAME:<SetupName>", "FilePath")

		<p>Name of the setup.</p> <p><FilePath></p> <p>Type : <string: file path></p> <p>Must be a valid file path and name.</p>
Return Value	None	

Python Syntax	<code>ImportSetup (<SetupTypeName>, <SetupInfo>)</code>
Python Example	<pre>oModule.ImportSetup ("OptiStatistical", ["NAME:StatisticalSetup1", "c:/importdir/mySetupInfoFile"])</pre>

VB Syntax	<code>ImportSetup <SetupTypeName>, <SetupInfo></code>
VB Example	<pre>oModule.ImportSetup "OptiStatistical", Array("NAME:StatisticalSetup1", "c:/importdir/mySetupInfoFile")</pre>

PasteSetup [Optimetrics]

Pastes the specified Optimetrics setup.

UI Access	NA						
Parameters	<table border="1"><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><SetupName></td><td>String</td><td>Name of the Setup</td></tr></table>	Name	Type	Description	<SetupName>	String	Name of the Setup
Name	Type	Description					
<SetupName>	String	Name of the Setup					
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	<table border="1"><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><OldName></td><td>String</td><td>The name that needs to be replaced</td></tr><tr><td><NewName></td><td>String</td><td>Replacement name</td></tr></table>	Name	Type	Description	<OldName>	String	The name that needs to be replaced	<NewName>	String	Replacement name
Name	Type	Description								
<OldName>	String	The name that needs to be replaced								
<NewName>	String	Replacement name								
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	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
	None		
Return Value	None		

Python Syntax	SolveAllSetup()
Python Example	oModule.SolveAllSetup ()

VB Syntax	SolveAllSetup
VB Example	oModule.SolveAllSetup

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	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><SetupName></td> <td>String</td> <td>Name of the setup to be solved</td> </tr> </tbody> </table>			Name	Type	Description	<SetupName>	String	Name of the setup to be solved
Name	Type	Description							
<SetupName>	String	Name of the setup to be solved							
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 <code><SetupName></code>	Type String	Description Name of the setup.
Return Value	None.		

Python Syntax	CopySetup (<SetupName>)
----------------------	-------------------------

Python Example	<code>oModule.CopySetup ("OptimizationSetup1")</code>
-----------------------	---

VB Syntax	<code>CopySetup <SetupName></code>
------------------	--

VB Example	<code>oModule.CopySetup "OptimizationSetup1"</code>
-------------------	---

DeleteSetups [Optimetrics]

Deletes the specified Optimetrics setups.

UI Access	Right-click the setup in the project tree, and then click Delete on the shortcut menu		
Parameters	Name <code><NameArray></code>	Type Array of Strings	Description 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 <ParametricSetupName>	Type String	Description Name of the Setup
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 <SetupName>	Type String	Description Name of the solve setup being edited.

		Array ("NAME:<NewSetupName>", <NamedParameters>) See the InsertSetup command for details and examples.
Return Value	None.	

Python Syntax	EditSetup (<SetupName>, <Attributes>)
	<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>
Python Example	

```
"Residual:=", 1E-005,  
"SolveResOnly:=", false,  
[ "NAME:Cond",  
    "MaxPass:=", 10,  
    "MinPass:=", 1,  
    "MinConvPass:=", 1,  
    "PerError:=", 1,  
    "PerRefine:=", 30),  
[ "NAME:Mult",  
    "MaxPass:=", 1,  
    "MinPass:=", 1,  
    "MinConvPass:=", 1,  
    "PerError:=", 1,  
    "PerRefine:=", 30]],  
[ "NAME:AC",  
    "MaxPass:=", 10,  
    "MinPass:=", 1,  
    "MinConvPass:=", 2,  
    "PerError:=", 1,  
    "PerRefine:=", 30]]
```

```
)  
  
oModule.InsertSetup("HfssDrivenAuto",  
    [ "NAME:Setup1",  
        "IsEnabled:=", True,  
        "AutoSolverSetting:=", "Balanced",  
        [ "NAME:Sweeps",  
            [ "NAME:Sweep",  
                "RangeType:=", "LinearStep",  
                "RangeStart:=", "1GHz",  
                "RangeEnd:=", "10GHz",  
                "RangeStep:=", "1GHz"  
            ]  
        ],  
        "SaveRadFieldsOnly:=", False,  
        "SaveAnyFields:=", True,  
        "Type:=", "Discrete"  
    ])  
  
oModule.InsertSetup("HfssDrivenAuto",  
    [  
        "NAME:Setup2",  
    ]
```

```
        "SolveType:="           , "Auto",
        "IsEnabled:="          , True,
        [
            "NAME:MeshLink",
            "ImportMesh:="        , False
        ],
        "AutoSolverSetting:="   , "Balanced",
        [
            "NAME:Sweeps",
            [
                "NAME:Sweep",
                "RangeType:="         , "LinearCount",
                "RangeStart:="         , "0GHz",
                "RangeEnd:="           , "10GHz",
                "RangeCount:="         , 501
            ]
        ],
        "SaveRadFieldsOnly:="   , False,
        "SaveAnyFields:="       , True,
        "InfiniteSphereSetup:=" , "Infinite Sphere1",
```

```
    "ListsForFields:" , [ "Objectlist2" ],
    "Type:" , "Interpolating"
  ] )

oModule.InsertSetup("HfssDriven",
[ "NAME:Setup3",
  "AdaptMultipleFreqs:", False,
  "Frequency:", "5GHz",
  "MaxDeltaS:", 0.02,
  "PortsOnly:", False,
  "UseMatrixConv:", False,
  "MaximumPasses:", 6,
  "MinimumPasses:", 1,
  "MinimumConvergedPasses:", 1,
  "PercentRefinement:", 30,
  "IsEnabled:", True,
  "BasisOrder:", 1,
  "DoLambdaRefine:", True,
  "DoMaterialLambda:", True,
  "SetLambdaTarget:", False,
  "Target:", 0.3333,
```

```
"UseMaxTetIncrease:=", False,
"PortAccuracy:=", 2,
"UseABCOnPort:=", False,
"SetPortMinMaxTri:=", False,
"UseDomains:=", True,
"UseIterativeSolver:=", False,
"IterativeResidual:=", 1E-06,
"DDMSolverResidual:=", 0.0001,
"EnhancedLowFreqAccuracy:=", True,
"SaveRadFieldsOnly:=", False,
"SaveAnyFields:=", True,
"IESolverType:=", "Auto",
"LambdaTargetForIESolver:=", 0.15,
"UseDefaultLambdaTgtForIESolver:=", True,
"SkipPIERegionSolveDuringAdaptivePasses:=", True
"RayDensityPerWavelength:=", 4,
"MaxNumberOfBounces:=", 5,
"InfiniteSphereSetup:=", "Infinite Sphere1",
"SkipSBRSSolveDuringAdaptivePasses:=", True,
"PTDUTDSimulationSettings:=", "PTD Correction + UTD Rays",
```

```
        "PTDEdgeDensity:="          , 20
    ])
### Edit an SBR+ Setup with Fast Frequency Looping
oModule.EditSetup("HfssDriven",
[
    "NAME:Setup1",
    "IsEnabled:="              , True,
    [
        "NAME:MeshLink",
        "ImportMesh:="           , False
    ],
    "IsSbrRangeDoppler:="     , False,
    "RayDensityPerWavelength:=", 4,
    "MaxNumberOfBounces:="     , 5,
    "IsMonostaticRCS:="       , True,
    "EnableCW Rays:="         , False,
    "RadiationSetup:="        , "",
    "PTDUTDSimulationSettings:=", "None",
    "FastFrequencyLooping:="   , True,
    [
        "NAME:Sweeps",
        [

```

```
        "NAME:Sweep",
        "RangeType:="           , "LinearStep",
        "RangeStart:="          , "1GHz",
        "RangeEnd:="            , "10GHz",
        "RangeStep:="           , "1GHz"
    ],
    "ComputeFarFields:="     , True
    "UseSBREnhancedRadiatedPowerCalculation:=", True,
    "IsGOBlockageEnabled:=" , False,
    "GOBlockageSurfaceSelfBlock:=", False
))

##### Edit and RF Discharge Setup for HFSS
import ScriptEnv
ScriptEnv.Initialize("Ansoft.ElectronicsDesktop")
oDesktop.RestoreWindow()
oProject = oDesktop.SetActiveProject("coaxbend_discharge_r212")
oDesign = oProject.SetActiveDesign("HFSSDesign60degBendTeflon")
oModule = oDesign.GetModule("AnalysisSetup")
oModule.EditSetup("RFDischarge1",
```

```
[  
    "NAME:RFDischarge1",  
    "Enabled:=" , True,  
    [  
        "NAME:MeshLink",  
        "ImportMesh:=" , True,  
        "Project:=" , "This Project*",  
        "Product:=" , "HFSS",  
        "Design:=" , "This Design*",  
        "Soln:=" , "Setup1 : Sweep",  
        [  
            "NAME:Params",  
            "bend_angle:=" , "bend_angle"  
        ],  
        "ForceSourceToSolve:=" , True,  
        "PreservePartnerSoln:=" , False,  
        "PathRelativeTo:=" , "SourceProduct",  
        "ApplyMeshOp:=" , True  
    ],  
    [  
        "NAME:Excitations",  
    ]]
```

```
[  
    "NAME:1:1",  
    "Magnitude:=" , "1",  
    "Phase:=" , "0deg"  
,  
[  
    "NAME:2:1",  
    "Magnitude:=" , "0",  
    "Phase:=" , "0deg"  
,  
[  
    "NAME:Frequencies",  
    "10GHz"  
,  
    "Minimum Power:=" , "0.01",  
    "Maximum Power:=" , "1000000",  
    "Minimum Pressure:=" , "100pascal",  
    "Maximum Pressure:=" , "101325pascal",  
    "Postproc Sampling:=" , 500,
```

	<pre> "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", "Residual:=", 1E-005, </pre>

```
"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 optometrics 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.
Return Value	None.		

Python Syntax	EnableSetup(<SetupName>, <Enable>)
Python Example	oModule.EnableSetup ("OptimizationSetup1", True)

VB Syntax	EnableSetup <SetupName>, <Enable>
VB Example	oModule.EnableSetup "OptimizationSetup1", true

ExportDXConfigFile

Create an xml file with the setup information for Design 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	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><SetupName></td> <td>String</td> <td>Must be one of the existing Parametric, Optimization, Sensitivity, Statistical or DesignXplorer setup names</td></tr> </tbody> </table>			Name	Type	Description	<SetupName>	String	Must be one of the existing Parametric, Optimization, Sensitivity, Statistical or DesignXplorer setup names
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	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><SetupName></td> <td>String</td> <td>Must be one of the existing Optimization, Sensitivity, Statistical, or DesignXplorer setup names</td> </tr> <tr> <td><FileName></td> <td>String</td> <td>Must be a valid file path and name with extension of csv, tab, dat, or txt..</td> </tr> <tr> <td>[useFullOutputName]</td> <td>Boolean</td> <td>Optional: defaulted to false. If set to true values will be printed with units.</td> </tr> </tbody> </table>			Name	Type	Description	<SetupName>	String	Must be one of the existing Optimization, Sensitivity, Statistical, or DesignXplorer setup names	<FileName>	String	Must be a valid file path and name with extension of csv, tab, dat, or txt..	[useFullOutputName]	Boolean	Optional: defaulted to false. If set to true values will be printed with units.
Name	Type	Description													
<SetupName>	String	Must be one of the existing Optimization, Sensitivity, Statistical, or DesignXplorer setup names													
<FileName>	String	Must be a valid file path and name with extension of csv, tab, dat, or txt..													
[useFullOutputName]	Boolean	Optional: defaulted to false. If set to true values will be printed with units.													

		This parameter is ignored for Optimization and Statistical results.
Return Value	None	

Python Syntax	ExportOptimetricsResult (<SetupName>, <FileName>, [useFullOutputName])
Python Example	<pre>oModule.ExportOptimetricsResult ("StatisticalSetup1", "c:/exportdir/test.csv", false)</pre>

VB Syntax	ExportOptimetricsResult <SetupName>, <FileName>, [useFullOutputName]
VB Example	<pre>oModule.ExportOptimetricsResult "StatisticalSetup1", "c:/exportdir/test.csv", false</pre>

ExportParametricResults

Export existing Parametric results.

UI Access	Right click on the desired Parametric setup in the project tree and choose View Analysis Result... On the Post Analysis Display dialog box, click the Result tab, then select Table view, and click on the Export button.														
Parameters	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><SetupName></td> <td>String</td> <td>Must be one of the existing Parametric setup names</td> </tr> <tr> <td><FileName></td> <td>String</td> <td>Must be a valid file path and name with extension of csv, tab, dat or txt</td> </tr> <tr> <td><bOutputUnits></td> <td>Boolean</td> <td>If set to true, values will be printed with units</td> </tr> </tbody> </table>			Name	Type	Description	<SetupName>	String	Must be one of the existing Parametric setup names	<FileName>	String	Must be a valid file path and name with extension of csv, tab, dat or txt	<bOutputUnits>	Boolean	If set to true, values will be printed with units
Name	Type	Description													
<SetupName>	String	Must be one of the existing Parametric setup names													
<FileName>	String	Must be a valid file path and name with extension of csv, tab, dat or txt													
<bOutputUnits>	Boolean	If set to true, values will be printed with units													
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	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><SetupName></td> <td>String</td> <td>Name of the setup.</td> </tr> <tr> <td><filePath></td> <td>String</td> <td>Full path for file export.</td> </tr> </tbody> </table>	Name	Type	Description	<SetupName>	String	Name of the setup.	<filePath>	String	Full path for file export.
Name	Type	Description								
<SetupName>	String	Name of the setup.								
<filePath>	String	Full path for file export.								
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	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><DOEName></td> <td>String</td> <td>Name of the Design of Experiments (DOE) setup.</td> </tr> <tr> <td><FileName></td> <td>String</td> <td>Output file name with path.</td> </tr> </tbody> </table>	Name	Type	Description	<DOEName>	String	Name of the Design of Experiments (DOE) setup.	<FileName>	String	Output file name with path.
Name	Type	Description								
<DOEName>	String	Name of the Design of Experiments (DOE) setup.								
<FileName>	String	Output file name with path.								
Return Value	None.									

Python Syntax	ExportRespSurfaceMinMaxTable(<DOEName>, <FileName>)
Python Example	oModule.ExportRespSurfaceMinMaxTable ("DesignOfExperimentsSetup1", "C:/temp/DesignOfExperimentsSetup1_Min-Max_Search.csv")

VB Syntax	ExportRespSurfaceMinMaxTable <DOEName>, <FileName>
VB Example	<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	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><DOEName></td> <td>String</td> <td>Name of the Design of Experiments (DOE) setup.</td> </tr> <tr> <td><FileName></td> <td>String</td> <td>Output file name with path.</td> </tr> </tbody> </table>			Name	Type	Description	<DOEName>	String	Name of the Design of Experiments (DOE) setup.	<FileName>	String	Output file name with path.
Name	Type	Description										
<DOEName>	String	Name of the Design of Experiments (DOE) setup.										
<FileName>	String	Output file name with path.										
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", _ </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	<table border="1"> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> <tr> <td><DOEName></td> <td>String</td> <td>Name of the Design of Experiments (DOE) setup.</td> </tr> <tr> <td><FileName></td> <td>String</td> <td>Output file name with path.</td> </tr> </table>			Name	Type	Description	<DOEName>	String	Name of the Design of Experiments (DOE) setup.	<FileName>	String	Output file name with path.
Name	Type	Description										
<DOEName>	String	Name of the Design of Experiments (DOE) setup.										
<FileName>	String	Output file name with path.										
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	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><DOEName></td> <td>String</td> <td>Name of the Design of Experiments (DOE) setup.</td> </tr> <tr> <td><FileName></td> <td>String</td> <td>Output file name with path.</td> </tr> </tbody> </table>	Name	Type	Description	<DOEName>	String	Name of the Design of Experiments (DOE) setup.	<FileName>	String	Output file name with path.
Name	Type	Description								
<DOEName>	String	Name of the Design of Experiments (DOE) setup.								
<FileName>	String	Output file name with path.								
Return Value	None.									

Python Syntax	ExportRespSurfaceVerificationPoints (<DOEName>, <FileName>)
Python Example	<pre>oModule.ExportRespSurfaceVerificationPoints("DesignOfExperimentsSetup1", " C:/temp/DesignOfExperimentsSetup1_Veri_Points.csv")</pre>

VB Syntax	ExportRespSurfaceVerificationPoints <DOEName>, <FileName>
VB Example	<pre>oModule.ExportRespSurfaceVerificationPoints_ "DesignOfExperimentsSetup1", _ " C:/temp/DesignOfExperimentsSetup1_Veri_Points.csv"</pre>

GenerateVariationData [Parametric]

Generate variation data before parametric solve for CAD integrated project

Command: Right click on the parametric setup in the project tree and choose "Generate Variation Data"

Syntax: GenerateVariationData <SetupName>

Return Value: None

Parameters: <SetupName>

Name of the setup.

VB Example:

```
oModule.GenerateVariationData "ParametricSetup1"
```

GetChildNames [Optimetrics]

If used without a specific optimization setup name, gets a list of all setups for all types. If a with a specific setup name, returns names for that optimization setup.

UI Access	NA						
Parameters	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>typeName</td><td>text string</td><td>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".</td></tr></tbody></table>	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".
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

```
oOptimModule = oDesign.GetChildObject("Optimetrics")
arrSetupTypes = oOptimModule.GetChildTypes()
```

GetName

Returns the name of the active design.

UI Access	N/A
Parameters	None.
Return Value	String indicating the name of the active design.

Python Syntax

```
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	<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	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><code><SetupName></code></td> <td>String</td> <td>Optimetrics setup name.</td> </tr> <tr> <td><code><vars></code></td> <td>Array</td> <td>Array containing string variable names. Use the Sweep Definitions tab in the UI or the <code><SweepDefs></code> parameter in the InsertSetup script to determine appropriate inputs.</td> </tr> <tr> <td><code><values></code></td> <td>Array</td> <td><i>Optional.</i> Array containing string values. When multiple variables and values are provided, the order must be the same in both the <code><vars></code> and <code><values></code> arrays. The first variable is paired with the first value, the second variable is paired with the second value, and so on.</td> </tr> </tbody> </table>	Name	Type	Description	<code><SetupName></code>	String	Optimetrics setup name.	<code><vars></code>	Array	Array containing string variable names. Use the Sweep Definitions tab in the UI or the <code><SweepDefs></code> parameter in the InsertSetup script to determine appropriate inputs.	<code><values></code>	Array	<i>Optional.</i> Array containing string values. When multiple variables and values are provided, the order must be the same in both the <code><vars></code> and <code><values></code> arrays. The first variable is paired with the first value, the second variable is paired with the second value, and so on.		
Name	Type	Description													
<code><SetupName></code>	String	Optimetrics setup name.													
<code><vars></code>	Array	Array containing string variable names. Use the Sweep Definitions tab in the UI or the <code><SweepDefs></code> parameter in the InsertSetup script to determine appropriate inputs.													
<code><values></code>	Array	<i>Optional.</i> Array containing string values. When multiple variables and values are provided, the order must be the same in both the <code><vars></code> and <code><values></code> 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	<pre>oModule.GetOptimetricResult('ParametricSetup1', ['AR', 'Re'], ['4.64', '6e+04'])</pre>

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	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>property-path</td> <td></td> <td>a child object's property path. See property path discussion here.</td> </tr> </tbody> </table>	Name	Type	Description	property-path		a child object's property path. See property path discussion here .
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")</pre> <p>Returns True if Enabled, or False if disabled.</p>

GetSetupNames [Optimetrics]

Gets a list of Optimetrics setup names

UI Access	NA						
Parameters	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>None</td><td></td><td></td></tr></tbody></table>	Name	Type	Description	None		
Name	Type	Description					
None							
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	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><Optimetrics type></td> <td>String</td> <td>Examples: parametric, optimization, statistical, sensitivity</td> </tr> </tbody> </table>	Name	Type	Description	<Optimetrics type>	String	Examples: parametric, optimization, statistical, sensitivity
Name	Type	Description					
<Optimetrics type>	String	Examples: parametric, optimization, statistical, sensitivity					
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	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><SetupTypeName></td> <td>String</td> <td>Must be one of "OptiParametric", "OptiOptimization", "OptiSensitivity", "OptiStatistical", or "OptiDesignExplorer".</td> </tr> <tr> <td><SetupInfo></td> <td>Array</td> <td>Array("NAME:<SetupName>", "FilePath")</td> </tr> </tbody> </table>	Name	Type	Description	<SetupTypeName>	String	Must be one of "OptiParametric", "OptiOptimization", "OptiSensitivity", "OptiStatistical", or "OptiDesignExplorer".	<SetupInfo>	Array	Array("NAME:<SetupName>", "FilePath")
Name	Type	Description								
<SetupTypeName>	String	Must be one of "OptiParametric", "OptiOptimization", "OptiSensitivity", "OptiStatistical", or "OptiDesignExplorer".								
<SetupInfo>	Array	Array("NAME:<SetupName>", "FilePath")								

			<p><SetupName> Type: <string> Name of the setup.</p> <p><FilePath> Type : <string: file path> Must be a valid file path and name.</p>
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	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><SetupName></td> <td>String</td> <td>Name of the Setup</td> </tr> </tbody> </table>	Name	Type	Description	<SetupName>	String	Name of the Setup
Name	Type	Description					
<SetupName>	String	Name of the Setup					
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	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><OldName></td> <td>String</td> <td>The name that needs to be replaced</td> </tr> <tr> <td><NewName></td> <td>String</td> <td>Replacement name</td> </tr> </tbody> </table>	Name	Type	Description	<OldName>	String	The name that needs to be replaced	<NewName>	String	Replacement name
Name	Type	Description								
<OldName>	String	The name that needs to be replaced								
<NewName>	String	Replacement name								
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
Return Value	True if the property is found and the new value is valid. Otherwise return False.		

Python Syntax	SetPropValue(propPath, newValue)
Python Example	<code>oOptModule.SetPropValue("ParametricSetup1\Enabled", False) //disable ParametricSetup1</code>

	<code>oOptModule.SetPropValue("OptimizationSetup1/Optimizer", "Quasi Newton")</code>
--	--

VB Syntax	<code>SetPropValue(propPath, newValue)</code>
VB Example	<code>SetPropValue("ParametricSetup1\Enabled", False)</code>

SolveAllSetup

Solves all Optimetrics setups

UI Access	Right-click on Optimetrics in Project Manager and select Analyze>All from context menu								
Parameters	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>None</td> <td></td> <td></td> </tr> </tbody> </table>			Name	Type	Description	None		
Name	Type	Description							
None									
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	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td><SetupName></td><td>String</td><td>Name of the setup to be solved</td></tr></tbody></table>			Name	Type	Description	<SetupName>	String	Name of the setup to be solved
Name	Type	Description							
<SetupName>	String	Name of the setup to be solved							
Return Value	None								

Python Syntax	SolveSetup (<SetupName>)
Python Example	<pre>oModule.SolveSetup ("OptimizationSetup1")</pre>

VB Syntax	SolveSetup <SetupName>
VB Example	<pre>oModule.SolveSetup "OptimizationSetup1"</pre>

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
Return Value	None		

Python Syntax	EditSetup (<SetupName>, <ParametricParams>)
Python Example	See <code>EditSetup [Optimization]</code>

VB Syntax	EditSetup <SetupName>, <ParametricParams>
VB Example	See <code>EditSetup [Optimization]</code>

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.
Return Value	None		

Python Syntax	ExportParametricSetupTable (<SetupName>, <filePath>)
Python Example	<code>oModule.ExportParametricSetupTable('ParametricSetup1', 'E:/Files/ParametricSetup1_Table.csv')</code>

VB Syntax	ExportParametricSetupTable <SetupName>, <filePath>
VB Example	<code>obj.ExportParametricSetupTable "ParametricSetup1", "E:/Files/ParametricSetup1_Table.csv"</code>

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	SweepDatas with the same value are synchronized.
	<SweepOps>		Array("NAME:Sweep Operations", "<OpType>:=, Array(<VarValue>, ..., <VarValue>), ...

		<OpType>:=, Array(<VarValue>, ..., <VarValue>))
	<OpType>	String The sweep operation type.
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], ["NAME:SweepDefinition", _ "Variable:=", "\$length", _ "Data:=", "LIN 12mm 17mm 2.5mm", _ "OffsetF1:=", false, _ "Synchronize:=", 0]]) </pre>

```
"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, _ "Synchronize:=", 0), Array("NAME:SweepDefinition", _ "Variable:=", "\$length", _ "Data:=", "LIN 8mm 12mm 2mm", _ "OffsetF1:=", false, _ "Synchronize:=", 0)), Array("NAME:Sweep Operations")) </pre>

```
Array("NAME:Goals", _  
    Array("NAME:Goal", _  
        "Solution:=", "Setup1 : LastAdaptive", _  
        "Calculation:=", "returnloss", _  
        "Context:=", "", _  
        Array("NAME:Ranges", _  
            "Range:=", Array("Var:=", "Freq", "Type:=", "s", _  
                "Start:=", "8GHz", "Stop:=", "8GHz"))), _  
    Array("NAME:Goal", _  
        "Solution:=", "Setup1 : LastAdaptive", _  
        "Calculation:=", "reflect", _  
        "Context:=", "", _  
        Array("NAME:Ranges", _  
            "Range:=", Array("Var:=", "Freq", "Type:=", "s", _  
                "Start:=", "8GHz", "Stop:=", "8GHz"))))
```

Optimization Script Commands

[EditSetup \[Optimization\]](#)

[InsertSetup \[Optimization\]](#)

EditSetup [Optimization]

Modifies an existing optimization setup.

UI Access	Right-click the setup in the project tree, and then click Properties on the shortcut menu									
Parameters	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><SetupName></td> <td>String</td> <td>Name of the Setup</td> </tr> <tr> <td></td> <td></td> <td></td> </tr> </tbody> </table>	Name	Type	Description	<SetupName>	String	Name of the Setup			
Name	Type	Description								
<SetupName>	String	Name of the Setup								
Return Value	None									

Python Syntax	EditSetup (<SetupName>, <OptimizationParams>)
Python Example	<pre> oModule.EditSetup("OptimizationSetup1", ["NAME:OptimizationSetup1", "UseFastCalculationUpdateAlgo:=", False, "FastCalcOptCtrlledByUser:=", False, ".IsEnabled:=", True, "SaveSolutions:=", False, ["NAME:StartingPoint"], "Optimizer:=", "Quasi Newton", []) </pre>

```
"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, "FastCalcOptCtrlledByUser:=", false, "IsEnabled:=", true, "SaveSolutions:=", false, Array("NAME:StartingPoint"), "Optimizer:=", </pre>

```
"Quasi Newton", Array("NAME:AnalysisStopOptions", "StopForNumIteration:=", _  
    true, "StopForElapsTime:=", false,  
    "StopForSlowImprovement:=", false, "StopForGrdTolerance:=", _  
    false, "MaxNumIteration:=", 1000, "MaxSolTimeInSec:=",  
    3600, "RelGradientTolerance:=", _  
    0, "MinNumIteration:=", 10), "CostFuncNormType:=",  
    "L2", "PriorPSetup:=", "", "PreSolvePSetup:=", _  
    true, Array("NAME:Variables"), Array("NAME:LCS"),  
    Array("NAME:Goals", Array("NAME:Goal", "ReportType:=", _  
        "Standard", "Solution:=", "TR", Array("NAME:SimValueContext",  
        "SimValueContext:=", Array( _  
            1, 0, 2, 0, false, false, -1, 1, 0, 1, 1, "", 0, 0)),  
        "Calculation:=", "Time", "Name:=", _  
        "Time", Array("NAME:Ranges", "Range:=",  
        Array("Var:=", "Time", "Type:=", "a")), "Condition:=", _  
        "==" , Array("NAME:GoalValue", "GoalValueType:=",  
        "Independent", "Format:=", _  
        "Real/Imag", "bG:=", Array("v:=", "[1;]")),  
        "Weight:=", "[1;"])), "Acceptable_Cost:=", _  
        0, "Noise:=", 0.0001, "UpdateDesign:=", false,
```

```
"UpdateIteration:=", 5, "KeepReportAxis:=", _  
true, "UpdateDesignWhenDone:=", true)
```

InsertSetup [Optimization]

Use: Inserts a new optimization setup.

UI Access	Right-click the Optimetrics folder in the project tree, and then click Add>Optimization on the shortcut menu.		
Parameters	Name <code><OptimizationParams></code>	Type Array	Description <pre>Array("NAME:<SetupName>", "SaveFields:=", <SaveField>, <StartingPoint>, "Optimizer:=", <Optimizer>, "MaxIterations:=", <MaxIter>, "PriorPSetup:=", <PriorSetup>, "PreSolvePSetup:=", <Preceed>, <OptimizationVars>, <Constraint>, Array("NAME:Goals", Array("NAME:Goal", <OptiGoalSpec>, <OptimizationGoalSpec>), ... Array("NAME:Goal", <OptiGoalSpec>, <OptimizationGoalSpec>)), "Acceptable_Cost:=", <AcceptableCost>, "Noise:=", <Noise>, "UpdateDesignWhenDone:=", <UpdateDesign>")</pre>
	<code><OptimizationVars></code>	Array	<pre>Array ("NAME:Variables", "VarName:=", Array ("i:=", <IncludeVar>, "Min:=", <MinV>, "Max:=", <MaxV>, "MinStep:=", <MinStepV>, "MaxStep:=", <MaxStepV>),</pre>

	 "VarName:=", Array("i:=", <IncludeVar>, "Min:=", <MinV>, "Max:=", <MaxV>, "MinStep:=", <MinStepV>, "MaxStep:=", <MaxStepV>))
<MinStepV>	VarValue	The minimum step of the variable.
<MaxStepV>	VarValue	The maximum step of the variable.
<AcceptableCost>	Double	The acceptable cost value for the optimizer to stop.
<Noise>	Double	The noise of the design.
<UpdateDesign>	Boolean	Specifies whether or not to apply the optimal variation to the design after the optimization is done.
<OptimizationGoalSpec>	Array	"Condition:=", <OptimizationCond>, Array("NAME:GoalValue", "GoalValeType:=", <GoalValueType>, "Format:=", <GoalValueFormat>, "bG:=", Array("v:=", <GoalValue>)), "Weight:=", <Weight>)
<OptimizationCond>	String	Either "<=", "==" , or ">="
<GoalValueType>	String	Either "Independent" or "Dependent"
<GoalValueFormat>	String	Either "Real/Imag" or "Mag/Ang".
<GoalValue>	String	Value in string. Value can be a real number, complex number, or expression.
Return Value	None	

Command: Right-click the **Optimetrics** folder in the project tree, and then click **Add>Optimization** on the shortcut menu.

Syntax: InsertSetup "OptiOptimization", <OptimizationParams>

Return Value: None

Parameters: <OptimizationParams>

```
Array("NAME:<SetupName>", "SaveFields:=",
<SaveField>, <StartingPoint>, "Optimizer:=",
<Optimizer>,
"MaxIterations:=", <MaxIter>, "PriorPSetup:=",
<PriorSetup>, "PreSolvePSetup:=", <Preceed>,
<OptimizationVars>, <Constraint>,
Array("NAME:Goals", Array("NAME:Goal",
<OptiGoalSpec>, <OptimizationGoalSpec>), ...
Array("NAME:Goal", <OptiGoalSpec>,
<OptimizationGoalSpec>)),
"Acceptable_Cost:=", <AcceptableCost>, "Noise:=",
<Noise>, "UpdateDesignWhenDone:=", <UpdateDesign"

<OptimizationVars>
Array("NAME:Variables", "VarName:=", Array("i:=",
<IncludeVar>, "Min:=", <MinV>, "Max:=", <MaxV>,
"MinStep:=", <MinStepV>, "MaxStep:=", <MaxStepV>),
..... "VarName:=", Array("i:=", <IncludeVar>, "Min:=", <MinV>, "Max:=", <MaxV>,
"MinStep:=", <MinStepV>, "MaxStep:=", <MaxStepV>))

<MinStepV>
```

Type : <VarValue>

The minimum step of the variable.

<MaxStepV>

Type: <VarValue>

The maximum step of the variable.

<AcceptableCost>

Type: <double>

The acceptable cost value for the optimizer to stop.

<Noise>

Type: <double>

The noise of the design.

<UpdateDesign>

Type: <bool>

Specifies whether or not to apply the optimal variation to the design after the optimization is done.

<OptimizationGoalSpec>

```
"Condition:=", <OptimizationCond>,
Array("NAME:GoalValue", "GoalValueType:=",
<GoalValueType>,
"Format:=", <GoalValueFormat>, "bG:=",
Array("v:=", <GoalValue>)), "Weight:=", <Weight>)

<OptimizationCond>
Type: <string>
Either "<=", "==" or ">="

<GoalValueType>
Type: <string>
Either "Independent" or "Dependent"

<GoalValueFormat>
Type:<string>
Either "Real/Imag" or "Mag/Ang".

<GoalValue>
Type: <string>
Value in string. Value can be a real number, complex number, or expression.
```

VB Example:

```
oModule.InsertSetup "OptiOptimization",_
    Array("NAME:OptimizationSetup1", _
        "SaveFields:=", false, _
        Array("NAME:StartingPoint", "$length:=", "8mm", _
            "$width:=", "14.5mm"), _
        "Optimizer:=", "Quasi Newton", _
        "MaxIterations:=", 100, _
        "PriorPSetup:=", "ParametricSetup1", _
        "PreSolvePSetup:=", true, _
        Array("NAME:Variables", _
            "$length:=", Array("i:=", true, "Min:=", "6mm", _
                "Max:=", "18mm", _
                "MinStep:=", "0.001mm", "MaxStep:=", _
                "1.2mm"), _
            "$width:=", Array("i:=", true, "Min:=", _
                "6.5mm", "Max:=", "19.5mm", _
                "MinStep:=", "0.001mm", "MaxStep:=", _
                "1.3mm")), _
        Array("NAME:LCS"))
```

```
Array("NAME:Goals", _  
Array("NAME:Goal", _  
"Solution:=", "Setup1 : LastAdaptive", _  
"Calculation:=", "reflect", _  
"Context:=", "", _  
Array("NAME:Ranges", _  
"Range:=", Array("Var:=", "Freq", _  
"Type:=", "s", _  
"Start:=", "8GHz", "Stop:=", "8GHz")), _  
"Condition:=", "<=", _  
Array("NAME:GoalValue", _  
"GoalValueType:=", "Independent", _  
"Format:=", "Real/Imag", _  
"bG:=", Array("v:=", "[0.0001"])), _  
"Weight:=", "[1]),  
"Acceptable_Cost:=", 0.0002, _  
"Noise:=", 0.0001, _  
"UpdateDesign:=", true, _  
"UpdateIteration:=", 5, _  
"KeepReportAxis:=", true, _  
"UpdateDesignWhenDone:=", true)
```

Python Syntax	InsertSetup ("OptiOptimization", <OptimizationParams>)
Python Example	<pre>oModule.InsertSetup "OptiOptimization",_ ["NAME:OptimizationSetup1", _ "SaveFields:=", false, _ ["NAME:StartingPoint", "\$length:=", "8mm",_ "\$width:=", "14.5mm"], _ "Optimizer:=", "Quasi Newton", _ "MaxIterations:=", 100, _ "PriorPSetup:=", "ParametricSetup1", _ "PreSolvePSetup:=", true, _ ["NAME:Variables", _ "\$length:=", [{"i:=", true, "Min:=", "6mm",_ "Max:=", "18mm", _} "MinStep:=", "0.001mm", "MaxStep:=", _ "1.2mm"], _ "\$width:=", [{"i:=", true, "Min:=", _} "6.5mm", "Max:=", "19.5mm", _]</pre>

```
"MinStep:=", "0.001mm", "MaxStep:=", __
"1.3mm"]], __
["NAME:LCS"], __
["NAME:Goals"], __
["NAME:Goal"], __
"Solution:=", "Setup1 : LastAdaptive", __
"Calculation:=", "reflect", __
"Context:=", "", __
["NAME:Ranges"], __
"Range:=", ["Var:=", "Freq", __
>Type:=", "s", __
"Start:=", "8GHz", "Stop:=", "8GHz"]], __
"Condition:=", "<=", __
["NAME:GoalValue"], __
"GoalValueType:=", "Independent", __
"Format:=", "Real/Imag", __
"bG:=", ["v:=", "[0.0001]"]], __
"Weight:=", "[1]"], __
"Acceptable_Cost:=", 0.0002, __
"Noise:=", 0.0001, __
```

```
"UpdateDesign:=", true, _  
"UpdateIteration:=", 5, _  
"KeepReportAxis:=", true, _  
"UpdateDesignWhenDone:=", true)
```

VB Syntax	InsertSetup "OptiOptimization", <OptimizationParams>
VB Example	<pre>oModule.InsertSetup "OptiOptimization",_ Array("NAME:OptimizationSetup1",_ "SaveFields:=", false, _ Array("NAME:StartingPoint", "\$length:=", "8mm",_ "\$width:=", "14.5mm"),_ "Optimizer:=", "Quasi Newton", _ "MaxIterations:=", 100, _ "PriorPSetup:=", "ParametricSetup1", _ "PreSolvePSetup:=", true, _ Array("NAME:Variables", _ "\$length:=", Array("i:=", true, "Min:=", "6mm", _ "Max:=", "18mm", _</pre>

```
"MinStep:=", "0.001mm", "MaxStep:=", __
"1.2mm"), __
"$width:=", Array("i:=", true, "Min:=", __
"6.5mm", "Max:=", "19.5mm", __
"MinStep:=", "0.001mm", "MaxStep:=", __
"1.3mm")), __
Array("NAME:LCS"), __
Array("NAME:Goals"), __
Array("NAME:Goal", __
"Solution:=", "Setup1 : LastAdaptive", __
"Calculation:=", "reflect", __
"Context:=", "", __
Array("NAME:Ranges", __
"Range:=", Array("Var:=", "Freq", __
>Type:=", "s", __
"Start:=", "8GHz", "Stop:=", "8GHz")), __
"Condition:=", "<=", __
Array("NAME:GoalValue", __
"GoalValueType:=", "Independent", __
"Format:=", "Real/Imag", __
```

```
"bG:=", Array("v:=", "[0.0001]"), _  
"Weight:=", "[1]"), _  
"Acceptable_Cost:=", 0.0002, _  
"Noise:=", 0.0001, _  
"UpdateDesign:=", true, _  
"UpdateIteration:=", 5, _  
"KeepReportAxis:=", true, _  
"UpdateDesignWhenDone:=", true)
```

Sensitivity Script Commands

[EditSetup \[Sensitivity\]](#)

[InsertSetup \[Sensitivity\]](#)

EditSetup [Sensitivity]

Modifies an existing sensitivity setup.

UI Access	Right-click the setup in the project tree, and then click Properties on the shortcut menu		
Parameters	Name <SetupName>	Type String	Description Name of the Setup
Return Value	None		

Python Syntax	EditSetup (<SetupName>, <SensitivityParams>)
Python Example	<pre> oModule.EditSetup("OptimizationSetup1", ["NAME:OptimizationSetup1", "UseFastCalculationUpdateAlgo:=", False, "FastCalcOptCtrlledByUser:=", False, "IsEnabled:=", True, "SaveSolutions:=", False, ["NAME:StartingPoint"], "Optimizer:=", "Quasi Newton", ["NAME:AnalysisStopOptions", "StopForNumIteration:=", True, "StopForElapsTime:=", False, "StopForSlowImprovement:=", False, "StopForGrdTolerance:=", False, "MaxNumIteration:=", 1001, "MaxSolTimeInSec:=", 3600, "RelGradientTolerance:=", 0,]]) </pre>

```
"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, "FastCalcOptCtrlledByUser:=", False, "IsEnabled:=", True, "SaveSolutions:=", False, Array("NAME:StartingPoint"), "Optimizer:=", "Quasi Newton", Array(</pre>

```
"NAME:AnalysisStopOptions",
"StopForNumIteration:=", True,
"StopForElapsTime:=", False,
"StopForSlowImprovement:=", False,
"StopForGrdTolerance:=", False,
"MaxNumIteration:=", 1001,
"MaxSolTimeInSec:=", 3600,
"RelGradientTolerance:=", 0,
"MinNumIteration:=", 10
),
"CostFuncNormType:=", "L2",
"PriorPSetup:=", "",
"PreSolvePSetup:=", True,
Array(
"NAME:Variables"
),
Array(
"NAME:LCS"
),
Array(
"NAME:Goals",
```

```
Array(
    "NAME:Goal",
    "ReportType:=", "Standard",
    "Solution:=", "TR3",
    Array(
        "NAME:SimValueContext",
        "SimValueContext:=",
        Array(1,0,2,0,False,False,-1,1,0,1,1,"",0,0)
    ),
    "Calculation:=", "mag(DIFF1.VAL)",
    "Name:=", "DIFF1.VAL",
    Array(
        "NAME:Ranges",
        "Range:=", Array("Var:=", "Time", "Type:=", "a")
    ),
    "Condition:=", "==",
    Array(
        "NAME:GoalValue",
        "GoalValueType:=", "Independent",
```

```
"Format:=", "Real/Imag",
"bG:=", Array("v:=", "Array(1;])"),
),
"Weight:=", "Array(1;]")
)
),
"Acceptable_Cost:=", 0,
"Noise:=", 0.0001,
"UpdateDesign:=", False,
"UpdateIteration:=", 5,
"KeepReportAxis:=", True,
"UpdateDesignWhenDone:=", True
))
```

InsertSetup [Sensitivity]

Inserts a new sensitivity setup.

UI Access	Right-click Optimetrics in the project tree, and then click Add>Sensitivity on the shortcut menu.						
Parameters	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><SensitivityParams></td> <td>Array</td> <td>Array("NAME:<SetupName>", "SaveFields:=", <SaveField>, <StartingPoint>, "MaxIterations:=", <MaxIter>, "PriorPSetup:=", <PriorSetup>,</td> </tr> </tbody> </table>	Name	Type	Description	<SensitivityParams>	Array	Array("NAME:<SetupName>", "SaveFields:=", <SaveField>, <StartingPoint>, "MaxIterations:=", <MaxIter>, "PriorPSetup:=", <PriorSetup>,
Name	Type	Description					
<SensitivityParams>	Array	Array("NAME:<SetupName>", "SaveFields:=", <SaveField>, <StartingPoint>, "MaxIterations:=", <MaxIter>, "PriorPSetup:=", <PriorSetup>,					

		<pre>"PreSolvePSetup:=", <Preceed>, <SensitivityVars>, <Constraint>, Array("NAME:Goals", Array("NAME:Goal", <OptiGoalSpec>), ..., Array("NAME:Goal", <OptiGoalSpec>)), "Primary Goal:=". <PrimaryGoalID>, "PrimaryError:=", <PrimaryError>)</pre>
	<SensitivityVars>	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
	<PrimaryError>	Double
Return Value	None	

Python Syntax	InsertSetup ("OptiSensitivity", <SensitivityParams>)
Python Example	<pre>oModule.InsertSetup ("OptiSensitivity", _</pre>

```
[ "NAME:SensitivitySetup1",_
  "SaveFields:=", true,_
  [ "NAME:StartingPoint"],_
  "MaxIterations:=", 20,_
  "PriorPSetup:=", "",_
  "PreSolvePSetup:=", true,_
  [ "NAME:Variables"],_
  [ "NAME:LCS"],_
  "NAME:Goals",_
  [ "NAME:Goal",_
    "Solution:=", "Setup1 : LastAdaptive",_
    "Calculation:=", "returnloss",_
    "Context:=", "",_
    [ "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"))</pre>

```
Array("NAME:Goals",  
      Array("NAME:Goal", _  
             "Solution:=", "Setup1 : LastAdaptive",_  
             "Calculation:=", "returnloss",_  
             "Context:=", "",_  
             Array("NAME:Ranges",_  
                   "Range:=", Array("Var:=", "Freq", " _  
                           Type:=", "s",_  
                           "Start:=", "8GHz", "Stop:=", "8GHz"))),_  
             Array("NAME:Goal",_  
                   "Solution:=", "Setup1 : LastAdaptive",_  
                   "Calculation:=", "reflect",_  
                   "Context:=", "",_  
                   Array("NAME:Ranges",_  
                         "Range:=", Array("Var:=", "Freq", _  
                                         Type:=", "s",_  
                                         "Start:=", "8GHz", "Stop:=", "8GHz"))),_  
                   "Primary Goal:=", 1,_  
                   "PrimaryError:=", 0.001)
```

Statistical Script Commands

[EditSetup \[Statistical\]](#)

[InsertSetup Statistical](#)

EditSetup [Statistical]

Modifies an existing statistical setup.

UI Access	Right-click the setup in the project tree, and clickProperties on the shortcut menu.		
Parameters	Name <SetupName>	Type String	Description 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("optiguides")
Set oDesign = oProject.SetActiveDesign("HFSSModel1")
Set oModule = oDesign.GetModule("Optimetrics")

oModule.EditSetup "StatisticalSetup1", Array("NAME:StatisticalSetup1",
Array("NAME:ProdOptiSetupData",
"SaveFields:=", true, "CopyMesh:=", false),
Array("NAME:StartingPoint", "$length:=", "7.824547736mm",
"$width:=", "14.8570192mm"),
"MaxIterations:=", 50,
"PriorPSetup:=", "",

Array("NAME:Variables",
"$length:=", Array("i:=", true,
"int:=", false,
"Dist:=", "Uniform",
"Tol:=", "10%", "StdD:=", "0.2mm", "Min:=", "-3",
"Max:=", "3", "Shape:=", "1", "Scale:=", "0.04mm",
"Location:=", "0.4mm",
"Dataset:=", "", "LatinHypercube:=", "true", "VarMin:=", "0.2mm", "VarMax:=", "0.6mm", "Prob:=",
```

```
"0.01",
"Mean:=", "0.4mm"),

"$width:=", Array("i:=", true,
"int:=", false,
"Dist:=", "Gaussian",
"Tol:=", "10%",
"StdD:=", "0.2mm",
"Min:=", "-3", "Max:=", "3",
"Shape:=", "1",
"Scale:=", "0.04mm",
"Location:=", "0.4mm",
"Dataset:=", "",
"LatinHypercube:=", "true",
"VarMin:=", "0.2mm", "VarMax:=", "0.6mm",
"Prob:=", "0.02",
"Mean:=", "0.4mm")),

Array("NAME:Goals", Array("NAME:Goal",
"ReportType:=", "Modal Solution Data",
"Solution:=", "Setup1 : PortOnly",
Array("NAME:SimValueContext", "Domain:=", "Sweep"),
"Calculation:=", "returnloss",
"Name:=", "returnloss",

Array("NAME:Ranges",
"Range:=", Array("Var:=", "Freq",
"Type:=", "s",
"Start:=", "8.2GHz", "Stop:=", "0"))),

Array("NAME:Goal",
"ReportType:=", "Modal Solution Data",
"Solution:=", "Setup1 : PortOnly",
Array("NAME:SimValueContext",
```

```
"Domain:=", "Sweep"),
"Calculation:=", "reflect",
"Name:=", "reflect",
Array("NAME:Ranges",
"Range:=", Array("Var:=", "Freq",
>Type:=", "s",
"Start:=", "8.2GHz", "Stop:=", "0"))))
```

InsertSetup [Statistical]

Inserts a new statistical setup.

UI Access	Right-click Optimetrics in the project tree, and then click Add>Statistical on the shortcut menu.		
Parameters	Name	Type	Description
	<StatisticalParams>	Array	Array("NAME:<SetupName>", "SaveFields:=", <SaveField>, <StartingPoint>, "MaxIterations:=", <MaxIter>, "PriorPSetup:=", <PriorSetup>, "PreSolvePSetup:=", <Preceed>, <StatisticalVars>, Array("NAME:Goals", Array("NAME:Goal", <OptiGoalSpec>), ..., Array("NAME:Goal", <OptiGoalSpec>)))
	<StatisticalVars>	Array	Array("NAME:Variables", "VarName:=", Array("i:=", <IncludeVar>, "Dist:=", <DistType>, "Tol:=", <Tolerance>, "StdD:=", <StdD>, "Min:=", <MinCutoff>, "Max:=", <MaxCutoff>, ... "VarName:=", Array("i:=", <IncludeVar>, "Dist:=",

		<DistType>, "Tol:=", <Tolerance>, "StdD:=", <StdD>, "Min:=", <MinCutoff>, "Max:=", <MaxCutoff>))
<DistType>	String	Distrbution can be "Gaussian" or "Uniform".
<Tolerance>	VarValue	The tolerance for the variable when distribution is Uniform
<StdD>	VarValue	The standard deviation for the variable when distribution is Gaussian.
<MinCutoff>	Double	The minimum cut-off for the variable when distribution is Gaussian.
<MaxCutoff>	Double	The maximum cut-off for the variable when distribution is Gaussian.
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", _</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"}]]])

```

VB Syntax	InsertSetup "OptiStatistical", <StatisticalParams>
VB Example	<pre> oModule.InsertSetup "OptiStatistical", _ Array("NAME:StatisticalSetup1", _</pre>

```
"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"))))
```

Example:

```
Dim oAnsoftApp  
Dim oDesktop  
Dim oProject  
Dim oDesign  
Dim oEditor  
Dim oModule  
  
Set oAnsoftApp = CreateObject("Ansoft.ElectronicsDesktop")  
Set oDesktop = oAnsoftApp.GetAppDesktop()  
oDesktop.RestoreWindow  
  
Set oProject = oDesktop.SetActiveProject("OptimTee")  
Set oDesign = oProject.SetActiveDesign("TeeModel")  
  
oDesign.ChangeProperty Array("NAME:AllTabs",  
Array("NAME:LocalVariableTab",  
Array("NAME:PropServers", "LocalVariables"),  
Array("NAME:ChangedProps",  
Array("NAME:offset",  
Array("NAME:Statistical", "Included:=", true))))  
  
Set oModule = oDesign.GetModule("Optimetrics")
```

```
oModule.InsertSetup "OptiStatistical", Array("NAME:StatisticalSetup1",
Array("NAME:ProdOptiSetupData",
"SaveFields:=", false, "CopyMesh:=", false),
Array("NAME:StartingPoint", "offset:=", "0in"),
"MaxIterations:=", 50, "PriorPSetup:=", "",

Array("NAME:Variables",
"offset:=", Array("i:=", true,
"int:=", false,
"Dist:=", "Gaussian",
"Tol:=", "10%",
"StdD:=", ".5in",
"Min:=", "-3",
"Max:=", "3",
"Shape:=", "1",
"Scale:=", "0in",
"Location:=", "0in",
"Dataset:=", "",
"LatinHypercube:=", "true",
"VarMin:=", "-lin",
"VarMax:=", "lin",
"Prob:=", "0.01",
"Mean:=", "0in")),

Array("NAME:Goals", Array("NAME:Goal",
"ReportType:=", "Modal Solution Data",
"Solutions:=", "Setup1 : LastAdaptive", Array("NAME:SimValueContext"),
"Calculation:=", "Power11",
"Name:=", "Power11",
Array("NAME:Ranges",
"Range:=", Array("Var:=", "Freq", "Type:=", "d",
"DiscreteValues:=", "10GHz")))))
```

For Q3D Extractor and Circuit the command details are as follows:

Inserts a new statistical setup.

Command: Right-click **Optimetrics** in the project tree, and then click **Add>Statistical** on the shortcut menu.

Syntax: InsertSetup "OptiStatistical", <StatisticalParams>

Return Value: None

Parameters: <StatisticalParams>

```
Array("NAME:<SetupName>", "SaveFields:=",
<SaveField>, <StartingPoint>, "MaxIterations:=",
<MaxIter>, "PriorPSetup:=", <PriorSetup>,
"PreSolvePSetup:=", <Preceed>, <StatisticalVars>,
Array("NAME:Goals", Array("NAME:Goal",
<OptiGoalSpec>), ..., Array("NAME:Goal",
<OptiGoalSpec>)) ,
<StatisticalVars>
Array("NAME:Variables",
"VarName:=", Array("i:=", <IncludeVar>, "Dist:=",
<DistType>, "Tol:=", <Tolerance>,
"StdD:=", <StdD>, "Min:=", <MinCutoff>, "Max:=",
<MaxCutoff>, ...
"VarName:=", Array("i:=", <IncludeVar>, "Dist:=",
```

```
<DistType>, "Tol:=", <Tolerance>, "StdD:=",
<StdD>, "Min:=", <MinCutoff>, "Max:=",
<MaxCutoff>))
```

Parameters:

```
<DistType>
```

Type : <string>

Distrbution can be "Gaussian" or "Uniform".

```
<Tolerance>
```

Type: <VarValue>

The tolerance for the variable when distribution is Uniform.

```
<StdD>
```

Type: <VarValue>

The standard deviation for the variable when distribution is Gaussian.

```
<MinCutoff>
```

Type: <double>

The minimum cut-off for the variable when distribution is Gaussian.

```
<MaxCutoff>
```

Type: <double>

The maximum cut-off for the variable when distribution is Gaussian.

Example: oModule.InsertSetup "OptiStatistical", _

```
Array("NAME:StatisticalSetup1", _  
    "SaveFields:=", true, _  
    Array("NAME:StartingPoint"), _  
    "MaxIterations:=", 50, _  
    "PriorPSetup:=", "", _  
    Array("NAME:Variables"), _  
    Array("NAME:Goals", _  
        Array("NAME:Goal", _  
            "Solution:=", "Setup1 : LastAdaptive", _  
            "Calculation:=", "returnloss", _  
            "Context:=", "", _  
            Array("NAME:Ranges", _  
                "Range:=", Array("Var:=", "Freq", _  
                    "Type:=", "s", _  
                    "Start:=", "8GHz", "Stop:=", "8GHz"))), _  
        Array("NAME:Goal", _  
            "Solution:=", "Setup1 : LastAdaptive", _  
            "Calculation:=", "reflect", _  
            "Context:=", "", _  
            Array("NAME:Ranges", _  
                "Range:=", Array("Var:=", "Freq", "Type:=", _
```

```
"s", "Start:=", "8GHz", "Stop:=", "8GHz"))))
```

17 - Solutions Module Script Commands

Solutions commands should be executed by the "Solutions" module.

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

```
oModule.CommandName <args>
```

[DeleteSolutionVariation](#)

ExportNetworkData

GetAvailableVariations

[TDROnReport](#)

DeleteSolutionVariation

Deletes all solution data for specific solutions and design variations. This is obsolete and is supported only for backward compatibility. You should use DeleteFullVariation.

UI Access	Right-click on Results , select Browse Solutions... , click Delete button in the dialog.		
Parameters	Name <SoluParams>	Type Array	Description Structured array. Array (<DataSpecifierArray>, ...)
	<DataSpecifierArray>	Array	Structured array. Array (<DesignVariationKey>, <SetupName>, <SolnName>)
Return Value	None.		

Python Syntax	DeleteSolutionVariation(<SoluParam>)
Python Example	<pre>oModule.DeleteSolutionVariation([["width='2in'", "Setup1", "Adaptive_1"], ["width='2in'", "Setup1", "Sweep1"]])</pre>

VB Syntax	DeleteSolutionVariation <SoluParam>
VB Example	<pre>oModule.DeleteSolutionVariation Array(_ Array("width='2in'", "Setup1", "Adaptive_1"), _ Array("width='2in'", "Setup1", "Sweep1"))</pre>

EditSources

Indicates which source excitations should be used for fields post-processing.

ExportNMFDATA [HFSS]

Exports matrix solution data to a file in neutral model format. Available only for Driven solution types with ports. Variables can be held constant by setting their values in the variation field. For example: "length='50mm' width='30mm'". All other independent variables will be treated as NMF parameters.

UI Access	N/A		
Parameters	Name <SolnSelectionArray>	Type Array	Description Array of selected solutions.

	<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.
	<DesignVariationKey>	String	Design variation.
	<DoRenorm>	Boolean	Specifies whether to renormalize the data before export.
	<RenormImped>	Double	Real impedance value in ohms, for renormalization. Required in syntax, but ignored if DoRenorm is false.
	<Pass>	Integer	Optional. The pass to export. This is ignored if the sourceName is a frequency sweep. Leaving out this value or specifying -1 gets all passes.
Return Value	None.		

Python Syntax	ExportNMFDATA(<SolnSelectionArray>, <OutFile>, <FreqsArray> <DesignVariationKey>, <DoRenorm>, <RenormImped>)
Python Example	<pre>oModule.ExportNMFDATA(["Setup1:Sweep1"], "c:\mydir\out.nmf", ["all"], "", False, 0)</pre>

VB Syntax	ExportNMFDATA <SolnSelectionArray>, <OutFile>, <FreqsArray> <DesignVariationKey>, <DoRenorm>, <RenormImped>
VB Example	<pre>oModule.ExportNMFDATA Array("Setup1:Sweep1"), _ "c:\mydir\out.nmf", Array("all"), "", false, 0</pre>

ExportTransientData

Exports transient solution data to a file.

UI Access	HFSS > Results > Solution Data..., then click Export.																		
Parameters	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><SoluName></td> <td>String</td> <td>Name of specified solution.</td> </tr> <tr> <td><Variation></td> <td>String</td> <td>Design variation key. Pass empty string for the current nominal variation.</td> </tr> <tr> <td><FileName></td> <td>String</td> <td>Full path of output file.</td> </tr> <tr> <td><DataType></td> <td>String</td> <td>Optional. Type of output data.</td> </tr> <tr> <td><DataFormat></td> <td>String</td> <td> Optional. Format of output data. .csv - Comma Delimited Data. .tab - Tab Delimited Data files. .dat - Ansys Plot Data files .txt - Post Processing Files </td> </tr> </tbody> </table>	Name	Type	Description	<SoluName>	String	Name of specified solution.	<Variation>	String	Design variation key. Pass empty string for the current nominal variation.	<FileName>	String	Full path of output file.	<DataType>	String	Optional. Type of output data.	<DataFormat>	String	Optional. Format of output data. .csv - Comma Delimited Data. .tab - Tab Delimited Data files. .dat - Ansys Plot Data files .txt - Post Processing Files
Name	Type	Description																	
<SoluName>	String	Name of specified solution.																	
<Variation>	String	Design variation key. Pass empty string for the current nominal variation.																	
<FileName>	String	Full path of output file.																	
<DataType>	String	Optional. Type of output data.																	
<DataFormat>	String	Optional. Format of output data. .csv - Comma Delimited Data. .tab - Tab Delimited Data files. .dat - Ansys Plot Data files .txt - Post Processing Files																	
Return Value	None.																		

Python Syntax	ExportTransientData(<SoluName>, <Variation>, <FileName>)
Python Example	oModule.ExportTransientData("Setup1:Transient", "", "C:/transient_sol1.csv")

VB Syntax	ExportTransientData <SoluName>, <Variation>, <FileName>
VB Example	oModule.ExportTransientData "Setup1:Transient", "", "C:/transient_sol1.csv"

FFTOnReport

Performs an FFT on a selected report.

UI Access	Right-click on Results in the project tree, select Perform FFT On Report...												
Parameters	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><ReportName></td> <td>String</td> <td>Name of specified report.</td> </tr> <tr> <td><WindowName></td> <td>String</td> <td>Name of window to apply for FFT. Possible values are "Rectangular", "Tri", "Van Hann", "Hamming", "Blackman", "Lanczos", "Weber", "Welch".</td> </tr> <tr> <td><Function></td> <td>String</td> <td>Function to apply on transformation values. Possible values are "none", "ang_deg", "ang_rad", "arg", "cang_deg", "cang_rad", "dB", "dB1 normalize", "dB2normalize", "dBc", "im", "mag", "normalize", "re".</td> </tr> </tbody> </table>	Name	Type	Description	<ReportName>	String	Name of specified report.	<WindowName>	String	Name of window to apply for FFT. Possible values are "Rectangular", "Tri", "Van Hann", "Hamming", "Blackman", "Lanczos", "Weber", "Welch".	<Function>	String	Function to apply on transformation values. Possible values are "none", "ang_deg", "ang_rad", "arg", "cang_deg", "cang_rad", "dB", "dB1 normalize", "dB2normalize", "dBc", "im", "mag", "normalize", "re".
Name	Type	Description											
<ReportName>	String	Name of specified report.											
<WindowName>	String	Name of window to apply for FFT. Possible values are "Rectangular", "Tri", "Van Hann", "Hamming", "Blackman", "Lanczos", "Weber", "Welch".											
<Function>	String	Function to apply on transformation values. Possible values are "none", "ang_deg", "ang_rad", "arg", "cang_deg", "cang_rad", "dB", "dB1 normalize", "dB2normalize", "dBc", "im", "mag", "normalize", "re".											
Return Value	None.												

Python Syntax	FFTOnReport <ReportName>, <WindowName>, <Function>
Python Example	oModule.FFTOnReport("XY Plot 1", "Rectangular", "dB")

VB Syntax	FFTOnReport(<ReportName>, <WindowName>, <Function>)
VB Example	oModule.FFTOnReport "XY Plot 1", "Rectangular", "dB"

GetAdaptiveSettings

Obtains the adaptive frequency settings of a specified setup.

UI Access	N/A						
Parameters	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><SetupName></td> <td>String</td> <td>Name of specified setup.</td> </tr> </tbody> </table>	Name	Type	Description	<SetupName>	String	Name of specified setup.
Name	Type	Description					
<SetupName>	String	Name of specified setup.					
Return Value	Array of strings represents adaptive frequency settings.						

Python Syntax	GetAdaptiveSettings(<SetupName>)
Python Example	<code>oModule.GetAdaptiveSettings ("Setup1")</code>

VB Syntax	GetAdaptiveSettings <SetupName>
VB Example	<code>oModule.GetAdaptiveSettings "Setup1"</code>

GetAllSourceMagnitudes

Obtains the magnitude values of all defined sources.

UI Access	N/A
Parameters	None.
Return Value	Array of strings containing magnitude values.

Python Syntax	GetAllSourceMagnitudes()
Python Example	<code>oModule.GetAllSourceMagnitudes ()</code>

VB Syntax	GetAllSourceMagnitudes
VB Example	<code>oModule.GetAllSourceMagnitudes</code>

GetAllSourceModes

Obtains mode information of all defined sources.

UI Access	N/A
Parameters	None.
Return Value	Array of strings containing mode information.

Python Syntax	GetAllSourceModes()
Python Example	<code>oModule.GetAllSourceModes ()</code>

VB Syntax	GetAllSourceModes
VB Example	<code>oModule.GetAllSourceModes</code>

GetAllSourcePhases

Obtains the phase values of all defined sources.

UI Access	N/A
Parameters	None.
Return Value	Array of strings containing phase values.

Python Syntax	GetAllSourcePhases()
Python Example	<code>oModule.GetAllSourcePhases ()</code>

VB Syntax	GetAllSourcePhases
VB Example	<code>oModule.GetAllSourcePhases</code>

GetAllSources

Retrieves all sources defined in the current solution setup.

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

Python Syntax	GetAllSources()
Python Example	<code>oModule.GetAllSources ()</code>

VB Syntax	GetAllSources
VB Example	<code>oModule.GetAllSources</code>

GetAntennaParameters

Retrieves antenna parameters defined in a specified setup.

UI Access	N/A								
Parameters	<table border="1"> <tr> <td>Name</td> <td>Type</td> <td>Description</td> </tr> <tr> <td><SetupName></td> <td>String</td> <td>Name of specified setup.</td> </tr> </table>			Name	Type	Description	<SetupName>	String	Name of specified setup.
Name	Type	Description							
<SetupName>	String	Name of specified setup.							
Return Value	Array of strings containing antenna parameters.								

Python Syntax	GetAntennaParameters(<SetupName>)
Python Example	oModule.GetAntennaParameters ("Setup1")

VB Syntax	GetAntennaParameters <SetupName>
VB Example	oModule.GetAntennaParameters "Setup1"

GetFieldType

Gets the field type of a driven modal design solution.

UI Access	N/A
Parameters	None.
Return Value	String containing field type.

Python Syntax	GetFieldType()
Python Example	<code>oModule.GetFieldType ()</code>

VB Syntax	GetFieldType
VB Example	<code>oModule.GetFieldType</code>

GetIncludePortPostProcessing

Determines whether the Include Port Post Processing Effects option is enabled.

UI Access	N/A
Parameters	None.
Return Value	<p>Integer.</p> <ul style="list-style-type: none">• 1 - Include Port Post Processing Effects is enabled.• 0 - Include Port Post Processing Effects is disabled.

Python Syntax	GetIncludePortPostProcessing()
Python Example	<code>oModule.GetIncludePortPostProcessing ()</code>

VB Syntax	GetIncludePortPostProcessing
------------------	------------------------------

VB Example	<code>oModule.GetIncludePortPostProcessing</code>
-------------------	---

GetMultipactionBreakdown

Gets multipaction breakdown from the specified setup.

UI Access	N/A		
Parameters	Name	Type	Description
	<code><SetupName></code>	String	Name of specified setup.
Return Value	Array of string containing the multipaction breakdown.		

Python Syntax	<code>GetMultipactionBreakdown(<SetupName>, <Variation>)</code>
Python Example	<code>oModule.GetMultipactionBreakdown("Setup1", "")</code>

VB Syntax	<code>GetMultipactionBreakdown <SetupName>, <Variation></code>
VB Example	<code>oModule.GetMultipactionBreakdown "Setup1", ""</code>

GetNetworkDataSolution

Gets matrix data solution from a specified setup.

UI Access	N/A		
Parameters	Name	Type	Description
	<code><SoluName></code>	String	Name of solution.

	<code><VariationKey></code>	String	Design variation key. Pass empty string for the current nominal variation.
Return Value	String containing matrix data.		

Python Syntax	<code>GetNetworkDataSolution(<SoluName>, <VariationKey>)</code>
Python Example	<code>oModule.GetNetworkDataSolution("Setup1:Sweep1", "")</code>

VB Syntax	<code>GetNetworkDataSolution <SoluName>, <VariationKey></code>
VB Example	<code>oModule.GetNetworkDataSolution "Setup1:Sweep1", ""</code>

GetNetworkDataSolutionDefinition

Gets definition of network data solution from a specified setup.

UI Access	N/A		
Parameters	Name	Type	Description
	<code><SoluName></code>	String	Name of specified solution.
Return Value	String containing network data definition.		

Python Syntax	<code>GetNetworkDataSolutionDefinition(<SoluName>)</code>
Python Example	<code>oModule.GetNetworkDataSolutionDefinition("Setup1:Sweep1")</code>

VB Syntax	GetNetworkDataSolutionDefinition < <i>SoluName</i> >
VB Example	<code>oModule.GetNetworkDataSolutionDefinition "Setup1:Sweep1"</code>

GetNetworkPostprocSetup

Obtains network post-processing setup.

UI Access	N/A
Parameters	None.
Return Value	String containing post-processing setup.

Python Syntax	GetNetworkPostprocSetup()
Python Example	<code>oModule.GetNetworkPostprocSetup()</code>

VB Syntax	GetNetworkPostprocSetup
VB Example	<code>oModule.GetNetworkPostprocSetup</code>

GetSourceContexts

Obtains sources currently enabled as context in the Edit Sources dialog Source Context tab.

UI Access	N/A
------------------	-----

Parameters	None.
Return Value	Array of enabled source names

Python Syntax	GetSourceContexts()
Python Example	<code>oModule.GetSourceContexts ()</code>

VB Syntax	GetSourceContexts
VB Example	<code>oModule.GetSourceContexts</code>

GetTerminalExcitationType

Obtains the type for terminal excitation in a driven terminal design.

UI Access	N/A
Parameters	None.
Return Value	String containing excitation type.

Python Syntax	GetTerminalExcitationType()
Python Example	<code>oModule.GetTerminalExcitationType ()</code>

VB Syntax	GetTerminalExcitationType
VB Example	<code>oModule.GetTerminalExcitationType</code>

GetTransientSolveTimes

Gets the transient solution solve time.

UI Access	N/A		
Parameters	Name	Type	Description
	<code><SoluName></code>	String	Name of specified solution.
Return Value	Array of strings containing solve time.		

Python Syntax	GetTransientSolveTimes(<SoluName>, <Variation>)
Python Example	<code>oModule.GetTransientSolveTimes("Setup1:Transient1", "")</code>

VB Syntax	GetTransientSolveTimes <SoluName>, <Variation>
VB Example	<code>oModule.GetTransientSolveTimes "Setup1:Transient1", ""</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 <code><SourceId></code>	Type Array	Description Name of modes/terminals to be set as source context.
Return Value	None.		

Python Syntax	<code>SetSourceContexts(<SourceId>)</code>
Python Example	<pre>oModule.SetSourceContexts (["Box1_T1", "Box1_T2", "Box1_T3", "Current1", "IncPWave1"])</pre>

VB Syntax	<code>SetSourceContexts <SourceId></code>
VB Example	<pre>oModule.SetSourceContexts _ Array("Box1_T1", "Box1_T2", "Box1_T3", "Current1", "IncPWave1")</pre>

TDROnReport

Performs a time-domain reflectometry (TDR) analysis on a specified report.

UI Access	Right-click on Results > Perform TDR On Report...			
Parameters	<table border="1"><tr><td>Name <code><ReportName></code></td><td>Type String</td><td>Description Name of specified report.</td></tr></table>	Name <code><ReportName></code>	Type String	Description Name of specified report.
Name <code><ReportName></code>	Type String	Description Name of specified report.		

	<i><IsStep></i>	Boolean	Specifies the input signal as Step or Impulse.
	<i><RiseTime></i>	Double	Specifies rise time with unit in 'ps'.
	<i><WindowName></i>	String	Name of window to apply. Possible values are "Rectangular", "Bartlett", "Blackman", "Hamming", "Hanning", "Kaiser", "Welch".
	<i><WindowWidth></i>	Integer	Width of window in percentage.
	<i><KaiserParams></i>	Double	Optional. If Kaiser window is specified, set corresponding parameters.
Return Value	None.		

Python Syntax	<code>TDROnReport(<ReportName>, <IsStep>, <RiseTime>, <WindowName>, <WindowWidth>, [Optional <KaiserParams>])</code>
Python Example	<code>oModule.TDROnReport ("Rept2DRectFreq", True, 1E-12, "Hanning", 100, 1)</code>

VB Syntax	<code>TDROnReport <ReportName>, <IsStep>, <RiseTime>, <WindowName>, <WindowWidth>, [Optional <KaiserParams>]</code>
VB Example	<code>oModule.TDROnReport "Rept2DRectFreq", true, 1E-12, "Hanning", 100, 1</code>

This page intentionally
left blank.

18 - Field Overlays Module Script Commands

Field overlay commands should be executed by the Field Overlays module, which is called "FieldsReporter" in scripts.

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

```
oModule.CommandName <args>
```

[AddMarkerToPlot](#)

[AddNamedExpr](#)

[DeleteFieldPlot](#)

[EditSurfaceMeshSummaryData](#)

[ExportPlotImageWithViewToFile](#)

[ExportSurfaceMeshSummary](#)

```
ModifyInceptionParameters
```

[RenamePlotFolder](#)

[SetFieldPlotSettings](#)

[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	<table border="1"> <tr> <td>Name</td> <td>Type</td> <td>Description</td> </tr> <tr> <td><Position></td> <td>Array</td> <td>Array containing X, Y and Z coordinates.</td> </tr> </table>			Name	Type	Description	<Position>	Array	Array containing X, Y and Z coordinates.
Name	Type	Description							
<Position>	Array	Array containing X, Y and Z coordinates.							

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 <Location>	Type Array	Description Array of strings containing X,Y, and Z coordinates for the marker.

	<i><PlotName></i>	String	Name of the field plot.
Return Value	None.		

Python Syntax	AddMarkerToPlot(<i><Location></i> , <i><PlotName></i>)
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 <i><Location></i> , <i><PlotName></i>
VB Example	<pre> oModule.AddMarkerToPlot _ Array("0.290455877780914in", _ "-0.616900205612183in", "1.77635683940025e-015in"), _ "Mag_E1" oModule.AddMarkerToPlot _ Array("-0.317279517650604in", _ </pre>

	"1.22481322288513in", "0in"), "Mag_E1"
--	---

AddNamedExpr

Creates a named expression using the expression at the top of the stack.

UI Access	Click Add in the Fields Calculator dialogue.
Parameters	Name Type Description <i><ExprName></i> String Name for the new named expression.
Return Value	None.

Python Syntax	AddNamedExpr(<ExprName>)
Python Example	oModule.AddNamedExpr ("Mag_JxE")

VB Syntax	AddNamedExpr <ExprName>
VB Example	oModule.AddNamedExpr "Mag_JxE"

AddNamedExpression

Creates a named expression using the expression at the top of the stack.

UI Access	Click Add in the Fields Calculator dialogue.
------------------	--

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	oModule.AddNamedExpression ("Mag_JxE", "Fields")

VB Syntax	AddNamedExpression <ExprName>, <FieldType>
VB Example	oModule.AddNamedExpression "Mag_JxE", "Fields"

CalcOp

Performs a calculator operation.

UI Access	Operation commands like Mag , + , etc.
Parameters	Name
	Type
	<OpString>
Return Value	The text on the corresponding calculator button.
	String
	None.

Python Syntax	CalcOp(<OpString>)
Python Example	oModule.CalcOp ("+")

VB Syntax	CalcOp < <i>OpString</i> >
VB Example	<code>oModule.CalcOp "+"</code>

CalcStack

Performs an operation on the stack.

UI Access	Stack operation buttons such as Push and Pop .		
Parameters	Name < <i>OpString</i> >	Type String	Description The text on the corresponding calculator button.
Return Value	None.		

Python Syntax	CalcStack(< <i>OpString</i> >)
Python Example	<code>oModule.CalcStack("push")</code>

VB Syntax	CalcStack < <i>OpString</i> >
VB Example	<code>oModule.CalcStack "push"</code>

CalcWrite

Writes contents of top register to file.

UI Access	N/A												
Parameters	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><FilePath></td> <td>String</td> <td>Path to output and including name of output register file.</td> </tr> <tr> <td><SoluName></td> <td>String</td> <td>Name of solution.</td> </tr> <tr> <td><VariablesArray></td> <td>Array</td> <td>Array of variable names, value pairs.</td> </tr> </tbody> </table>	Name	Type	Description	<FilePath>	String	Path to output and including name of output register file.	<SoluName>	String	Name of solution.	<VariablesArray>	Array	Array of variable names, value pairs.
Name	Type	Description											
<FilePath>	String	Path to output and including name of output register file.											
<SoluName>	String	Name of solution.											
<VariablesArray>	Array	Array of variable names, value pairs.											
Return Value	None.												

Python Syntax	CalcWrite(<FilePath>, <SoluName>, <VariablesArray>)
Python Example	<pre>oModule.CalcWrite("C:\Ansoft\smoothedTemp.fld", "Setup1 : LastAdaptive", ["\$conductivity:=", "50000000"])</pre>

VB Syntax	CalcWrite <FilePath>, <SoluName>, <VariablesArray>
VB Example	<pre>oModule.CalcWrite "C:\Ansoft\smoothedTemp.fld", "Setup1 : LastAdaptive", _ Array("\$conductivity:=", "50000000")</pre>

CalculatorRead

Gets a register file and applies it to the calculator stack.

UI Access	Click Read... in the Fields Calculator dialog.						
Parameters	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><FileName></td> <td>String</td> <td>Path to and including name of input register file.</td> </tr> </tbody> </table>	Name	Type	Description	<FileName>	String	Path to and including name of input register file.
Name	Type	Description					
<FileName>	String	Path to and including name of input register file.					

	<i><SoluName></i>	String	Specified solution to read in.
	<i><FieldType></i>	String	Type of specified field.
	<i><VarArray></i>	Array	Array of variable names, value pairs.
Return Value	None.		

Python Syntax	CalculatorRead(<FileName>, <SoluName>, <FieldType>, <VarArray>)
Python Example	<pre>oModule.CalculatorRead("c:\\example.reg", "Setup1: LastAdaptive", "Fields", ["Freq:=", "10GHz", "Phase:=", "0deg"])</pre>

VB Syntax	CalculatorRead <FileName>, <SoluName>, <FieldType>, <VarArray>
VB Example	<pre>oModule.CalculatorRead _ "c:\\example.reg", _ "Setup1: LastAdaptive", "Fields", _ Array("Freq:=", "10GHz", "Phase:=", "0deg")</pre>

CalculatorWrite

Writes contents of top register to file.

UI Access	Click Write... in the Fields Calculator dialog.												
Parameters	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><code><OutputFilePath></code></td> <td>String</td> <td>Path to and including name of output register file.</td> </tr> <tr> <td><code><SoluNameArray></code></td> <td>Array</td> <td>Array of strings containing solution names.</td> </tr> <tr> <td><code><VarArray></code></td> <td>Array</td> <td>Array of variable names, value pairs.</td> </tr> </tbody> </table>	Name	Type	Description	<code><OutputFilePath></code>	String	Path to and including name of output register file.	<code><SoluNameArray></code>	Array	Array of strings containing solution names.	<code><VarArray></code>	Array	Array of variable names, value pairs.
Name	Type	Description											
<code><OutputFilePath></code>	String	Path to and including name of output register file.											
<code><SoluNameArray></code>	Array	Array of strings containing solution names.											
<code><VarArray></code>	Array	Array of variable names, value pairs.											
Return Value	None.												

Python Syntax	<code>CalculatorWrite(<OutputFilePath>, <SoluNameArray>, <VarArray>)</code>
Python Example	<pre> oModule.CalculatorWrite("c:\test.reg", ["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>

ChangeGeomSettings

Changes the line discretization setting.

UI Access	In the Fields Calculator dialog box, click on Geom Settings...						
Parameters	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><i><LineDiscr></i></td> <td>Integer</td> <td>Line discretization value.</td> </tr> </tbody> </table>	Name	Type	Description	<i><LineDiscr></i>	Integer	Line discretization value.
Name	Type	Description					
<i><LineDiscr></i>	Integer	Line discretization value.					
Return Value	None.						

Python Syntax	ChangeGeomSettings(<i><LineDiscr></i>)
Python Example	<code>oModule.ChangeGeomSettings(500)</code>

VB Syntax	ChangeGeomSettings <i><LineDiscr></i>
VB Example	<code>oModule.ChangeGeomSettings 500</code>

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	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><i><SoluName></i></td> <td>String</td> <td>Name of specified solution.</td> </tr> <tr> <td><i><VarArray></i></td> <td>Array</td> <td>Array of variable name, value pairs.</td> </tr> <tr> <td><i><FieldType></i></td> <td>String</td> <td>Optional. Type of specified field.</td> </tr> </tbody> </table>	Name	Type	Description	<i><SoluName></i>	String	Name of specified solution.	<i><VarArray></i>	Array	Array of variable name, value pairs.	<i><FieldType></i>	String	Optional. Type of specified field.
Name	Type	Description											
<i><SoluName></i>	String	Name of specified solution.											
<i><VarArray></i>	Array	Array of variable name, value pairs.											
<i><FieldType></i>	String	Optional. Type of specified field.											
Return Value	None.												

Python Syntax	ClcEval(<SoluName>, <VarArray>, [Optional <FieldType>])
Python Example	<pre> oModule.ClcEval("Setup1: LastAdaptive", ["Freq:=", "10GHz", "Phase:=", "0deg"]) </pre>

VB Syntax	ClcEval <SoluName>, <VarArray>, [Optional <FieldType>]
VB Example	<pre> oModule.ClcEval "Setup1: LastAdaptive", _ Array ("Freq:=", "10GHz", _ "Phase:=", "0deg") </pre>

ClcMaterial

Performs a material operation on the top stack element.

UI Access	Click Matl... in the Fields Calculator dialog.									
Parameters	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><MatString></td> <td>String</td> <td>The material property to apply.</td> </tr> <tr> <td><OpString></td> <td>String</td> <td>Name of operation. Possible values are "mult", or "div".</td> </tr> </tbody> </table>	Name	Type	Description	<MatString>	String	The material property to apply.	<OpString>	String	Name of operation. Possible values are "mult", or "div".
Name	Type	Description								
<MatString>	String	The material property to apply.								
<OpString>	String	Name of operation. Possible values are "mult", or "div".								
Return Value	None.									

Python Syntax	ClcMaterial(<MatString>, <OpString>)
----------------------	--------------------------------------

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	<table border="1"><tr><td>Name <code><MaterialName></code></td><td>Type String</td><td>Description Name of specified material property.</td></tr></table>	Name <code><MaterialName></code>	Type String	Description Name of specified material property.
Name <code><MaterialName></code>	Type String	Description 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	<table border="1"><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><code><ExprName></code></td><td>String</td><td>Name of the expression to be copied to the top of the stack.</td></tr></table>	Name	Type	Description	<code><ExprName></code>	String	Name of the expression to be copied to the top of the stack.
Name	Type	Description					
<code><ExprName></code>	String	Name of the expression to be copied to the top of the stack.					
Return Value	None.						

Python Syntax	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: **HFSS>Fields>Plot Fields><field_quantity>**

Command: **Maxwell3D or Maxwell2D>Fields>Fields><field_quantity>**

Command: **Maxwell3D>Fields>Fields>Field Line Trace** (for 3D Electrostatic designs - see [Examples](#) below)

Command: **Icepak>Fields>Plot Fields><field_quantity>**

Syntax: `CreateFieldPlot <PlotParameterArray> ["Field" | "VRT"]`

Return Value: None

Parameters: `<PlotParameterArray> "Field"`

```
Array("NAME:<PlotName>",
      "SolutionName:=", <string>,
      "QuantityName:=", <string>,
      "PlotFolder:=", <string>,
      "UserSpecifyName:=", <int>,
```

```
"UserSpecifyFolder:=", <int>,
"IntrinsicVar:=", <string>,
"PlotGeomInfo:=", <PlotGeomArray>,
"FilterBoxes:=", <FilterBoxArray>,
<PlotOnPointsSettings>,
<PlotOnLineSettings>,
<PlotOnSurfaceSettings>,
<PlotOnVolumeSettings>)
```

SolutionName

Name of the solution setup and solution formatted as:

```
"<SolveSetupName>: <WhichSolution>",
where <WhichSolution> can be "Adaptive_<n>",
"LastAdaptive", or "PortOnly".
```

For example: "Setup1 : Adaptive_2"

HFSS and Maxwell require a space on either side of the ':' character. If it is missing, the plot will not be created.

QuantityName

Type of plot to create. Possible values include:

Mesh plots: "Mesh"

Field plots (HFSS) include:

```
"Mag_E", "Mag_H", "Mag_Jvol", "Mag_Jsurf",
"ComplexMag_E", "ComplexMag_H", "ComplexMag_Jvol",
"ComplexMag_Jsurf", "Vector_E", "Vector_H",
"Vector_Jvol", "Vector_Jsurf", "Vector_RealPoynting",
"Local_SAR", "Average_SAR"
```

Field plots (Maxwell) include:

```
"Mag_E", "Mag_H", "E_Vector", "H_Vector", "Mag_J", "J_Vector",
"ComplexMag_J", "Mag_Jc", "Jc_Vector", "ComplexMag_Jc",
"Energy", "coEnergy", "appEnergy", "Ohmic_Loss", "Total_Loss",
""Hysteresis_Loss", "Dielectric_Loss", "Temperature", "Volume_Force_Density", Average_
Surface_Loss_Density
"Surface_Force_Density", "Demag_Coef", "QuantityName_FieldLineTrace", "Surface_Loss_
Density", QSurf
```

Field plots (Mechanical) include - (dependent on solution type):

```
"Mag_Displacement", "Displacement_Vector", "Temperature", "HeatFlux",
"Mag_HeatFlux", "Surface Loss Density", "Volume Loss Density",
"Linked Heat Transfer Coefficient", "Equivalent Stress"
```

PlotFolder

Name of the folder to which the plot should be added. (Values vary with the design type.) Some possible values include:

"E Field", "H Field", "Jvol", "Jsurf", "SAR Field", "Jc", "Surface-Loss", QSurf, Temperature, Energy, Average-Surface-Loss-Density, "Dielectric_Loss", "MeshPlots", "Heat Flux", and "Displacement".

UserSpecifyName

0 if default name for plot is used, 1 otherwise.

Not needed. <PlotName> will be respected regardless of whether this flag is set.

UserSpecifyFolder

0 if default folder for plot is used, 1 otherwise.

Not needed. The specified PlotFolder will be respected regardless of whether this flag is set.

IntrinsicVar

Formatted string that specifies the frequency and phase at which to make the plot.

For example: "Freq='1GHz' Phase='30deg'"

IntrinsicVar (*Maxwell*)

Formatted string that specifies the skew slice number at which to make the plot.

For example: "IntrinsicVar:=" , "Slice='2' Time='0.00020000000000000001s'"

PlotGeomInfo

Creating field plot on selected layer-net pairs:

For example, Maxwell: "PlotGeomInfo :=", [1,"Volume","ObjList",2,"LC1_1:Top","LC1_1:Top#L1"]

HFSS example with "Plot on surface" option is not checked

```
"PlotGeomInfo:=" , [1,"Volume","LayerNets",2, "Top",2,"net1","net2",_
"Ground",1,"GND"]
```

The command creates field plot on 2 layer-nets combinations. It starts with type "Volume", and subtype "LayerNets", followed by the number of layer-nets combinations which is 2 in this example.

The two layer-nets combinations are (1) "Top" layer, 2 nets, net names are "net1" and "net2", i.e., [Top, net1] pair, and [Top, net2] pair. (2) "Ground" layer and 1 net, net name is "GND", i.e., [Ground, GND] pair.

HFSS example with "Plot on surface" option is checked

```
"PlotGeomInfo:=" , [1,"Surface","LayerNetsExtFace",2,"Top",2,"net1","net2",_
"Ground",1,"GND"],
```

The command creates field plot on 2 layer-nets combinations. It starts with type "Surface", and subtype "LayerNetsExtFace", followed by the number of layer-nets combinations which is 2 in this example.

The two layer-nets combinations are (1) "Top" layer, 2 nets, net names are "net1" and "net2", i.e., [Top, net1] pair, and [Top, net2] pair. (2) "Ground" layer and 1 net, net name is "GND", i.e., [Ground, GND] pair.

Limitation:

No Layer/Nets name is supported for field plot command recording

Should be able to add it based on this change.

No Layer/Nets name support for pure Layout design field plot command.

```
<PlotGeomArray>
```

```
Array(<NumGeomTypes>, <GeomTypeData>,
<GeomTypeData>, ...)
```

For example: Array(4, "Volume", "ObjList", 1, "Box1",
"Surface", "FacesList", 1, "12", "Line", 1,
"Polyline1", "Point", 2, "Point1", "Point2")

```
<NumGeomTypes>
```

Type: **<int>**

Number of different geometry types (volume, surface, line, point)
plotted on at the same time.

```
<GeomTypeData>
```

```
<GeomType>, <ListType>, <NumIDs>, <ID>, <ID>, ...)
```

```
<GeomType>
```

Type: **<string>**

Possible values are "Volume", "Surface", "Line", "Point".

```
<ListType>
```

Type: **<string>**

Possible values are "ObjList", or "FacesList".

These are used for the GeomType of "Line" or "Point".

<NumIDs>

Type: <int>

Number of IDs or object names that will follow.

<ID>

Type: <int> or <string>

ID of a face or name of an object, line, or point on which to plot.

<FilterBoxArray>

Array of names of objects to use to restrict the plot range.

Array(<NumFilters>, <ObjName>, <ObjName>, ...)

Example: Array(1, "Box1")

Example: Array(0) no filtering

<PlotOnPointSettings>

Array("NAME:PlotOnPointSettings",

"PlotMarker:=", <bool>,

"PlotArrow:=", <bool>)

```
<PlotOnLineSettings>
    Array("NAME:PlotOnLineSettings",
        Array("NAME:LineSettingsID",
            "Width:=", <int>,
            "Style:=", <string>),
        "IsoValType:=", <string>,
        "ArrowUniform:=", <bool>,
        "NumofArrow:=", <int>)
```

Style

*Possible values are "Cylinder", "Solid", "Dashdash",
"Dotdot", "Dotdash"*

IsoValType

Possible values are "Tone", "Fringe", "Gourard"

<PlotOnSurfaceSettings>

```
    Array("NAME:PlotOnSurfaceSettings",
        "Filled:=", <bool>,
```

```
"IsoValType:=", <string>,
"SmoothShade:=", <bool>,
"AddGrid:=", <bool>,
"MapTransparency:=", <bool>,
"Transparency:=", <double>,
"ArrowUniform:=", <bool>
"ArrowSpacing:=", <double>
"GridColor:=", Array(<int>, <int>, <int>)
```

IsoValType

Possible values are: "Tone", "Line", "Fringe", "Gourard"

GridColor

Array containing the R, G, B components of the color. Components should be in the range 0 to 255.

<PlotOnVolumeSettings>

```
Array("NAME:PlotOnVolumeSettings",
"PlotIsoSurface:=", <bool>,
"CloudDensity:=", <double>,
"PointSize:=", <int>,
```

```
"ArrowUniform:=", <bool>,  
"ArrowSpacing:=", <double>)
```

Example Field Plot:

```
oModule.CreateFieldPlot Array("NAME:Mag_E1", _  
    "SolutionName:=", "Setup1 : LastAdaptive", _  
    "QuantityName:=", "Mag_E", _  
    "PlotFolder:=", "E Field1", _  
    "UserSpecifyName:=", 0, _  
    "UserSpecifyFolder:=", 0, _  
    "IntrinsicVar:=", "Freq='1GHz' Phase='0deg'", _  
    "PlotGeomInfo:=", Array( 1, "Surface", _  
        "FacesList", 1, "7"), _  
    "FilterBoxes:=", Array(0),  
    Array("NAME:PlotOnSurfaceSettings", _  
        "Filled:=", false, _  
        "IsoValType:=", "Fringe", _  
        "SmoothShade:=", true, _  
        "AddGrid:=", false, _  
        "MapTransparency:=", true, _
```

```
"Transparency:=", 0, _
"ArrowUniform:=", true, _
"ArrowSpacing:=", 0.100000001490116, _
"GridColor:=", Array(255, 255, 255)))
```

Example Demag_Coef Field Plot:

```
oModule.CreateFieldPlot Array("NAME:Demag_Coef2", "SolutionName:=", _
"Setup1 : Transient", "UserSpecifyName:=", 0, "UserSpecifyFolder:=", 0, _
"QuantityName:=", "Demag_Coef", "PlotFolder:=", "Demag-Coef", "StreamlinePlot:=", _
false, "AdjacentSidePlot:=", false, "FullModelPlot:=", false, "IntrinsicVar:=", _
"Time=" & Chr(39) & "0s" & Chr(39) & "", "PlotGeomInfo:=", Array( 1, _
"Volume", "ObjList", 1, "Mag2_0"), "FilterBoxes:=", Array(1, "Mag1_0"), _
Array("NAME:PlotOnVolumeSettings", "PlotIsoSurface:=", true, "PointSize:=", 1, _
"Refinement:=", 0, "CloudSpacing:=", 0.5, "CloudMinSpacing:=", -1, _
"CloudMaxSpacing:=", -1, Array( "NAME:Arrow3DSpacingSettings", "ArrowUniform:=", _
true, "ArrowSpacing:=", 0, "MinArrowSpacing:=", 0, "MaxArrowSpacing:=", 0)), _
"EnableGaussianSmoothing:=", false), "Field"
```

Parameters: <PlotParameterArray> "VRT"

```
Array("NAME:<PlotName>",
"SolutionName:=", <string>,
```

```
"QuantityName:=", <string>,
"PlotFolder:=", <string>,
"UserSpecifyName:=", <int>,
"UserSpecifyFolder:=", <int>,
"IntrinsicVar:=", <string>,
"MaxFrequency:=", "<Number><FreqUnits>", _
"LaunchFrom:=", ["Launch from custom", | "LaunchFromPointID:"], _
18007, "RayDensity:=", <int>, _
"NumberBounces:=", <int>, _
"Multi-Bounce Ray Density Control:=", <Boolean>, _
"MBRD Max sub divisions:=", <int>, _
"ShootFilterType:=", ["All Rays" | _
"Rays by index", "start index:=", <int>, "stop index:=", <int>, | _
"index step:=", <int> "Rays in box", "FilterBoxID:=", <objID> | _
"Single ray", "Ray elevation:=", "<real>deg", "Ray azimuth:=", "<real>deg"
"),
"VRT"
```

Example VRT Plot:

```
Dim oAnsoftApp
```

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

Set oAnsoftApp = CreateObject("Ansoft.ElectronicsDesktop")

Set oDesktop = oAnsoftApp.GetAppDesktop()

oDesktop.RestoreWindow

Set oProject = oDesktop.SetActiveProject("abrams_1p5")

Set oDesign = oProject.SetActiveDesign("UseSbr")

Set oModule = oDesign.GetModule("FieldsReporter")

oModule.CreateFieldPlot Array("NAME:VRT_Plot1", _
"SolutionName:=", "Setup1 : LastAdaptive", _
"QuantityName:=", "QuantityName_SBR", _
"PlotFolder:=", "Visual Ray Trace SBR", "UserSpecifyName:=", 0, _
"UserSpecifyFolder:=", 0, "IntrinsicVar:=", "", "MaxFrequency:=", "1GHz", _
"LaunchFrom:=", "Launch from custom", "LaunchFromPointID:=", 18007, _
"RayDensity:=", 2, "NumberBounces:=", 5, "Multi-Bounce Ray Density Control:=", _
false, "MBRD Max sub divisions:=", 2, "ShootFilterType:=", "All Rays"), "VRT"
```

VB Example:

```
oModule.CreateFieldPlot Array("NAME:Mag_E1", _  
"SolutionName:=", "Setup1 : LastAdaptive", _  
"QuantityName:=", "Mag_E", _  
"PlotFolder:=", "E Field1", _  
"UserSpecifyName:=", 0, _  
"UserSpecifyFolder:=", 0, _  
"IntrinsicVar:=", "Freq='1GHz' Phase='0deg'", _  
"PlotGeomInfo:=", Array( 1, "Surface", _  
    "FacesList", 1, "7"), _  
"FilterBoxes:=", Array(0),  
Array("NAME:PlotOnSurfaceSettings", _  
    "Filled:=", false, _ "IsoValType:=", "Fringe", _  
    "SmoothShade:=", true, _  
    "AddGrid:=", false, _  
    "MapTransparency:=", true, _  
    "Transparency:=", 0, _  
    "ArrowUniform:=", true, _  
    "ArrowSpacing:=", 0.10000001490116, _  
    "GridColor:=", Array(255, 255, 255))), "Field"
```

Example Electron Density Plot

```
oModule = oDesign.GetModule("FieldsReporter")
oModule.CreateFieldPlot(
    [
        "NAME:Electron_Density3",
        "SolutionName:=" , "AIR : RFDischarge",
        "UserSpecifyName:=" , 0,
        "UserSpecifyFolder:=" , 0,
        "QuantityName:=" , "Electron_Density",
        "PlotFolder:=" , "RF Discharge Fields",
        "StreamlinePlot:=" , False,
        "AdjacentSidePlot:=" , False,
        "FullModelPlot:=" , False,
        "IntrinsicVar:=" , "Freq='0.2000000000000001GHz' GasPressure='0.02kPascal'",
        "PlotGeomInfo:=" , [1,"Surface","FacesList",1,"48"],
        "FilterBoxes:=" , [0],
        [
            "NAME:PlotOnSurfaceSettings",
            "Filled:=" , False,
            "IsoValType:=" , "Tone",
            "AddGrid:=" , False,
```

```
"MapTransparency:=" , True,
"Refinement:=" , 0,
"Transparency:=" , 0,
"SmoothingLevel:=" , 0,
"ShadingType:=" , 0,
[
    "NAME:Arrow3DSpacingSettings",
    "ArrowUniform:=" , True,
    "ArrowSpacing:=" , 0,
    "MinArrowSpacing:=" , 0,
    "MaxArrowSpacing:=" , 0
],
"GridColor:=" , [255,255,255]
],
"EnableGaussianSmoothing:=" , False,
"SurfaceOnly:=" , False
], "Field")
```

Example Mesh Plot

```
Dim oAnsoftApp
```

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

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

Set oProject = oDesktop.SetActiveProject("abrams_1p5")
Set oDesign = oProject.SetActiveDesign("UseSbr")
Set oModule = oDesign.GetModule("FieldsReporter")

oModule.CreateFieldPlot Array("NAME:Mesh1", "SolutionName:=", "Setup1 : LastAdaptive", _
"QuantityName:=", "Mesh", "PlotFolder:=", "MeshPlots", "FieldType:=", "Fields", _
>UserSpecifyName:=", 0, "UserSpecifyFolder:=", 0, "StreamlinePlot:=", false, _
"IntrinsicVar:=", "Freq=" & Chr(39) & "2GHz" & Chr(39) & " Phase=" _
& Chr(39) & "0deg" & Chr(39) & "" & "", "PlotGeomInfo:=", Array(1, _
"Surface", "FacesList", 1, "10568"), "FilterBoxes:=", Array(0), "Real time mode:=", _
true, Array("NAME:MeshSettings", "Scale factor:=", 100, "Transparency:=", 0, _
"Mesh type:=", "Shaded", "Surface only:=", true, "Add grid:=", true, "Refinement:=", _
0, "Use geometry color:=", true, "Mesh line color:=", Array(0, 0, 255), _
"Filled color:=", Array( 255, 255, 255))), "Field"
```

VB Example: Maxwell Plot with Skew Slice

```
oModule.CreateFieldPlot Array("NAME:energy2", "SolutionName:=", "Setup1 : Transient", "User-
SpecifyName:=", 0, "UserSpecifyFolder:=", 0, "QuantityName:=", "energy", "PlotFolder:=",
"Energy", "StreamlinePlot:=", false, "AdjacentSidePlot:=", false, "FullModelPlot:=", false,
"IntrinsicVar:=", "Slice=" & Chr(39) & "2" & Chr(39) & " Time=" & Chr(39) &
"0.00020000000000000001s" & Chr(39) & "",...)
```

Python Example: Maxwell Plot with Skew Slice

```
oModule.CreateFieldPlot(
[
    "NAME:Mesh1",
    "SolutionName:=", "Setup1 : Transient",
    "UserSpecifyName:=", 0,
    "UserSpecifyFolder:=", 0,
    "QuantityName:=" "Mesh",
    "PlotFolder:=", "MeshPlots",
    "FieldType:=", "Fields",
    "StreamlinePlot:=", False,
    "AdjacentSidePlot:=", False,
    "FullModelPlot:=", False,
    "IntrinsicVar:=", "Slice='2' Time='0.00020000000000000001s'",
```

```
"PlotGeomInfo:=", [1,"Surface","FacesList",1,"2955"],
"FilterBoxes:=", [0],
...]
```

Python Example: Maxwell Average_Surface_Loss_Density Plot

```
oModule.CreateFieldPlot(
[
    "NAME:Average_Surface_Loss_Density1",
    "SolutionName:=" , "Setup1 : Transient",
    "UserSpecifyName:=" , 0,
    "UserSpecifyFolder:=" , 0,
    "QuantityName:=" , "Average_Surface_Loss_Density",
    "PlotFolder:=" , "Average-Surface-Loss-Density",
    "StreamlinePlot:=" , False,
    "AdjacentSidePlot:=" , False,
    "FullModelPlot:=" , False,
    "IntrinsicVar:=" , "Time=\'4us\' Time0=\'0s\'",
    "PlotGeomInfo:=", [1,"Surface","FacesList",1,"6"],
    ...
])
```

Python Examples: Maxwell ACConduction Design Plot

```
oModule.CreateFieldPlot(  
[  
    "NAME:Mag_Jc1",  
    "SolutionName:=", "Setup1 : LastAdaptive",  
    "UserSpecifyName:=", 0,  
    "UserSpecifyFolder:=", 0,  
    "QuantityName:=", "Mag_Jc",  
    "PlotFolder:=", "Jc",  
    "StreamlinePlot:=", False,  
    "AdjacentSidePlot:=", False,  
    "FullModelPlot:=", False,  
    "IntrinsicVar:=", "Freq=\\"100000Hz\\" Phase=\\"0deg\\\"",  
    "PlotGeomInfo:=", [1, "Volume", "ObjList", 1, "Box1"],  
    "FilterBoxes:=", [1, ""],  
    [  
        "NAME:PlotOnVolumeSettings",  
        "PlotIsoSurface:=", True,  
        "PointSize:=", 1,  
        "Refinement:=", 0,
```

```
"CloudSpacing:=", 0.5,
"CloudMinSpacing:=", -1,
"CloudMaxSpacing:=", -1,
"ShadingType:=", 0,
[
    "NAME:Arrow3DSpacingSettings",
    "ArrowUniform:=", True,
    "ArrowSpacing:=", 0,
    "MinArrowSpacing:=", 0,
    "MaxArrowSpacing:=", 0
],
],
"EnableGaussianSmoothing:=", False
], "Field")
```

```
oModule.CreateFieldPlot(
[
    "NAME:ComplexMag_Jc1",
    "SolutionName:=", "Setup1 : LastAdaptive",
    "UserSpecifyName:=", 0,
    "UserSpecifyFolder:=", 0,
```

```
"QuantityName:=", "ComplexMag_Jc",
"PlotFolder:=", "Jc",
...)

oModule.CreateFieldPlot(
[
    "NAME:Jc_Vector2",
    "SolutionName:=", "Setup1 : LastAdaptive",
    "UserSpecifyName:=", 0,
    "UserSpecifyFolder:=", 0,
    "QuantityName:=", "Jc_Vector",
    "PlotFolder:=", "Jc",
]
...)

oModule.CreateFieldPlot(
[
    "NAME:Dielectric_Loss1",
    "SolutionName:=", "Setup1 : LastAdaptive",
    "UserSpecifyName:=", 0,
```

```
"UserSpecifyFolder:=", 0,  
"QuantityName:=", "Dielectric_Loss",  
"PlotFolder:=", "Dielectric-Loss",  
]  
...)
```

Maxwell Field Line Trace Plot Examples

VB Example: Maxwell Field Line Trace Plot

```
oModule.CreateFieldPlot  
Array("NAME:FieldLineTrace_Plot3",  
"SolutionName:=", "Setup1 : LastAdaptive",  
"UserSpecifyName:=", 0,  
"UserSpecifyFolder:=", 0,  
"QuantityName:=", "QuantityName_FieldLineTrace",  
"PlotFolder:=", "Field line trace plot",  
"IntrinsicVar:=", "",  
"Trace Step Length:=", "0.001mm",  
"Use Adaptive Step:=", true,  
"Seeding Faces:=", Array( 1, 15091),  
"Seeding Markers:=", Array(0),  
"Surface Tracing Objects:=", Array(1, 15),  
"Volume Tracing Objects:=", Array(1, 36),  
"Seeding Sampling Option:=", true,  
"Seeding Points Number:=", 15,  
"Fractional of Maximal:=", 0.8,  
"Discrete Seeds Option:=", "Marker Point",  
Array("NAME:InceptionEvaluationSettings",  
"Gas Type:=", 0, "Gas Pressure:=", 1),  
Array("NAME:FieldLineTracePlotSettings",
```

```
Array("NAME:LineSettingsID",
"Width:=", 1,
"Style:=", "Solid"),
"IsoValType:=", "Tone"))
"FieldLineTrace")

oModule.ModifyInceptionParameters "FieldLineTrace_Plot3",
Array("NAME:InceptionEvaluationSettings",
"Gas Type:=", 2, "Gas Pressure:=", 1,
"Use Inception:=", True,
"Potential U0:=", 0,
"Potential K:=", 1,
"Potential A:=", 1
"Critical Value:=", 5.9426,
"Streamer Constant:=", 6.5,
"Ionization Coefficient Dataset:=", Array( 0))
```

Python Example: Maxwell Field Line Trace Plot

```
oModule.CreateFieldPlot(
[
    "NAME:FieldLineTrace_Plot3",
    "SolutionName:=", "Setup1 : LastAdaptive",
    "UserSpecifyName:=", 0,
    "UserSpecifyFolder:=", 0,
    "QuantityName:=", "QuantityName_FieldLineTrace",
    "PlotFolder:=", "Field line trace plot",
```

```
"IntrinsicVar:="", "",  
"Trace Step Length:=", "0.001mm",  
"Use Adaptive Step:=", True,  
"Seeding Faces:=", [1,15091],  
"Seeding Markers:=", [0],  
"Surface Tracing Objects:=", [1,15],  
"Volume Tracing Objects:=", [1,36],  
"Seeding Sampling Option:=", True,  
"Seeding Points Number:=", 15,  
"Fractional of Maximal:=", 0.8,  
"Discrete Seeds Option:=", "Marker Point",  
[  
    "NAME:InceptionEvaluationSettings",  
    "Gas Type:=", 0,  
    "Gas Pressure:=", 1  
    "Use Inception:=", True,  
    "Potential U0:=", 0,  
    "Potential K:=", 1,  
    "Potential A:=", 1  
],  
[
```

```
"NAME:FieldLineTracePlotSettings",
[
    "NAME:LineSettingsID",
    "Width:=", 1,
    "Style:=", "Solid"
],
"IsoValType:=", "Tone"
],
"FieldLineTrace")
```

For Q3D Extractor and 2D Extractor, the command details are as follows:

Use: Creates a field/mesh plot.

Command: **Q3D Extractor or 2D Extractor>Fields**

Syntax: CreateFieldPlot <PlotParameterArray>

Return Value: None

Parameters: <PlotParameterArray>

```
Array("NAME:<PlotName>",
"SolutionName:=", <string>,
"QuantityName:=", <string>,
```

```

"PlotFolder:=", <string>,
"UserSpecifyName:=", <int>,
"UserSpecifyFolder:=", <int>,
"IntrinsicVar:=", <string>,
"PlotGeomInfo:=", <PlotGeomArray>,
"FilterBoxes:=", <FilterBoxArray>,
<PlotOnPointSettings>, <PlotOnLineSettings>,
<PlotOnSurfaceSettings>, <PlotOnVolumeSettings>)

```

SolutionName

Name of the solution setup and solution formatted as:

"<SolveSetupName> : <WhichSolution>" ,

where <WhichSolution> can be "Adaptive_<n>" ,

"LastAdaptive" , or "PortOnly" .

For example: "Setup1 : Adaptive_2"

HFSS requires a space on both sides of the ':' character. Otherwise, the plot is not be created.

QuantityName

Type of plot to create. Possible values are:

Mesh plots: "Mesh"

Q3D Field Plots:

Field type	Plot quantity names
------------	---------------------

AC R/L Fields	"SurfaceJac", "Mag_SurfaceJac"
DC R/L PEC Fields	"SurfaceJdc", "Mag_SurfaceJdc"
DC R/L Fields	"VolumeJdc", "Mag_VolumeJdc", "Phidc"
C Fields	"SmoothQ", "ABS_Q"

2D Extractor Field Plots:

Field type	Plot quantity names
CG Fields	"Mag_Phi", "PhiAtPhase", "Mag_E", "VectorE", "Mag_Jcg", "VectorJcg" and "energyCG"
RL Fields	"Flux Lines", "VectorA", "Mag_B", "VectorB", "Mag_H", "VectorH", "Jrl", "VectorJrl", "energyRL", "coenergy", "appenergy" and "emloss"

PlotFolder

Name of the folder to which the plot should be added. Possible values are: "Q", "ABS_Q", "JDC Vol", "Phi", "JDC Surf", and "JAC".

UserSpecifyName

0 if default name for plot is used, 1 otherwise.

This parameter is not essential. <PlotName> is respected regardless of whether this flag is set.

UserSpecifyFolder

0 if the default folder for plot is used, 1 otherwise.

This parameter is not essential. The specified PlotFolder is respected regardless of whether this flag is set.

IntrinsicVar

Formatted string that specifies the frequency and phase at which to create the plot.

For example: "Freq='1GHz' Phase='30deg'"

<PlotGeomArray>

Array(<NumGeomTypes>, <GeomTypeData>,

<GeomTypeData>, ...)

For example: *Array(4, "Volume", "ObjList", 1, "Box1",*

"Surface", "FacesList", 1, "12", "Line", 1,

"Polyline1", "Point", 2, "Point1", "Point2")

<NumGeomTypes>

Type: <int>

Number of different geometry types (volume, surface, line, point) plotted at the same time.

<GeomTypeData>

<GeomType>, <ListType>, <NumIDs>, <ID>, <ID>, ...)

<GeomType>

Type: <string>

Possible values are "Volume", "Surface", "Line", "Point".

<ListType>

Type: <string>

Possible values are "ObjList" or "FacesList".

These are used for GeomType values "Line" or "Point"

<NumIDs>

Type: <int>

Number of IDs or object names that will follow.

<ID>

Type: <int> or <string>

ID of a face or name of an object, line, or point on which to plot.

<FilterBoxArray>

Array of object names used to restrict the plot range.

Array(<NumFilters>, <ObjName>, <ObjName>, ...)

Example: Array(1, "Box1")

Example: Array(0) no filtering

<PlotOnPointSettings>

Array("NAME:PlotOnPointSettings",

"PlotMarker:=", <bool>,

"PlotArrow:=", <bool>)

<PlotOnLineSettings>

Array("NAME:PlotOnLineSettings",

Array("NAME:LineSettingsID",

"Width:=", <int>,

```
"Style:=", <string>),
"IsoValType:=", <string>,
"ArrowUniform:=", <bool>,
"NumofArrow:=", <int>

Style
    Possible values are "Cylinder", "Solid", "Dashdash",
    "Dotdot", "Dotdash".

IsoValType
    Possible values are "Tone", "Fringe", "Gourard".

<PlotOnSurfaceSettings>
    Array("NAME:PlotOnSurfaceSettings",
        "Filled:=", <bool>,
        "IsoValType:=", <string>,
        "SmoothShade:=", <bool>,
        "AddGrid:=", <bool>,
        "MapTransparency:=", <bool>,
        "Transparency:=", <double>,
        "ArrowUniform:=", <bool>
        "ArrowSpacing:=", <double>
        "GridColor:=", Array(<int>, <int>, <int>)

IsoValType
```

Possible values are: "Tone", "Line", "Fringe", "Gourard".

GridColor

Array containing the R, G, B components of the color. Components should be in the range 0 to 255.

<PlotOnVolumeSettings>

```
Array("NAME:PlotOnVolumeSettings",
"PlotIsoSurface:=", <bool>,
"CloudDensity:=", <double>,
"PointSize:=", <int>,
"ArrowUniform:=", <bool>,
"ArrowSpacing:=", <double>)
```

DeleteFieldPlot

Deletes one or more field plots.

UI Access	Right-click on one filed plot, select Delete .							
Parameters	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td><NameArray></td><td>Array</td><td>Array of strings containing the names of the plots to delete.</td></tr></tbody></table>		Name	Type	Description	<NameArray>	Array	Array of strings containing the names of the plots to delete.
Name	Type	Description						
<NameArray>	Array	Array of strings containing the names of the plots to delete.						
Return Value	None.							

Python Syntax

```
DeleteFieldPlot(<NameArray>)
```

Python Example	<code>oModule.DeleteFieldPlot(["Mag_E1", "Vector_E1"])</code>
-----------------------	---

VB Syntax	<code>DeleteFieldPlot <NameArray></code>
VB Example	<code>oModule.DeleteFieldPlot Array("Mag_E1", "Vector_E1")</code>

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 <code><MarkerNames></code>	Type Array	Description Array of strings containing marker names.
Return Value	None.		

Python Syntax	<code>DeleteMarker(<MarkerNames>)</code>
Python Example	<code>oModule.DeleteMarker (["m1", "m2"])</code>

VB Syntax	<code>DeleteMarker <MarkerNames></code>
VB Example	<code>oModule.DeleteMarker Array("m1", "m2")</code>

DeleteNamedExpr

Deletes the selected named expression from the list.

UI Access	Select a named expression and then click Delete .						
Parameters	<table border="1"><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><ExprName></td><td>String</td><td>Name of specified named expression.</td></tr></table>	Name	Type	Description	<ExprName>	String	Name of specified named expression.
Name	Type	Description					
<ExprName>	String	Name of specified named expression.					
Return Value	None.						

Python Syntax	<code>DeleteNamedExpr(<ExprName>)</code>
Python Example	<code>oModule.DeleteNamedExpr ("Mag_JxE")</code>

VB Syntax	<code>DeleteNamedExpr <ExprName></code>
VB Example	<code>oModule.DeleteNamedExpr "Mag_JxE"</code>

DeleteUneditablePlot

Deletes one or more uneditable plots.

UI Access	N/A						
Parameters	<table border="1"><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><NameArray></td><td>Array</td><td>Array of the plot names to delete.</td></tr></table>	Name	Type	Description	<NameArray>	Array	Array of the plot names to delete.
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	<table border="1"> <tr> <td>Name <ExpName></td> <td>Type String</td> <td>Description Name of the specified named expression.</td> </tr> </table>	Name <ExpName>	Type String	Description Name of the specified named expression.
Name <ExpName>	Type String	Description Name of the specified named expression.		
Return Value	<p>Boolean:</p> <ul style="list-style-type: none"> • 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

```
oModule.DoesNamedExpressionExists "Mag_JxE"
```

EditSurfaceMeshSummaryData

Use: Defines or Edits Surface Mesh Summary Data.

Command: Create Surface Mesh Summary, **Setup...**

Syntax: EditSurfaceMeshSummaryData(Array(<Parameters Array>))

Return Value: None

Parameters:

"NAME:SurfaceMeshSummary"

Type: <string>

"NAME:SurfaceMeshSummary"

"SolutionName:=" <string>

Type: <string>

Solution Name.

"Variation Name:=" <string>

Type: <string>

Variation name.

<RowItemsArray>

Array<MeshRowItems>

Row data.

<ItemPerRow>

Array<EntityPerRow>

Entity ID.

<MeshDataPer Item>

Array<Mesh Data Category and Stats>

VB Example:

```
Set oModule = oDesign.GetModule("FieldsReporter")
oModule.EditSurfaceMeshSummaryData Array(Array("NAME:SurfaceMeshSummary", "SolutionName:=", _
"Setup1 : LastAdaptive", "Variation:=", "Nominal", Array("NAME:MeshRowItems", Array
("NAME:ItemPerRow", "EntityPerRow:=", _
"annular_rng", Array("NAME:MeshDataPerItem", "Min Edge Len:=", 0.000166961133900917, "Max Edge
Len:=", _
0.00145730991671888, "Mean Edge Len:=", 0.000524160923260581, "Std Devn Edge Len:=", _
0.000217758706109324, "Min Tri Area:=", 3.4982570283924E-09, "Max Tri Area:=", _
3.73116346661249E-07, "Mean Tri Area:=", 8.91526236529066E-08, "Std Devn Tri Area:=", _
6.89482807990471E-08, "DataState:=", 2)), Array("NAME:ItemPerRow", "EntityPerRow:=", _
"gap", Array("NAME:MeshDataPerItem", "Min Edge Len:=", 0.000105681286042456, "Max Edge Len:=", _
0.000362084671538928, "Mean Edge Len:=", 0.000233090954707978, "Std Devn Edge Len:=", _
7.55387369793154E-05, "Min Tri Area:=", 1.74640378345369E-09, "Max Tri Area:=", _
3.78166789228631E-08, "Mean Tri Area:=", 1.56497981245286E-08, "Std Devn Tri Area:=", _
1.04691637802914E-08, "DataState:=", 2)))))
```

Python Syntax	RenameReport (<OldReportName>, <NewReportName>)
Python Example	<pre>oModule>EditSurfaceMeshSummaryData([["NAME:SurfaceMeshSummary", "SolutionName:=" , "Setup1 : LastAdaptive", "Variation:=" , "Nominal", ["NAME:MeshRowItems", ["NAME:ItemPerRow", "EntityPerRow:=" , "annular_rng", ["NAME:MeshDataPerItem", "Min Edge Len:=" , 0.000166961133900917, "Max Edge Len:=" , 0.00145730991671888, "Mean Edge Len:=" , 0.000524160923260581, "Std Devn Edge Len:=" , 0.000217758706109324, "Min Tri Area:=" , 3.4982570283924E-09, "Max Tri Area:=" , 3.73116346661249E-07,</pre>

```
        "Mean Tri Area:=" , 8.91526236529066E-08,
        "Std Devn Tri Area:=" , 6.89482807990471E-08,
        "DataState:=" , 2
    ]
],
[
    "NAME:ItemPerRow",
    "EntityPerRow:=" , "gap",
    [
        "NAME:MeshDataPerItem",
        "Min Edge Len:=" , 0.000105681286042456,
        "Max Edge Len:=" , 0.000362084671538928,
        "Mean Edge Len:=" , 0.000233090954707978,
        "Std Devn Edge Len:=" , 7.55387369793154E-05,
        "Min Tri Area:=" , 1.74640378345369E-09,
        "Max Tri Area:=" , 3.78166789228631E-08,
        "Mean Tri Area:=" , 1.56497981245286E-08,
        "Std Devn Tri Area:=" , 1.04691637802914E-08,
        "DataState:=" , 2
    ]
]
```

```
    ]  
    ]  
])
```

EnterComplex

Enters a complex number onto the stack.

UI Access	Click Number , and then click Scalar. Complex option is selected.						
Parameters	<table border="1"><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><ComplexNum></td><td>String</td><td>String of complex value. Ex. "1 + 2j".</td></tr></table>	Name	Type	Description	<ComplexNum>	String	String of complex value. Ex. "1 + 2j".
Name	Type	Description					
<ComplexNum>	String	String of complex value. Ex. "1 + 2j".					
Return Value	None.						

Python Syntax	EnterComplex(<ComplexNum>)
Python Example	oModule.EnterComplex ("1 + 2 j")

VB Syntax	EnterComplex <ComplexNum>
VB Example	oModule.EnterComplex "1 + 2 j"

EnterComplexVector

Enters a complex vector onto the stack.

UI Access	Click Number , and then click Vector. Complex option is selected.						
Parameters	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><ComplexVector></td> <td>Array</td> <td>Array of strings containing X, Y and Z complex values.</td> </tr> </tbody> </table>	Name	Type	Description	<ComplexVector>	Array	Array of strings containing X, Y and Z complex values.
Name	Type	Description					
<ComplexVector>	Array	Array of strings containing X, Y and Z complex values.					
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	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td><CoordName></td><td>String</td><td>Name of a coordinate system defined in the 3D Modeler editor.</td></tr></tbody></table>			Name	Type	Description	<CoordName>	String	Name of a coordinate system defined in the 3D Modeler editor.
Name	Type	Description							
<CoordName>	String	Name of a coordinate system defined in the 3D Modeler editor.							
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	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td><EdgeName></td><td>String</td><td>Name of an edge defined in the 3D Modeler editor.</td></tr></tbody></table>			Name	Type	Description	<EdgeName>	String	Name of an edge defined in the 3D Modeler editor.
Name	Type	Description							
<EdgeName>	String	Name of an edge defined in the 3D Modeler editor.							
Return Value	None.								

Python Syntax	EnterEdge(<EdgeName>)
Python Example	<code>oModule.EnterPoint ("Edge_1")</code>

VB Syntax	EnterEdge <EdgeName>
VB Example	<code>oModule.EnterPoint "Edge_1"</code>

EnterLine

Enters a line defined in the 3D Modeler editor.

UI Access	Click Geometry and then select Line						
Parameters	<table border="1"> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> <tr> <td><LineName></td> <td>String</td> <td>Name of a line defined in the 3D Modeler editor.</td> </tr> </table>	Name	Type	Description	<LineName>	String	Name of a line defined in the 3D Modeler editor.
Name	Type	Description					
<LineName>	String	Name of a line defined in the 3D Modeler editor.					
Return Value	None.						

Python Syntax	EnterLine(<LineName>)
Python Example	<code>oModule.EnterLine ("Line1")</code>

VB Syntax	EnterLine <LineName>
VB Example	<code>oModule.EnterLine "Line1"</code>

EnterOutputVar

Enters Output Vars, only valid for Eigenmode problems.

UI Access	Click Geometry and then select Output Vars .		
Parameters	Name	Type	Description
	<VarName>	String	Name of the output vars. Only 'freq' is supported.
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	<code>oModule.EnterPoint ("Point1")</code>

VB Syntax	EnterPoint <PointName>
VB Example	<code>oModule.EnterPoint "Point1"</code>

EnterQty

Enters a field quantity.

UI Access	Click Quantity , and then select from the list.						
Parameters	<table border="1"> <tr> <td>Name</td> <td>Type</td> <td>Description</td> </tr> <tr> <td><FieldQty></td> <td>String</td> <td>The field quantity to be entered onto the stack.</td> </tr> </table>	Name	Type	Description	<FieldQty>	String	The field quantity to be entered onto the stack.
Name	Type	Description					
<FieldQty>	String	The field quantity to be entered onto the stack.					
Return Value	None.						

Python Syntax	EnterQty(<FieldQty>)
Python Example	<code>oModule.EnterQty ("E")</code>

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	<table border="1"><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><Scalar></td><td>Double</td><td>The real number to enter onto the stack.</td></tr></table>			Name	Type	Description	<Scalar>	Double	The real number to enter onto the stack.
Name	Type	Description							
<Scalar>	Double	The real number to enter onto the stack.							
Return Value	None.								

Python Syntax

```
EnterScalar(<Scalar>)
```

Python Example

```
oModule.EnterScalar(3.14159265358979)
```

VB Syntax

```
EnterScalar <Scalar>
```

VB Example

```
oModule.EnterScalar 3.14159265358979
```

EnterScalarFunc

Enters a scalar function.

UI Access	Click Function and then select Scalar .					
Parameters	<table border="1"><tr><td>Name</td><td>Type</td><td>Description</td></tr></table>			Name	Type	Description
Name	Type	Description				

	<code><VarName></code>	String	Name of a variable to enter as a scalar function onto the stack.
Return Value	None.		

Python Syntax	<code>EnterScalarFunc(<VarName>)</code>
Python Example	<code>oModule.EnterScalarFunc ("Phase")</code>

VB Syntax	<code>EnterScalarFunc <VarName></code>
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	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><code><SurfName></code></td> <td>String</td> <td>Name of a surface defined in the 3D Modeler editor.</td> </tr> </tbody> </table>			Name	Type	Description	<code><SurfName></code>	String	Name of a surface defined in the 3D Modeler editor.
Name	Type	Description							
<code><SurfName></code>	String	Name of a surface defined in the 3D Modeler editor.							
Return Value	None.								

Python Syntax	<code>EnterSurf(<SurfName>)</code>
Python Example	<code>oModule.EnterSurf ("Rectangle1")</code>

VB Syntax	EnterSurf < <i>SurfName</i> >
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	<table border="1"><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><code><VectorArray></code></td><td>Array</td><td>Array of strings containing X, Y and Z components of the vector.</td></tr></table>	Name	Type	Description	<code><VectorArray></code>	Array	Array of strings containing X, Y and Z components of the vector.
Name	Type	Description					
<code><VectorArray></code>	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 <i><VarNames></i>	Type Array	Description Array of strings containing names of a variable for the X, Y, and Z coordinates, respectively, to enter as a vector function on the stack.
Return Value	None.		

Python Syntax	EnterVectorFunc(<i><VarNames></i>)
Python Example	<code>oModuleEnterVectorFunc(["X", "Y", "Z"])</code>

VB Syntax	EnterVectorFunc < <i>VarNames</i> >
VB Example	<code>oModuleEnterVectorFunc Array("X", "Y", "Z")</code>

EnterVol

Enters a volume defined in the 3D Modeler editor.

UI Access	Click Geometry and then select Volume .		
Parameters	Name <i><VolumeName></i>	Type String	Description Name of a volume defined in the 3D Modeler editor.
Return Value	None.		

Python Syntax	EnterVol(<i><VolumeName></i>)
Python Example	<code>oModule.EnterVol ("Box1")</code>

VB Syntax	EnterVol <VolumeName>
VB Example	oModule.EnterVol "Box1"

ExportFieldPlot

Exports a field plot to a file.

UI Access	N/A												
Parameters	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td><PlotName></td><td>String</td><td>Name of the field plot to export.</td></tr><tr><td><>ShowHeader></td><td>Boolean</td><td>True - export field plot header to a file. False - export field plot.</td></tr><tr><td><FileName></td><td>String</td><td>Name of file to save as, including the file path.</td></tr></tbody></table>	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.
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	<pre>oModule.ExportFieldPlot "Mag_E1", _ true, "C:/field_report.dsp"</pre>
-------------------	--

ExportMarkerTable

Exports the marker table to a .csv or .tab file.

UI Access	[product] > Fields > Plot Fields > Marker > Export Marker Table .						
Parameters	<table border="1"> <tr> <td>Name</td> <td>Type</td> <td>Description</td> </tr> <tr> <td><FileName></td> <td>String</td> <td>Name of export file include path.</td> </tr> </table>	Name	Type	Description	<FileName>	String	Name of export file include path.
Name	Type	Description					
<FileName>	String	Name of export file include path.					
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	<table border="1"> <tr> <td>Name</td> <td>Type</td> <td>Description</td> </tr> </table>	Name	Type	Description
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", </pre>

```

["Freq:=", "4.5GHz", "Phase:=" , "0deg"],

[
    "NAME:ExportOption",
    "IncludePtInOutput:=" , True,
    "RefCSName:=" , "offset",
    "PtInSI:=" , False,
    "FieldInRefCS:=" , True
],
"Cartesian", ["0mm", "0mm", "0mm"], False
)

```

VB Syntax	ExportOnGrid(<OutputFile>, <Min>, <Max>, <Spacing>, <SolName>, <SolParameters>, [<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", _ </pre>

```
Array("0mm", "0mm", "0mm")

oModule.ExportOnGrid("C:\offset_grid_model_unit_ref.fld",
    Array("-1mm", "16mm", "0mm"),
    Array("1mm", "18mm", "1mm"),
    Array("2mm", "2mm", "1mm"),
    "4500MHz : LastAdaptive",
    Array("Freq:=", "4.5GHz", "Phase:=" , "0deg"),
    [
        "NAME:ExportOption",
        "IncludePtInOutput:=" , True,
        "RefCSName:=" , "offset",
        "PtInSI:=" , False,
        "FieldInRefCS:=" , True
    ],
    "Cartesian", Array("0mm", "0mm", "0mm"), False
)
```

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.
Return Value	<ViewOrientation>		
	String Optional. Name of orientation to use for plot.		
Return Value		Image file is exported to the specified path.	

Python Syntax	ExportPlotImageWithViewToFile(<FileName>, <PlotQuantityName>, <PlotItemName>, <PixelSizeX>, <PixelSizeY>, <ViewOrientation>)
Python Example	oModule.ExportPlotImageWithViewToFile("E:/MyDir/OptimToutput/magE2.gif", "E Field", "Mag_E2", 1920, 1080, "newot2")

VB Syntax	ExportPlotImageWithViewToFile <FileName>, <PlotQuantityName>, <PlotItemName>, <PixelSizeX>, <PixelSizeY>, <ViewOrientation>
VB Example	<pre>oModule.ExportPlotImageWithViewToFile "E:/MyDir/OptimToutput/magE2.gif", "E Field", _ "Mag_E2", 1920, 1080, "newot2"</pre>

ExportSurfaceMeshSummary

Use: Exports a Surface Mesh Summary.

Command: **Create Surface Mesh Summary.**

Syntax: ExportSurfaceMeshSummary Array(<SolutionName>, <DesignVariationKey> <ExportFileName>)

Return Value: None

Parameters: <SolutionName>

Type: <string>

Original name of the plot.

<DesignVariationKey>

Type: <string>

New name of the plot.

VB Example:

```
oModule.ExportSurfaceMeshSummary Array("SolutionName:=", "Setup1 : LastAdaptive", "DesignVariationKey:=", _
"Nominal", "ExportFileName:=", "D:\documents\surfaceMeshSummaryReport.csv")
```

Python Syntax	ExportSurfaceMeshSummary (<SolutionName>, <DesignVariationKey>, <ExportFileName>)
----------------------	---

Python Example

```

oModule.ExportSurfaceMeshSummary(
    [
        "SolutionName:=" , "Setup1 : LastAdaptive",
        "DesignVariationKey:=" , "Nominal",
        "ExportFileName:=" , "D:\\Program Files\\Documents\\surfaceMeshSummaryReport.csv"
    ]
)

```

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 .		
	Name	Type	Description
Parameters	<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/cftt.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	<code>oModule.GetFieldFolderNames ()</code>

VB Syntax	GetFieldFolderNames
VB Example	<code>oModule.GetFieldFolderNames</code>

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	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><PlotName></td> <td>String</td> <td>Name of specified plot.</td> </tr> </tbody> </table>			Name	Type	Description	<PlotName>	String	Name of specified plot.
Name	Type	Description							
<PlotName>	String	Name of specified plot.							
Return Value	String plot quantity name.								

Python Syntax	GetFieldPlotQuantityName(<PlotName>)
Python Example	oModule.GetFieldPlotQuantityName ("Mag_H1")

VB Syntax	GetFieldPlotQuantityName <PlotName>
VB Example	oModule.GetFieldPlotQuantityName "Mag_H1"

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	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td><code><SolnName></code></td><td>String</td><td>Name of specified solution.</td></tr><tr><td><code><VarVals></code></td><td>Array</td><td>Array of variable name, value pairs.</td></tr></tbody></table>	Name	Type	Description	<code><SolnName></code>	String	Name of specified solution.	<code><VarVals></code>	Array	Array of variable name, value pairs.
Name	Type	Description								
<code><SolnName></code>	String	Name of specified solution.								
<code><VarVals></code>	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 <SolName>, <VarVals>
VB Example	<pre>oModule.GetTopEntryValue "Setup1:LastAdaptive", _ Array("Freq:=", "1GHz", "Phase:=", "0deg", _ "x_size:=", "2mm")</pre>

HidePolarPlot

Hides a currently visible polar plot.

UI Access	N/A						
Parameters	<table border="1"> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> <tr> <td><PlotName></td> <td>String</td> <td>Name of the polar plot.</td> </tr> </table>	Name	Type	Description	<PlotName>	String	Name of the polar plot.
Name	Type	Description					
<PlotName>	String	Name of the polar plot.					
Return Value	None.						

Python Syntax	HidePolarPlot(<PlotName>)
Python Example	<pre>oModule.HidePolarPlot("rE Plot 1")</pre>

VB Syntax	HidePolarPlot <PlotName>
VB Example	<pre>oModule.HidePolarPlot "rE Plot 1"</pre>

HideRadiatedPlotOverlay

Hides overlay radiation fields plot.

UI Access	N/A		
Parameters	Name <i><PlotName></i>	Type String	Description Name of specified plot.
Return Value	None.		

Python Syntax	HideRadiatedPlotOverlay(<i><PlotName></i>)
Python Example	oModule.HideRadiatedPlotOverlay("Gain Plot 1")

VB Syntax	HideRadiatedPlotOverlay <PlotName>
VB Example	oModule.HideRadiatedPlotOverlay "Gain Plot 1"

LoadNamedExpressions

Loads a named expression definition from a saved file.

UI Access	In the Fields Calculator, click Load From... in the Library area.		
Parameters	Name <i><FileName></i>	Type String	Description Filename and full path to the file to hold the named expression definition.
	<i><FieldType></i>	String	For products with just one filed type, it is set to "Fields".

	<code><NamedExpr></code>	Array	rray of strings containing the names of expression definitions to load from the file.
Return Value	None.		

Python Syntax	<code>LoadNamedExpressions(<FileName>, <FieldType>, <NamedExpr>)</code>
Python Example	<code>oModule.LoadNamedExpressions("C:\Ansoft\PersonalLib\smth.clc", "Fields", ["smoothedtemp"])</code>

VB Syntax	<code>LoadNamedExpressions <FileName>, <FieldType>, <NamedExpr></code>
VB Example	<code>oModule.LoadNamedExpressions _ "C:\Ansoft\PersonalLib\smth.clc", _ "Fields", Array("smoothedtemp")</code>

ModifyFieldPlot

Modifies a plot definition.

UI Access	<code>[product] > Fields > Modify Plot</code>		
Parameters	Name	Type	Description
	<code><OriginalName></code>	String	Name of the plot to be modified.
	<code><PlotParams></code>	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], ["NAME:PlotOnSurfaceSettings", "Filled:=", False, "IsoValType:=", "Fringe", "SmoothShade:=", True, "AddGrid:=", False, "MapTransparency:=", True, "Transparency:=", 0, _</pre>

```

    "ArrowUniform:=", True,
    "ArrowSpacing:=", 0.100000001490116,
    "GridColor:=", [255, 255, 255]
)

```

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, _ "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, _ </pre>

```
"MapTransparency:=", true, _  
"Transparency:=", 0, _  
"ArrowUniform:=", true, _  
"ArrowSpacing:=", 0.100000001490116, _  
"GridColor:=", Array(255, 255, 255)))
```

ReassignFieldPlot

Reassigns a field plot.

UI Access	Right-click on a field plot > Reassign .	
Parameters	Name < <i>PlotName</i> >	Type String Description Name of specified plot to reassign.
	< <i>PlotParameterArray</i> >	Array See CreateFieldPlot
Return Value	None.	

Python Syntax	ReassignFieldPlot(< <i>PlotName</i> >, < <i>PlotParameterArray</i> >)
Python Example	<pre>oModule.ReassignFieldPlot "Mag_H1", Array("NAME:Mag_H1", _ "SolutionName:=", "Setup1 : LastAdaptive", _ "UserSpecifyName:=", 0, _ "UserSpecifyFolder:=", 0, _</pre>

```
"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,  
    "Refinement:=", 0,  
    "Transparency:=", 0,  
    "SmoothingLevel:=", 0,  
    "ShadingType:=", 0,  
    ["NAME:Arrow3DSpacingSettings",  
        "ArrowUniform:=", True,  
        "ArrowSpacing:=", 0,  
        "MinArrowSpacing:=", 0,
```

```
    "MaxArrowSpacing:=", 0
    ],
    "GridColor:=", [255,255,255]
],
"EnableGaussianSmoothing:=", False
])
```

VB Syntax	ReassignFieldPlot <PlotName>, <PlotParameterArray>
VB Example	<pre>oModule.ReassignFieldPlot "Mag_E1", _ Array("NAME:Mag_E1", "SolutionName:=", "Setup1 : LastAdaptive", "UserSpecifyName:=", 0, _ "UserSpecifyFolder:=", 0, "QuantityName:=", "Mag_E", "PlotFolder:=", "E Field", _ "StreamlinePlot:=", false, "AdjacentSidePlot:=", false, "FullModelPlot:=", false, "IntrinsicVar:=", "Freq=" & Chr(39) & "20GHz" & Chr(39) & " " Phase=" & Chr(39) & "0deg" & Chr(39) & "", "PlotGeomInfo:=", Array(1, "Surface", "FacesList", 1, "149"), "FilterBoxes:=", Array(0), _</pre>

```

Array("NAME:PlotOnSurfaceSettings", "Filled:=", _  

    false, "IsoValType:=", "Fringe", _  

    "AddGrid:=", false, "MapTransparency:=", true, "Refinement:=", _  

    0, "Transparency:=", 0, "SmoothingLevel:=", 0, _  

    "ShadingType:=", 0, Array("NAME:Arrow3DSpacingSettings", "ArrowUniform:=", _  

    true, "ArrowSpacing:=", 0, _  

    "MinArrowSpacing:=", 0, "MaxArrowSpacing:=", 0), "GridColor:=", Array( _  

    255, 255, 255)), "EnableGaussianSmoothing:=", false)

```

RenameFieldPlot

Renames a plot.

UI Access	Right-click the plot you want to rename in the project tree, and then click Rename on the shortcut menu.									
Parameters	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><OldName></td> <td>String</td> <td>Original name of the plot.</td> </tr> <tr> <td><NewName></td> <td>String</td> <td>New name of the plot.</td> </tr> </tbody> </table>	Name	Type	Description	<OldName>	String	Original name of the plot.	<NewName>	String	New name of the plot.
Name	Type	Description								
<OldName>	String	Original name of the plot.								
<NewName>	String	New name of the plot.								
Return Value	None.									

Python Syntax	RenameFieldPlot(<OldName>, <NewName>)
Python Example	<pre> oModule.RenameFieldPlot("Vector_E1", "Vector_E2") </pre>

VB Syntax	RenameFieldPlot <OldName>, <NewName>
VB Example	<pre>oModule.RenameFieldPlot _ "Vector_E1", "Vector_E2"</pre>

RenamePlotFolder

Renames a plot folder.

UI Access	Right-click a plot folder in the project tree, and then click Rename on the shortcut menu.									
Parameters	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td><OldName></td><td>String</td><td>Original name of the folder.</td></tr><tr><td><NewName></td><td>String</td><td>New name of the folder.</td></tr></tbody></table>	Name	Type	Description	<OldName>	String	Original name of the folder.	<NewName>	String	New name of the folder.
Name	Type	Description								
<OldName>	String	Original name of the folder.								
<NewName>	String	New name of the folder.								
Return Value	None.									

Python Syntax	RenamePlotFolder(<OldName>, <NewName>)
Python Example	<pre>oModule.RenamePlotFolder("E Field", "Surface Plots")</pre>

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.
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.
Return Value	Specifies whether to overwrite the file.		

Python Syntax	SaveNamedExpressions(<FileName>, <NamedExprs>, <Overwrite>)
Python Example	<pre>oModule.SaveNamedExpressions("C:\Ansoft\PersonalLib\smth.clc", ["smoothedtemp"], True)</pre>

VB Syntax	SaveNamedExpressions <FileName>, <NamedExprs>, <Overwrite>
VB Example	<pre>oModule.SaveNamedExpressions _ "C:\Ansoft\PersonalLib\smth.clc", _ Array("smoothedtemp"), true</pre>

SetFieldPlotSettings

Sets plot attributes.

UI Access	[product] > Fields > Modify Plot Attributes
-----------	---

Parameters	Name	Type	Description
	<PlotName>	String	Name of the plot to modify.
	<PlotItemAttributes>	Array	<p>Structured array.</p> <pre>Array ("NAME:FieldsPlotItemSettings", <PlotOnPointsSettings>, <PlotOnLineSettings>, <PlotOnSurfaceSettings>, <PlotOnVolumeSettings>)</pre> <p><i>See description of CreateFieldPlot command for details.</i></p>
Return Value	None.		

Python Syntax	SetFieldPlotSettings(<PlotName> <PlotItemAttributes>)
Python Example	<pre>oModule.SetFieldPlotSettings ("Mag_E2", ["NAME:FieldsPlotItemSettings", ["NAME:PlotOnLineSettings", ["NAME:LineSettingsID", "Width:=", 4, "Style:=", "Cylinder"), "IsoValType:=", "Tone", "ArrowUniform:=", True, "NumofArrow:=", 100),</pre>

```
[ "NAME:PlotOnSurfaceSettings",
  "Filled:=", False,
  "IsoValType:=", "Tone",
  "SmoothShade:=", True,
  "AddGrid:=", False,
  "MapTransparency:=", True,
  "Transparency:=", 0,
  "ArrowUniform:=", True,
  "ArrowSpacing:=", 0.100000001490116,
  "GridColor:=", [255, 255, 255]]]
)
```

VB Syntax	SetFieldPlotSettings <PlotName> <PlotItemAttributes>
VB Example	<pre>oModule.SetFieldPlotSettings "Mag_E2", _ Array("NAME:FieldsPlotItemSettings", _ Array("NAME:PlotOnLineSettings", _ Array("NAME:LineSettingsID", _ "Width:=", 4, _ "Style:=", "Cylinder"), _</pre>

```

"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 <i><PlotFolderName></i>	Type String	Description Name of the folder with the attributes to modify.

			<pre> <ColorMapSettings>, <Scale3DSettings>, <Marker3DSettings>, <Arrow3DSettings>) </pre>
<code><ColorMapSettings></code>	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>
<code><Scale3DSettings></code>	Array	Structured array.	<pre> Array("NAME:Scale3DSettings", "m_nLevels:=", <int>, "m_autoScale:=", <bool>, "minvalue:=", <double>, "maxvalue:=", <double>, "log:=", <bool>, </pre>

		<pre>"IntrinsicMin:=", <double>, "IntrinsicMax:=", <double>)</pre>
<code><Marker3DSettings></code>	Array	<p>Structured array.</p> <pre>Array ("NAME:Marker3DSettings", "MarkerType:=", <integer 9:Sphere 10: Box 11: Tetrahedron 12: Octahedron default: Sphere>, "MarkerMapSize:=", <bool>, "MarkerMapColor:=", <bool>, "MarkerSize:=", <double>)</pre>
<code><Arrow3DSettings></code>	Array	<p>Structured array.</p> <pre>Array ("NAME:Arrow3DSettings", "ArrowType:=", <integer 0: Line 1: Cylinder 2: Umbrella default: Line>, "ArrowMapSize:=", <bool>, "ArrowMapColor:=", <bool>,</pre>

			"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, "m_autoScale:=", True, "minvalue:=", 9.34379863739014, "maxvalue:=", 13683.755859375, "log:=", False,</pre>

```

    "IntrinsicMin:=", 9.34379863739014,
    "IntrinsicMax:=", 13683.755859375),
    [ "NAME:Marker3DSettings",
      "MarkerType:=", 0,
      "MarkerMapSize:=", True,
      "MarkerMapColor:=", False,
      "MarkerSize:=", 0.25],
    [ "NAME:Arrow3DSettings",
      "ArrowType:=", 1,
      "ArrowMapSize:=", True,
      "ArrowMapColor:=", True,
      "ShowArrowTail:=", True,
      "ArrowSize:=", 0.25]]
)

```

VB Syntax	SetPlotFolderSettings <PlotFolderName>, <PlotFolderAttributes>
VB Example	<pre> oModule.SetPlotFolderSettings "E Field1", _ Array("NAME:FieldsPlotSettings", _ "Real time mode:=", true, _ Array("NAME:ColorMapSettings", _ "ColorMapType:=", "Spectrum", _ </pre>

```
"SpectrumType:=", "Rainbow", _  
"UniformColor:=", Array(127, 255, 255), _  
"RampColor:=", Array(255, 127, 127)), _  
Array("NAME:Scale3DSettings", _  
"m_nLevels:=", 27, _  
"m_autoScale:=", true, _  
"minvalue:=", 9.34379863739014, _  
"maxvalue:=", 13683.755859375, _  
"log:=", false, _  
"IntrinsicMin:=", 9.34379863739014, _  
"IntrinsicMax:=", 13683.755859375), _  
Array("NAME:Marker3DSettings", _  
"MarkerType:=", 0, _  
"MarkerMapSize:=", true, _  
"MarkerMapColor:=", false, _  
"MarkerSize:=", 0.25), _  
Array("NAME:Arrow3DSettings", _  
"ArrowType:=", 1, _  
"ArrowMapSize:=", true, _  
"ArrowMapColor:=", true, _
```

	<pre>"ShowArrowTail:=", true, _ "ArrowSize:=", 0.25))</pre>
--	---

ShowPolarPlot

Delete this text and replace it with your own content.

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>

UpdateQuantityFieldsPlots

Updates the Quantity Fields Plots.

UI Access	N/A								
Parameters	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td><i><FolderName></i></td><td>String</td><td>Folder containing the plotted quantities.</td></tr></tbody></table>			Name	Type	Description	<i><FolderName></i>	String	Folder containing the plotted quantities.
Name	Type	Description							
<i><FolderName></i>	String	Folder containing the plotted quantities.							
Return Value	None.								

Python Syntax	UpdateQuantityFieldsPlots(<i><FolderName></i>)
Python Example	<code>oModule.UpdateQuantityFieldsPlots ("fldplt")</code>

VB Syntax	UpdateQuantityFieldsPlots <FolderName>
VB Example	<code>oModule.UpdateQuantityFieldsPlots "fldplt"</code>

19 - Fields Calculator Script Commands

Fields Calculator commands should be executed by the Field Overlays module, which is called "FieldsReporter" in Icepak scripts.

```
Set oModule = oDesign.GetModule("FieldsReporter")
oModule.CommandName <args>
```

The command associated with each of the following scripting commands will be a button pressed in the Fields Calculator.

[AddNamedExpression](#)

[AddNamedExpr](#)

[CalcOp](#)

[CalculatorRead](#)

[CalcStack](#)

[CalculatorWrite](#)

[ChangeGeomSettings](#)

[ClcEval](#)

[ClcMaterial](#)

[ClearAllNamedExpr](#)

[CopyNamedExprToStack](#)

[DeleteNamedExpr](#)

[EnterComplex](#)

[EnterComplexVector](#)

[EnterLine](#)

[EnterPoint](#)[EnterQty](#)[EnterScalar](#)[EnterScalarFunc](#)[EnterSurf](#)[EnterVector](#)[EnterVectorFunc](#)[EnterVol](#)[ExportOnGrid \[Fields Calculator\]](#)[ExportToFile \[Fields Calculator\]](#)[GetTopEntryValue](#)[LoadNamedExpressions](#)[SaveNamedExpressions](#)

AddNamedExpression

Creates a named expression using the expression at the top of the stack.

UI Access	Click Add in the Fields Calculator dialogue.		
Parameters	Name	Type	Description
	<ExprName>	String	Name for the new named expression.
Return Value	None.		

Python Syntax	AddNamedExpression(< <i>ExprName</i> >, < <i>FieldType</i> >)
Python Example	<code>oModule.AddNamedExpression ("Mag_JxE", "Fields")</code>

VB Syntax	AddNamedExpression < <i>ExprName</i> >, < <i>FieldType</i> >
VB Example	<code>oModule.AddNamedExpression "Mag_JxE", "Fields"</code>

AddNamedExpr

Creates a named expression using the expression at the top of the stack.

UI Access	Click Add in the Fields Calculator dialogue.						
Parameters	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><<i>ExprName</i>></td> <td>String</td> <td>Name for the new named expression.</td> </tr> </tbody> </table>	Name	Type	Description	< <i>ExprName</i> >	String	Name for the new named expression.
Name	Type	Description					
< <i>ExprName</i> >	String	Name for the new named expression.					
Return Value	None.						

Python Syntax	AddNamedExpr(< <i>ExprName</i> >)
Python Example	<code>oModule.AddNamedExpr ("Mag_JxE")</code>

VB Syntax	AddNamedExpr < <i>ExprName</i> >
VB Example	<code>oModule.AddNamedExpr "Mag_JxE"</code>

CalcOp

Performs a calculator operation.

UI Access	Operation commands like Mag , + , etc.		
Parameters	Name <i><OpString></i>	Type String	Description The text on the corresponding calculator button.
Return Value	None.		

Python Syntax	CalcOp(<i><OpString></i>)
Python Example	<code>oModule.CalcOp ("+")</code>

VB Syntax	CalcOp <i><OpString></i>
VB Example	<code>oModule.CalcOp "+"</code>

CalcRead(deprecated)

Reads a file that is written out by the CalcWrite command, and puts the result into a calculator numeric.

UI Access	Click Read... in the Fields Calculator dialog.		
Parameters	Name <i><FileName></i>	Type String	Description Name of the file to be read.
	<i><SoluName></i>	Type String	Description Specified solution to read in.

	<code><VarArray></code>	Array	Array of variables to read.
Return Value	None.		

Python Syntax	<code>CalcRead(<FileName>, <SoluName>, <VarArray>)</code>
Python Example	<pre> oModule.CalcRead("c:\example.reg", "Setup1: LastAdaptive", ["Freq:=", "10GHz", "Phase:=", "0deg"]) </pre>

VB Syntax	<code>CalcRead <FileName>, <SoluName>, <VarArray></code>
VB Example	<pre> oModule.CalcRead _ "c:\example.reg", _ "Setup1: LastAdaptive", _ Array("Freq:=", "10GHz", "Phase:=", "0deg") </pre>

CalculatorRead

Gets a register file and applies it to the calculator stack.

UI Access	Click Read... in the Fields Calculator dialog.								
Parameters	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><code><FileName></code></td> <td>String</td> <td>Path to and including name of input register file.</td> </tr> </tbody> </table>			Name	Type	Description	<code><FileName></code>	String	Path to and including name of input register file.
Name	Type	Description							
<code><FileName></code>	String	Path to and including name of input register file.							

	<i><SoluName></i>	String	Specified solution to read in.
	<i><FieldType></i>	String	Type of specified field.
	<i><VarArray></i>	Array	Array of variable names, value pairs.
Return Value	None.		

Python Syntax	CalculatorRead(<FileName>, <SoluName>, <FieldType>, <VarArray>)
Python Example	<pre>oModule.CalculatorRead("c:\\example.reg", "Setup1: LastAdaptive", "Fields", ["Freq:=", "10GHz", "Phase:=", "0deg"])</pre>

VB Syntax	CalculatorRead <FileName>, <SoluName>, <FieldType>, <VarArray>
VB Example	<pre>oModule.CalculatorRead _ "c:\\example.reg", _ "Setup1: LastAdaptive", "Fields", _ Array("Freq:=", "10GHz", "Phase:=", "0deg")</pre>

CalcStack

Performs an operation on the stack.

UI Access	Stack operation buttons such as Push and Pop .								
Parameters	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><code><OpString></code></td> <td>String</td> <td>The text on the corresponding calculator button.</td> </tr> </tbody> </table>			Name	Type	Description	<code><OpString></code>	String	The text on the corresponding calculator button.
Name	Type	Description							
<code><OpString></code>	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>

CalculatorWrite

Writes contents of top register to file.

UI Access	Click Write... in the Fields Calculator dialog.														
Parameters	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><code><OutputFilePath></code></td> <td>String</td> <td>Path to and including name of output register file.</td> </tr> <tr> <td><code><SoluNameArray></code></td> <td>Array</td> <td>Array of strings containing solution names.</td> </tr> <tr> <td><code><VarArray></code></td> <td>Array</td> <td>Array of variable names, value pairs.</td> </tr> </tbody> </table>			Name	Type	Description	<code><OutputFilePath></code>	String	Path to and including name of output register file.	<code><SoluNameArray></code>	Array	Array of strings containing solution names.	<code><VarArray></code>	Array	Array of variable names, value pairs.
Name	Type	Description													
<code><OutputFilePath></code>	String	Path to and including name of output register file.													
<code><SoluNameArray></code>	Array	Array of strings containing solution names.													
<code><VarArray></code>	Array	Array of variable names, value pairs.													
Return Value	None.														

Python Syntax	CalculatorWrite(<OutputFilePath>, <SoluNameArray>, <VarArray>)
Python Example	<pre> oModule.CalculatorWrite("c:\test.reg", ["Solution:=", "Setup1 : LastAdaptive"], ["Freq:=", "1GHz", "Phase:=", "0deg"]) </pre>

VB Syntax	CalculatorWrite <OutputFilePath>, <SoluNameArray>, <VarArray>
VB Example	<pre> oModule.CalculatorWrite "c:\test.reg", _ Array("Solution:=", "Setup1 : LastAdaptive"), _ Array("Freq:=", "1GHz", "Phase:=", "0deg") </pre>

CalcWrite

Writes contents of top register to file.

UI Access	N/A		
Parameters	Name	Type	Description
	<FilePath>	String	Path to output and including name of output register file.
	<SoluName>	String	Name of solution.
	<VariablesArray>	Array	Array of variable names, value pairs.
Return Value	None.		

Python Syntax	CalcWrite(<FilePath>, <SoluName>, <VariablesArray>)
Python Example	<pre>oModule.CalcWrite("C:\Ansoft\smoothedTemp.fld", "Setup1 : LastAdaptive", ["\$conductivity:=", "50000000"])</pre>

VB Syntax	CalcWrite <FilePath>, <SoluName>, <VariablesArray>
VB Example	<pre>oModule.CalcWrite "C:\Ansoft\smoothedTemp.fld", "Setup1 : LastAdaptive", _ Array("\$conductivity:=", "50000000")</pre>

ChangeGeomSettings

Changes the line discretization setting.

UI Access	In the Fields Calculator dialog box, click on Geom Settings...						
Parameters	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><LineDiscr></td> <td>Integer</td> <td>Line discretization value.</td> </tr> </tbody> </table>	Name	Type	Description	<LineDiscr>	Integer	Line discretization value.
Name	Type	Description					
<LineDiscr>	Integer	Line discretization value.					
Return Value	None.						

Python Syntax	ChangeGeomSettings(<LineDiscr>)
Python Example	<pre>oModule.ChangeGeomSettings(500)</pre>

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	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td><SoluName></td><td>String</td><td>Name of specified solution.</td></tr><tr><td><VarArray></td><td>Array</td><td>Array of variable name, value pairs.</td></tr><tr><td><FieldType></td><td>String</td><td>Optional. Type of specified field.</td></tr></tbody></table>	Name	Type	Description	<SoluName>	String	Name of specified solution.	<VarArray>	Array	Array of variable name, value pairs.	<FieldType>	String	Optional. Type of specified field.
Name	Type	Description											
<SoluName>	String	Name of specified solution.											
<VarArray>	Array	Array of variable name, value pairs.											
<FieldType>	String	Optional. Type of specified field.											
Return Value	None.												

Python Syntax	ClcEval(<SoluName>, <VarArray>, [Optional <FieldType>])
Python Example	oModule.ClcEval("Setup1: LastAdaptive", ["Freq:=", "10GHz", "Phase:=", "0deg"])

VB Syntax	ClcEval <SoluName>, <VarArray>, [Optional <FieldType>]
------------------	--

VB Example <pre>oModule.ClcEval "Setup1: LastAdaptive", _ Array ("Freq:=", "10GHz", _ "Phase:=", "0deg")</pre>
--

ClcMaterial

Performs a material operation on the top stack element.

UI Access	Click Matl... in the Fields Calculator dialog.										
Parameters	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><MatString></td> <td>String</td> <td>The material property to apply.</td> </tr> <tr> <td><OpString></td> <td>String</td> <td>Name of operation. Possible values are "mult", or "div".</td> </tr> </tbody> </table>	Name	Type	Description	<MatString>	String	The material property to apply.	<OpString>	String	Name of operation. Possible values are "mult", or "div".	
Name	Type	Description									
<MatString>	String	The material property to apply.									
<OpString>	String	Name of operation. Possible values are "mult", or "div".									
Return Value	None.										

Python Syntax	ClcMaterial(<MatString>, <OpString>)
Python Example	oModule.ClcMaterial("Permeability (mu)" "mult")

VB Syntax	ClcMaterial <MatString>, <OpString>
VB Example	oModule.ClcMaterial "Permeability (mu)" "mult"

ClcMaterialValue

Shows the value of the material property without performing any operation.

UI Access	Select None in the Material Operation dialog.						
Parameters	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td><MaterialName></td><td>String</td><td>Name of specified material property.</td></tr></tbody></table>	Name	Type	Description	<MaterialName>	String	Name of specified material property.
Name	Type	Description					
<MaterialName>	String	Name of specified material property.					
Return Value	None.						

Python Syntax	ClcMaterialValue(<MaterialName>)
Python Example	<pre>oModule.ClcMaterialValue("Mass Density")</pre>

VB Syntax	ClcMaterialValue <MaterialName>
VB Example	<pre>oModule.ClcMaterialValue "Mass Density"</pre>

ClearAllNamedExpr

Clears all user-defined named expressions from the list.

UI Access	Click ClearAll in the Fields Calculator dialog.
Parameters	None.
Return Value	None.

Python Syntax	ClearAllNamedExpr()
---------------	---------------------

Python Example	<code>oModule.ClearAllNamedExpr()</code>
-----------------------	--

VB Syntax	<code>ClearAllNamedExpr</code>
------------------	--------------------------------

VB Example	<code>oModule.ClearAllNamedExpr</code>
-------------------	--

CopyNamedExprToStack

Copies the named expression selected to the calculator stack.

UI Access	Select a named expression and then click Copy to stack .		
Parameters	Name <code><ExprName></code>	Type String	Description Name of the expression to be copied to the top of the stack.
Return Value	None.		

Python Syntax	<code>CopyNamedExprToStack(<ExprName>)</code>
----------------------	---

Python Example	<code>oModule.CopyNamedExprToStack ("Mag_JxE")</code>
-----------------------	---

VB Syntax	<code>CopyNamedExprToStack <ExprName></code>
------------------	--

VB Example	<code>oModule.CopyNamedExprToStack "Mag_JxE"</code>
-------------------	---

DeleteNamedExpr

Deletes the selected named expression from the list.

UI Access	Select a named expression and then click Delete .						
Parameters	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td><code><ExprName></code></td><td>String</td><td>Name of specified named expression.</td></tr></tbody></table>	Name	Type	Description	<code><ExprName></code>	String	Name of specified named expression.
Name	Type	Description					
<code><ExprName></code>	String	Name of specified named expression.					
Return Value	None.						

Python Syntax	<code>DeleteNamedExpr(<ExprName>)</code>
Python Example	<code>oModule.DeleteNamedExpr ("Mag_JxE")</code>

VB Syntax	<code>DeleteNamedExpr <ExprName></code>
VB Example	<code>oModule.DeleteNamedExpr "Mag_JxE"</code>

DoesNamedExpressionExists

Determines whether specified named expression exists.

UI Access	N/A						
Parameters	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td><code><ExpName></code></td><td>String</td><td>Name of the specified named expression.</td></tr></tbody></table>	Name	Type	Description	<code><ExpName></code>	String	Name of the specified named expression.
Name	Type	Description					
<code><ExpName></code>	String	Name of the specified named expression.					
Return Value	Boolean: <ul style="list-style-type: none">• 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 <ComplexNum>	Type String	Description 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	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td><ComplexVector></td><td>Array</td><td>Array of strings containing X, Y and Z complex values.</td></tr></tbody></table>			Name	Type	Description	<ComplexVector>	Array	Array of strings containing X, Y and Z complex values.
Name	Type	Description							
<ComplexVector>	Array	Array of strings containing X, Y and Z complex values.							
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", _</pre>

	"1 + 2 j", "1 + 2 j")
--	--------------------------

EnterCoord

Enters a coordinate system defined in the 3D Modeler editor.

UI Access	Click Geometry and then select Coord .		
Parameters	Name <CoordName>	Type String	Description Name of a coordinate system defined in the 3D Modeler editor.
Return Value	None.		

Python Syntax	EnterCoord(<CoordName>)
Python Example	oModule.EnterCoord ("Global")

VB Syntax	EnterCoord <CoordName>
VB Example	oModule.EnterCoord "Global"

EnterEdge

Enters an edge defined in the 3D Modeler editor.

UI Access	N/A		
Parameters	Name	Type	Description

	<i><EdgeName></i>	String	Name of an edge defined in the 3D Modeler editor.
Return Value	None.		

Python Syntax	EnterEdge(<EdgeName>)
Python Example	<code>oModule.EnterPoint ("Edge_1")</code>

VB Syntax	EnterEdge <EdgeName>
VB Example	<code>oModule.EnterPoint "Edge_1"</code>

EnterLine

Enters a line defined in the 3D Modeler editor.

UI Access	Click Geometry and then select Line		
Parameters	Name	Type	Description
	<i><LineName></i>	String	Name of a line defined in the 3D Modeler editor.
Return Value	None.		

Python Syntax	EnterLine(<LineName>)
Python Example	<code>oModule.EnterLine ("Line1")</code>

VB Syntax	EnterLine <LineName>
VB Example	<code>oModule.EnterLine "Line1"</code>

EnterOutputVar

Enters Output Vars, only valid for Eigenmode problems.

UI Access	Click Geometry and then select Output Vars .									
Parameters	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><VarName></td> <td>String</td> <td>Name of the output vars. Only 'freq' is supported.</td> </tr> <tr> <td><VarType></td> <td>String</td> <td>Type of the output vars. 'Complex' type.</td> </tr> </tbody> </table>	Name	Type	Description	<VarName>	String	Name of the output vars. Only 'freq' is supported.	<VarType>	String	Type of the output vars. 'Complex' type.
Name	Type	Description								
<VarName>	String	Name of the output vars. Only 'freq' is supported.								
<VarType>	String	Type of the output vars. 'Complex' type.								
Return Value	None.									

Python Syntax	EnterOutputVar(<VarName>, <VarType>)
Python Example	<code>oModule.EnterOutputVar ("Freq", "Complex")</code>

VB Syntax	EnterOutputVar <VarName>, <VarType>
VB Example	<code>oModule.EnterOutputVar "Freq", "Complex"</code>

EnterPoint

Enters a point defined in the 3D Modeler editor.

UI Access	Click Geometry and then select Point .								
Parameters	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td><PointName></td><td>String</td><td>Name of a point defined in the 3D Modeler editor.</td></tr></tbody></table>			Name	Type	Description	<PointName>	String	Name of a point defined in the 3D Modeler editor.
Name	Type	Description							
<PointName>	String	Name of a point defined in the 3D Modeler editor.							
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	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td><FieldQty></td><td>String</td><td>The field quantity to be entered onto the stack.</td></tr></tbody></table>			Name	Type	Description	<FieldQty>	String	The field quantity to be entered onto the stack.
Name	Type	Description							
<FieldQty>	String	The field quantity to be entered onto the stack.							
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	<table border="1"> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> <tr> <td><Scalar></td> <td>Double</td> <td>The real number to enter onto the stack.</td> </tr> </table>	Name	Type	Description	<Scalar>	Double	The real number to enter onto the stack.
Name	Type	Description					
<Scalar>	Double	The real number to enter onto the stack.					
Return Value	None.						

Python Syntax	EnterScalar(<Scalar>)
Python Example	oModule.EnterScalar(3.14159265358979)

VB Syntax	EnterScalar <Scalar>
VB Example	oModule.EnterScalar 3.14159265358979

EnterScalarFunc

Enters a scalar function.

UI Access	Click Function and then select Scalar .		
Parameters	Name <VarName>	Type String	Description Name of a variable to enter as a scalar function onto the stack.
Return Value	None.		

Python Syntax	EnterScalarFunc(<VarName>)
Python Example	oModule.EnterScalarFunc ("Phase")

VB Syntax	EnterScalarFunc <VarName>
VB Example	oModule.EnterScalarFunc "Phase"

EnterSurf

Enters a surface defined in the 3D Modeler editor.

UI Access	Click Geometry and then select Surface .		
Parameters	Name <SurfName>	Type String	Description 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	<table border="1"> <tr> <td>Name</td> <td>Type</td> <td>Description</td> </tr> <tr> <td><VectorArray></td> <td>Array</td> <td>Array of strings containing X, Y and Z components of the vector.</td> </tr> </table>			Name	Type	Description	<VectorArray>	Array	Array of strings containing X, Y and Z components of the vector.
Name	Type	Description							
<VectorArray>	Array	Array of strings containing X, Y and Z components of the vector.							
Return Value	None.								

Python Syntax	EnterVector(<VectorArray>)
Python Example	<code>oModule.EnterVector ([1.0, 1.0, 1.0])</code>

VB Syntax	EnterVector <VectorArray>
------------------	---------------------------

VB Example	<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 <code><VarNames></code>	Type Array	Description Array of strings containing names of a variable for the X, Y, and Z coordinates, respectively, to enter as a vector function on the stack.
Return Value	None.		

Python Syntax	<code>EnterVectorFunc(<VarNames>)</code>
Python Example	<code>oModuleEnterVectorFunc(["X", "Y", "Z"])</code>

VB Syntax	<code>EnterVectorFunc <VarNames></code>
VB Example	<code>oModuleEnterVectorFunc Array("X", "Y", "Z")</code>

EnterVol

Enters a volume defined in the 3D Modeler editor.

UI Access	Click Geometry and then select Volume .
------------------	---

Parameters	<table border="1"> <tr> <th>Name</th><th>Type</th><th>Description</th></tr> <tr> <td><VolumeName></td><td>String</td><td>Name of a volume defined in the 3D Modeler editor.</td></tr> </table>	Name	Type	Description	<VolumeName>	String	Name of a volume defined in the 3D Modeler editor.
Name	Type	Description					
<VolumeName>	String	Name of a volume defined in the 3D Modeler editor.					
Return Value	None.						

Python Syntax	EnterVol(<VolumeName>)
Python Example	oModule.EnterVol ("Box1")

VB Syntax	EnterVol <VolumeName>
VB Example	oModule.EnterVol "Box1"

ExportOnGrid [Fields Calculator]

Evaluates the top stack element at a set of points specified by a grid and exports the data to a file.

UI Access	Click Export , and then click On Grid .																											
Parameters	<table border="1"> <tr> <td>Name</td> <td>Type</td> <td>Description</td> </tr> <tr> <td><OutputFile></td> <td>String</td> <td>Name of the output file.</td> </tr> <tr> <td><Min></td> <td>Array</td> <td>Minimum values for the coordinate components of the grid system</td> </tr> <tr> <td><Max></td> <td>Array</td> <td>Maximum values for the coordinate components of the grid system</td> </tr> <tr> <td><Spacing></td> <td>Array</td> <td>Spacing values for the coordinate components of the grid system</td> </tr> <tr> <td><SolnName></td> <td>String</td> <td>Name of the simulation setup</td> </tr> <tr> <td><VarVals></td> <td>Array</td> <td>Array of strings containing setup definitions.</td> </tr> <tr> <td><IncludePoints></td> <td>Boolean</td> <td>Optional. Specifies whether include points in the output file.</td> </tr> <tr> <td><CSType></td> <td>String</td> <td>Optional. Type of coordinate system. "Cartesian" (default) "Cylindrical" "Spherical"</td> </tr> </table>	Name	Type	Description	<OutputFile>	String	Name of the output file.	<Min>	Array	Minimum values for the coordinate components of the grid system	<Max>	Array	Maximum values for the coordinate components of the grid system	<Spacing>	Array	Spacing values for the coordinate components of the grid system	<SolnName>	String	Name of the simulation setup	<VarVals>	Array	Array of strings containing setup definitions.	<IncludePoints>	Boolean	Optional. Specifies whether include points in the output file.	<CSType>	String	Optional. Type of coordinate system. "Cartesian" (default) "Cylindrical" "Spherical"
Name	Type	Description																										
<OutputFile>	String	Name of the output file.																										
<Min>	Array	Minimum values for the coordinate components of the grid system																										
<Max>	Array	Maximum values for the coordinate components of the grid system																										
<Spacing>	Array	Spacing values for the coordinate components of the grid system																										
<SolnName>	String	Name of the simulation setup																										
<VarVals>	Array	Array of strings containing setup definitions.																										
<IncludePoints>	Boolean	Optional. Specifies whether include points in the output file.																										
<CSType>	String	Optional. Type of coordinate system. "Cartesian" (default) "Cylindrical" "Spherical"																										

	<code><Offsets></code>	Array	Optional. Origin for the offset coordinate system. For Cartesian, x, y, z, for Cylindrical, R, Phi, Z, for Spherical, Rho, Theta, Phi.
	<code><ByCount></code>	Boolean	Optional.
Return Value	None.		

Note: Regarding the **ExportOnGrid** legacy script which only has “IncludePtInOutput” argument (the last Boolean one), AEDT can still read it and assign other new arguments as default values. Those default values are RefCSName = “global”, PtInSI = “True”, FieldInRefCS = “False”

Python Syntax	<code>ExportOnGrid(<OutputFile>, <Min>, <Max>, <Spacing>, <SolName>, <SolParameters>, [<ExportOption>, "IncludePointsInOutput:=", <boolean>], "RefCSName:=", <CSName>, "PtsInSI:=", <boolean>, "FieldRefCS:=", <boolean>], "<CSType>", [<Offsets>, <ByCount>])</code>
Python Example	<pre> oModule.ExportOnGrid("C:\offset_grid_model_unit_ref.fld", ["-1mm", "16mm", "0mm"], ["1mm", "18mm", "1mm"], ["2mm", "2mm", "1mm"], "4500MHz : LastAdaptive", ["Freq:=", "4.5GHz", "Phase:=" , "0deg"], ["NAME:ExportOption", "IncludePtInOutput:=" , True, "RefCSName:=" , "offset", "PtInSI:=" , False,] </pre>

	<pre> "FieldInRefCS:=" , True] , "Cartesian", ["0mm", "0mm", "0mm"], False) </pre>
--	--

VB Syntax	ExportOnGrid(<OutputFile>, <Min>, <Max>, <Spacing>, <SolnName>, <SolnParameters>, [<ExportOption>, "IncludePointsInOutput:=", <boolean>], "RefCSName:=", <CSName>, "PtsInSI:=", <boolean>, "FieldRefCS:=", <boolean>], "<CSType>", [<Offsets>, <ByCount>])
VB Example	<pre> oModule.ExportOnGrid "C:/Grid.fld", _ Array("0mm", "0deg", "-25mm"), _ Array("20mm", "90deg", "125mm"), _ Array("10mm", "45deg", "50mm"), _ "Setup1 : LastAdaptive", _ Array("Freq:=", "10000Hz", "Phase:=", "0deg"), _ true, "Cylindrical", _ Array("0mm", "0mm", "0mm") oModule.ExportOnGrid("C:\offset_grid_model_unit_ref.fld", Array("-1mm", "16mm", "0mm"), Array("1mm", "18mm", "1mm"), Array("2mm", "2mm", "1mm"), </pre>

```
"4500MHz : LastAdaptive",
    Array("Freq:=", "4.5GHz", "Phase:=" , "0deg"),
    [
        "NAME:ExportOption",
        "IncludePtInOutput:=" , True,
        "RefCSName:=" , "offset",
        "PtInSI:=" , False,
        "FieldInRefCS:=" , True
    ],
    "Cartesian", Array("0mm", "0mm", "0mm"), False
)
```

ExportToFile [Fields Calculator]

Evaluates the top stack element at a set of points specified in an external file and exports the data to a file.

Note:

Regarding to legacy **ExportToFile** script which only has “IncludePtInOutput” argument (the last Boolean one), AEDT can still read it and assign other new arguments as default values. Those default values are RefCSName = “global”, PtInSI = “True”, FieldInRefCS = “False”

UI Access

Click **Export**, and then click **To File**.

	Name	Type	Description
Parameters	<ReportName>	String	Name of report to be exported.
	<FileName>	String	Full path of the exported image file name; with extension of .txt - Post processor format file .csv - Comma-delimited data file .tab - Tab-separated file .dat - Ansys plot data file
	<solution>		Solution Name
	<freq>		frequency
	<phase>		phase
	<Export Options>		Whether to include points in output, RefCSName, Pts in SI units <boolean>, Field in RefCS, <boolean>.
	Return Value	None.	

Python Syntax	ExportToFile(<ReportName>, <FileName>, <solution>, ["Freq:=", <freq>, "Phase:=", "<phase>"], <ExportOptions>)
Python Example	<pre>oModule.ExportToFile("C:\\\\offset_file_model_unit_ref.fld", "C:\\\\offset_SI.pts", "4500MHz : LastAdaptive", ["Freq:=" , "4.5GHz", "Phase:=" , "0deg"],</pre>

```
[  

    "NAME:ExportOption",  

    "IncludePtInOutput:=" , True,  

    "RefCSName:=" , "offset",  

    "PtInSI:=" , False,  

    "FieldInRefCS:=" , True  

])
```

VB Syntax	ExportToFile <ReportName>, <FileName>
VB Example	<pre>oModule.ExportToFile("C:\\\\offset_file_model_unit_ref.fld", "C:\\\\offset_SI.pts", "4500MHz : LastAdaptive", ["Freq:=" , "4.5GHz", "Phase:=" , "0deg"], ["NAME:ExportOption", "IncludePtInOutput:=" , True, "RefCSName:=" , "offset",</pre>

```

    "PtInSI:=" , False,
    "FieldInRefCS:=" , True
]
)

```

GetTopEntryValue

Gets the value of the top entry of the calculator stack.

UI Access	N/A									
Parameters	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><SolnName></td> <td>String</td> <td>Name of specified solution.</td> </tr> <tr> <td><VarVals></td> <td>Array</td> <td>Array of variable name, value pairs.</td> </tr> </tbody> </table>	Name	Type	Description	<SolnName>	String	Name of specified solution.	<VarVals>	Array	Array of variable name, value pairs.
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	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><FileName></td> <td>String</td> <td>Filename and full path to the file to hold the named expression definition.</td> </tr> <tr> <td><FieldType></td> <td>String</td> <td>For products with just one filed type, it is set to "Fields".</td> </tr> <tr> <td><NamedExpr></td> <td>Array</td> <td>rray of strings containing the names of expression definitions to load from the file.</td> </tr> </tbody> </table>	Name	Type	Description	<FileName>	String	Filename and full path to the file to hold the named expression definition.	<FieldType>	String	For products with just one filed type, it is set to "Fields".	<NamedExpr>	Array	rray of strings containing the names of expression definitions to load from the file.
Name	Type	Description											
<FileName>	String	Filename and full path to the file to hold the named expression definition.											
<FieldType>	String	For products with just one filed type, it is set to "Fields".											
<NamedExpr>	Array	rray of strings containing the names of expression definitions to load from the file.											
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>
-------------------	---

SaveNamedExpressions

Saves a named expression definition to a file.

UI Access	In the Fields Calculator, click Save To... in the Library area.												
Parameters	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><FileName></td> <td>String</td> <td>Filename and full path to the file to hold the named expression definition.</td> </tr> <tr> <td><NamedExprs></td> <td>Array</td> <td>Array of strings containing the names of expression definitions to load from the file.</td> </tr> <tr> <td><Overwrite></td> <td>Boolean</td> <td>Specifies whether to overwrite the file.</td> </tr> </tbody> </table>	Name	Type	Description	<FileName>	String	Filename and full path to the file to hold the named expression definition.	<NamedExprs>	Array	Array of strings containing the names of expression definitions to load from the file.	<Overwrite>	Boolean	Specifies whether to overwrite the file.
Name	Type	Description											
<FileName>	String	Filename and full path to the file to hold the named expression definition.											
<NamedExprs>	Array	Array of strings containing the names of expression definitions to load from the file.											
<Overwrite>	Boolean	Specifies whether to overwrite the file.											
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 _ </pre>

	"C:\Ansoft\PersonalLib\smth.clc", _ Array("smoothedtemp"), true
--	--

20 - 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 an Icepak design.

UI Access	Icepak > Fields > Create Fields Summary		
Parameters	Name	Type	Description
	<Calculation>	list	"Entity Type", "Geometry Type", "Selected Geometry", "Selected Quantity", "Normal", "Side"
	<Entity Type>	bool	Boundary or Object
	<Geometry Type>	bool	Surface or Volume
	<Selected Geometry>	string	Name of selected geometry
	<Selected Quantity>	string	Name of selected quantity
	<Normal>	int	Coordinate values for direction relative to normal
Return Value	None		

Python Syntax	EditFieldsSummarySetting (<Calculation>, <Calculation>, <Calculation>, <Calculation>, <Calculation>)
Python Example	<pre>oModule.EditFieldsSummarySetting(["Calculation:=" , ["Boundary", "Surface", "CPU", "Temperature", "", "Default"], "Calculation:=" , ["Boundary", "Surface", "Grille1", "Speed", "1.00,-0.00,-0.00", "Default"], "Calculation:=" , ["Boundary", "Surface", "Opening1", "Speed", "-1.00,-0.00,-0.00", "De-</pre>

	<pre> fault"], "Calculation:=" , ["Object", "Sur- face", "AllObjects", "SurfTemperature", "", "Default"], "Calculation:=" , ["Object", "Surface", "ALPHA_MAI[N_- PCB", "SurfTemperature", "", "Adjacent"]])) </pre>
--	---

VB Syntax	EditFieldsSummarySetting (<Calculation>, <Calculation>, <Calculation>, <Calculation>, <Calculation>)
VB Example	<pre> oModule.EditFieldsSummarySetting Array("Calculation:=", Array("Boundary", "Surface", "CPU", "Temperature", "", "Default")) </pre>

ExportFieldsSummary

Exports a fields summary report as a CSV file.

UI Access	Icepak > 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	ExportFieldsSummary (<SolutionName>, <DesignVariationKey>, <ExportFileName>, <IntrinsicValue>)
Python Example	<pre> oModule.ExportFieldsSummary (["SolutionName:=" , "Setup1 : SteadyState", "DesignVariationKey:=" , "Nominal", "ExportFileName:=" , "C:\\Users\\spsander\\OneDrive - A Inc\\Desktop\\AEDT\\R22.1\\summaryReport.csv", "IntrinsicValue:=" , ""]) </pre>

VB Syntax	ExportFieldsSummary (<SolutionName>, <DesignVariationKey>, <ExportFileName>, <IntrinsicValue>)
VB Example	<pre>oModule.ExportFieldsSummary Array("SolutionName:=", "Setup1 : SteadyState", _ "DesignVariationKey:=", "Nominal", "ExportFileName:=", _ "C:\Users\spsander\OneDrive - ANSYS, Inc\Desktop\AEDT\R22.1\summaryReport.csv", _ "IntrinsicValue:=", "")</pre>

This page intentionally
left blank.

21 - 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	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td><data></td><td>Array</td><td>Data that defines the document.</td></tr><tr><td><traces></td><td>Array</td><td>trace data for the inputs in the document.</td></tr></tbody></table>	Name	Type	Description	<data>	Array	Data that defines the document.	<traces>	Array	trace data for the inputs in the document.
Name	Type	Description								
<data>	Array	Data that 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 Syn- tax	AddDocument <data>, <traces>
----------------	------------------------------

VB Example

In this example, the document names come from the UserDefinedDocument folder in the syslib, userlib, and personallib folders. This creates a document and places it in the Documents folder under results.

```
oModule.AddDocument _
    Array("NAME:Design Summary", _
        "", "SysLib", "DesignSummary", _
        Array("NAME:Inputs")), _
    Array("NAME:DocTraces")
```

The following example is explained in [Explication of a Sample UDD Script](#).

```
oModule.AddDocument Array("NAME:Test Report1", "Test Report", "SysLib", _
    "Examples/TestUDDInputs", Array("NAME:Inputs", Array("NAME:DLMetrics", "Solution", _
        "Data Line Metrics", -1, -1), Array("NAME:DQ0", "Trace", "DQ0", -1, -1), _
        Array("NAME:DQS", "Trace", "DQS", -1, -1), _
        Array("NAME:Name", "Text", "User Name", Array("Sita Ramesh")), _
        Array("NAME:Summary", "Bool", "Display Summary", Array(true)), _
        Array("NAME:Version", "Number", "Script Version"))), _
    Array("NAME:DocTraces", Array("NAME:DLMetrics", _
        Array("User Defined", "", "DDR3 AC-Timing 4-DQ1", Array("Context:=", ""), _
            Array("Index:=", Array("All"), "Trise:=", Array("Nominal"), "Tfall:=", _
                Array("Nominal"), "Pulse_Width:=", Array("Nominal")), _
                "Data_Rate:=", Array("Nominal"), "Length:=", Array("Nominal"))), _
            Array("Probe Component:=", Array(""), Array()), _
            Array("NAME:DQ0", Array( _
```

```
"Standard", "DQ0", "NexximTransient", Array("NAME:Context", "SimValueContext:=", Array(_  
1, 0, 2, 0, false, false, -1, 1, 0, 1, 1, "", 0, 0, _  
"DE", false, "0", "DP", false, "20000000", "DT", false, "0.001", "WE", _  
false, "100ns", "WM", false, "100ns", "WN", false, "0ps", "WS", false, "0ps")), _  
Array("Time:=", Array("All"), "Trise:=", Array("Nominal"), "Tfall:=", Array("Nominal"), _  
"Pulse_Width:=", Array("Nominal"), "Data_Rate:=", Array("Nominal"), _  
"Length:=", Array("Nominal")), Array("Probe Component:=", Array("DQ0")), Array())))
```

DeleteAllDocuments

Deletes all documents in the object.

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	<table border="1"> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> <tr> <td><code><name></code></td> <td>String</td> <td>Name of the document to be deleted.</td> </tr> </table>	Name	Type	Description	<code><name></code>	String	Name of the document to be deleted.
Name	Type	Description					
<code><name></code>	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	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><separator></td> <td>String</td> <td>Separator used to convey the directory "level"</td> </tr> </tbody> </table>	Name	Type	Description	<separator>	String	Separator used to convey the directory "level"
Name	Type	Description					
<separator>	String	Separator used to convey the directory "level"					
Return Value	None.						

Python Syntax	GetDocumentDefinitionNames(<separator>)
Python Example	<code>oModule.GetDocumentDefinitionNames ("")</code>

VB Syntax	GetDocumentDefinitionNames <separator>
VB Example	<code>oModule.GetDocumentDefinitionNames "</code>

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	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td><code><oldName></code></td><td>String</td><td>Current name of the document</td></tr><tr><td><code><newName></code></td><td>String</td><td>New name of the document</td></tr></tbody></table>			Name	Type	Description	<code><oldName></code>	String	Current name of the document	<code><newName></code>	String	New name of the document
Name	Type	Description										
<code><oldName></code>	String	Current name of the document										
<code><newName></code>	String	New name of the document										

Return Value	None.
---------------------	-------

Python Syntax	RenameDocument(<oldName>, <newName>)
Python Example	<code>oModule.RenameDocument ("Design Summary", "Project Design Summary")</code>

VB Syntax	RenameDocument <oldName>, <newName>
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	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><name></td> <td>String</td> <td>Name of the document to be saved.</td> </tr> <tr> <td><saveTo></td> <td>String</td> <td>File path to save the document to.</td> </tr> </tbody> </table>	Name	Type	Description	<name>	String	Name of the document to be saved.	<saveTo>	String	File path to save the document to.
Name	Type	Description								
<name>	String	Name of the document to be saved.								
<saveTo>	String	File path to save the document to.								
Return Value	none									

Python Syntax	SaveHtmlDocumentAs(<name>, <saveTo>)
Python Example	<code>oModule.SaveHtmlDocumentAs ("Design Summary 1", "DS1.html")</code>

VB Syntax	SaveHtmlDocumentAs <name> <saveTo>
VB Example	oModule.SaveHtmlDocumentAs "Design Summary 1" "DS1.html"

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	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td><name></td><td>String</td><td>Name of the document to be saved.</td></tr><tr><td><saveTo></td><td>String</td><td>File path to save the document at.</td></tr></tbody></table>	Name	Type	Description	<name>	String	Name of the document to be saved.	<saveTo>	String	File path to save the document at.
Name	Type	Description								
<name>	String	Name of the document to be saved.								
<saveTo>	String	File path to save the document at.								
Return Value	None.									

Python Syntax	SavePdfDocumentAs(<name>, <saveTo>)
Python Example	oModule.SavePdfDocumentAs("Design Summary 1", "DS1.pdf")

VB Syntax	SavePdfDocumentAs <name>, <saveTo>
VB Example	oModule.SavePdfDocumentAs "Design Summary 1", "DS1.pdf"

UpdateAllDocuments

Refreshes the contents of all created documents. This action is made on the folder rather than the individual document.

UI Access	Right click on Results > Documents in the Project Manager and click Update All Documents .
Parameters	None.
Return Value	None.

Python Syntax	<code>UpdateAllDocuments()</code>
Python Example	<code>oModule.UpdateAllDocuments ()</code>

VB Syntax	<code>UpdateAllDocuments</code>
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	<table border="1"> <tr> <td>Name</td> <td>Type</td> <td>Description</td> </tr> <tr> <td><code><name></code></td> <td>String</td> <td>Name of the document to be updated</td> </tr> </table>			Name	Type	Description	<code><name></code>	String	Name of the document to be updated
Name	Type	Description							
<code><name></code>	String	Name of the document to be updated							
Return Value	None.								

Python Syntax	UpdateDocument(<name>)
Python Example	<code>oModule.UpdateDocument ("Test UDD Report")</code>

VB Syntax	UpdateDocument <name>
VB Example	<code>oModule.UpdateDocument "Test UDD Report"</code>

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	<table border="1"><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><name></td><td>String</td><td>Name of the document to be viewed as a HTML</td></tr></table>			Name	Type	Description	<name>	String	Name of the document to be viewed as a HTML
Name	Type	Description							
<name>	String	Name of the document to be viewed as a HTML							
Return Value	None								

Python Syntax	ViewHtmlDocument(<name>)
Python Example	<code>oModule.ViewHtmlDocument ("Design Summary 1")</code>

VB Syntax	ViewHtmlDocument <name>
VB Example	<code>oModule.ViewHtmlDocument "Design Summary 1"</code>

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	<table border="1"> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> <tr> <td><name></td> <td>String</td> <td>Name of the document to be viewed as a PDF</td> </tr> </table>			Name	Type	Description	<name>	String	Name of the document to be viewed as a PDF
Name	Type	Description							
<name>	String	Name of the document to be viewed as a PDF							
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",  
      "Solution",  
      "Data Line Metrics",  
      -1,  
      -1),  
      ' Input name  
      ' Solution Input Type  
      ' Input Description  
      ' Solution ID  
      ' Report ID  
  
'This array contains the trace input.  
Array("NAME:DQ0",  
      "Trace",  
      "DQ0",  
      -1,  
      -1),  
      ' Input name  
      ' Trace Input Type  
      ' Input Description  
      ' Solution ID  
      ' Report ID  
  
'This array contains the text input.  
Array("NAME:Name",  
      "Text",  
      "User Name",  
      Array("Sita Ramesh")),  
      ' Input name  
      ' Text Input Type  
      ' Input Description  
      ' Default Value  
  
'This array contains the Bool input.  
Array("NAME:Summary",  
      "Bool",  
      "Display Summary",  
      Array(true)),  
      ' Input name  
      ' Boolean Input Type  
      ' Input Description  
      ' Default Value  
  
'This array contains the number input.
```

```

Array("NAME:Version",
      ' Input name
      "Number",
      ' Number Input Type
      "Script Version",
      ' Input Description
      Array(1021))),           ' Default Value

'This array contains trace selection for the solution and trace inputs.
Array("NAME:DocTraces",           ' Document traces keyword

'This array has input for "DLMetrics".
Array("NAME:DLMetrics",           ' Input name

'This array defines a trace similar to the UDO. This trace definition is a User defined solution
Array("User Defined", "", "DDR3 AC-Timing 4-DQ1", Array("Context:=", ""), Array("Index:=", Array
("All"), "Trise:=", Array("Nominal"), "Tfall:=", Array("Nominal"), "Pulse_Width:=", Array("Nom-
inal"), "Data_Rate:=", Array("Nominal"), "Length:=", Array("Nominal")), Array("Probe Com-
ponent:=", Array(""))), Array()),

'This array is for input "DQ0".
Array("NAME:DQ0",

'This array defines a trace similar to the UDO. This trace definition is a Standard solution.
Array("Standard", "DQ0", "NexximTransient", Array("NAME:Context", "SimValueContext:=", Array(1,
0, 2, 0, false, false, -1, 1, 0, 1, 1, "", 0, 0, "DE", false, "0", "DP",
false, "20000000", "DT", false, "0.001", "WE", false, "100ns", "WM", false,
"100ns", "WN", false, "0ps", "WS", false, "0ps")), Array("Time:=", Array("All"), "Trise:=",
Array(
"Nominal"), "Tfall:=", Array("Nominal"), "Pulse_Width:=", Array("Nominal"), "Data_Rate:=", Array
("Nominal"), "Length:=", Array("Nominal")), Array("Probe Component:=", Array(
"DQ0")), Array())))

```

Example Python Script: Defining a Document

This script creates a user-defined solution and a document.

```
import ScriptEnv
ScriptEnv.Initialize("Ansoft.ElectronicsDesktop")
oDesktop.RestoreWindow()
oProject = oDesktop.SetActiveProject("BJT_Inverter")
oDesign = oProject.SetActiveDesign("Nexxim1")
oModule = oDesign.GetModule("UserDefinedSolutionModule")

oModule.CreateUserDefinedSolution("UDS Distance Trace Arithmetic Result1", "SysLib", "TraceArith-
metic/Distance Sweep Trace Arithmetic",
[
    "Offset 1:=", "0",
    "Scale 1:=", "1",
    "Offset 2:=", "0",
    "Scale 2:=", "1",
    "Operation:=", "Add"
],
[
[
    [
        "Standard",
        "probe1",
        "Transient",
        [
            style="font-family: monospace;">"NAME:Context",
            style="font-family: monospace;">"SimValueContext:=", [1,0,2,0,False,False,-
1,1,0,1,1,"",0,0,"DE",False,"0","DP",False,"500000000","DT",False,"0.001","NUMLEVELS",False,"0","-
WE",False,"10us","WM",False,"10us","WN",False,"0ns","WS",False,"0ns"]
        ],
        [
            "Time:=", ["All"]
        ],
        [
            "Probe Component:=", ["V(Port1)"]
        ]
    ]
]
```

```
        ],
        []
    ],
    [
        "Standard",
        "probe2",
        "Transient",
        [
            "NAME:Context",
            "SimValueContext:=", [1,0,2,0,False,False,-
1,1,0,1,1,"",0,0,"DE",False,"0","DP",False,"500000000","DT",False,"0.001","NUMLEVELS",False,"0",-"
"WE",False,"10us","WM",False,"10us","WN",False,"0ns","WS",False,"0ns"]
        ],
        [
            "Time:=", ["All"]
        ],
        [
            "Probe Component:=", ["V(Port1)"]
        ],
        []
    ]
],
[])
oModule = oDesign.GetModule("UserDefinedDocuments")
oModule.AddDocument(
[
    "NAME:Test Report",
    "Test Report",
    "SysLib",
    "TestUDDInputs",
    [
        "NAME:Inputs",
        [
            "NAME:UDS1",

```

```
        "Solution",
        "UDS Distance Trace Arithmetic Result1",
        1000000,
        0
    ],
    [
        "NAME:UDS2",
        "Solution",
        "UDS Distance Trace Arithmetic Result2",
        1000000,
        2
    ]
],
[
    "NAME:DocTraces",
    [
        "NAME:UDS1",
        [
            "User Defined",
            "",
            "UDS Distance Trace Arithmetic Result1",
            [
                "Context:=", ""
            ],
            [
                "Distance:=", ["All"]
            ],
            [
                "Probe Component:=", []
            ],
            []
        ]
    ]
]
```

```
        ],
    ],
    [
        "NAME:UDS2",
        [
            "User Defined",
            "",
            "UDS Distance Trace Arithmetic Result1",
            [
                "Context:=", ""
            ],
            [
                "Distance:=", ["All"]
            ],
            [
                "Probe Component:=", []
            ],
            []
        ]
    ],
    []
)
])
```

This page intentionally
left blank.

22 - 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.

		<p>For example:</p> <pre>Array("multiply_factor:=", "2.0", "component_name:=", "resistor1")</pre>
<ProbeSelections>	Array	Name of the probe being specified. Note: this must match a probe name specified in the UDS plugin file.
<DynamicProbes>	Array	Array of <ProbeSelection>'s, representing the probes that are used by dynamic-probes.
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>

```

Array(), Array("Freq:=", Array( "All")), _
Array("Probe Component:=", Array("dB(S(1,1))"), Array()), _
Array( "Modal Solution Data", "Probe 2",
"Setup1 : LastAdaptive", Array(), _
Array("Freq:=", Array( "All")), _
Array("Probe Component:=", Array("mag(S(1,1))"), Array()), _
Array(Array("Modal Solution Data", _
"Dynamic Probe 1", "Setup1 : LastAdaptive", Array(), _
Array("Freq:=", Array( "All")), _
Array("Probe Component:=", Array("Freq")), Array())))

```

DeleteUserDefinedSolutions

Deletes one or more user defined solutions.

UI Access	Delete button from the User Defined Solutions dialog.						
Parameters	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><SolNames></td> <td>Array</td> <td>Array of strings containing names of User Defined Solutions to be deleted.</td> </tr> </tbody> </table>	Name	Type	Description	<SolNames>	Array	Array of strings containing names of User Defined Solutions to be deleted.
Name	Type	Description					
<SolNames>	Array	Array of strings containing names of User Defined Solutions to be deleted.					
Return Value	None.						

Python Syntax	DeleteUserDefinedSolutions(<SolNames>)
Python Example	oModule.DeleteUserDefinedSolutions(["Solution1", "Solution2"])

VB Syntax	DeleteUserDefinedSolutions <SolNames>
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	<table border="1"> <tr> <td>Name</td> <td>Type</td> <td>Description</td> </tr> <tr> <td><SolName></td> <td>String</td> <td>Name of user defined solution to be edited.</td> </tr> <tr> <td><NewSolName></td> <td>String</td> <td>New name for the specified user defined solution.</td> </tr> <tr> <td><LibType></td> <td>String</td> <td>Indicates the library where the UDS plugin file is located. This parameter must be one of the following values: "SysLib", "UserLib", "PersonalLib".</td> </tr> <tr> <td><RelativePath></td> <td>String</td> <td>The path of the UDS plugin file relative to the "UserDefinedOutputs" sub-directory of the library specified by <LibType>.</td> </tr> <tr> <td><PropList></td> <td>Array</td> <td> Strings specify name-value pairs corresponding to the UDS properties specified in the plugin file. For example: <pre>Array("multiply_factor:=", "2.0", "component_name:=", "resistor1")</pre> </td> </tr> <tr> <td><ProbeSelections></td> <td>Array</td> <td>Name of the probe being specified. Note: this must match a probe name specified in the UDS plugin file.</td> </tr> <tr> <td><DynamicProbes></td> <td>Array</td> <td>Array of <ProbeSelection>'s, representing the probes that are used by dynamic-probes.</td> </tr> </table>			Name	Type	Description	<SolName>	String	Name of user defined solution to be edited.	<NewSolName>	String	New name for the specified user defined solution.	<LibType>	String	Indicates the library where the UDS plugin file is located. This parameter must be one of the following values: "SysLib", "UserLib", "PersonalLib".	<RelativePath>	String	The path of the UDS plugin file relative to the "UserDefinedOutputs" sub-directory of the library specified by <LibType>.	<PropList>	Array	Strings specify name-value pairs corresponding to the UDS properties specified in the plugin file. For example: <pre>Array("multiply_factor:=", "2.0", "component_name:=", "resistor1")</pre>	<ProbeSelections>	Array	Name of the probe being specified. Note: this must match a probe name specified in the UDS plugin file.	<DynamicProbes>	Array	Array of <ProbeSelection>'s, representing the probes that are used by dynamic-probes.
Name	Type	Description																									
<SolName>	String	Name of user defined solution to be edited.																									
<NewSolName>	String	New name for the specified user defined solution.																									
<LibType>	String	Indicates the library where the UDS plugin file is located. This parameter must be one of the following values: "SysLib", "UserLib", "PersonalLib".																									
<RelativePath>	String	The path of the UDS plugin file relative to the "UserDefinedOutputs" sub-directory of the library specified by <LibType>.																									
<PropList>	Array	Strings specify name-value pairs corresponding to the UDS properties specified in the plugin file. For example: <pre>Array("multiply_factor:=", "2.0", "component_name:=", "resistor1")</pre>																									
<ProbeSelections>	Array	Name of the probe being specified. Note: this must match a probe name specified in the UDS plugin file.																									
<DynamicProbes>	Array	Array of <ProbeSelection>'s, representing the probes that are used by dynamic-probes.																									
Return Value	String name of update user defined solution.																										

Python Syntax	EditUserDefinedSolution(<SolName>, <NewSolName>, <LibType>, <RelativePath>, <PropList>, <ProbeSelections>, <DynamicProbes>)
Python Example	<pre> oModule.EditUserDefinedSolution("ConstantTimestep1" "ConstantTimestep1After", "SysLib", "ConstantTimestep", [], [], []) </pre>

VB Syntax	EditUserDefinedSolution <SolName>, <NewSolName>, <LibType>, <RelativePath>, <PropList>, <ProbeSelections>, <DynamicProbes>
VB Example	<pre> oModule.CreateUserDefinedSolution "User Defined Solution 1", _ "User Defined Solution 1", "SysLib", "Example", _ Array("multiplication_factor:=", "1.2"), _ Array(Array("Modal Solution Data", "Probe 1", "Setup1 : LastAdaptive", _ Array(), Array("Freq:=", Array("All"))), _ Array("Probe Component:=", Array("dB(S(1,1))))), Array(), _ Array("Modal Solution Data", "Probe 2", "Setup1 : LastAdaptive", Array(), _ 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>

	Array("Freq:=", Array("All")), _ Array("Probe Component:=", Array("Freq")), Array()))
--	---

GetUserDefinedSolutionNames

Retrieves user defined solution names.

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

Python Syntax	GetUserDefinedSolutionNames()
Python Example	oModule.GetUserDefinedSolutionNames ()

VB Syntax	GetUserDefinedSolutionNames
VB Example	oModule.GetUserDefinedSolutionNames

GetUserDefinedSolutionProperties

Obtains properties for a specified user defined solution.

UI Access	N/A
------------------	-----

Parameters	Name	Type	Description
	<SoluName>	String	Name of a specified user defined solution.
Return Value	Array of strings containing properties values.		

Python Syntax	GetUserDefinedSolutionProperties(<SoluName>)
Python Example	oModule.GetUserDefinedSolutionProperties ("ConstantTimestep1")

VB Syntax	GetUserDefinedSolutionProperties <SoluName>
VB Example	oModule.GetUserDefinedSolutionProperties "ConstantTimestep1"

This page intentionally
left blank.

23 - 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:

[CloneMaterial](#)

[DoesMaterialExist](#)

[ExportMaterial](#)

[RemoveMaterial](#)

[RemoveUnusedDefinitions](#)

Related Topics:

[Component Manager Script Commands](#)

[Material Manager Script Commands](#)

[Model Manager Script Commands](#)

[Network Data Explorer Manager Script Commands](#)

[Script and Library Scripts](#)

[Symbol Manager Script Commands](#)

Add [component manager]

Use: Add a component

Command: Tools > Edit Configured Libraries > Components > Add Component

Syntax: Add Array("NAME:<ComponentName>","

```
"Info:=", <ComponentInfo>,
"RefBase:=", <string>, // reference designator
"NumParts:=", <int>, // parts per component
"OriginalComponent:=", <string>
"Terminal:=", <TerminalInfo>,
"Terminal:=", <TerminalInfo>, ...
// The remaining parameters are optional
Array("NAME:Parameters", // any combo of the following
"VariableProp:=", <VariableInfo>,
"CheckboxProp:=", <CheckBoxInfo>,
"ButtonProp:=", <ButtonInfo>,
"TextProp:=", <TextInfo>,
"NumberProp:=", <NumberInfo>,
"SeparatorProp:=", <SeparatorInfo>,
"ValueProp:=", <ValueInfo>,
"MenuProp:=", <MenuInfo>),
```

```
Array("NAME:Properties", // any combo of the following
      "CheckboxProp:=", <CheckBoxInfo>,
      "TextProp:=", <TextInfo>,
      "NumberProp:=", <NumberInfo>,
      "SeparatorProp:=", <SeparatorInfo>,
      "ValueProp:=", <ValueInfo>,
      "MenuProp:=", <MenuInfo>),
      "VPointProp:=", <VPointInfo>,
      "PointProp:=", <PointInfo>),
      Array("Quantities",
            "QuantityProp:=", <QuantityPropInfo>...),
      Array("NAME:CosimDefinitions",
            <CosimDefInfo>,
            <CosimDefInfo>...)
```

Return Value:<string>

```
// composite name of the component.  
// If the name requested conflicts with the name of an existing  
// component, the requested name is altered to be unique.  
// The name returned reflects any change made to be unique.
```

Parameters:<ComponentName>:

```
<string> // simple name of the component

<ComponentInfo>:
    Array("Type:=", <TypeInfo>,
          "NumTerminals:=", <int>,
          "DataSource:=", <string>,
          "ModifiedOn:=", <ModifiedOnInfo>,
          "Manufacturer:=", "<string>,
          "Symbol:=", <string>,
          "Footprint:=", <string>,
          "Description:=", <string>,
          "InfoTopic:=", <string>,
          "InfoHelpFile:=", <string>,
          "IconFile:=", <string>,
          "LibraryName:=", """",
          "OriginalLocation:=", "Project", // Project Location
          "Author:=", <string>,
          "OriginalAuthor:=", <string>,
          "CreationDate:=", <int>)
```

<TypeInfo>:

An integer that is the or-ing of bits for each product listed below. The default setting is 0xffffffff (4294967295) which indicates valid for all products. In the component editing dialog, checking different boxes in the "Specify products for which this component is valid" grid control sets specific flags that correspond to the following hex/decimal settings:

Nexxim -- 100 binary, 4 decimal, 0x4

SIwaveDeNovo -- 1000 binary, 8 decimal, 0x8

Simplorer -- 10000 binary, 16 decimal, 0x10

MaxwellCircuit -- 100000 binary, 32 decimal, 0x20

<ModifiedOnInfo>:

An integer that corresponds to the number of seconds that have elapsed since 00:00 hours, Jan 1, 1970 UTC from the system clock.

<TerminalInfo>:

```
Array(<string>, // symbol pin  
<string> // footprint pin  
<string>, // gate name  
<bool>, // shared  
<int>, // equivalence number  
<int>, // what to do if unconnected: flag as error:0, ignore:1  
<string> // description  
<Nature>)
```

<Nature>:

<string> // content varies as follows

Nexxim/Circuit:

"Electrical" // the only choice

Simplorer:

// several choices

"Electrical", "Magnetic", "Fluidic", "Translational",
"Translational_V", "Rotational", "Rotational_V",
""Radian", "Thermal", or <VHDLPackageName>

<VHDLPackageName>:

<string> // in the form <Library>.<Package>

<Library>:

<string> // name of the VHDL library

<Package>:

<string> // name of the VHDL package

```
<VariableInfo>:  
    Array(<string>, // name  
          <FlagLetters>,  
          <string>, // description  
          "CB:=", <string>, // optional - script for call back  
          <string>) // value: number, variable, or expression  
  
<FlagLetters>:  
    <string> // "D" - has description parameter,  
    // "RD" - readonly & has description parameter,  
    // or "RHD" - readonly, hidden, & has description parameter  
  
<CheckBoxInfo>:  
    Array(<string>, // name  
          <FlagLetters>,  
          <string>, // description  
          "CB:=", <string>, // optional - script for call back  
          <bool>) // value: true or false  
  
<ButtonInfo>:
```

```
Array<string>, // name
<FlagLetters>,
<string>, // description
<string>, // button title
<string>, // extra text
<ClientID>,
"ButtonPropClientData:= ", <ClientdataArray>

<ClientID>:
<int> // specifies Button Prop Client
// 0 - unknown, ButtonPropClientData
// array will be empty
// 1 - Netlist Prop Client
// 2 - not used
// 3 - File Name Prop Client

<ClientdataArray>:
varies with <ClientID>

<ClientId> is 0 or 1: empty array
```

Array()

<ClientID> is 3:

Array("InternalFormatText:=", "<prefix><RelativePath>")

<prefix>:

<string> // "<Project>", "<PersonalLib>", "<UserLib>", or "<SysLib>"

<RelativePath>:

<string> // relative path to file from <prefix>

<TextInfo>:

Array(<string>, // name
<FlagLetters>,
<string>, // description
"CB:=", <string>, // optional - script for call back
<string>) // value: a text string

<NumberInfo>:

Array(<string>, // name
<FlagLetters>,

```
<string>, // description
"CB:=", <string>, // optional - script for call back
<real>, // value: a number
<string>) // units

<SeparatorInfo>:
Array(<string>, // name
<FlagLetters>,
<string>, // description
"CB:=", <string>, // optional - script for call back
<string>) // value: a text string

<ValueInfo>:
Array(<string>, // name
<FlagLetters>,
<string>, // description
"CB:=", <string>, // optional - script for call back
<string>) // value: a number, variable or expression

<MenuPropInfo>:
Array(<string>, // name
```

```
<FlagLetters>,
<string>, // description
<string>, // menu choices - separated by commas
<int>) // 0 based index of current menu choice
```

```
<VPointInfo>:
Array(<string>, // name
<FlagLetters>,
<string>, // description
"CB:=", <string>, // optional - script for call back
<string>, // x value: number with length units
<string>) // y value: number with length units
```

```
<PointInfo>:
Array(<string>, // name
<FlagLetters>,
<string>, // description
"CB:=", <string>, // optional - script for call back
<real>, // x value
<real>) // y value
```

```
<QuantityPropInfo>:
```

```
Array<string>, // name  
<FlagLetters>,  
<string>, // description  
<string>, // value  
<TypeString>,  
<TypeStringDependentInfo>  
  
<TypeString>:  
<string> // "Across", "Through", or "Free"  
  
<TypeStringDependentInfo>:  
  
<TypeString> is "Free":  
<string>, // direction: "In", "Out", "InOut", or "DontCare"  
// Following <string> is not present if direction is "DontCare"  
<string> // when to calculate: "BeforeAnalogSolver",  
// "BeforeStateGraph", "AfterStateGraph", or "DontCareWhen"  
  
<TypeString> is "Across" or "Through":  
<int>, // terminal 1  
<int> // terminal 2
```

```
<CosimDefInfo>
  Array("NAME:CosimDefinition",
    "CosimulatorType:=", <int>,
    "CosimDefName:=", <string> // "HFSS 3D Layout", "Circuit",
    // "Custom", or "Netlist"
    "IsDefinition:=", <bool>,
    final array member(s) vary with CosimDefName)
```

final array members for HFSS 3D Layout:

```
"CosimStackup:=", <string>,
"CosimDmbedRatio:=", <int>
```

final array members for Circuit:

```
"ExportAsNport:=", <int>,
"UsePjt:=", <int>
```

final array member for Custom:

```
"DefinitionCompName:=", <string>
```

final array member for Netlist:

"NetlistString:=", <string>

VB Example:

```
Dim name  
  
oComponentManager.Add (Array("NAME:MyComponent", _  
"Info:=", Array("Type:=", 4294901767, _  
"NumTerminals:=", 2, _  
"DataSource:=", "", _  
"ModifiedOn:=", 1071096503, _  
"Manufacturer:=", "Ansys", _  
"Symbol:=", "bendo", _  
"Footprint:=", "BENDO", _  
"Description:=", "", _  
"InfoTopic:=", "", _  
"InfoHelpFile:=", "", _  
"IconFile:=", "", _  
"LibraryName:=", "", _  
"OriginalLocation:=", "Project", _  
"Author:=", "", _  
"OriginalAuthor:=", "", _
```

```
"CreationDate:=", 1147460679), _  
"Refbase:=", "U", _  
"NumParts:=", 1, _  
"OriginalComponent:=", "", _  
"Terminal:=", Array("n1", _  
"n1", _  
"A", _  
false, _  
0, _  
1, _  
"", _  
"Electrical"), _  
"Terminal:=", Array("n2", _  
"n2", _  
"A", _  
false, _  
1, _  
0, _  
"", _  
"Electrical"), _  
Array("NAME:Parameters", _
```

```
"MenuProp:=", Array("CoSimulator", _  
"D", _  
"", _  
"Default, HFSS 3D Layout,Circuit,Custom,Netlist", _  
0), _  
"ButtonProp:=", Array("CosimDefinition", _  
"D", _  
"", _  
"", _  
"Edit", _  
0, _  
"ButtonPropClientData:=", Array()), _  
Array("NAME:CosimDefinitions", _  
Array("NAME:CosimDefinition", _  
"CosimulatorType:=", 0, _  
"CosimDefName:=", "HFSS 3D Layout", _  
"IsDefinition:=", true, _  
"CosimStackup:=", "Layout stackup", _  
"CosimDmbedRatio:=", 3), _  
Array("NAME:CosimDefinition", _
```

```
"CosimulatorType:=", 1, _  
"CosimDefName:=", "Circuit", _  
"IsDefinition:=", true, _  
"ExportAsNport:=", 0, _  
"UsePjt:=", 0), _  
Array("NAME:CosimDefinition", _  
"CosimulatorType:=", 2, _  
"CosimDefName:=", "Custom", _  
"IsDefinition:=", true, _  
"DefinitionCompName:=", ""), _  
Array("NAME:CosimDefinition", _  
"CosimulatorType:=", 3, _  
"CosimDefName:=", "Netlist", _  
"IsDefinition:=", true, _  
"NetlistString:=", "")))
```

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>

```
"IconFile:="", "",  
"Library:="", "",  
"OriginalLocation:=", "Project",  
"IEEE:="", "",  
"Author:="", "",  
"OriginalAuthor:="", "",  
"CreationDate:=", 1467910746,  
"ExampleFile:="", ""],  
"Refbase:=", "U",  
"NumParts:=", 1,  
"ModSinceLib:=", True,  
"CompExtID:=", 2  
])
```

AddDataset

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

UI Access	Project > Datasets > Add.		
Parameters	Name	Type	Description
	< DatasetdataArray>	Array	Array("NAME:<DatasetName>", Array("NAME:Coordinates", <CoordinateArray>,

		<CoordinateArray>, ...)
<DatasetName>	String	Name of the dataset.
<CoordinateArray>	Array	Array("NAME:Coordinate", "X:=", <double>, "Y:=", <double>)
Return Value	None.	

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

	<code><defBlock></code>	String	Text of the new material definition in block form.
	<code><defFolderName></code>	String	Library type (by definition folder name)
	<code><newTimeStamp></code>	String	New timestamp (time_t as integer number of seconds since 1/1/1970 12:00am, as string), default is current time
	<code><replaceExisting></code>	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	<p>For simple material:</p> <pre>"<PropertyName>:=", <value></pre> <p>For anisotropic material:</p> <pre>["NAME:<PropertyName>", "property_type:=", "AnisoProperty", "unit:=", <Unit>, "component1:=", <value>, "component2:=", <value>, "component3:=", <value>)]</pre>
<PropertyName>	String	<p>Should be one of the following (depending on the material, design, and solution types):</p> <p>Electromagnetic (Maxwell-exclusive material properties omitted, see Maxwell Scripting help):</p> <pre>"permittivity", "permeability", "conductivity", "dielectric_loss_tangent", "magnetic_loss_tangent", "electric_coercivity", "magnetic_coer-</pre>

		<pre> civity", "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", "vis- cosity", "diffusivity", "molecular_mass", "clarity_type" Structural: "mass_density", "youngs_modulus", "poissons_ratio", "thermal_expansion_coefficient" </pre>
<Unit>	String	<p>Possible values (Maxwell-exclusive properties omitted, see Maxwell Scripting Help; other missing entries are unitless):</p> <pre> 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" </pre>

			specific_heat: "J/kg-C" youngs_modulus: "N/m^2" thermal_expansion_coefficient: "1/C"
Return Value	None		

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

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

Add [padstack manager]

Use: Add a padstack

Command: Tools > Edit Configured Libraries > Padstacks > Add Padstack

Syntax: Add Array("NAME:<PadstackName>",<PadstackProperty>, ...)

```
"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", "Sz:=", Array("2.5mm"), "X:=", "0mm", "Y:=", _
"0mm", "R:=", "0deg"), "ant:=", Array("shp:=", "Cir", "Sz:=", 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)))
```

DeleteDataset

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

UI Access	Project > Datasets > Remove.		
Parameters	Name <i><DatasetName></i>	Type String	Description 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>

Edit [component manager]

Modifies an existing component

Command: Tools > Edit Configured Libraries > Components > Edit Component

Syntax: Edit <ComponentName>,

```
Array("NAME:<NewComponentName>",
  "Info:=", <ComponentInfo>,
  "RefBase:=", <string>, // reference designator
  "NumParts:=", <int>, // parts per component
  "OriginalComponent:=", <string>
  "Terminal:=", <TerminalInfo>,
  "Terminal:=", <TerminalInfo>, ...
  // The remaining parameters are optional
  Array("NAME:Parameters", // any combo of the following
    "VariableProp:=", <VariableInfo>,
    "CheckboxProp:=", <CheckBoxInfo>,
    "ButtonProp:=", <ButtonInfo>,
    "TextProp:=", <TextInfo>,
    "NumberProp:=", <NumberInfo>,
    "SeparatorProp:=", <SeparatorInfo>,
    "ValueProp:=", <ValueInfo>,
    "MenuProp:=", <MenuInfo>),
  Array("NAME:Properties", // any combo of the following
    "CheckboxProp:=", <CheckBoxInfo>,
    "TextProp:=", <TextInfo>,
    "NumberProp:=", <NumberInfo>,
```

```
"SeparatorProp:=", <SeparatorInfo>,
"ValueProp:=", <ValueInfo>,
"MenuProp:=", <MenuInfo>),
"VPointProp:=", <VPointInfo>,
"PointProp:=", <PointInfo>),
Array("Quantities",
"QuantityProp:=", <QuantityPropInfo>...),
Array("NAME:CosimDefinitions",
<CosimDefInfo>,
<CosimDefInfo>...)
```

Return Value: <string>

```
// composite name of the component.
// If the name requested conflicts with the name of an existing
// component, the requested name is altered to be unique.
// The name returned reflects any change made to be unique.
```

Parameters: <ComponentName>:

```
<string> // composite name of the component to edit
```

```
<NewComponentName>:
```

```
<string> // new simple name for the component
```

```
<ComponentInfo>:  
  Array("Type:=", <TypeInfo>,  
        "NumTerminals:=", <int>,  
        "DataSource:=", <string>,  
        "ModifiedOn:=", <ModifiedOnInfo>,  
        "Manufacturer:=", "<string>,  
        "Symbol:=", <string>,  
        "Footprint:=", <string>,  
        "Description:=", <string>,  
        "InfoTopic:=", <string>,  
        "InfoHelpFile:=", <string>,  
        "IconFile:=", <string>,  
        "LibraryName:=", <string>,  
        "OriginalLocation:=", <string>, // Project Location  
        "Author:=", <string>,  
        "OriginalAuthor:=", <string>,  
        "CreationDate:=", <int>)
```

```
<TypeInfo>:
```

An integer that is the or-ing of bits for each product listed below. The default setting is 0xffffffff (4294967295) which indicates valid for all products. In the component editing dialog, checking different boxes in the "Specify products for which this component is valid" grid control sets specific flags that correspond to the following hex/decimal settings:

Nexxim -- 100 binary, 4 decimal, 0x4

SIwaveDeNovo -- 1000 binary, 8 decimal, 0x8

Simplorer -- 10000 binary, 16 decimal, 0x10

MaxwellCircuit -- 100000 binary, 32 decimal, 0x20

<ModifiedOnInfo>:

An integer that corresponds to the number of seconds that have elapsed since 00:00 hours, Jan 1, 1970 UTC from the system clock.

<TerminalInfo>:

```
Array(<string>, // symbol pin  
<string> // footprint pin  
<string>, // gate name  
<bool>, // shared  
<int>, // equivalence number  
<int>, // what to do if unconnected: flag as error:0, ignore:1  
<string>, // description  
<Nature>)
```

<Nature>:

<string> // content varies as follows

Nexxim/Circuit:

"Electrical" // the only choice

Simplorer:

// several choices

"Electrical", "Magnetic", "Fluidic", "Translational",
"Translational_V", "Rotational", "Rotational_V",
""Radian", "Thermal", or <VHDLPackageName>

<VHDLPackageName>:

<string> // in the form <Library>.<Package>

<Library>:

<string> // name of the VHDL library

<Package>:

<string> // name of the VHDL package

```
<VariableInfo>:  
    Array(<string>, // name  
          <FlagLetters>,  
          <string>, // description  
          "CB:=", <string>, // optional - script for call back  
          <string>) // value: number, variable, or expression  
  
<FlagLetters>:  
    <string> // "D" - has description parameter,  
    // "RD" - readonly & has description parameter,  
    // or "RHD" - readonly, hidden, & has description parameter  
  
<CheckBoxInfo>:  
    Array(<string>, // name  
          <FlagLetters>,  
          <string>, // description  
          "CB:=", <string>, // optional - script for call back  
          <bool>) // value: true or false
```

```
<ButtonInfo>:  
    Array(<string>, // name  
        <FlagLetters>,  
        <string>, // description  
        <string>, // button title  
        <string>, // extra text  
        <ClientID>,  
        "ButtonPropClientData:= ", <ClientdataArray>)
```

```
<ClientID>:  
    <int> // specifies Button Prop Client  
    // 0 - unknown, "ButtonPropClientData  
    // array will be empty  
    // 1 - Netlist Prop Client  
    // 2 - not used  
    // 3 - File Name Prop Client
```

```
<ClientdataArray>:  
varies with <ClientID>
```

<ClientID> is 0 or 1: empty array

Array()

<ClientID> is 3:

Array("InternalFormatText:", "<prefix><RelativePath>")

<prefix>:

<string> // "<Project>", "<PersonalLib>", "<UserLib>", or "<SysLib>"

<RelativePath>:

<string> // relative path to file from <prefix>

<TextInfo>:

Array(<string>, // name

<FlagLetters>,

<string>, // description

"CB:=", <string>, // optional - script for call back

<string>) // value: a text string

<NumberInfo>:

```
Array(<string>, // name
      <FlagLetters>,
      <string>, // description
      "CB:=", <string>, // optional - script for call back
      <real>, // value: a number
      <string>) // units

<SeparatorInfo>:
  Array(<string>, // name
        <FlagLetters>,
        <string>, // description
        "CB:=", <string>, // optional - script for call back
        <string>) // value: a text string

<ValueInfo>:
  Array(<string>, // name
        <FlagLetters>,
        <string>, // description
        "CB:=", <string>, // optional - script for call back
        <string>) // value: a number, variable or expression

<MenuPropInfo>:
```

```
Array(<string>, // name  
<FlagLetters>,  
<string>, // description  
<string>, // menu choices - separated by commas  
<int>) // 0 based index of current menu choice  
  
<VPointInfo>:  
Array(<string>, // name  
<FlagLetters>,  
<string>, // description  
"CB:=", <string>, // optional - script for call back  
<string>, // x value: number with length units  
<string>) // y value: number with length units  
  
<PointInfo>:  
Array(<string>, // name  
<FlagLetters>,  
<string>, // description  
"CB:=", <string>, // optional - script for call back  
<real>, // x value  
<real>) // y value
```

```
<QuantityPropInfo>:  
    Array(<string>, // name  
        <FlagLetters>,  
        <string>, // description  
        <string>, // value  
        <TypeString>,  
        <TypeStringDependentInfo>)  
  
<TypeString>:  
    <string> // "Across", "Through", or "Free"  
  
<TypeStringDependentInfo>:  
  
    "Free":  
        <string>, // direction: "In", "Out", "InOut", or "DontCare"  
        // Following <string> is not present if direction is "DontCare"  
        <string> // when to calculate: "BeforeAnalogSolver",  
        // "BeforeStateGraph", "AfterStateGraph", or "DontCareWhen"  
  
    "Across" or "Through"  
    <int>, // terminal 1
```

<int> // terminal 2

<CosimDefInfo>:

```
Array("NAME:CosimDefinition",
"CosimulatorType:=", <int>,
"CosimDefName:=", <string> // "HFSS3D", "Circuit",
// "Custom", or "Netlist"
"IsDefinition:=", <bool>,
final array member(s) vary with CosimDefName)
```

final array members for HFSS 3D Layout:

```
"CosimStackup:=", <string>,
"CosimDmbedRatio:=", <int>
```

final array members for Circuit:

```
"ExportAsNport:=", <int>,
"UsePjt:=", <int>
```

final array member for Custom:

```
"DefinitionCompName:=", <string>
```

final array member for Netlist:

"NetlistString:=", <string>

VB Example:

```
Dim name

name = oComponentManager.Edit ("Nexxim Circuit Elements\BJTs:Level01_NPN", _
Array("NAME:Level01_NPN", _
"Info:=", Array("Type:=", 4294901764,_
"NumTerminals:=", 3, _
"DataSource:=", "Ansys built-in component", _
"ModifiedOn:=", 1152722112, _
"Manufacturer:=", "", _
"Symbol:=", "nexx_bjt_npn", _
"Footprint:=", "", _
"Description:=", "BJT, GP, NPN", _
"InfoTopic:=", "NXBJT1.htm", _
"InfoHelpFile:=", "nexximcomponents.chm", _
"IconFile:=", "bjtsn.bmp", _
"Library:=", "Nexxim Circuit Elements\BJTs", _
"OriginalLocation:=", "SysLibrary ", _
"Author:=", "", _
```

```
"OriginalAuthor:=", "", _  
"CreationDate:=", 1152722102), _  
"Refbase:=", "Q", _  
"NumParts:=", 1, _  
"Terminal:=", Array("collector", _  
"collector", _  
"A", _  
false, _  
6, _  
0, _  
"", _  
"Electrical"), _  
"Terminal:=", Array("base", _  
"base", _  
"A", _  
false, _  
7, _  
0, _  
"", _  
"Electrical"), _
```

```
"Terminal:=", Array("emitter", _  
"emitter", _  
"A", _  
false, _  
8, _  
0, _  
"", _  
"Electrical"), _  
Array("NAME:Parameters", _  
"TextProp:=", Array("LabelID", _  
"HD", _  
"Property string for netlist ID", _  
"Q@ID"), _  
"TextProp:=", Array("MOD", _  
"D", _  
"Name of model data reference", _  
"required"), _  
"VariableProp:=", Array("AREA", _  
"D", _  
"Emitter area multiplying factor, which affects currents, resistances, and capacitances", _ "1"), _  
"VariableProp=", Array("AREAB", _
```

```
"D", _  
"Base AREA", _  
"1"), _  
"VariableProp:=", Array( "AREAC", _  
"D", _  
Collector AREA", _  
"1"), _  
"VariableProp:=", Array("DTEMP", _  
"D", _  
"The difference between element and circuit temperature deg Cel)", _  
"0"), _  
"VariableProp:=", Array("M", _  
"D", _  
"Multiplier factor to simulate multiple BJTs in parallel", _ "1"), _  
"ButtonProp:=", Array("NexximNetlist", _  
"HD", _  
"", _  
"Q@ID %0 %1 %2 *MOD(@MOD) *AREA(AREA=@AREA)" & _  
" *AREAB (AREAB=@AREAB) *AREAC (AREAC=@" & _  
"AREAC) *DTEMP (DTEMP=@DTEMP) *M (M=@M)", _
```

```
"Q@ID %0 %1 %2 *MOD(@MOD) " & _ "*AREA(AREA=@AREA) AREAB(AREAB=@AREAB) *AREAC(AREAC=@" & _  
"AREAC) *DTEMP(DTEMP=@DTEMP) *M(M=@M)", _  
1, _  
"ButtonPropClientData:=", Array(), _  
"TextProp:=", Array( "modelName", _  
"HD", _  
"", _  
"Q")))
```

VB Example:

```
Dim name2  
  
name2 = oComponentManager.Edit "MyComponent", _  
(Array("NAME:MyOtherComponent", _  
"Info:=", Array("Type:=", 4294901767, _  
"NumTerminals:=", 2, _  
"DataSource:=", "", _  
"ModifiedOn:=", 1071096503, _  
"Manufacturer:=", "Ansys", _  
"Symbol:=", "bendo", _  
"Footprint:=", "BENDO", _  
"Description:=", "", _  
"InfoTopic:=", "", _
```

```
"InfoHelpFile:="", "", _
"IconFile:="", "", _
"LibraryName:="", "", _
"OriginalLocation:=", "Project", _
"Author:="", "", _
"OriginalAuthor:="", "", _
"CreationDate:= ", 1147460679), _
"Refbase:=", "U", _
"NumParts:=", 1, _
"OriginalComponent:="", "", _
"Terminal:=", Array("n1", _
"n1", _
"A", _
false, _
0, _
0, _
"", _
"Electrical"), _
"Terminal:=", Array("n2", _
"n2", _
```

```
"A", _  
false, _  
1, _  
0, _  
"", _  
Electrical"), _  
Array("NAME:Parameters", _  
"MenuProp:=", Array("CoSimulator", _  
"D", _  
"", _  
"Default, HFSS3D, Circuit, Custom, Netlist", _  
0), _  
"ButtonProp:=", Array("CosimDefinition", _  
"D", _  
"", _  
"", _  
"Edit", _  
0, _  
"ButtonPropClientData:=", Array()), _  
Array("NAME:CosimDefinitions", _  
Array("NAME:CosimDefinition", _
```

```
"CosimulatorType:=", 0, _  
"CosimDefName:=", "HFSS3D", _  
"IsDefinition:=", true, _  
"CosimStackup:=", "Layout stackup", _  
"CosimDmbedRatio:=", 3), _  
Array("NAME:CosimDefinition", _  
"CosimulatorType:=", 1, _  
"CosimDefName:=", "Circuit", _  
"IsDefinition:=", true, _  
"ExportAsNport:=", 0, _  
"UsePjt:=", 0), _  
Array("NAME:CosimDefinition", _  
"CosimulatorType:=", 2, _  
"CosimDefName:=", "Custom", _  
"IsDefinition:=", true, _  
"DefinitionCompName:=", ""), _  
Array("NAME:CosimDefinition", _  
"CosimulatorType:=", 3, _  
"CosimDefName:=", "Netlist", _  
"IsDefinition:=", true, _
```

```
"NetlistString:=", "") ))
```

	<pre>Edit <ComponentName>, ["NAME:<NewComponentName>", "Info:=", <ComponentInfo>, "RefBase:=", <string>, // reference designator "NumParts:=", <int>, // parts per component "OriginalComponent:=", <string> "Terminal:=", <TerminalInfo>, "Terminal:=", <TerminalInfo>, ... #The remaining parameters are optional ["NAME:Parameters", // any combo of the following "VariableProp:=", <VariableInfo>, "CheckboxProp:=", <CheckBoxInfo>, "ButtonProp:=", <ButtonInfo>, "TextProp:=", <TextInfo>, "NumberProp:=", <NumberInfo>, "SeparatorProp:=", <SeparatorInfo>, "ValueProp:=", <ValueInfo>, "MenuProp:=", <MenuInfo>], ["NAME:Properties", # any combo of the following</pre>
--	---

	<pre> "CheckboxProp:=", <CheckBoxInfo>, "TextProp:=", <TextInfo>, "NumberProp:=", <NumberInfo>, "SeparatorProp:=", <SeparatorInfo>, "ValueProp:=", <ValueInfo>, "MenuProp:=", <MenuInfo>), "VPointProp:=", <VPointInfo>, "PointProp:=", <PointInfo>), ["Quantities", "QuantityProp:=", <QuantityPropInfo>...], ["NAME:CosimDefinitions", <CosimDefInfo>, <CosimDefInfo>...]) </pre>
Python Example	<pre> name = oComponentManager.Edit ("Simplorer Circuit Elements\BJTs:Level01_NPN", _ ["NAME:Level01_NPN", "Info:=", ["Type:=", 4294901764, _ "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", _</pre>

```
"Library:=", "Nexxim Circuit Elements\BJTs",_
"OriginalLocation:=", "SysLibrary ", "Author:=", "", _
"OriginalAuthor:=", "", "CreationDate:=", 1152722102], _
"Refbase:=", "Q", "NumParts:=", 1, "Terminal:=", ["collector", _
"collector", "A", false, 6, 0, "", "Electrical"], _
"Terminal:=", ["base", "base", "A", false, _
7, 0, "", "Electrical"], "Terminal:=", ["emitter", _
"emitter", "A", false, 8, 0, "", "Electrical"], _
["NAME:Parameters", "TextProp:=", ["LabelID", _
"HD", "Property string for netlist ID", _
"Q@ID"], "TextProp:=", ["MOD", "D", _
"Name of model data reference", "required"], _
"VariableProp:=", ["AREA", "D", _
"Emitter area multiplying factor, which affects
currents, resistances, and capacitances", "1"], _
"VariableProp:=", ["AREAB", "D", "Base AREA", _
"1"], "VariableProp:=", ["AREAC", "D", "Collector AREA", _
"1"], "VariableProp:=", ["DTEMP", "D", _
"The difference between element and circuit temperature (deg Cel)", _
"0"], "VariableProp:=", ["M", "D", _
```

```
"Multiplier factor to simulate multiple BJTs in parallel", _  
"1"], "ButtonProp:=", ["NexximNetlist", "HD", "", _  
"Q@ID %0 %1 %2 *MOD(@MOD) *AREA(AREA=@AREA)" & _  
" *AREAB (AREAB=@AREAB) *AREAC (AREAC=@" & _  
"AREAC) *DTEMP (DTEMP=@DTEMP) *M(M=@M)", _  
"Q@ID %0 %1 %2 *MOD(@MOD) " & "*AREA(AREA=@AREA)  
*AREAB (AREAB=@AREAB) *AREAC (AREAC=@" & _  
"AREAC) *DTEMP (DTEMP=@DTEMP) *M(M=@M)", 1, _  
"ButtonPropClientData:", []],  
"TextProp:=", ["modelName", "HD", "", "Q"]]))
```

```
name2 = oComponentManager.Edit ("MyComponent", _  
  
( [ "NAME:MyOtherComponent", "Info:=", [ "Type:=", 4294901767, _  
"NumTerminals:=", 2, "DataSource:=", "", _  
"ModifiedOn:=", 1071096503, "Manufacturer:=", "Ansoft", _
```

Python Example 2

```
"Symbol:=", "bendo", "Footprint:=", "BENDO", _

"Description:=", "", "InfoTopic:=", "", _

"InfoHelpFile:=", "", "IconFile:=", "", _

"LibraryName:=", "", "OriginalLocation:=", "Project", _

"Author:=", "", "OriginalAuthor:=", "", _

"CreationDate:= ", 1147460679], "Refbase:=", "U", _

"NumParts:=", 1, "OriginalComponent:=", "", _

"Terminal:=", ["n1", "n1", "A", false, 0, 0, "", _

"Electrical"], "Terminal:=", ["n2", "n2", "A", _

false, 1, 0, "", Electrical], ["NAME:Parameters", _
```

```
"MenuProp:=", ["CoSimulator", "D", "", _  
  
"Default,Custom,Netlist", 0], "ButtonProp:=", ["CosimDefinition", _  
  
"D", "", "", "Edit", 0, "ButtonPropClientData:=", []]], _  
  
["NAME:CosimDefinitions", ["NAME:CosimDefinition", _  
  
"CosimulatorType:=", 0, "CosimDefName:=", "HFSS3D", _  
  
"IsDefinition:=", true, "CosimStackup:=", "Layout stackup", _  
  
"CosimDmbedRatio:=", 3], ["NAME:CosimDefinition", _  
  
"CosimulatorType:=", 1, "CosimDefName:=", "", _  
  
"IsDefinition:=", true, "ExportAsNport:=", 0, _
```

```

"UsePjt:=", 0], ["NAME:CosimDefinition", _

"CosimulatorType:=", 2, "CosimDefName:=", "Custom", _

"IsDefinition:=", true, "DefinitionCompName:=", ""], _

["NAME:CosimDefinition", "CosimulatorType:=", 3, _

"CosimDefName:=", "Netlist", "IsDefinition:=", true, _

"NetlistString:=", "")]
)

```

EditDataset

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

UI Access	Project > Datasets > Edit.		
Parameters	Name	Type	Description
	<OriginalName>	String	Name of the original 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", [[</pre>

```

        "NAME:Coordinate",
        "X:=", 1, "Y:=", 2
    ],
    [
        "NAME:Coordinate",
        "X:=", 3, "Y:=", 4
    ]
]
)

```

VB Syntax	EditDataset < <i>OriginalName</i> > < <i>DatasetdataArray</i> >
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		
	Name	Type	Description
Parameters	<OriginalName>	String	Name of the material before editing.
	<MatProperties>	Array	<p>Structured array containing material properties:</p> <pre>["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>,]</pre>

		<pre> "specific_heat:=" , <string containing value>, "youngs_modulus:=" , <string containing value>, "poissons_ratio:=" , <string containing value>, "thermal_expansion_coefficient:=" , <string con- taining value>] </pre>
	<p><i><ModifierdataArray></i></p>	<p>Array</p> <p>Optional structured array containing thermal or spatial modifiers:</p> <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 "spatial_modifier">,]]]]</pre>

			<pre> "use_free_form:=" , <Boolean>, "free_form_value:=" , <string modifier value>,]]]]</pre>
Return Value	None.		

Python Syntax	EditMaterial (<OriginalName>, <MatProperties>)
Python Example	<p>Without Modifiers:</p> <pre> oDefinitionManager.EditMaterial("alumina_92pct", ["NAME:alumina_92pct", "CoordinateSystemType:=" , "Cartesian", "BulkOrSurfaceType:=" , 1, ["NAME:PhysicsTypes", "set:=" , ["Electromagnetic","Thermal","Structural"]]]</pre>

```
],
"permittivity:=" , "9.3",
"dielectric_loss_tangent:=" , "0.008",
"thermal_conductivity:=" , "26",
"mass_density:=" , "3720",
"specific_heat:=" , "790",
"youngs_modulus:=" , "267000000000",
"poissons_ratio:=" , "0.26",
"thermal_expansion_coeffcient:=" , "7.2e-006"
]
)
```

With Thermal Modifier:

```
oDefinitionManager>EditMaterial("copper",
[
"NAME:copper",
"CoordinateSystemType:=" , "Cartesian",
"BulkOrSurfaceType:=" , 1,
[
"NAME:PhysicsTypes",
"set:=" , ["Electromagnetic","Thermal","Structural"]
],
```

```
[  
  "NAME:ModifierData",  
    [  
      "NAME:ThermalModifierData",  
        ["modifier_data:=", "thermal_modifier_data",  
          [  
            "NAME:all_thermal_modifiers",  
              [  
                "NAME:one_thermal_modifier",  
                  "Property:::", "permittivity",  
                  "Index:::", 0,  
                  "prop_modifier:::", "thermal_modifier",  
                  "use_free_form:::", True,  
                  "free_form_value:::", "if(Temp > 1000cel, 1, if(Temp < -273.15cel,  
1, 1))"  
                ]  
              ]  
            ]  
        ],  
        "permeability:::", "0.999991",
```

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

Transient Solve, Non-linear Drude Data Plasma

```
# -----
# Script Recorded by Ansys Electronics Desktop Version 2023.2.0
# 14:23:56 Jun 17, 2022
# -----
import ScriptEnv
ScriptEnv.Initialize("Ansoft.ElectronicsDesktop")
oDesktop.RestoreWindow()
oProject = oDesktop.SetActiveProject("Drude_plasma_parameters_r231")
oDefinitionManager = oProject.GetDefinitionManager()
oDefinitionManager.EditMaterial("Drude",
[
    "NAME:Drude",
```

```
"CoordinateSystemType:=", "Cartesian",
"BulkOrSurfaceType:=" , 1,
[

    "NAME:PhysicsTypes",
    "set:=" , ["Electromagnetic"]

] ,
[

    "NAME:AttachedData",
    [

        "NAME:MatNonLinearDrudeFreqDepData",
        "property_data:=" , "nonlinear_drude_data",
        "EpsilonInfinity:=" , "1",
        "PlasmaFrequency:=" , "4.62348462366278GHz",
        "CollisionFrequency:=" , "0.00054491190162662GHz",
        "FieldBreakdown:=" , "1000V_per_meter",
        "PlasmaMaintainFrequency:=" , "2.31174231183139GHz",
        "NeutralDensity:=" , 2.65164580488373E+20,
        "ElectronDensity:=" , 2.65164580488373E+17,
        "CollisionRateConstant:=" , 2.05499505485618E-15

    ]
]
```

]])
--	---------

VB Syn-tax	EditMaterial <OriginalName>, <MatProperties>
VB Example	<p>Without Modifiers:</p> <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", "specific_heat:=", "790", "youngs_modulus:=", "267000000000", "poissons_ratio:=", "0.26", "thermal_expansion_coeffcient:=", "7.2e-006"</pre>

```
)
```

With Thermal Modifier:

```
oDefinitionManager>EditMaterial "copper",
    Array("NAME:copper",
        "CoordinateSystemType:=", "Cartesian",
        "BulkOrSurfaceType:=" , 1,
            Array("NAME:PhysicsTypes",
                "set:=" , Array("Electromagnetic","Thermal","Structural")),
        Array("NAME:ModifierData",
            Array("NAME:ThermalModifierData",
                "modifier_data:=" , "thermal_modifier_data",
                    Array("NAME:all_thermal_modifiers",
                        Array("NAME:one_thermal_modifier",
                            "Property:::" , "permittivity",
                            "Index:::" , 0,
                            "prop_modifier:=" , "thermal_modifier",
                            "use_free_form:=" , True,
                            "free_form_value:=" , "if(Temp > 1000cel, 1, if(Temp < -273.15cel,
1, 1))"
                        )
                    )
                )
            )
        )
    )
)
```

```
)  
)  
,  
"permeability:=" , "0.999991",  
"conductivity:=" , "58000000",  
"thermal_conductivity:=" , "400",  
"mass_density:=" , "8933",  
"specific_heat:=" , "385",  
"youngs_modulus:=" , "120000000000",  
"poissons_ratio:=" , "0.38",  
"thermal_expansion_coefficient:=" , "1.77e-05"  
)
```

Edit [padstack manager]

Use: Edit an existing padstack.

Command: Tools > Edit Configured Libraries > Padstacks > Edit Padstack

Syntax: Edit <PadstackName>,

```
Array("NAME:<NewPadstackName>",  
"ModTime:=", <ModifiedOnInfo>,  
"Library:="", "", // name of the library  
"LibLocation:=", "Project", // location of the named library  
Array("NAME:psd",
```

```
"nam:=", <PadstackName>,
"lib:="", "", // name of the library
"mat:=", "", // hole plating material
"plt:=", "0", // percent of hole's radius filled by plating
Array("NAME:pds",
<LayerGeometryArray>,
<LayerGeometryArray....>),
"hle:=", <PadInfo>
"hRg:=", <HoleRange>,
"sbsb:=", <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 <datasetFilePath>	Type String	Description The full path to the file.
Return Value	None.		

Python Syntax	ExportDataset (<datasetFilePath>)
Python Example	oProject.ExportDataset ('e:/tmp/dsdata.txt')

	<code>oDesign.ExportDataset('e:/tmp/dsdata.txt')</code>
--	---

VB Syntax	<code>ExportDataset <datasetFullPath></code>
VB Example	<code>oProject.ExportDataset "e:/tmp/dsdata.txt"</code> <code>oDesign.ExportDataset "e:/tmp/dsdata.txt"</code>

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	<pre>oDefinitionManager.ExportMaterial (["NAME:mylib", "Material1", "Material2", "Material3"], "PersonalLib")</pre>

VB Syntax	ExportMaterial < <i>ExportData</i> >, < <i>LibraryLocation</i> >
VB Example	<pre>oDefinitionManager.ExportMaterial Array ("NAME:mylib",_ "Material1", "Material2", "Material3"), "PersonalLib"</pre>

Export [padstack manager]

Use: Export a padstack to a library

Command: Tools > Edit Configured Libraries > Padstacks > Export to Library

Syntax: Export Array("NAME:<LibraryName>",

```
<PadstackName>,
<PadstackName>...),
<LibraryLocation>
```

Return Value: None

Parameters: <LibraryName>:

```
<string> // name of the library
```

<PadstackName>:

```
<string> // simple name of padstack to export
```

<LibraryLocation>:

```
<string> // location of the library in <LibraryName>
```

```
// One of "Project", "PersonalLib", or "UserLib"
```

VB Example:

```
oPadstackManager.Export Array("NAME:mylib", "myPadstack"), "PersonalLib"
```

ExportScript

Use: Export to Library in the script definition manager

Command: None

Syntax: ExportScript <ExportData>,<Library location>

Return Value: None

Parameters: <ExportData>

```
Array("NAME:<LibraryName>",<ScriptName>,<ScriptName>,...)
```

VB Example:

```
oProject.ExportComponent Array("NAME:mylib", "myscript"), "PersonalLib"
```

Python Syntax	ExportScript(<ExportData>,<Library location>)
Python Example	<pre>oProject.ExportComponent (["NAME:mylib", "myscript"], "PersonalLib")</pre>

Export [symbol manager]

Use: Exports symbol(s) to a library

Command: Tools > Edit Configured Libraries > Symbols > Export to Library

Syntax: Export Array("NAME:<LibraryName>","

 <SymbolName>,

 <SymbolName>...),

 <LibraryLocation>

Return Value: None

Parameters: <LibraryName>:

 <string> // name of the library

<SymbolName>:

 <string> // composite name of symbol to export

<LibraryLocation>:

 <string> // location of the library in <LibraryName>

 // One of "Project", "PersonalLib", or "UserLib"

VB Example:

```
oSymbolManager.Export _  
Array("NAME Nexxim Circuit Elements\Distribution\Distribution:bendo:mylib", _ "mySymbol"), "Per-  
sonalLib"
```

GetProjectMaterialNames

Returns the material names belonging to an active Project.

UI Access	N/A						
Parameters	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>None</td><td></td><td></td></tr></tbody></table>	Name	Type	Description	None		
Name	Type	Description					
None							
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	<p>One of the following, where tab titles are shown in parentheses:</p> <ul style="list-style-type: none"> • 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>

```

val = oEditor.GetPropertyValues("PassedParameterTab", k, "R")
...

```

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

ImportDataset

Imports a dataset from a named file. This can be executed by the oProject, or oDesign variables. The name of the dataset is file-name+index number (e.g., dsdata1) unless the filename ends with a trailing number. When there is a trailing number at the end, we will remove the number and use first unused index. Alternatively, the name of the dataset can be explicitly defined by providing a string as an optional second argument.

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).
Return Value	None.		

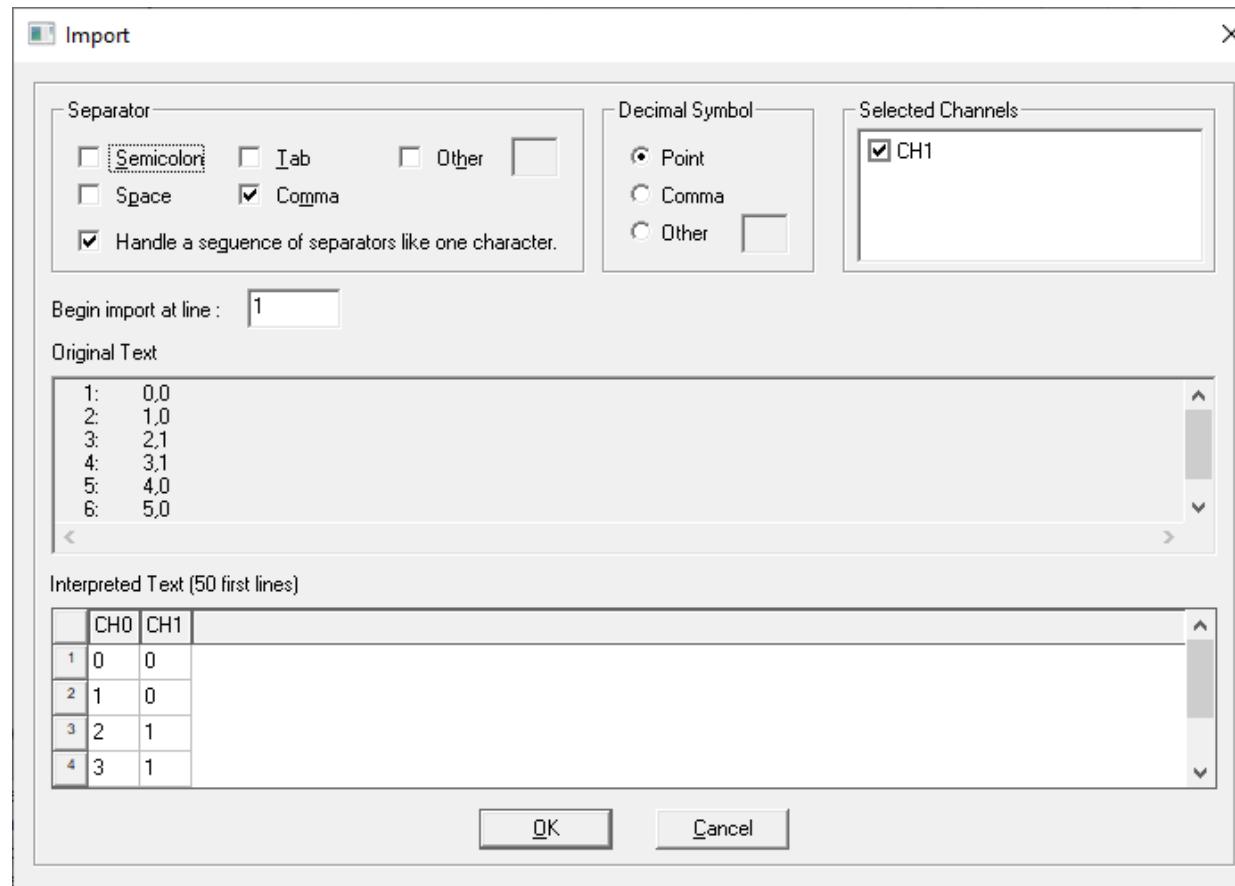
Python Syntax	ImportDataset (<datasetFileFullPath>,<optionalDatasetName>)
Python Example	<pre> oProject.ImportDataset('e:\tmp\dsdata.tab') oDesign.ImportDataset('e:\tmp\dsdata.tab') oProject.ImportDataset('e:\tmp\dsdata.tab', 'MyDatasetName') oDesign.ImportDataset('e:\tmp\dsdata.tab', 'MyDatasetName')</pre>

VB Syntax	ImportDataset <datasetFileFullPath>,<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

```
oDefinitionManager.RemoveMaterial ([  
    "Material1", false, "mo0907", "UserLib"])
```

VB Syntax

```
RemoveMaterial <MaterialName>, <IsProjectMaterial>,  
<LibraryName>, <LibraryLocation>
```

VB Example

```
oDefinitionManager.RemoveMaterial  
    "Material1", false, "mo0907", "UserLib"
```

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	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><Definitions></td> <td>Array</td> <td>Definitions to be removed, such as materials and surface materials.</td> </tr> </tbody> </table>			Name	Type	Description	<Definitions>	Array	Definitions to be removed, such as materials and surface materials.
Name	Type	Description							
<Definitions>	Array	Definitions to be removed, such as materials and surface materials.							
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	<p>One of the following, where tab titles are shown in parentheses:</p> <ul style="list-style-type: none"> • 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 ComplInst@R;2;3
	<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>

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
	<code><targetDefName></code>	String	Name of the target definition, i.e. the name of the material to update.
	<code><defBlock></code>	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-p-l-e	<pre> oProject = oDesktop.NewProject() oProject.InsertDesign("HFSS", "HFSSDesign1", "DrivenModal", "") oDesign = oProject.SetActiveDesign("HFSSDesign1") oEditor = oDesign.SetActiveEditor("3D Modeler") oDefinitionManager = oProject.GetDefinitionManager() defBlock = "\$begin 'vacuum2' \$begin 'AttachedData' \$begin 'MatAppearanceData' property_data-='appearance_data' Red=230 Green=230 Blue=230 Transparency=0.95 \$end 'MatAppearanceData'" \$end 'AttachedData' simple('permittivity', 1) ModTime=1499970477 \$end 'vacuum2' added = oDefinitionManager.AddDefinitionFromBlock(defBlock, "Materials", "10101010", True) addedName = '' if isinstance(added, basestring): </pre>

```
addedName = added

elif isinstance(added, list):

    addedName = added[0]

else:

    addedName = added.GetName().replace("Materials:", "")

AddInfoMessage(os.path.basename(__file__) + " result: " + addedName)

materialNameInQuotes = "\"" + addedName + "\""

# rename vacuum2 to vacuum3

newDefBlock = "$begin 'vacuum3' $begin 'AttachedData' $begin 'MatAppearanceData' property_
data='appearance_data' Red=230 Green=230 Blue=230 Transparency=0.95 $end 'MatAp-
pearanceData' $end 'AttachedData' simple('permittivity', 1) ModTime=1499970477 $end
'vacuum3'"

updatedObj = UpdateDefFromBlock(addedName, newDefBlock, "Materials")
```

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", "permittivity:=", "0.123") )
```

```

oMaterialMgr.Add( Array("NAME:MyMaterial2", "CoordinateSystemType:=", "Cartesian", "permittivity:=", "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 <material_name>	Type string Boolean	Description Name of the project material True or False. If you use False or only a single argument, then GetProperties returns only the properties that are different from the defaults.
Return Value	Array of material data		

Python Syntax	GetProperties (<materialname>)
Python Example	<pre> oMaterialManager.GetProperties("Gold", True) oMaterialManager.GetProperties("vacuum") </pre>

VB Syntax	GetProperties <material_name>
VB Example	materialProps = oMaterialMgr.GetProperties("MyMaterial2")

IsUsed [material manager]

Use: Checks if a project material is in use

Command: None

Syntax: IsUsed <material_name>

Return Value: Returns 'True' if the material is in use.

Parameters: <material_name>

Type: string

Value: Name of the project material to check.

VB Example:

```
used = oMaterialMgr.IsUsed( "MyMaterial2" )
```

Python Syntax	IsUsed(<ComboName>)
Python Example	isused = oMaterialManager.IsUsed("mylib:mymaterial")

RemoveUnused [material manager]

Use: Remove all unused materials from the project.

Command: None

Syntax: RemoveUnused

Return Value: None

Parameters: None

VB Example:

```
oMaterialMgr.RemoveUnused()
```

Python Syntax	RemoveUnused()
Python Example	<pre>thereWereExtras = oMaterialManager.RemoveUnused()</pre>

Model Manager Script Commands

The model manager provides access to models in a project. The manager object is accessed via the definition manager.

```
Set oDefinitionManager = oProject.GetDefinitionManager()  
Set oModelManager = oDefinitionManager.GetManager("Model")
```

The model manager script commands are listed below:

[Add](#)

[ConvertToDynamic](#)

[ConvertToParametric](#)

[Edit](#)

[EditWithComps](#)

[Export](#)

[GetData](#)

[GetNames](#)

[IsUsed](#)

[Remove](#)

[RemoveUnused](#)

Add [model manager]

Use: Add a model

Command: Tools > Edit Configured Libraries > Models > Add Model

Syntax: Add Array("NAME:<modelName>","

```
"ModTime:=", <ModifiedTimeInfo>,  
"Library:=", "", // Library name  
"LibLocation:=", "Project", // Project Location  
<PinDefInfo>,  
<PinDefInfo>,... // optional, to define pins  
<GraphicsDataInfo>, // optional, to define graphics  
<PropDisplayMapInfo>)) // optional, to define property displays
```

Return Value: <string>

```
// composite name of the model.  
// If the name requested conflicts with the name of an existing  
// model, the requested name is altered to be unique.  
// The name returned reflects any change made to be unique.
```

Parameters: <modelName>:

<string> // [simple name](#) of the model being added

<ModifiedOnInfo>:

An integer that corresponds to the number of seconds that have elapsed since 00:00 hours, Jan 1, 1970 UTC from the system clock.

<PinDefInfo>:

```
Array("NAME:PinDef",
"Pin:=", Array (<string>, // pin name
<real>, // x location
<real>, // y location
<real>, // angle in radians
<PinType>,
<real>, // line width
<real>, // line length
<bool>, // mirrored
<int>, // color
<bool>, // true if visible, false if not
<string>, // hidden net name
```

```
<OptionalPinInfo>, // optional info  
<PropDisplayMapInfo>)) // optional
```

```
<PinType>:  
  <string> // "N" : normal pin  
  // "I" : input pin  
  // "O" : output pin
```

```
<OptionalPinInfo>:  
  // Specify both or neither  
  <bool>, // true if name is to be shown  
  <bool>, // true if number is to be shown
```

```
<PropDisplayMapInfo>:  
  Array("NAME:PropDisplayMap",  
    <PropDisplayInfo>,  
    <PropDisplayInfo>,...)
```

```
<PropDisplayInfo>:
```

```
<NameString>, Array(<DisplayTypeInfo>,
<DisplayLocationInfo>,
<int>, // optional, level number
<TextInfo>)
<NameString>:
<string> // PropertyName:=, where PropertyName is the name of
// the property to be displayed

<DisplayTypeInfo>:
<int> // 0 : No display
// 1 : Display name only
// 2 : Display value only
// 3 : Display both name and value
// 4: Display evaluated value only
// 5: Display both name and evaluated value

<DisplayLocationInfo>:
<int> // 0 : Left
// 1 : Top
// 2 : Right
// 3 : Bottom
```

```
// 4 : Center  
// 5 : Custom placement  
  
<GraphicsDataInfo>:  
    Array("NAME:Graphics",  
        // one or more of the following  
        <RectInfo>,  
        <CircleInfo>,  
        <ArcInfo>,  
        <LineInfo>,  
        <PolygonInfo>,  
        <TextInfo>,  
        <ImageInfo>)  
  
<RectInfo>:  
    "Rect:=", Array(<real>, // line width  
                    <int>, // fill pattern  
                    <int>, // color  
                    <real>, // angle, in radians  
                    <real>, // x position of center
```

```
<real>, // y position of center  
<real>, // width  
< real>) // height  
  
<CircleInfo>:  
"Circle:=", Array(<real>, // line width  
<int>, // fill pattern  
<int>, // color  
<real>, // x position of center  
<real>, // y position of center  
< real>) // radius  
  
<ArcInfo>:  
"Arc:=", Array(<real>, // line width  
<int>, // line pattern  
<int>, // color  
<real>, // x position of center  
<real>, // y position of center  
< real>, // radius  
<real>, // start angle, in radians  
<end>) // end angle, in radians
```

```
<LineInfo>:  
  "Line:=", Array(<real>, // line width  
    <int>, // line pattern  
    <int>, // color  
    <PointInfo>, // must specify at least 2 points  
    <PointInfo>...)  
  <PointInfo>:  
    <real>, // x position  
    <real> // y position  
  
<PolygonInfo>:  
  "Polygon:=", Array(<real>, // line width  
    <int>, // fill pattern  
    <int>, // color  
    <PointInfo>, // must specify at least 3 points  
    <PointInfo>...)  
  
<TextInfo>:  
  "Text:=", Array(<real>, // x position
```

```
<real>, // y position  
<real>, // angle, in radians  
<Justification>,  
<bool>, // is plotter font  
<string>, // font name  
<int>, // color  
<string>) // text string
```

<Justification>:

<int> // 0 : left top

// 1 : left base

// 2 : left bottom

// 3 : center top

// 4 : center base

// 5 : center bottom

// 6 : right top

// 7 : right base

// 8 : right bottom

<ImageInfo>:

"Image:=", Array(<RectInfo>,

```
<ImageData>,
<bool>) // is mirrored

<ImageData>:
<string>, // file path
<int>, // 0 : use the file path and link to it
// 1 : ignore file path and use next parameter
<string> // text data, only present if preceding int is 1
```

VB Example:

```
oModelManager.Add Array("NAME:MyModel", _
"ModTime:=", 1070989137, _
"Library:=", "", _
"LibLocation:=", "Project", _
Array("NAME:PinDef", _
"Pin:=", Array ("newpin0", _
0.00254, _
0, _
0, _
"N", _
0, _
```

```
0.00254, _
false, _
0, _
true, _
"",_
false, _
false)),
Array("NAME:PinDef",
"Pin:=", Array ("newpin1",
-0.00254, _
0, _
3.14159265358979, _
"N",_
0, _
0.00254, _
false, _
0, _
true, _
"",_
false, _
false)),
```

```
Array("NAME:Graphics", _  
"Rect:=", Array(0, _  
0, _  
12566272, _  
0, _  
4.33680868994202e-019, _  
-0.000635, _  
0.00508, _  
0.002794), _  
"Circle:=", Array(0, _  
0, _  
12566272, _  
0.000127, _  
-0.000635, _  
0.000635)))
```

ConvertToDynamic

Use: Build a new dynamic model based on an existing parametric model.

Command: Right-click on a model under Definitions/Models in the Project Tree and choose ConvertToDynamic.

Syntax: ConvertToDynamic(defName, newname)

Return Value: <newname> // Name of the new model added

Parameters: <defName> // Model that is the base for the new conversion

VB Example:

```
Dim newname  
  
newname = oModelManager.ConvertToDynamic([in] BSTR defName, [out, retval] BSTR* newName);
```

ConvertToParametric

Use: Build a new parametric model based on an existing dynamic model.

Command: Right-click on a model under Definitions/Models in the Project Tree and choose ConvertToParametric.

Syntax: ConvertToParametric(defName, newname)

Return Value: <newname> // Name of the new model added

Parameters: <defName> // Model that is the base for the new conversion

VB Example: Dim newname

```
newname = oModelManager.ConvertToParametric([in] BSTR defName, [out, retval] BSTR* newName);
```

Edit [deprecated]

Deprecated command — please use [EditWithComps](#).

EditWithComps [model manager]

Use: Edit an existing model.

Command: None

Syntax: EditWithComps <modelName>,

```
Array("NAME:<New modelName>",  
"ModTime:=", <Modified Time Info>,  
"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	dataArray = oModelManager.GetData ("mylib:mymodel")

GetNames [model manager]

Use: Returns the names of the models (used and unused) in a design. The following script command, **IsUsed**, can then be used to separate used and unused models.

Command: None

Syntax: GetNames()

Return Value: An array of strings

Parameters: None

VB Example:

```
Dim modelNames
```

```
modelNames = oModelManager.GetNames()
```

Python Syntax	GetNames()
Python Example	modelnames = oModelManager.GetNames()

IsUsed [model manager]

Use: Used to determine if a model is used in the design.

Command: None

Syntax: IsUsed(<ModelName>)

Return Value: <Boolean> // true if the specified model is used in the design

Parameters: <ModelName>:

<string>

VB Example:

```
Dim isUsed
```

```
isUsed = oModelManager.IsUsed("MyModel")
```

Python Syntax	IsUsed(<ComboName>)
Python Example	isused = oModelManager.IsUsed ("mylib:mymodel")

Remove [model manager]

Use: Removes a model from a library

Command: Tools > Edit Configured Libraries > Models > Remove Model

Syntax: Remove <ModelName>,

```
<IsProjectModel>,
<LibraryName>,
<LibraryLocation>
```

Return Value: None

Parameters: <modelName>:

```
<string> // composite name of the model to remove
```

<IsProjectModel>:

```
<bool>
```

<LibraryName>:

```
<string> // name of the library
```

<LibraryLocation>:

```
<string> // location of the library in <LibraryName>
```

```
// One of "Project", "PersonalLib", or "UserLib"
```

VB Example:

```
oModelManager.Remove "Nexxim Circuit Elements\ Distributed\ Distributed:bendo", true, "Project"
```

Python Syntax

Remove (<modelName>, <IsLocal>, <LibraryName>, <LibraryLocation>)

Python Example

```
oModelManager.Export([  
    "NAME:mylib", "model1", "model2"])
```

RemoveUnused [model manager]

Use: Removes models that are not used in the design.

Command: **Project->Remove Unused Definitions** is similar but operates slightly different and does not record script commands.

Syntax: RemoveUnused()

Return Value: <bool> True if one or more models are removed.

Parameters: None

VB Example:

```
Dim removedDefs
```

```
removedDefs = oModelManager.RemoveUnused()
```

Note:

The order of calls to RemoveUnused is significant. As a result, removing definitions in an unordered fashion may cause other models in dependent definitions to be rendered unusable.

Also, the model and footprint of an unused component are not unusable until after the component itself is removed using the Component Manager Remove script.

Python Syntax	RemoveUnused()
Python Example	<code>thereWereExtras = oModelManager.RemoveUnused()</code>

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 <i><AddScriptArray></i>	Type Array	Description Structured array. Array ("NAME:<string script name>", "ScriptLang:=", <string "vbscript" or "javascript"> "ScriptText:=, <string text of script>")
Return Value	None.		

Python Syntax	AddScript(<AddScriptArray>)
Python Example	<pre>oDefinitionManager.AddScript(["NAME:MyScript", "ScriptLang:=", "vbscript", "ScriptText:=", "MsgBox(\"HelloWorld\")"])</pre>

VB Syntax	AddScript <AddScriptArray>
VB Example	<pre>oDefinitionManager.AddScript Array("NAME:MyScript", "ScriptLang:=", "vbscript", "ScriptText:=", "MsgBox(\"HelloWorld\")")</pre>

EditScript

Edits a script in the definition manager.

UI Access	N/A		
Parameters	Name <OriginalName> <EditScriptArray>	Type String Array	Description Name of the script to be edited. 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", "ScriptLang:=", "vbscript", "ScriptText:=", "MsgBox(""HelloAgain"")") </pre>

ExportScript

Use: Export to Library in the script definition manager

Command: None

Syntax: ExportScript <ExportData>,<Library location>

Return Value: None

Parameters: <ExportData>
Array ("NAME:<LibraryName>", <ScriptName>, <ScriptName>, ...)

VB Example:

```
oProject.ExportComponent Array ("NAME:mylib", "myscript"), "PersonalLib"
```

Python Syntax	ExportScript(<ExportData>,<Library location>)
Python Example	<pre>oProject.ExportComponent (["NAME:mylib", "myscript"], "PersonalLib")</pre>

ModifyLibraries

Use: Configure Libraries on the Tools menu

Command: None

Syntax: ModifyLibraries <DesignName>,Array(<ConfigLibArray>)

Return Value: None

Parameters: <DesignName>
Type: <string>
<ConfigLibArray>
Array ("NAME:<LibraryType>,<ConfiguredLib>,<ConfiguredLib>, ...), ...
<ConfiguredLib> // blank to leave unchanged
<DefinitionType>
Array ("<libraryname >","<libraryname>,...")

VB Example:

```
oDefinitionManager.ModifyLibraries "MyCircuit", _
Array("NAME:PersonalLib"), _
Array("NAME:UserLib"), _
Array("NAME:SystemLib", _
"Symbols:=", Array( "Circuit Elements", "Symbols", _
"ParamExtraElements\PE_Symbols", _
"Vendor Elements\Nonlinear"))
```

Python Syntax	ModifyLibraries (<DesignName>,[<ConfigLibArray>])
Python Example	<pre>oDefinitionManager.ModifyLibraries("MyCircuit", _ ["NAME:PersonalLib"], _ ["NAME:UserLib"], _ ["NAME:SystemLib", _ "Symbols:=", ["Circuit Elements", "Symbols",_ "ParamExtraElements\PE_Symbols", _ "Vendor Elements\Nonlinear"]])</pre>

RemoveScript

Use: Remove Script in the script definition manager

Command: None

Syntax: RemoveScript <ScriptName>,<IsProjectScript>, <LibraryName>,<LibraryLocation>

Return Value: None

Parameters: <ScriptName>

Type: <string>

<IsProjectScript>

Type: <bool>

<LibraryName>

Type: <string>

<LibraryLocation>

Type: <string>

VB Example:

```
oDefinitionManager.RemoveScript "myscript", true, "Local", "Project"
```

Python Syntax	RemoveScript (<ScriptName>,<IsProjectScript>, <LibraryName>,<LibraryLocation>)
Python Example	<pre>oDefinitionManager.RemoveScript("myscript", true, "Local", "Project")</pre>

Symbol Manager Script Commands

The symbol manager provides access to symbols in a project. The manager object is accessed via the definition manager.

```
Set oDefinitionManager = oProject.GetDefinitionManager()
Set oSymbolManager = oDefinitionManager.GetManager("Symbol")
```

The symbol manager script commands are listed below.

[Add](#)

[BringToFront](#)

[Edit](#)

[EditWithComps](#)

[Export](#)

[GetData](#)

[GetNames](#)

[IsUsed](#)

[Remove](#)

[RemoveUnused](#)

Add [symbol manager]

Use: Add a symbol

Command: Tools > Edit Configured Libraries > Symbols > Add Symbol

Syntax: Add Array("NAME:<SymbolName>,"

```
"ModTime:=", <ModifiedTimeInfo>,
"Library:=", "", // Library name
"LibLocation:=", "Project", // Project Location
<PinDefInfo>,
```

```
<PinDefInfo>,... // optional, to define pins  
<GraphicsDataInfo>, // optional, to define graphics  
<PropDisplayMapInfo>)) // optional, to define property displays
```

Return Value: <string>

```
// composite name of the symbol.  
// If the name requested conflicts with the name of an existing  
// symbol, the requested name is altered to be unique.  
// The name returned reflects any change made to be unique.
```

Parameters: <SymbolName>:

```
<string> // simple name of the symbol being added
```

<ModifiedOnInfo>:

An integer that corresponds to the number of seconds that have elapsed
since 00:00 hours, Jan 1, 1970 UTC from the system clock.

<PinDefInfo>:

```
Array("NAME:PinDef",  
"Pin:=", Array (<string>), // pin name
```

```
<real>, // x location  
<real>, // y location  
<real>, // angle in radians  
<PinType>,  
<real>, // line width  
<real>, // line length  
<bool>, // mirrored  
<int>, // color  
<bool>, // true if visible, false if not  
<string>, // hidden net name  
<OptionalPinInfo>, // optional info  
<PropDisplayMapInfo>)) // optional
```

```
<PinType>:  
<string> // "N" : normal pin  
// "I" : input pin  
// "O" : output pin
```

```
<OptionalPinInfo>:  
// Specify both or neither  
<bool>, // true if name is to be shown
```

```
<bool>, // true if number is to be shown

<PropDisplayMapInfo>:
    Array("NAME:PropDisplayMap",
        <PropDisplayInfo>,
        <PropDisplayInfo>,...)

    <PropDisplayInfo>:
        <NameString>, Array(<DisplayTypeInfo>,
            <DisplayLocationInfo>,
            <int>, // optional, level number
            <TextInfo>)
        <NameString>:
            <string> // PropertyName:=, where PropertyName is the name of
            // the property to be displayed

        <DisplayTypeInfo>:
            <int> // 0 : No display
            // 1 : Display name only
```

```
// 2 : Display value only  
// 3 : Display both name and value  
// 4: Display evaluated value only  
// 5: Display both name and evaluated value
```

```
<DisplayLocationInfo>:
```

```
<int> // 0 : Left  
// 1 : Top  
// 2 : Right  
// 3 : Bottom  
// 4 : Center  
// 5 : Custom placement
```

```
<GraphicsDataInfo>:
```

```
Array("NAME:Graphics",  
// one or more of the following  
<RectInfo>,  
<CircleInfo>,  
<ArcInfo>,  
<LineInfo>,  
<PolygonInfo>,
```

```
<TextInfo>,  
<ImageInfo>)  
  
<RectInfo>:  
"Rect:=", Array(<real>, // line width  
<int>, // fill pattern  
<int>, // color  
<real>, // angle, in radians  
<real>, // x position of center  
<real>, // y position of center  
<real>, // width  
< real>) // height
```

```
<CircleInfo>:  
"Circle:=", Array(<real>, // line width  
<int>, // fill pattern  
<int>, // color  
<real>, // x position of center  
<real>, // y position of center  
< real>) // radius
```

```
<ArcInfo>:  
"Arc:=", Array(<real>, // line width  
<int>, // line pattern  
<int>, // color  
<real>, // x position of center  
<real>, // y position of center  
< real>, // radius  
<real>, // start angle, in radians  
<end>) // end angle, in radians
```

```
<LineInfo>:  
"Line:=", Array(<real>, // line width  
<int>, // line pattern  
<int>, // color  
<PointInfo>, // must specify at least 2 points  
<PointInfo>...)  
<PointInfo>:  
<real>, // x position  
<real> // y position
```

```
<PolygonInfo>:  
  "Polygon:=", Array(<real>, // line width  
    <int>, // fill pattern  
    <int>, // color  
    <PointInfo>, // must specify at least 3 points  
    <PointInfo>...)
```

```
<TextInfo>:  
  "Text:=", Array(<real>, // x position  
    <real>, // y position  
    <real>, // angle, in radians  
    <Justification>,  
    <bool>, // is plotter font  
    <string>, // font name  
    <int>, // color  
    <string>) // text string
```

```
<Justification>:  
  <int> // 0 : left top  
  // 1 : left base
```

```
// 2 : left bottom  
// 3 : center top  
// 4 : center base  
// 5 : center bottom  
// 6 : right top  
// 7 : right base  
// 8 : right bottom
```

```
<ImageInfo>:  
"Image:=", Array(<RectInfo>,  
<ImageData>,  
<bool>) // is mirrored
```

```
<ImageData>:  
<string>, // file path  
<int>, // 0 : use the file path and link to it  
// 1 : ignore file path and use next parameter  
<string> // text data, only present if preceding int is 1
```

VB Example:

```
oSymbolManager.Add Array("NAME:MySymbol",_  
"ModTime:=", 1070989137, _
```

```
"Library:=", "", _
"LibLocation:=", "Project", _
Array("NAME:PinDef", _
"Pin:=", Array ("newpin0", _
0.00254, _ 0, _
0, _
"N", _
0, _
0.00254, _
false, _
0, _
true, _
"", _
false, _
false)), _
Array("NAME:PinDef", _
"Pin:=", Array ("newpin1", _
-0.00254, _
0, _
3.14159265358979, _
```

```
"N", _  
0, _  
0.00254, _  
false, _  
0, _  
true, _  
"", _  
false, _  
false)),  
Array("NAME:Graphics", _  
"Rect:=", Array(0, _  
0, _  
12566272, _  
0, _  
4.33680868994202e-019, _  
-0.000635, _  
0.00508, _  
0.002794), _  
"Circle:=", Array(0, _  
0, _  
12566272, _
```

```
0.000127, _  
0.000635, _  
0.000635) ))
```

BringToFront [symbol manager]

Use: Changes the drawing for the symbol so that the specified objects are drawn on top of other overlapping objects.

Command: Draw > Bring To Front

Syntax: BringToFront Array("NAME:Selections", "Selections:=", Array (<Object>, <Object>, ...))

Return Value: None

Parameters: <Object>

<string> // object to bring to the front

VB Example:

```
oDefinitionEditor.BringToFront Array("NAME:Selections", "Selections:=", Array( "SchObj@10"))
```

Edit [deprecated]

Deprecated command — please use [EditWithComps](#).

EditWithComps [symbol manager]

Use: Edit an existing symbol.

Command: None

Syntax: EditWithComps <SymbolName>,

```
Array("NAME:<NewSymbolName>",
      "ModTime:=", <ModifiedTimeInfo>,
      "Library:=", <string>, // Library name
      "LibLocation:=", <string>, // Project Location
      <PinDefInfo>,
      <PinDefInfo>,... // optional, to define pins
      <GraphicsDataInfo>, // optional, to define graphics
      <PropDisplayMapInfo>), // optional, to define property displays
      Array(<ListOfComponentNames>) // Component names
```

Return Value: <string>

```
// composite name of the symbol.  
// If the name requested conflicts with the name of an existing  
// symbol, the requested name is altered to be unique.  
// The name returned reflects any change made to be unique.
```

Parameters: <SymbolName>:

```
<string> // composite name of the symbol being edited
```

<NewSymbolName>:

```
<string> // new simple name for the symbol
```

<ModifiedOnInfo>:

An integer that corresponds to the number of seconds that have elapsed since 00:00 hours, Jan 1, 1970 UTC from the system clock.

<PinDefInfo>:

```
Array("NAME:PinDef",
"Pin:=", Array (<string>, // pin name
<real>, // x location
<real>, // y location
<real>, // angle in radians
<PinType>,
<real>, // line width
<real>, // line length
<bool>, // mirrored
<int>, // color
<bool>, // true if visible, false if not
<string>, // hidden net name
<OptionalPinInfo>, // optional info
<PropDisplayMapInfo>)) // optional
```

```
<PinType>:  
  <string> // "N" : normal pin  
  // "I" : input pin  
  // "O" : output pin  
  
<OptionalPinInfo>:  
  // Specify both or neither  
  <bool>, // true if name is to be shown  
  <bool>, // true if number is to be shown  
  
<PropDisplayMapInfo>:  
  Array("NAME:PropDisplayMap",  
    <PropDisplayInfo>,  
    <PropDisplayInfo>,...)  
  
<PropDisplayInfo>:  
  <NameString>, Array(<DisplayTypeInfo>,  
    <DisplayLocationInfo>,  
    <int>, // optional, level number  
    <TextInfo>)
```

```
<NameString>
<string> // PropertyName:=, where PropertyName is the name of
// the property to be displayed

<DisplayTypeInfo>
<int> // 0 : No display
// 1 : Display name only
// 2 : Display value only
// 3 : Display both name and value
// 4: Display evaluated value only
// 5: Display both name and evaluated value

<DisplayLocationInfo>
<int> // 0 : Left
// 1 : Top
// 2 : Right
// 3 : Bottom
// 4 : Center
// 5 : Custom placement
```

```
<GraphicsDataInfo>
  Array("NAME:Graphics",
    // one or more of the following
    <RectInfo>,
    <CircleInfo>,
    <ArcInfo>,
    <LineInfo>,
    <PolygonInfo>,
    <TextInfo>,
    <ImageInfo>

  <RectInfo>:
    "Rect:=", Array(<real>, // line width
    <int>, // fill pattern
    <int>, // color
    <real>, // angle, in radians
    <real>, // x position of center
    <real>, // y position of center
    <real>, // width
    <real>) // height
```

```
<CircleInfo>:  
  "Circle:=", Array(<real>, // line width  
    <int>, // fill pattern  
    <int>, // color  
    <real>, // x position of center  
    <real>, // y position of center  
    < real>) // radius
```

```
<ArcInfo>:  
  "Arc:=", Array(<real>, // line width  
    <int>, // line pattern  
    <int>, // color  
    <real>, // x position of center  
    <real>, // y position of center  
    < real>, // radius  
    <real>, // start angle, in radians  
    <end>) // end angle, in radians
```

```
<LineInfo>:
```

```
"Line:=", Array(<real>, // line width  
    <int>, // line pattern  
    <int>, // color  
    <PointInfo>, // must specify at least 2 points  
    <PointInfo>...)  
<PointInfo>:  
    <real>, // x position  
    <real> // y position  
  
<PolygonInfo>:  
"Polygon:=", Array(<real>, // line width  
    <int>, // fill pattern  
    <int>, // color  
    <PointInfo>, // must specify at least 3 points  
    <PointInfo>...)  
  
<TextInfo>:  
"Text:=", Array(<real>, // x position  
    <real>, // y position  
    <real>, // angle, in radians  
    <Justification>,
```

```
<bool>, // is plotter font  
<string>, // font name  
<int>, // color  
<string>) // text string
```

<Justification>:

```
<int> // 0 : left top  
// 1 : left base  
// 2 : left bottom  
// 3 : center top  
// 4 : center base  
// 5 : center bottom  
// 6 : right top  
// 7 : right base  
// 8 : right bottom
```

<ImageInfo>:

```
"Image:=", Array(<RectInfo>,  
<ImageData>,  
<bool>) // is mirrored
```

```
<ImageData>
<string>, // file path
<int>, // 0 : use the file path and link to it
// 1 : ignore file path and use next parameter
<string> // text data, only present if preceding int is 1

<ListOfComponentNames>
<string>,<string> ...
// The list may be empty. When not empty, each string that is listed is a component
// that references the symbol to be edited. Prior to editing, a clone of the symbol is
// made, and the components that are listed are modified so that they now refer to
// the clone.
```

VB Example:

```
Dim nam
oSymbolManager>EditWithComps _
("Nexxim Circuit Elements\ Distributed\ Distributed:bendo", _
Array("NAME:bendo new name", _
"ModTime:=", 1152722165, _
"Library:=", "Nexxim Circuit Elements\ Distributed\ Distributed", _
```

```
"LibLocation:=", "SysLibrary", _  
Array("NAME:PinDef", _  
"Pin:=", Array( "n1", _  
-0.00254, _  
0.00254, _  
0, _  
"N", _  
0, _  
0, _  
false, _  
0, _  
true, _  
"", _  
false, _  
false)), _  
Array("NAME:PinDef", _  
"Pin:=", Array( "n2", _  
0.00254, _  
-0.00254, _  
1.5708, _
```

```
"N", _
0, _
0, _
false, _
0, _
true, _
"", _
false, _
false)), _
Array("NAME:Graphics", _
"Polygon:=", Array( 0, 0, 12566272, 0, 0.00381, 0, .00127, 0.00127, _
0.00127, 0.00127, 0, 0.00381, 0, 0.00381, _
0.002032, 0.00127, 0.00381), _
"Line:=", Array(0, 1, 12566272, -0.00254, 0.00254, 0, .00254), _
"Line:=", Array(0, 1, 12566272, 0.00254, -0.00254, .00254, 0)), _
Array("NAME:PropDisplayMap", _
"W:=", Array(3, _
5, _
0, _
"Text:=", Array( 2.1684E-019, _
0.00504119, _
```

```
0, _
1, _
5, _
false, _
"Arial", _
0, _
"W=***") )) ), _
Array("MY_COMP")
```

Export [symbol manager]

Use: Exports symbol(s) to a library

Command: Tools > Edit Configured Libraries > Symbols > Export to Library

Syntax: Export Array("NAME:<LibraryName>","

```
<SymbolName>,
<SymbolName>...),
<LibraryLocation>
```

Return Value: None

Parameters: <LibraryName>:

```
<string> // name of the library
```

<SymbolName>:

<string> // [composite name](#) of symbol to export

<LibraryLocation>:

<string> // location of the library in <LibraryName>

// One of "Project", "PersonalLib", or "UserLib"

VB Example:

```
oSymbolManager.Export _  
Array("NAME NEXXIM Circuit Elements\Distribution\Distribution:bendo:mylib", _ "mySymbol"), "Per-  
sonalLib"
```

GetData [symbol manager]

Use: Gets data that describes the definition.

Command: None

Syntax: GetData(<DefinitionName>)

Return Value: <DefinitionData> This is an array of data for the definition.

Parameters: <DefinitionName>:

<string> // [composite name](#) of the definition to edit

VB Example:

```
Dim symbolData  
  
symbolData = oSymbolManager.GetData("NEXXIM Circuit Elements\HSPICE:nexx_bjt_npn")
```

Note:

GetData allows the user to access definition information, make modifications, and then use the Edit or EditWithComps script commands to save the modified definition. Accordingly, for each type of definition, the array data returned to GetData should be the same array information that is supplied to the [Edit](#) or [EditWithComps](#) commands.

GetNames [symbol manager]

Use: Returns the names of the symbols (used and unused) in a design. The following script command, **IsUsed**, can then be used to separate used and unused symbols.

Command: None

Syntax: GetNames()

Return Value: An array of strings

Parameters: None

VB Example:

```
Dim symbolNames  
symbolNames = oSymbolManager.GetNames()
```

IsUsed [symbol manager]

Use: Used to determine if a symbol is used in the design.

Command: None

Syntax: IsUsed(<SymbolName>)

Return Value: <Boolean> // true if the specified symbol is used in the design

Parameters: <SymbolName>:

<string>

VB Example: Dim isUsed

```
isUsed = oSymbolManager.IsUsed("MySymbol")
```

Remove [symbol manager]

Use: Removes a symbol from a library

Command: Tools > Edit Configured Libraries > Symbols > Remove Symbol

Syntax: Remove <SymbolName>,

```
<IsProjectSymbol>,  
<LibraryName>,  
<LibraryLocation>
```

Return Value: None

Parameters: <SymbolName>:

<string> // [composite name](#) of the symbol to remove

<IsProjectSymbol>:

<bool>

<LibraryName>:

<string> // name of the library

<LibraryLocation>:

<string> // location of the library in <LibraryName>

```
// One of "Project", "PersonalLib", or "UserLib"
```

VB Example:

```
oSymbolManager.Remove "Nexxim Circuit Elements\ Distributed\ Distributed:bendo", true, "Project"
```

RemoveUnused [symbol manager]

Use: Removes symbols that are not used in the design.

Command: **Project->Remove Unused Definitions** is similar but operates slightly different and does not record script commands.

Syntax: RemoveUnused()

Return Value: <bool> True if one or more symbols are removed.

Parameters: None

VB Example:

```
Dim removedDefs  
removedDefs = oSymbolManager.RemoveUnused()
```

Note:

The order of calls to RemoveUnused is significant. As a result, removing definitions in an unordered fashion may cause other symbols in dependent definitions to be rendered unusable.

Also, the symbol and footprint of an unused component are not unusable until after the component itself is removed using the Component Manager Remove script.

This page intentionally
left blank.

24 - 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 <i><message></i>	Type String	Description 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 <i><message></i>	Type String	Description Error message.
Return Value	None.		

Python Syntax	AddFatalMessage(<i><message></i>)
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 <i><message></i>	Type String	Description Informational message.
Return Value	None.		

Python Syntax	AddInfoMessage(<i><message></i>)
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	<table border="1"> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> <tr> <td><message></td> <td>String</td> <td>Warning message.</td> </tr> </table>			Name	Type	Description	<message>	String	Warning message.
Name	Type	Description							
<message>	String	Warning message.							
Return Value	None.								

Python Syntax	AddWarningMessge(<message>)
Python Example	<pre>import CoreGlobalScriptContextFunctions CoreGlobalScriptContextFunctions.AddWarningMessage('My warning.')</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	<table border="1"> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> <tr> <td><message></td> <td>String</td> <td>Debug message.</td> </tr> </table>			Name	Type	Description	<message>	String	Debug message.
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 <error>	Type String	Description 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.

25 - Example Scripts

This section contains [VBScript](#) and [IronPython](#) example scripts.

VBScript Example Scripts

VBScript Examples:

- [Variable Helix Script](#)
- [HFSS Data Export Script](#)
- [ExampleGetLayeredImpedanceBoundaryPropertyNamesandValues.htm](#)

Data Export Script

The following sample script demonstrates how to export data and save it to a file. The output data in the example script is in 3 columns. The first column is frequency in GHz, the second is the Real part of S11, and the third is the Img part of S11. It uses a tab-delimited format. The output is done using output variables.

The frequency sweep data must be entered correctly. If it is incorrect, the script will request a freq point that does not exist and execution will stop.

The script includes comment lines, which are preceded by an apostrophe (') and offer explanations for each subsequent line or lines.

```
Dim oAnsoftApp
Dim oDesktop
Dim oProject
Dim oDesign
Dim oEditor
Dim oModule
Set oAnsoftApp = CreateObject("Ansoft.ElectronicsDesktop")
Set oDesktop = oAnsoftApp.GetAppDesktop()
set oProject = oDesktop.GetActiveProject
set oDesign = oProject.GetActiveDesign()
Dim oFS,ofile,x,y,z,path,range,
Dim arr2,del_f,freq,cfreq,val,temp,stn,stw,i,line
'
'
' Input the desired file name.
'
```

```
path = inputbox("Input the file name" & chr(13) & _
"Note: If you do not specify a path the file will " & _
"be placed in the script directory", _
"File","C:\hfss_export.txt",50,50)

'

' If the user clicks Cancel the path will be blank, in which case
the script should just exit.

If path <>"" then

'

' Create the file, open it for data entry, and output the column
labels.

'

Set oFS = CreateObject("Scripting.FileSystemObject")
Set ofile = oFS.CreateTextFile (path)
line = "Freq" & chr(9) & "RE(S11)" & chr(9) & "IMG(S11)"
ofile.WriteLine line

'

' Input the needed freq, solution, and sweep data and clean it
up.

'

msgbox("For the following input make sure it matches " & _
"the frequencies defined in your sweep")
range = inputbox("Input the range of frequencies in GHz " & _
"and number of points",_
"Frequency","8,12,10",50,50)

'

' The following 2 lines define the 2 output variables.

'

oDesign.CreateOutputVariable "re_S", "re(S(port1,port1))"
oDesign.CreateOutputVariable "im_S", "im(S(port1,port1))"
arr = split (range, ",")
```

```
arr(0) = Trim(arr(0))
arr(1) = Trim(arr(1))
arr(2) = Trim(arr(2))

if cint(arr(2)) <> 1 then
    del_f = (arr(1)-arr(0))/(arr(2)-1)
else
    del_f = 0
end if

temp = InputBox("Input the Setup and Sweep number to use:_" & chr(13) & "(e.g. input 1,2 for Setup1 and Sweep2)", _ "Solution Data", "1,1", 50, 50)

arr2 = split(temp, ",")  

stn = arr2(0)
swn = arr2(1)
stn = Trim(stn)
swn = Trim(swn)

'  

' Loop through the freq points.  

'  

for i=1 to arr(2) step 1
freq = arr(0) + (cint(i)-1)*del_f
x=freq
cfreq="Freq=""" & freq & "Ghz""'  

'  

' Get the values of the output variables for the desired freq.  

'  

val = oDesign.GetOutputVariableValue("re_S", "Setup" & stn & " :Sweep" & swn, cfreq, "")  

y = val
val = oDesign.GetOutputVariableValue("im_S", "Setup" &
```

```
stn & " : Sweep" & swn,cfreq, "")  
z = val  
  
'  
  
' Create the line of text to send to the file and write it to the  
file.  
  
'  
  
line = x & chr(9) & y & chr(9) &z  
ofile.WriteLine line  
  
Next  
  
'  
  
' Delete the 2 output variables before finishing.  
  
'  
  
oDesign.DeleteOutputVariable "re_S"  
oDesign.DeleteOutputVariable "im_S"  
  
'  
  
' Close the file.  
  
'  
  
ofile.close  
End if
```

Variable Helix Script

This sample script creates a tapered helix. Tapering helices is not supported from the interface.

The script includes comment lines, which are preceded by an apostrophe (') and offer explanations for each subsequent line or lines.

```
Dim oAnsoftApp  
Dim oDesktop  
Dim oProject  
Dim oDesign  
Dim oEditor  
Dim oModule  
  
Set oAnsoftApp = CreateObject("Ansoft.ElectronicsDesktop")
```

```
Set oDesktop = oAnsoftApp.GetAppDesktop()
Set oProject = oDesktop.GetActiveProject()
Set oDesign = oProject.GetActiveDesign()
Set oEditor = oDesign.SetActiveEditor("3D Modeler")

' Declare the arrays and variables needed for building the polyline.
'

Dim points(), segments()

Dim NumPoints, R(2), P(2), PointsPerTurn, Turns, Units
'

' Establish the constant Pi.

Pi = 4*Atn(1)

' Retrieve the variable helix parameters from the user.

' Start with the input for unit selection.

'

Units = InputBox("Select the units:"&Chr(13)&_
"(cm,mm,um,in,mil)", "Variable Helix","mil",50,50)
'

' Check to make sure it is a valid unit.

'

Select Case Units
Case "m"
    Units = ""
Case "cm"
Case "mm"
Case "um"
Case "in"
Case "mil"
Case Else
    MsgBox("Invalid Units - defaults to m")
```

```
Units = ""

End Select
,
' Obtain the other user-defined parameters.
,
Turns = InputBox("Select the number of turns (must be
integer):", "Variable Helix", 2,50,50)
PointsPerTurn = InputBox("Select the points per turn:", _
"Variable Helix",16,50,50)
R(0) = InputBox("Select the initial Radius: ", _
"Variable Helix",10,50,50)
R(1) = InputBox("Select the final Radius: ", _
"Variable Helix",10,50,50)
P(0) = InputBox("Select the initial Pitch: ", _
"Variable Helix", 4,50,50)
P(1) = InputBox("Select the final Pitch: ", _
"Variable Helix", 4,50,50)
NumPoints = Turns*PointsPerTurn
,
' Initialize the points and segments arrays.
,
Redim points(NumPoints+1)
Redim segments(NumPoints)
points(0) = "NAME:PolylinePoints"
segments(0) = "NAME:PolylineSegments"
,
' Build the Point and Segment Arrays needed in the HFSS polyline call.
,
For n = 1 To (NumPoints+1)
    Angle = (n-1)*2*Pi/PointsPerTurn
```

```

Radius = R(0) + ((n-1)/NumPoints)*(R(1)-R(0))
Pitch = P(0) + ((n-1)/NumPoints)*(P(1)-P(0))
Rise = (n-1)*Pitch/PointsPerTurn

XValue = cstr(Radius*cos(Angle)) & Units
YValue = cstr(Radius*sin(Angle)) & Units
ZValue = cstr(Rise) & Units
points(n) = Array("NAME:PLPoint", "X:=", XValue, "Y:=", _
YValue, "Z:=", ZValue)
,
' Create the line segments between each of the pairs of points.
,
If n<=NumPoints Then
segments(n) = Array("NAME:PLSegment", "SegmentType:=", _
"Line", "StartIndex:=", (n-1), "NoOfPoints:=", 2)
End If
Next
,
' Create the polyline.
,
oEditor.CreatePolyline _
Array("NAME:PolylineParameters", "IsPolylineCovered:=", true, _
"IsPolylineClosed:=", false, points, segments), _
Array("NAME:Attributes", "Name:=", "Line_Helix", "Flags:=", _
"", "Color:=", "(132 132 193)", "Transparency:=", 0.4, _
"PartCoordinateSystem:=", "Global", "MaterialName:=", _
"vacuum", "SolveInside:=", true)
,
' Create the helix cross-section.
,
```

```
oEditor.CreateCircle _  
    Array("NAME:CircleParameters", "IsCovered:=", true, "XCenter:=", _  
        cstr(R(0))&Units, "YCenter:=", 0, "ZCenter:=", 0, "Radius:=", _  
        "1"&Units, "WhichAxis:=", "Y"), _  
    Array("NAME:Attributes", "Name:=", "Circle_Helix", "Flags:=", _  
        "", "Color:=", "(132 132 193)", "Transparency:=", 0.4, _  
        "PartCoordinateSystem:=", "Global", "MaterialName:=", "vacuum", _  
        "SolveInside:=", true)  
,  
' Sweep the cross-section along the path.  
,  
  
oEditor.SweepAlongPath _  
    Array("NAME:Selections", "Selections:=", _  
        "Circle_Helix,Line_Helix"),  
    Array("NAME:PathSweepParameters", "DraftAngle:=", "0deg", _  
        "DraftType:=", "Round", "TwistAngle:=", "0deg")
```

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",
```

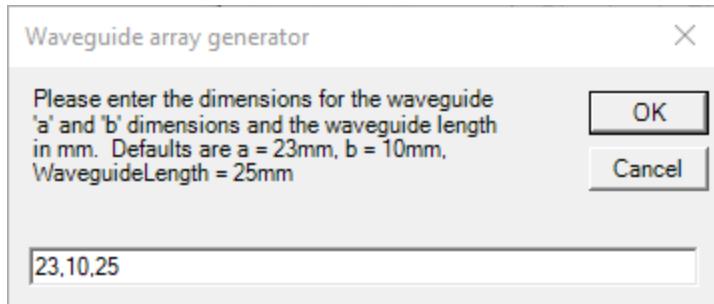
```
"XSectionNumSegments:=" , "0",
"XSectionBendType:=" , "Corner"
]
],
[
"NAME:Attributes",
"Name:=" , "Polyline"+str(n),
"Flags:=" , "",
"Color:=" , "(132 132 193)",
"Transparency:=" , 0,
"PartCoordinateSystem:=", "Global",
"UDMId:=" , "",
"MaterialValue:=" , "\"vacuum\"",
"SurfaceMaterialValue:=" , "\"\"",
"SolveInside:=" , True,
"IsMaterialEditable:=" , True,
"UseMaterialAppearance:=" , False,
"IsLightweight:=" , False
])
)
```

HFSS Waveguide Array Python Script

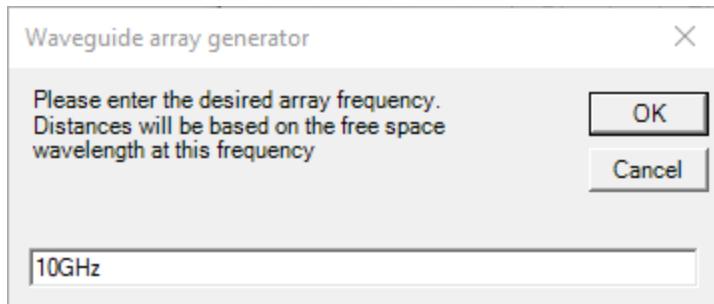
This sample Python script creates an HFSS Waveguide array. The script includes comment lines, which are preceded by an apostrophe ('), that offer explanations for each subsequent line or lines. The script includes examples of creating a 3D model, boundaries and excitation, solution setup and sweep, as 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.

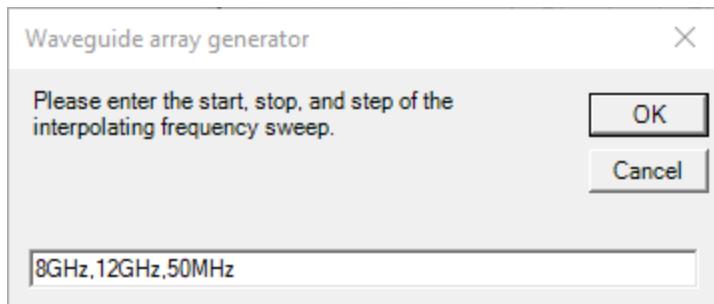
The first dialog asks for input for a and b dimensions and the waveguide length.



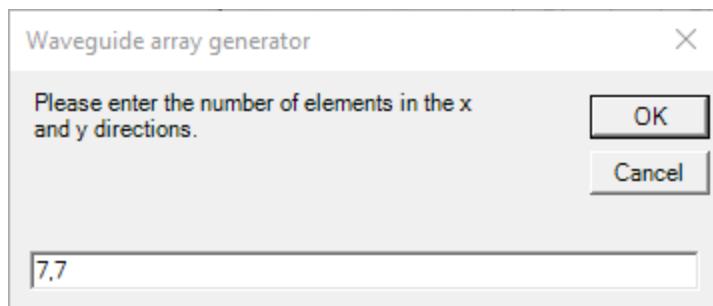
The next asks for the array frequency.



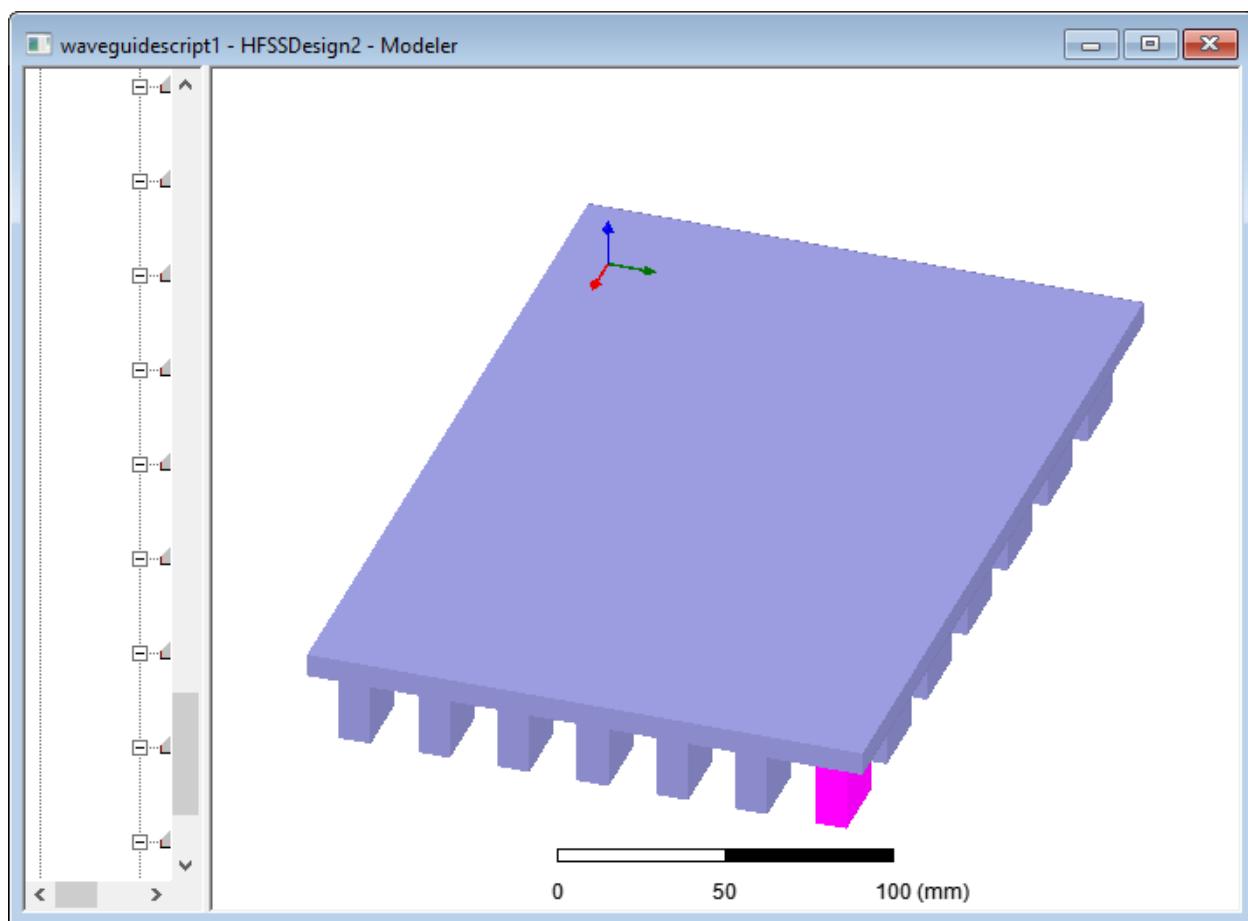
The next asks for start, stop and step values for in an interpolating frequency sweep.



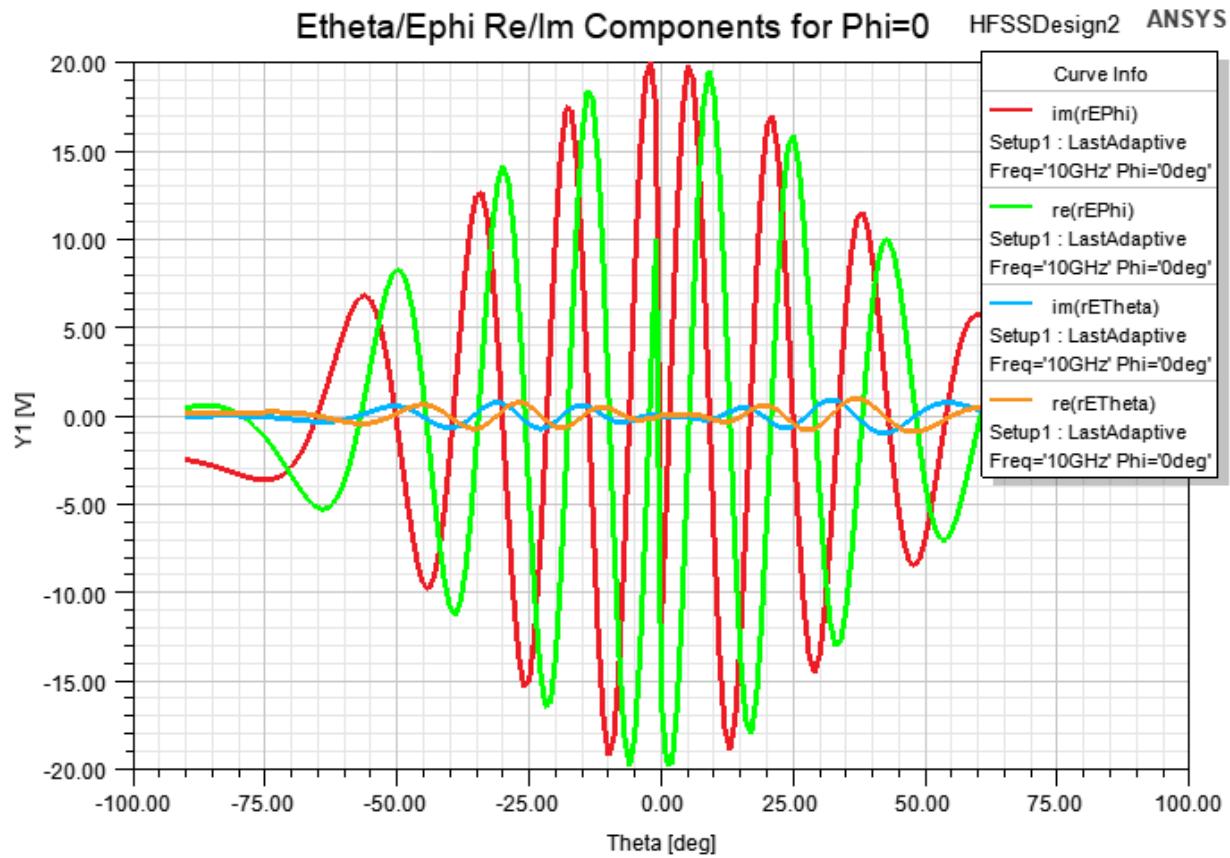
The last one asks for the array definition.

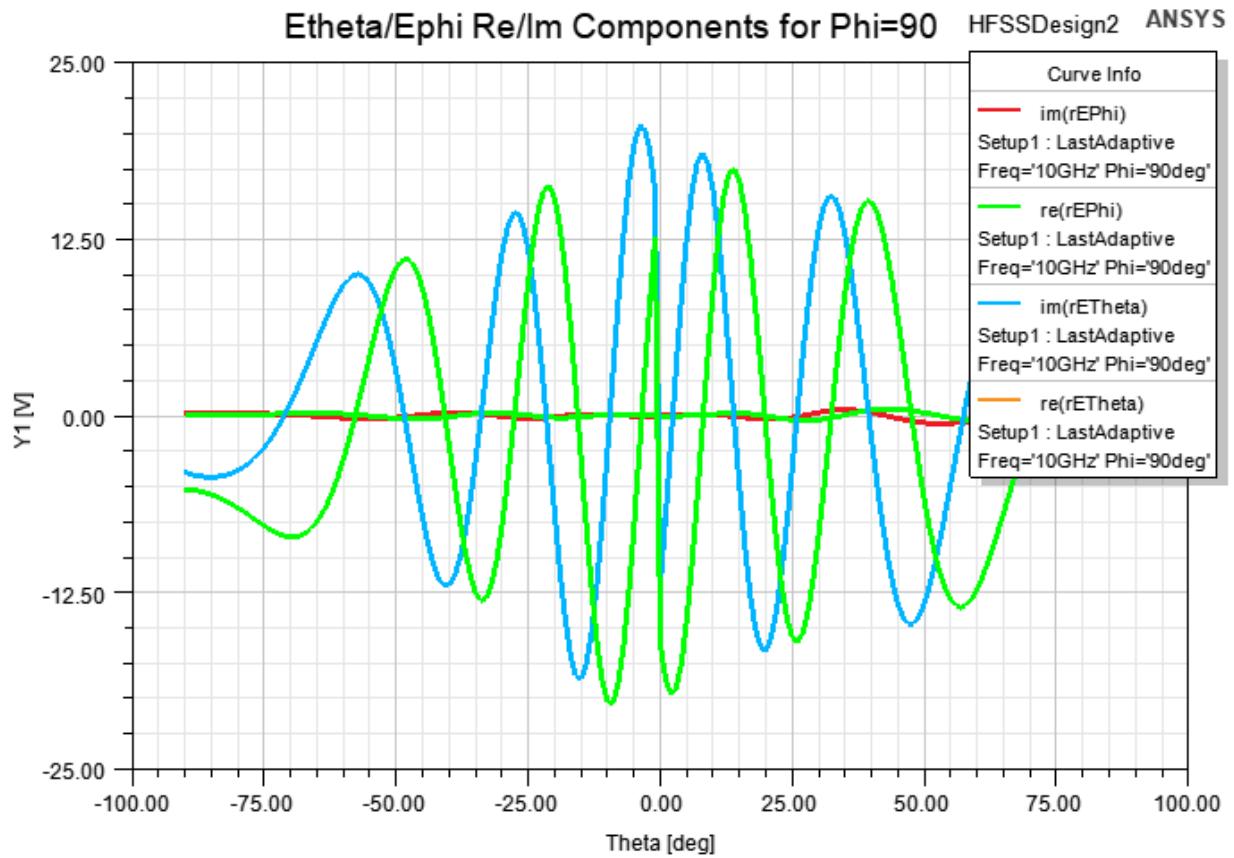


The following figure shows the waveguide array generated by the defaults.



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:Mode1",
        "ModeNum:=", 1,
        "UseIntLine:=", True,
        [
            "NAME:IntLine",

```

```

"Start:=" , ["0mm", "-" +str(b_over2) + "mm", "-" + Wave-
guideLength_str],
"End:=" , ["0mm", str(b_over2) + "mm", "-" + WaveguideLength_str]
],
"AlignmentGroup:=" , 0,
"CharImp:=" , "Zpi"
],
],
"ShowReporterFilter:=" , False,
"ReporterFilter:=" , [True],
"UseAnalyticAlignment:=" , False
])

#Set the radiation boundary
#-----
# Get face IDs for further assignment
top_face_id = oEditor.GetFaceByPosition([ "NAME:FaceParameters",
    "BodyName:=", "RadiationBox",
    "XPosition:=", "0mm",
    "YPosition:=", "NumY*b-b/2+(NumY-1)*Lambda/2+RadBoundDist",
    "ZPosition:=", "0mm"])

faceIDs = [int(elem) for elem in oEditor.GetFaceIDs("RadiationBox")]
if elem != str(top_face_id)]

oModule.AssignRadiation(
[
    "NAME:Radiation",
    "Faces:=" , faceIDs,
    "IsFssReference:=" , False,
    "IsForPML:=" , False
])

```

```
# Copy and paste elements/ports into a rectangular array.

# Duplicate boundaries with geometry" must be turned on under Tools-
>Options->HFSS Options

# -----
-----

ElementNum = 1

for i in range(1, int(numX)+1):

    for j in range(1, int(numY)+1):

        if ElementNum == 1:

            pass

        elif ElementNum <= numY: #If in the first column, only

            oEditor.Copy(["NAME:Selections", "Selections:=", "Element1"])

            oEditor.Paste()

            oEditor.Move(["NAME:Selections", "Selections:=", "Element" + str(ElementNum)],

                        ["NAME:TranslateParameters", "CoordinateSystemID:=", -1,
                         "TranslateVectorX:=", "0mm",
                         "TranslateVectorY:=", str(j-1) + "* (b+Lambda/2)",
                         "TranslateVectorZ:=", "0mm"])

        elif ElementNum > numY:

            oEditor.Copy(["NAME:Selections", "Selections:=", "Element1"])

            oEditor.Paste()

            oEditor.Move(["NAME:Selections", "Selections:=", "Element" + str(ElementNum)],

                        ["NAME:TranslateParameters", "CoordinateSystemID:=", -1,
                         "TranslateVectorX:=", str(i-1) + "* (a+Lambda/2)",
                         "TranslateVectorY:=", str(j-1) + "* (b+Lambda/2)",
                         "TranslateVectorZ:=", "0mm"])
```

```
ElementNum += 1

#Create the setup and interpolating sweep
#-----
oModule = oDesign.GetModule("AnalysisSetup")
oModule.InsertSetup("HfssDriven",
[
    "NAME:Setup1",
    "AdaptMultipleFreqs:=" , False,
    "Frequency:=" , Frequency,
    "MaxDeltaS:=" , 0.02,
    "PortsOnly:=" , False,
    "UseMatrixConv:=" , False,
    "MaximumPasses:=" , 15,
    "MinimumPasses:=" , 1,
    "MinimumConvergedPasses:=" , 1,
    "PercentRefinement:=" , 50,
    "IsEnabled:=" , True,
    "BasisOrder:=" , 1,
    "DoLambdaRefine:=" , True,
    "DoMaterialLambda:=" , True,
    "SetLambdaTarget:=" , False,
    "Target:=" , 0.3333,
    "UseMaxTetIncrease:=" , False,
    "PortAccuracy:=" , 2,
    "UseABCOnPort:=" , False,
    "SetPortMinMaxTri:=" , False,
    "UseDomains:=" , False,
    "UseIterativeSolver:=" , False,
    "SaveRadFieldsOnly:=" , False,
```

```
"SaveAnyFields:=" , True,
"IESolverType:=" , "Auto",
"LambdaTargetForIESolver:=" , 0.15,
"UseDefaultLambdaTgtForIESolver:=" , True
])

oModule.InsertFrequencySweep("Setup1",
[
    "NAME:InterpolatingSweep",
    "IsEnabled:=" , True,
    "RangeType:=" , "LinearStep",
    "RangeStart:=" , StartFrequency,
    "RangeEnd:=" , StopFrequency,
    "RangeStep:=" , StepFrequency,
    "Type:=" , "Interpolating",
    "SaveFields:=" , False,
    "InterpTolerance:=" , 0.5,
    "InterpMaxSolns:=" , 50,
    "InterpMinSolns:=" , 0,
    "InterpMinSubranges:=" , 1,
    "ExtrapToDC:=" , False,
    "InterpUseS:=" , True,
    "InterpUsePortImped:=" , False,
    "InterpUsePropConst:=" , True,
    "UseDerivativeConvergence:=" , False,
    "InterpDerivTolerance:=" , 0.2,
    "UseFullBasis:=" , True,
    "EnforcePassivity:=" , True,
    "PassivityErrorTolerance:=" , 0.0001
])
```

```

#Create a relative coordinate system centered on the array for radi-
ation pattern calculations

#-----
-----

oEditor.CreateRelativeCS (
[
    "NAME:RelativeCSParameters",
    "Mode:=" , "Axis/Position",
    "OriginX:=" , "-a/2+NumX*a/2+(NumX-1)*Lambda/4",
    "OriginY:=" , "-b/2+NumY*b/2+(NumY-1)*Lambda/4",
    "OriginZ:=" , "0mm",
    "XAxisXvec:=" , "1mm",
    "XAxisYvec:=" , "0mm",
    "XAxisZvec:=" , "0mm",
    "YAxisXvec:=" , "0mm",
    "YAxisYvec:=" , "1mm",
    "YAxisZvec:=" , "0mm"
],
[
    "NAME:Attributes",
    "Name:=" , "RelativeCS1"
])

#Create an infinite sphere with fine theta resolution and phi cuts at
0 and 90 degrees

#-----
-----

oModule = oDesign.GetModule("RadField")
oModule.InsertFarFieldSphereSetup (
[
    "NAME:Infinite Sphere1",
    "UseCustomRadiationSurface:=", False,
]
)

```

```
"ThetaStart:=" , "-90deg",
"ThetaStop:=" , "90deg",
"ThetaStep:=" , "1deg",
"PhiStart:=" , "0deg",
"PhiStop:=" , "90deg",
"PhiStep:=" , "90deg",
"UseLocalCS:=" , True,
"CoordSystem:=" , "RelativeCS1"

])

#Create output plots for Ephi/Etheta real and imaginary components
for Phi = 0 and separately for Phi = 90
-----
oModule = oDesign.GetModule("ReportSetup")

#For phi = 0
oModule.CreateReport("Etheta/Ephi Re/Im Components for Phi=0", "Far
Fields",
"Rectangular Plot", "Setup1 : LastAdaptive",
[
"Context:=" , "Infinite Sphere1"
],
[
"Theta:=" , ["All"],
"Phi:=" , ["0deg"],
"Freq:=" , ["All"],
"a:=" , ["Nominal"],
"b:=" , ["Nominal"],
"NumX:=" , ["Nominal"],
"NumY:=" , ["Nominal"],
"WaveguideLength:=" , ["Nominal"],
```

```
"Frequency:=" , ["Nominal"]
],
[
"X Component:=" , "Theta",
"Y Component:=" , ["im(rEPhi)","re(rEPhi)","im(rETheta)","re
(rETheta)"]
], [])

#For phi = 90
#-----
oModule.CreateReport("Etheta/Ephi Re/Im Components for Phi=90", "Far
Fields",
"Rectangular Plot", "Setup1 : LastAdaptive",
[
"context:=" , "Infinite Sphere1"
],
[
"Theta:=" , ["All"],
"Phi:=" , ["90deg"],
"Freq:=" , ["All"],
"a:=" , ["Nominal"],
"b:=" , ["Nominal"],
"NumX:=" , ["Nominal"],
"NumY:=" , ["Nominal"],
"WaveguideLength:=" , ["Nominal"],
"Frequency:=" , ["Nominal"]
],
[
"X Component:=" , "Theta",
"Y Component:=" , ["im(rEPhi)","re(rEPhi)","im(rETheta)","re
(rETheta)"]]
```

], [])

Index

3

3D Modeler editor commands 10-1
 AlignFaces 10-116
 AssignMaterial 10-117
 AssignSurfaceMaterial 10-253
 Chamfer 10-120
 CleanUpModel 10-122
 CloseAllWindows 10-257
 Connect 10-123
 Copy 10-95
 CoverLines 10-124
 CoverSurfaces 10-125
 CreateBondwire 10-6
 CreateBox 10-9
 CreateCircle 10-12
 CreateCone 10-15
 CreateCutplane 10-18
 CreateCylinder 10-19
 CreateEllipse 10-22
 CreateEntityList 10-126
 CreateEquationCurve 10-25
 CreateEquationSurface 10-28
 CreateFaceCS 10-127, 10-133
 CreateGroup 10-132
 CreateHelix 10-31

CreateObjectFromedges 10-140
 CreateObjectFromFace 10-142
 CreateObjectFromFaces 10-144
 CreatePoint 10-34
 CreatePolyline 10-35
 CreateRectangle 10-41
 CreateRegularPolygon 10-43
 CreateRegularPolyhedron 10-46
 CreateRelativeCS 10-145
 CreateSpiral 10-49
 CreateTorus 10-52
 CreateUserDefinedModel 10-54
 CreateUserDefinedPart 10-57
 Defeature 10-258
 Delete 10-259
 DeleteEmptyGroups 10-147
 DeleteLastOperation 10-148
 DeleteOperation 10-149
 DeletePolylinePoint 10-96
 DetachEdges 10-151
 DetachFaces 10-152
 DuplicateAlongLine 10-97
 DuplicateAroundAxi 10-99
 DuplicateMirror 10-102
 EditEntityList 10-154

EditFaceCS 10-155	GetEdgePositionAtNormalizedParameter 9-45
EditObjectCS 10-161	GetEntityListIDByName 10-278
EditPolyline 10-69	GetFaceArea 10-280
EditRelativeCS 10-168	GetFaceByPosition 10-282
Export 10-170	GetFaceCenter 10-281
ExportModelImageToFile 10- 172, 12-60	GetFaceIDFromNameForFirstOperation 10-283
Fillet 10-176	GetFaceIDs 10-284
FlattenGroup 10-178	GetFaceIDsOfSheet 10-285
GenerateAllUserDefinedModels 10-260	GetGeometryModelerMode 10-285
GenerateHistory 10-179	GetGroupSubmodelNames 10-286
GenerateUserDefinedModel 10- 261	GetModelBoundingBox 10-287
Get3DComponentParameters 10-76	GetPartsForUserDefinedModel 10-297
Get3DComponentPartNames 10-77	GetPoints 10-298
GetActiveCoordinateSystem 10- 180	GetProperties 5-42, 7-17, 9-53
GetAct- iveCoordi- nateSystemTransform 10-180	GetPropertyValue 5-44, 7-18, 9-54, 10- 299, 12-85, 23-102
GetBodyNamesByPosition 10- 264	GetPropEvaluatedValue 10-298
GetCoordinateSystems 10-181	GetPropSIValue 10-302
GetEdgeByPosition 10-274	GetRelativeCoordinateSystems 10-304
GetEdgeIDFromNameForFirstOper- ation 10-276	GetUser Position 10-306
GetEdgeIDsfromFace 10-276	GetVertexIDFromNameForFirstOperation 10-307
GetEdgeIDsfromObject 10-277	GetVertexIDsFromEdge 10-308
GetEdgeLength 10-278	GetVertexIDsFromFace 10-308
	GetVertexIDsFromObject 10-309
	GetVertexPosition 10-310
	GetWireBodyNames 10-310
	HealObject 10-182

Import 10-186
ImportDXF 10-190
ImportFromClipboard 10-194
ImportGDSII 10-195
Imprint 10-198
ImprintProjection 10-199
Insert3DComponent 10-77
InsertComponent 10-78
Intersect 10-201
Mirror 10-105
Move 10-107
MoveCSToEnd 10-203
MoveEntityToGroup 10-204
MoveFaces 10-205
OffsetFaces 10-109
PageSetup 10-312
ProjectSheet 10-207
RemoveBadEdges 10-314
RemoveBadFaces 10-314
RemoveBadVertices 10-315
RenamePart 10-316
Replacewith3DComponent 10-209
Rotate 10-111
Scale 10-112
Section 10-212
SeparateBody 10-214
SetModelUnits 10-215
SetPropValue Modeler 10-317

SetTopDownViewDirectionForActiveView 10-318
SetTopDownViewDirectionForAllViews 10-319
SetWCS 10-216
Simplify 10-218
Split 10-221
Stitch 10-223
Subtract 10-225
SweepAlongPath 10-82
SweepAlongVector 10-84
SweepAroundAxis 10-86
SweepFacesAlongNormal 10-88, 10-88, 10-226, 10-226
Sweep-FacesAlongNormalWithAttributes 10-90
ThickenSheet 10-228
UncoverFaces 10-230
Ungroup 10-232
Unite 10-231
UpdateComponentDefinition 10-93
UpdatePriorityList 10-319
UpgradeVersion 10-320
Validate3DComponent 10-321
WrapSheet 10-233
WriteHistoryTreeLayoutForTest 10-322

3D Modeler editor object commands
GetNumObjects 10-289

GetObjectIDByName 10-289	AddDefinitionFromLibFile 10-242
GetObjectName 10-290	AddDeltaMarker 12-12
GetObject NameByEdgeID 10-291	Additional Property Scripting Example 7-25
GetObject NameByFaceID 10-291	AddMarker 12-13
GetObject NameByID 10-292	AddMarker[Fields Reporter] 18-1
GetObject NameByVertexID 10-293	AddMarkerToPlot 18-2
GetObjectShapeType 10-294	AddMaterial 5-6, 23-28
GetObjectVolume 10-296	AddMessage 3-6
GetObjPath 10-296	AddModelingProperties 9-7
	AddNamedExpr 18-4, 19-3
	AddNote 12-14
	AddTraceCharacteristics 12-16
	AddTraces 12-18
	AlignFaces 10-116
	Analysis module commands
	AnalyzeAll 9-7
	AnalyzeAllNominal 9-8
	CopySetup 15-2, 16-6, 16-44
	DeleteSetups 15-3
	EditSetup 15-4, 16-8, 16-46
	GetSetupCount 15-16
	GetSetups 15-17
	PasteSetup 15-39
	RenameSetup 15-39
	RevertAllToInitial 15-40
	RevertSetupToInitial 15-41
	Analysis Setup Module Script Commands 15-1

Ansoft Application Object commands	Boundary/Excitation module commands
GetAppDesktop 2-2	AddAssignmentToBoundary 13-2
ApplyReportTemplate 12-20	DeleteAllBoundaries 13-3
arithmetic operators 1-6	DeleteBoundaries 13-3
array variables 1-4	GetBoundaries 13-4
AssignAdiabaticPlateBoundary 14-8	GetBoundariesOfType 13-4
AssignBlockBoundary 14-1	GetBoundaryAssignment 13-5
AssignBlowerBoundary_1 14-22	GetDefaultBaseName 13-5
AssignConductingPlateBoundary 14-10	GetNumBoundaries 13-6
AssignEMLoss 14-21	GetNumBoundariesOfType 13-6
AssignGrilleBoundary 14-3	ReassignBoundaries 13-2, 13-7
AssignMaterial 10-117	ReassignBoundary 13-7
AssignNetworkBoundary 14-5	RemoveAssignmentFromBoundary 13-7
AssignOpeningBoundary 14-6	RenameBoundary 13-8
AssignRecircBoundary 14-7	ReprioritizeBoundaries 13-9
AssignResistanceBoundary 14-13	SetDefaultBaseName 13-10
AssignSourceBoundary 14-17	BreakUDMConnection 10-256
AssignStationaryWallBoundary 14-18	BringToFront 23-170
AssignSurfaceMaterial 10-253	C
AssignSymmetryWallBoundary 14-20	Cable Modeling commands
AssignVirtualMeshRegion 14-27	AddCableToBundle 10-325
B	CreateCableBundle 10-326
Boundary and Excitation Module Script Commands 13-1	CreateCableHarness 10-328
boundary conditions script commands 13-1	CreateClockSource 10-330
	CreatePWLSource 10-332
	CreateStraightWireCable 10-334, 10-335

ExportCableLibrary	10-337	comparison operators	1-7
ImportCableLibrary	10-338	conditional statements	
RemoveCables	10-339	If...Then... Else	1-8
UpdateCableHarness	10-339	Select Case	1-8
Cable Modeling Commands	10-324	types of	1-8
CalcOp	18-5, 19-4	Connect	10-123
CalcRead (deprecated)	19-4	ConstructVariationString	9-8
CalcStack	18-6, 19-6	conventions	
CalculatorRead	18-7, 19-5	scripting help	1-1
CalculatorWrite	18-8, 19-7	converting data types	1-10
CalcWrite	18-6, 19-8	ConvertToDynamic	23-134
Chamfer	10-120	ConvertToParametric	23-135
ChangeGeomSettings	18-9, 19-9	Copy	10-95
ClcEval	18-10, 19-10	CopyDesign	5-12
ClcMaterial	18-11, 19-11	CopyItemCommand	9-9
ClcMaterialValue	18-12, 19-11	CopyNamedExprToStack	18-14, 19-13
CleanUpModel	10-122	CopyPlotSettings	12-23
ClearAllMarkers	12-21	CopyReportDefinition	12-24
ClearAllMarkers[Fields Reporter]	18-13	CopyReportsData	12-25
ClearAllNamedExpr	18-13, 19-12	Copyright and Trademark Information	2
ClearAllTraceCharacteristics	12-22	CopySetup	
ClearMessages	3-7, 5-11	Analysis module command	15-2, 16-6, 16-44
CloneReportsFromDatasetSolution	12-22	Optimetrics module command	15-2, 16- 6, 16-44
Close	5-12	CopyTraceDefinitions	12-25
CloseAllWindows	3-8, 10-257	CopyTracesData	12-26
CloseProject	3-8	Core Global Script Context Commands	24-1
CloseProjectNoForce	3-9	Count	3-10
comment lines	1-12, 1-14		

CoverLines 10-124	CreateRegularPolyhedron 10-46
CoverSurfaces 10-125	CreateRelativeCS 10-145
CPython 1-72	CreateReport 9-10, 12-27
CreateBondwire 10-6	CreateReport [Designer] 12-42
CreateBox 10-9	CreateReportFromTemplate 12-47
CreateCableBundle 10-326	CreateReportofAllQuantities 12-48
CreateCableHarness 10-328	CreateSpiral 10-49
CreateCircle 10-12	CreateTorus 10-52
CreateClockSource 10-330	CreateUserDefinedModel 10-54
CreateCone 10-15	CreateUserDefinedPart 10-57
CreateCutplane 10-18	CreateUserDefinedSolution 22-1
CreateCylinder 10-19	CutDesign 5-13
CreateEllipse 10-22	
CreateEntityList 10-126	D
CreateEquationCurve 10-25	dataset commands
CreateEquationSurface 10-28	AddDataset 5-3, 8-1, 9-4, 23-20
CreateFaceCS 10-127, 10-133	DeleteDataset 5-14, 8-4, 9-25, 23-50
CreateFieldPlot 18-15	EditDataset 5-16, 8-5, 23-77
CreateGroup 10-132	ExportDataset 5-28, 8-7, 9-36, 23-96
CreateHelix 10-31	HasDataset 8-8
CreateObjectFromEdges 10-140	ImportDataset 5-48, 8-9, 9-62, 23-104
CreateObjectFromFace 10-142	Dataset Script Commands 8-1
CreateObjectFromFaces 10-144	Defeature 10-258
CreateOutputVariable 11-1	Definition Manager commands
CreatePoint 10-34	AddDefinitionFromBlock 10-237, 23-23
CreatePolyline 10-35	AddDefinitionFromLibFile 10-242
CreatePWLSource 10-332	
CreateRectangle 10-41	
CreateRegularPolygon 10-43	

GetExtendedDefinitionObject 10-279
GetProjectMaterialNames 23-101
UpdateDefFromBlock 23-116
Definition Manager Script Commands 23-1
Delete 10-259
DeleteAllBoundaries 13-3
DeleteAllReports 12-51
DeleteBoundaries 13-3
DeleteDataset 5-14, 8-4, 9-25, 23-50
DeleteDesign 5-15
DeleteEmptyGroups 10-147
DeleteFieldPlot 18-46
DeleteFieldVariation 9-25
DeleteFullVariation 9-26
DeleteLastOperation 10-148
DeleteLinkedDataVariation 9-27
DeleteMarker 12-50
DeleteMarker[Fields Reporter] 18-47
DeleteNamedExpr 18-47, 19-13
DeleteOp 14-30
DeleteOperation 10-149
DeleteOutputVariable 9-28, 9-30, 11-2
DeletePolylinePoint 10-96
DeleteProject 3-11
DeleteReport 12-51
DeleteSetups 15-3
Analysis module command 15-3
Optimetrics module command 16-7, 16-45
DeleteSolutionVariation 9-29, 17-1
DeleteTraceCharacteristics 12-52
DeleteTraces 12-53
DeleteUneditablePlot 18-48
DeleteUserDefinedSolutions 22-3
Design object commands 12-72, 12-73, 12-75, 12-75, 12-76, 12-91
AddCartesianLimitLine 12-4
AddCartesianXMarker 12-10
AddCartesianYMarker 12-10
AddDeltaMarker 12-12
AddMarker 12-13
AddMarkerToPlot 18-2
AddNote 12-14
AddTraceCharacteristics 12-16
AddTraces 12-18
CalculatorRead 18-7, 19-5
ClearAllMarkers 12-21
ClearAllTraceCharacteristics 12-22
CloseAllWindows 3-8
CopyReportDefinition 12-24
CreateOutputVariable 11-1
CreateReportFromTemplate 12-47
DeleteAllReports 12-51
DeleteOutputVariable 9-28, 9-30, 11-2

DeleteReport 12-51
DeleteTraces 12-53
DoesOutputVariableExist 11-3
EditDesignSettings 9-31
EditMarker 12-57
EditOutputVariable 11-4
ExportConvergence 9-35
ExportModelMeshToFile 10-175,
12-63
ExportPlot3DToFile 12-63
ExportPlotImageToFile 18-68
ExportPlotImageWithViewToFile
18-70
ExportProfile 9-38
ExportReport 12-65
ExportToFile 12-69
GeometryCheckAndAutofix 10-
262
GetDisplayType 12-80
GetLibraryDirectory 3-25
GetMatchedObjectName 10-287
GetModelUnits 10-288
GetModule 9-47
GetName 9-48, 12-83, 16-32,
16-70
GetObjectsByMaterial 10-293
GetObjectsInGroup 10-295
GetObjPath 9-50, 12-84, 16-32,
16-70
GetOutputVariableValue 9-51,
11-7

GetProjectDirectory 3-31
GetProperties 5-42, 7-17, 9-53
GetPropertyValue 5-44, 7-18, 9-
54, 10-299, 12-85, 23-102
GetPropEvaluatedValue 10-298
GetPropNames 9-56
GetPropSIValue 10-302
GetRegistryInt 3-33, 3-73
GetRegistryString 3-34, 3-74
GetSelections 10-304
GetSolutionType 9-58
GetSolveInsideThreshold 9-59
GetSubGroupsInGroup 10-305
GetTempDirectory 3-38
GetVariables 5-47, 7-20, 9-59
GetVariableValue 5-46, 7-20, 9-60
GetVariationVariableValue 9-61
GetVersion 3-39
HasDataset 8-8
ImportIntoReport 12-108
IsFeaturEnabled 3-40
PasteDesign 9-65
PasteReports 12-113
PasteReportsWithLegacyNames
12-114
PasteTracesWithLegacyNames
12-115
Redo 9-66
RenameDesignInstance 9-67
RenameReport 12-116

RenameTraces	12-117	Count	3-10
RunToolkit	9-68	DeleteProject	3-11
SARSetup	9-69	DownloadJobResults	3-12
SavePlotSettingsAsDefault	12-118	EnableAutosave	3-15
SetActiveEditor	9-70	ExportOptionsFiles	3-16
SetAllowMaterialOverride	9-70	GetActiveProject	3-16
SetLibraryDirectory	3-62	GetAutosaveEnabled	3-17
SetProjectDirectoryVBCommand>	3-62	GetBuildTimeDateString	3-18
SetPropertyValue	5-65, 7-21, 9-74, 23-115	GetChildNames	5-31, 9-39
SetSolutionType	9-75	GetChildNames Modeler	10-265
SetSolveInsideThreshold	9-77	GetChildTypes	5-33, 9-41, 10-272
SetTempDirectory	3-66	GetChildTypes Optimetrics	16-31, 16-69
SetVariableValue	5-67, 7-23, 9-79	GetCustomMenuSet	3-19
ShowWindow	10-217	GetDesigns	5-37
Solve	9-80, 9-81	GetDesktopConfiguration	3-22
StopSimLink	9-81	GetDistributedAnalysisMachines	3-22
UpdateAllReports	12-122	GetDis-	
ValidateDesign	9-82	trib-	
Design Object Script Commands	9-1	utedAna-	
Desktop Commands For Registry		lysisMachinesForDesignType	3-23
Values	3-71	GetPPELicensingEnabled	3-29
Desktop object commands		GetProjectList	3-32
AddMessage	3-6	GetProjects	3-28, 3-32
ClearMessages	3-7, 5-11	GetPropNames	5-41, 16-35, 16-72
CloseProject	3-8	GetSchematicEnvironment	3-35
CloseProjectNoForce	3-9	KeepDesktopResponsive	3-41
		LaunchJobMonitor	3-42
		NewProject	3-42
		OpenMultipleProjects	3-43

OpenProject 3-44	DoesOutputVariableExist 11-3
PauseRecording 3-45	DoesSupportTraceCharacteristics 12-54
PauseScript 3-46	Draw Menu commands 10-4
Print 3-46	DumpAllReportsData 12-55
QuitApplication 3-47	DuplicateAlongLine 10-97
RefreshJobMonitor 3-48	DuplicateAroundAxis 10-99
ResetLogging 3-49	DuplicateMirror 10-102
RestoreWindow 3-51	
ResumeRecording 3-51	E
RunProgram 3-52	EcxmlExport_1 9-38
RunScript 3-53	Edit 23-50, 23-89, 23-135, 23-170
SelectScheduler 3-56	Edit Menu Commands 10-94
SetActiveProject 3-58	Edit3DComponent 10-64
SetActiveProjectByPath 3-59	Edit3DComponentDefinition 10-67
SetCustomMenuSet 3-59	EditCartesianXMarker 12-55
SetDesktopConfiguration 3-61	EditCartesianYMarker 12-56
SetSchematicEnvironment 3-65	EditCoSimulationOptions 9-31
Sleep 3-68	EditDataset 5-16, 8-5, 23-77
SubmitJob 3-69	EditDesignSettings 9-31
TileWindows 3-70	EditEntityList 10-154
Desktop Object Script Commands 3-1, 14-1	EditFaceCS 10-155
Desktop Scripting Conventions 1- 85	EditFieldsSummarySetting 20-1
DetachEdges 10-151	EditGlobalMeshRegion 14-24
DetachFaces 10-152	EditInfiniteArray 9-33
DistributedAnalyzeSetup, Opti- metrics module command 16- 7, 16-46	EditMarker 12-57
DoesNamedExpressionExists 18- 49, 19-14	EditMaterial 5-19, 23-80
	EditMeshRegion 14-29
	EditObjectCS 10-161

Editor object commands	EnterVector 18-62, 19-23
SetPropertyValue 5-65, 7-21, 9-74, 23-115	EnterVectorFunc 18-62, 19-24
EditOutputVariable 11-4	EnterVol 18-63, 19-24
EditPolyline 10-69	Event Callback Scripting 1-89
EditRelativeCS 10-168	Example Scripts 25-1
EditSetup 15-4, 16-8, 16-46	Export 10-170, 23-97, 23-99, 23-100, 23-147, 23-182
optimization command 16-87	ExportCableLibrary 10-337
parametric command 16-80	ExportConvergence 9-35
sensitivity command 16-104	ExportDataset 5-28, 8-7, 9-36, 23-96
statistical command 16-116	ExportDXConfigFile, Optimetrics module command 16-21, 16-59
EditSources 17-2	ExportEyeMaskViolation 12-58
EditSurfaceMeshSummaryData 18-50	ExportFieldPlot 18-64
EditUserDefinedSolution 22-4	ExportFieldsSummary 20-2
EditWithComps 23-135, 23-170	ExportImageToFile 12-59
EMDesignOptions 9-33	ExportLPVROM_1 9-37
EnableAutoSave 3-15	ExportMarkerTable 18-65
EnableSetup 16-20, 16-59	ExportMaterial 5-29, 23-98
EnterComplex 18-54, 19-15	ExportModelMeshToFile 10-175, 12-63
EnterComplexVector 18-54, 19-16	ExportNMFDData 17-2
EnterCoord 18-55, 19-17	ExportOptimetricsProfile, Optimetrics module command 16-22, 16-60
EnterEdge 18-56, 19-17	ExportOptimetricsResults, Optimetrics module command 16-23, 16-61
EnterLine 18-57, 19-18	ExportOptionsFiles 3-16
EnterOutputVar 18-58, 19-19	ExportOutputVariables 11-5
EnterPoint 18-58, 19-19	ExportParametricResults, Optimetrics module command 16-24, 16-62
EnterQty 18-59, 19-20	ExportPlot3DToFile 12-63
EnterScalar 18-60, 19-21	
EnterScalarFunc 18-60, 19-22	
EnterSurf 18-61, 19-22	

ExportPlotImageToFile 18-68	CalcOp 18-5, 19-4
ExportPlotImageWithViewToFile 18-70	CalcStack 18-6, 19-6
ExportProfile 9-38	ChangeGeomSettings 18-9, 19-9
ExportReport 12-65	ClcMaterial 18-11, 19-11
ExportReportDataToFile 12-66	ClearAllNamedExpr 18-13, 19-12
ExportRespSurfaceMinMaxTable 16-26, 16-64	CopyNamedExprToStack 18-14, 19-13
ExportRespSurfaceRefinePoints 16-27, 16-65	CreateFieldPlot 18-15
ExportRespSurfaceResponsePoints 16-27, 16-66	DeleteFieldPlot 18-46
ExportRespSur- faceVerificationPoints 16-28, 16-67	DeleteNamedExpr 18-47, 19-13
ExportScript 23-100, 23-155	DeleteUneditablePlot 18-48
ExportSurfaceMeshSummary 18- 71	EditSurfaceMeshSummaryData 18-50
ExportTableToFile 12-67	EnterComplex 18-54, 19-15
ExportToFile 12-68, 18-72	EnterComplexVector 18-54, 19-16
ExportTransientData 17-3	EnterCoord 18-55, 19-17
ExportUniformPointsToFile 12-70	EnterEdge 18-56, 19-17
F	
FFTONReport 17-4	EnterLine 18-57, 19-18
Field Calculator commands	EnterOutputVar 18-58, 19-19
DoesNamedExpressionExists 18-49, 19-14	EnterPoint 18-58, 19-19
Field Overlay module commands, GetFieldPlotNames 18-74	EnterQty 18-59, 19-20
field overlay script commands 18-1	EnterScalar 18-60, 19-21
Field Overlays module commands	EnterScalarFunc 18-60, 19-22
AddNamedExpr 18-4, 19-3	EnterSurf 18-61, 19-22
	EnterVector 18-62, 19-23
	EnterVectorFunc 18-62, 19-24
	EnteVol 18-63, 19-24
	ExportFieldPlot 18-64
	ExportOnGrid 18-65

ExportSurfaceMeshSummary 18-71	ClearAllNamedExpr 18-13, 19-12
GetFieldFolderNames 18-73	CopyNamedExprToStack 18-14, 19-13
GetFieldPlotQuantityName 18- 75	DeleteNamedExpr 18-47, 19-13
GetMeshPlotNames 18-75	EnterComplex 18-54, 19-15
HidePolarPlot 18-77	EnterComplexVector 18-54, 19-16
HideRadiatedPlotOverlay 18-78	EnterCoord 18-55, 19-17
ModifyFieldPlot 18-79	EnterEdge 18-56, 19-17
ReassignFieldPlot 18-82	EnterLine 18-57, 19-18
RenameFieldPlot 18-85	EnterOutputVar 18-58, 19-19
RenamePlotFolder 18-86	EnterPoint 18-58, 19-19
SaveFieldsPlots 18-87	EnterQty 18-59, 19-20
SetFieldPlotSettings 18-88	EnterScalar 18-60, 19-21
SetPlotFolderSettings 18-91	EnterScalarFunc 18-60, 19-22
UpdateAllFieldsPlots 18-97	EnterSurf 18-61, 19-22
UpdateQuantityFieldsPlots 18- 97	EnterVector 18-62, 19-23
Fields Calculator commands	EnterVectorFunc 18-62, 19-24
AddNamedExpr 18-4, 18-4, 19- 2, 19-3	EnterVol 18-63, 19-24
CalcOp 18-5, 19-4	ExportOnGrid 19-25
CalcStack 18-6, 19-6	ExportToFile 19-28
CalculatorWrite 18-8, 19-7	Fields Calculator Script Commands 19-1
CalcWrite 18-6, 19-8	Fields reporter module commands
ChangeGeomSettings 18-9, 19- 9	AddMarker[Fields Reporter] 18-1
ClcEval 18-10, 19-10	ClearAllMarkers[Fields Reporter] 18-13
ClcMaterial 18-11, 19-11	DeleteMarker[Fields Reporter] 18-47
ClcMaterialValue 18-12, 19-11	ExportMarkerTable 18-65
	Fields Summary Script Commands 20-1
	Fillet 10-176
	FlattenGroup 10-178

For...Next loop	1-9	GetAllCategories	12-72
functions		GetAllQuantities	12-73
VBScript procedures	1-10	GetAllReportNames	12-74
G			
GenerateAllUserDefinedModels	10-260	GetAllSourceMagnitudes	17-6
GenerateHistory	10-179	GetAllSourceModes	17-7
GenerateUserDefinedModel	10-261	GetAllSourcePhases	17-7
GenerateVariationData Parametric, parametric command	16-29, 16-67, 16-82	GetAllSources	17-8
GeometryCheckAndAutofix	10-262	GetAntennaParameters	17-9
Get3DComponentDefinitionNames	10-73	GetArrayVariables	5-31, 7-16
Get3DComponentInstanceNames	10-74	GetAutoSaveEnabled	3-17
Get3DComponentMaterialNames	10-75	GetAvailableDisplayTypes	12-75
Get3DComponentMaterialProperties	10-75	GetAvailableReportTypes	12-76
Get3DComponentParameters	10-76	GetAvailableSolutions	12-76
Get3DComponentPartNames	10-77	GetBodyNamesByPosition	10-264
GetActiveCoordinateSystem	10-180	GetBoundaries	13-4
GetActiveCoordinateSystemTransform	10-180	GetBoundariesOfType	13-4
GetActiveDesign	5-30	GetBoundaryAssignment	13-5
GetActiveProject	3-16	GetBuildDateString	3-18
GetAdaptiveSettings	17-5	GetChild Object (Report Setup), Report module command	12-77
		GetChildNames	5-31, 9-39
		GetChildNames (Optimetrics), Report module command	12-77
		GetChildNames Modeler	10-265
		GetChildObject Design	5-32, 9-40
		GetChildObject Modeler	10-269
		GetChildTypes	5-33, 9-41, 10-272
		GetChildTypes Optimetrics	16-31, 16-69
		GetChildTypes ReportSetup	12-79

GetCoordinateSystems 10-181
GetCurvePropServerName 12-79
GetData 23-119, 23-148, 23-183
GetDataExpressions 12-93
GetDataUnits 12-94
GetDefaultBaseName 13-5
GetDefinitionManager 5-35
GetDependentFiles 5-35
GetDesignID 9-43
GetDesignName 9-44
GetDesigns 5-37
GetDesignType 9-44
GetDesignVariableNames 12-95
GetDesignVariableUnits 12-96
GetDesignVariableValue 12-97
GetDesignVariationKey 12-98
GetDisplayType 12-80
GetDistributedAnalysisMachines 3-22
GetDis-
trib-
utedAna-
lysisMachinesForDesignType
3-23
GetDocumentNames 21-8
GetDynLinkIntrinsicVariables 12-81
GetDynLinkQtyValueState 12-81
GetDynLinkTraces 12-82
GetDynLinkVariableValues 12-83
GetEdgeByPosition 10-274
GetEdgeIDFromNameForFirstOperation
10-276
GetEdgeIDsFromFace 10-276
GetEdgeIDsFromObject 10-277
GetEdgeLength 10-278
GetEdgePositionAtNormalizedParameter
9-45
GetEntityListIDByName 10-278
GetExeDir 3-24
GetExtendedDefinitionObect 10-279
GetFaceArea 10-280
GetFaceByPosition 10-282
GetFaceCenter 10-281
GetFacelIDFromNameForFirstOperation
10-283
GetFacelIDs 10-284
GetFacelIDsOfSheet 10-285
GetFieldFolderNames 18-73
GetFieldPlotNames, Field Overlay module
command 18-74
GetFieldPlotQuantityName 18-75
GetFieldType 17-9
GetGeometryModelerMode 10-285
GetGroupSubmodelNames 10-286
GetImagDataValues 12-98
GetIncludePortPostProcessing 17-10
GetLibraryDirectory 3-25
GetMatchedObjectName 10-287
GetMeshPlotNames 18-75
GetModelBoundingBox 10-287

GetModelUnits 10-288
GetModule 9-47
GetMultipactionBreakdown 17-11
GetName 5-39, 9-48, 12-83, 16-32, 16-70
GetNames 23-119, 23-149, 23-184
GetNetworkDataSolution 17-11
GetNetworkDataSolutionDefinition 17-12
GetNetworkPostprocSetup 17-13
GetNumBoundaries 13-6
GetNumBoundariesOfType 13-6
GetNumObjects 10-289
GetObjectIDByName 10-289
GetObjectName 10-290
GetObjectByNameByID 10-291
GetObjectByNameByFacetID 10-291
GetObjectByNameByID 10-292
GetObjectByNameByVertexID 10-293
GetObjectsByMaterial 10-293
GetObjectShapeType 10-294
GetObjectsInGroup 10-295
GetObjectVolume 10-296
GetObjPath 5-40, 9-50, 10-296, 12-84, 16-32, 16-70
GetOperationNames 14-30
GetOptimetricsResults, Optimetrics module command 16-34
GetOutputVariableValue 9-51, 11-7
GetPartsForUserDefinedModel 10-297
GetPath 5-40
GetPerQuantityPrimarySweepValues 12-99
GetPersonalLibDirectory 3-29
GetPoints 10-298
GetPPELicensingEnabled 3-29
GetProcessID 3-30
GetProjectDirectory 3-31
GetProjectList 3-32
GetProjectMaterialNames 23-101
GetProjects 3-28, 3-32
GetProperties 5-42, 7-17, 9-53, 23-121
GetPropertyEvaluatedValue 10-298
GetPropNames 5-41, 9-56, 12-86, 16-35, 16-72
GetPropNames Modeler 10-301
GetPropSIValue 10-302
GetPropValue Modeler 10-303
GetPropValue Optimetrics 16-36, 16-73
GetPropValue Project 5-42, 9-57
GetPropValue Reporter 12-87
GetQtyExpressionsForSourceTrace 12-88
GetRealDataValues 12-100
GetRegistryInt 3-33, 3-73
GetRegistryString 3-34, 3-74

GetRelativeCoordinateSystems 10-304
GetReportSummaryForRegressionTesting 12-89
GetReportTraceNames 12-88
GetSelections 9-58, 10-304
GetSetupCount 15-16
GetSetupCount, Analysis module command 15-16
GetSetupNames (Optimetrics), Optimetrics module command 16-30, 16-30, 16-36, 16-68, 16-69, 16-73
GetSetupNamesByType (Optimetrics), Optimetrics module command 16-37, 16-74
GetSetups 15-17
 Analysis module command 15-17
GetSolutionContexts 12-90
GetSolutionDataPerVariation 12-91
GetSolutionType 9-58
GetSolveInsideThreshold 9-59
GetSourceContexts 17-13
GetSubGroupsInGroup 10-305
GetSweepNames 12-101
GetSweepUnits 12-102
GetSweepValues 12-103
GetSysLibDirectory 3-37
GetTempDirectory 3-38
GetTerminalExcitationType 17-14
GetTool 3-77
GetTopDesignList 5-45
GetTopEntryValue 18-76, 19-31
GetTransientSolveTimes 17-15
 GetUserDefinedSolutionNames 22-6
 GetUserDefinedSolutionProperties 22-6
 GetUserLibDirectory 3-38
 GetUserPosition 10-306
GetVariables 5-47, 7-20, 9-59
GetVariableValue 5-46, 7-20, 9-60
GetVariationVariableValue 9-61
GetVersion 3-39
GetVertexIDFromNameForFirstOperation 10-307
GetVertexIDsFromEdge 10-308
GetVertexIDsFromFace 10-308
GetVertexIDsFromObject 10-309
GetVertexPosition 10-310
GetWireBodyNames 10-310
GroupPlotCurvesByGroupingStrategy 12-107

H

HasDataset 8-8
HealObject 10-182
HidePolarPlot 18-77
HideRadiatedPlotOverlay 18-78
hierarchy of variables in HFSS 1-76

I	
Icepak boundary condition commands	ImportAutoCAD 3-81
AssignAdiabaticPlateBoundary 14-8	ImportAWRMicrowaveOffice 3-82
AssignBlockBoundary 14-1	ImportCableLibrary 10-338
AssignBlowerBoundary_1 14-22	ImportDataset 5-48, 8-9, 9-62, 23-104
AssignConductingPlateBoundary 14-10	ImportDXF 10-190
AssignEMLoss 14-21	ImportEDB 3-84
AssignGrilleBoundary 14-3	ImportExport Tool 3-77
AssignMeshRegion 14-27	ImportExtracta 3-85
AssignNetworkBoundary 14-5	ImportFromClipboard 10-194
AssignOpeningBoundary 14-6	ImportGDSII 3-86, 3-87, 10-195
AssignRecircBoundary 14-7	ImportIDF 3-88, 3-91
AssignResistanceBoundary 14-13	ImportIDX 3-92
AssignSourceBoundary 14-17	ImportIntoReport 12-108
AssignStationaryWallBoundary 14-18	ImportIPC 3-95
AssignSymmetryWallBoundary 14-20	ImportODB 3-96
EcxmlExport 9-38	ImportOutputVariables 11-8
EditGlobalMeshRegion 14-24	ImportReportDataIntoReport 12-109
ExportLPVROM_1 9-37	ImportSetup, Optimetrics module command 16-38, 16-75
ImportIDX 3-92	ImportXFL 3-97
InsertSetup 15-17, 15-28	Imprint 10-198
If...Then... Else statement 1-8	ImprintProjection 10-199
Import (3D Modeler command) 10-186	include files
scripts 1-11	
indentation, IronPython 1-21	
InputBox function 1-11	
Insert3DComponent 10-77	
InsertComponent 10-78	
InsertDesign 5-50, 9-64	
ImportANF 3-78, 3-79	

InsertDesignWithWorkflow 5-51

InsertSetup 15-17, 15-28

optimization command 16-93

parametric command 16-82

sensitivity command 16-111

statistical command 16-119

Intersect 10-201

IronPython 1-16

indentation in 1-21

IsDataComplex 12-104

IsFeatureEnabled 3-40

IsPerQuantityPrimarySweep 12-105

IsUsed 23-122, 23-150, 23-184

K

KeepDesktopResponsive 3-41

keywords, VBScript 1-12, 1-14

L

LaunchJobMonitor 3-42

Layout Scripting 1-89

LoadNamedExpressions 18-78, 19-32

logical operators 1-8

looping through code

Do ... Loop 1-9

For ... Next 1-9

M

material commands

AddDefinitionFromBlock 10-237, 23-23

AddDefinitionFromLibFile 10-242

AddMaterial 5-6, 23-28

EditMaterial 5-19, 23-80

ExportMaterial 5-29, 23-98

GetExtendedDefinitionObject 10-279

RemoveMaterial 5-55, 23-109

UpdateDefFromBlock 23-116

Materials Scripting Support 6-23

Mesh Operations module commands

DeleteOp 14-30

EditMeshRegion 14-29

GetOperationNames 14-30

ReassignOp 14-31

RenameOp 14-32

Mesh Operations Module Script Commands 14-24

mesh operations script commands 14-30

Microsoft

VBScript user's guide 1-12

Mirror 10-105

Model Manager Script Commands 23-123

Modeler Menu Commands 10-114

ModifyFieldPlot 18-79

ModifyLibraries 23-156

module script commands	OpenWindowForReports 12-112
boundary conditions 13-1	
field overlay 18-1	
mesh operations 14-30	
modules in HFSS scripting 1-76	
Move 10-107	
MoveCSToEnd 10-203	
MoveEntityToGroup 10-204	
MoveFaces 10-205	
MovePlotCurvestoGroup 12-109	
MovePlotCurvestoNewGroup 12-110	
MsgBox function 1-11	
N	
Named Arguments 1-86	
NewProject 3-42	
O	
oAnsoftApp object 1-76	
Object Oriented Property Scripting 6-1	
oDesign object 1-76	
oDesktop object 1-76	
oEditor object 1-76	
OffsetFaces 10-109	
oModule object 1-76	
OpenExternalEditor 10-311	
OpenMultipleProjects 3-43	
OpenProject 3-44	
OpenWindowForAllReports 12-111	
operators	
arithmetic 1-6	
categories in VBScript 1-5	
comparison 1-7	
concatenation 1-7	
logical 1-8	
precedence of 1-5	
oProject object 1-76	
Optimetrics module commands	
DeleteSetups 16-7, 16-45	
DistributeAnalyzeSetup 16-7, 16-46	
EnableSetup 16-20, 16-59	
ExportDXConfigFile 16-21, 16-59	
ExportOptimetricsProfile 16-22, 16-60	
ExportOptimetricsResults 16-23, 16-61	
ExportParametricResults 16-24, 16-62	
ExportRespSurfaceMinMaxTable 16-26, 16-64	
ExportRespSurfaceRefinePoints 16-27, 16-65	
ExportRespSurfaceResponsePoints 16-27, 16-66	
ExportRespSurfaceVerificationPoints 16-28, 16-67	
GetChildNames 16-30, 16-30, 16-68, 16-69	
GetOptimetricResult 16-33, 16-71	

GetOptimetricsResults 16-34
GetSetupNames 16-36, 16-74
GetSetupNamesByType 16-37,
16-74
ImportSetup 16-38, 16-75
RenameSetup 16-40, 16-77
SolveSetup 15-2, 16-6, 16-42,
16-42, 16-44, 16-79, 16-79
Optimetrics Module Script Com-
mands 16-1
optimization commands
EditSetup 16-87
InsertSetup 16-93
Optimization Script Commands 16-
87
Other oEditor Commands 10-234
output variable commands
CreateOutputVariable 11-1
DeleteOutputVariable 9-28, 9-
30, 11-2
DoesOutputVariableExist 11-3
EditOutputVariable 11-4
GetOutputVariableValue 9-51,
11-7
Output variable commands
ExportOutputVariables 11-5
ImportOutputVariables 11-8
Output Variable Script
Commands 11-1

P

PageSetup 10-312

parametric commands
EditSetup 16-80
GenerateVariationData Parametric 16-
29, 16-67, 16-82
InsertSetup 16-82
Parametric Script Commands 16-80
Paste 5-53, 10-334, 10-335
Paste (Model Editor) 10-110
Paste (Project Object) 5-53
PasteDesign 9-65
PastePlotSettings 12-113
PasteReports 12-113
PasteReportsWithLegacyNames 12-114
PasteSetup
Analysis module command 15-39
Optimetrics module command 16-39,
16-76
PasteTraces 12-115
PasteTracesWithLegacyNames 12-115
PauseRecording 3-45
PauseScript 3-46
Print 3-46
Project object commands
AddDataset 5-3, 8-1, 9-4, 23-20
AddMaterial 5-6, 23-28
Close 5-12
CopyDesign 5-12
CutDesign 5-13
DeleteDataset 5-14, 8-4, 9-25, 23-50

DeleteDesign 5-15
EditDataset 5-16, 5-28, 5-48, 8-5, 8-7, 8-9, 9-36, 9-61, 23-77, 23-96, 23-104
EditMaterial 5-19, 23-80
ExportMaterial 5-29, 23-98
GetActiveDesign 5-30
GetArrayVariables 5-31, 7-16
GetChildObject Design 5-32, 9-40
GetChildObject Modeler 10-269
GetDependentFiles 5-35
GetDesignID 9-43
GetDesignName 9-44
GetDesignType 9-44
GetName 5-39
GetObjPath 5-40
GetPath 5-40
GetProjectMaterialNames 23-101
GetProperties 5-42, 7-17, 9-53
GetPropertyValue 5-44, 7-18, 9-54, 10-299, 12-85, 23-102
GetPropEvaluatedValue 10-298
GetPropNames Modeler 10-301
GetPropSIValue 10-302
GetPropvalue Modeler 10-303
GetPropvalue Optimetrics 16-36, 16-73
GetPropvalue Project 5-42, 9-57
GetPropvalue Reporter 12-87
GetTopDesignList 5-45
GetTopEntryValue 18-76, 19-31
GetVariables 5-47, 7-20, 9-59
GetVariableValue 5-46, 7-20, 9-60
InsertDesign 5-50, 9-64
Paste 5-53, 5-53
Redo 5-54
RemoveAllUnusedDefinitions 5-54
RemoveMaterial 5-55, 23-109
RemoveUnusedDefinitions 5-56, 23-113
Rename 5-57
RestoreProjectArchive 3-50
Save 5-58
SaveAs 5-59
SaveAsStandAloneProject 5-61
SaveProjectArchive 5-62
SetActiveDesign 5-64
SetPropertyValue 5-65, 7-21, 9-74, 23-115
SetPropValue Design 9-73
SetPropValue Optimetrics 16-41, 16-78
SetPropValue Project 5-64, 12-120
SetVariableValue 5-67, 7-23, 9-79
SimulateAll 5-10, 5-68
Undo 5-69
UpdateDefinitions 5-69
Project Object Script Commands 5-1

- ProjectSheet 10-207
property commands
 GetArrayVariables 5-31, 7-16
 GetProjectMaterialNames 23-101
 GetProperties 5-42, 7-17, 9-53
 GetPropertyValue 5-44, 7-18, 9-54, 10-299, 12-85, 23-102
 GetPropEvaluatedValue 10-298
 GetPropSIValue 10-302
 GetTopEntryValue 18-76, 19-31
 GetVariables 5-47, 7-20, 9-59
 GetVariableValue 5-46, 7-20, 9-60
 SetPropertyValue 5-65, 7-21, 9-74, 23-115
 SetVariableValue 5-67, 7-23, 9-79
- Property Script Commands 7-1
 Conventions used in this Chapter 7-13
Object Script Property Function Summary 7-3
- PurgeHistory 10-208
- Q**
- QuitApplication 3-47
- R**
- ReassignBoundaries 13-2, 13-7
ReassignBoundary 13-7
ReassignFieldPlot 18-82
- ReassignOp 14-31
Record Script and Edit Properties 7-24
Redo
 design-level command 9-66
 project-level command 5-54
references, for VBScript 1-12
RefreshJobMonitor 3-48
ReleaseData 12-106
Remove 23-106, 23-108, 23-110, 23-112, 23-150, 23-185
RemoveAllUnusedDefinitions 5-54
RemoveAssignmentFromBoundary 13-7
RemoveBadEdges 10-314
RemoveBadFaces 10-314
RemoveBadVertices 10-315
RemoveCables 10-339
RemoveMaterial 5-55, 23-109
RemoveScript 23-111, 23-158
RemoveUnused 23-122, 23-152, 23-186
RemoveUnusedDefinitions 5-56, 23-113
Rename 5-57
RenameBoundary 13-8
RenameDesignInstance 9-67
RenameFieldPlot 18-85
RenameOp 14-32
RenamePart 10-316
RenamePlotFolder 18-86
RenameReport 12-116

RenameSetup	15-39	ApplyReportTemplate	12-20
Analysis module command	15-39	ClearAllMarkers	12-21
Optimetrics module command	16-40, 16-77	ClearAllTraceCharacteristics	12-22
RenameSource [Interface Source]	9-68	CloneReportsFromDatasetSolution	12-22
RenameTraces	12-117	CopyPlotSettings	12-23
Repeating a Statement Until a Condition Becomes True	1-10	CopyReportDefinition	12-24
Repeating Statements While a Condition is True	1-9	CopyReportsData	12-25
ReplaceWith3DComponent	10-209	CopyTraceDefinitions	12-25
Report module commands		CopyTracesData	12-26
GetChildNames	12-77	CreateReport	9-10, 12-27
Reporter editor commands		CreateReportFromTemplate	12-47
AddAllEyeMeasurements	12-4	CreateReportOfAllQuantities	12-48
AddCartesianLimitLine	12-4	DeleteAllReports	12-51
AddCartesianLimitLineFromCurve	12-6	DeleteReport	12-51
AddCartesianLimitLineFromEquation	12-8	DeleteTraceCharacteristics	12-52
AddCartesianXMarker	12-10, 12-10	DeleteTraces	12-53
AddCartesianYMarkerToStack	12-11	DoesSupportTraceCharacteristics	12-54
AddDeltaMarker	12-12	DumpAllReportsData	12-55
AddMarker	12-13	EditCartesianXMarker	12-55
AddMarkerToPlot	18-2	EditCartesianYMarker	12-56
AddNote	12-14	EditMarker	12-57
AddTraceCharacteristics	12-16	ExportEyeMaskViolation	12-58
AddTraces	12-18	ExportImageToFile	12-59
		ExportPlot3DToFile	12-63
		ExportPlotImageToFile	18-68, 18-70
		ExportReport	12-65

ExportReportDataToFile 12-66	GetSolutionContexts 12-90, 12-91
ExportTableToFile 12-67	GroupPlotCurvesByGroupingStrategy 12-107
ExportToFile 12-69, 12-108	ImportReportDataIntoReport 12-109
ExportUniformPointsToFile 12-70	MovePlotCurvestoGroup 12-109
FFTOnReport 17-4	MovePlotCurvestoNewGroup 12-110
GetAllCategories 12-72	OpenWindowForAllReports 12-111
GetAllQuantities 12-73	OpenWindowForReports 12-112
GetAllReportNames 12-74	PastePlotSettings 12-113
GetAvailableDisplayTypes 12-75	PasteReports 12-113
GetAvailableReportTypes 12-76	PasteReportsWithLegacyNames 12-114
GetAvailableSolutions 12-76	PasteTraces 12-115
GetChildTypes ReportSetup 12-79	PasteTracesWithLegacyNames 12-115
GetCurvePropServerName 12-79	RenameReport 12-116
GetDisplayType 12-80	RenameTraces 12-117
GetDynLinkIntrinsicVariables 12-81	ResetPlotSettings 12-118
GetDynLinkQtyValueState 12-81	SavePlotSettingsAsDefault 12-118
GetDynLinkTraces 12-82	SetLinkOutputTraces 12-119
GetDynLinkVariableValues 12-83	UnGroupPlotCurvesInGroup 12-121
GetPropNames 12-86	UpdateAllReports 12-122
GetQtyExpressionsForSourceTrace 12-88	UpdateReports 12-122
GetRe- portSum- maryForRegressionTesting 12-89	UpdateTraces 12-123
GetReportTraceNames 12-88	UpdateTracesContextAndSweeps 12-126
	Reporter Editor Script Commands 12-1
	Reporter module commands
	GetChildObject 12-78
	ReprioritizeBoundaries 13-9
	ResetLogging 3-49

ResetPlotSettings 12-118
RestoreWindow 3-51
ResumeRecording 3-51
RevertAllToInitial 15-40
RevertSetupToInitial 15-41
Rotate 10-111
Running Instance Manager Script Commands 4-1
RunProgram 3-52
RunScript 3-53
RunScriptWithArguments 3-55
RunToolkit 9-68

S

sample scripts
 simple HFSS 1-12
 simple Q3D Extractor 1-14
 variable helix 25-13
 waveguide 25-16
SARSetup 9-69
Save 5-58
SaveAs 5-59
SaveAsStandAloneProject 5-61
SaveFieldsPlots 18-87
SaveNamedExpressions 18-87, 19-33
SavePlotSettingsAsDefault 12-118
Scale 10-112
Scope and Lifetime of Variables 1-4
Script and Library Scripts 23-153

script commands
 boundary/excitation 13-1
 field overlay 18-1
 mesh operations 14-30

Script Commands for Creating and Modifying Mesh Regions 14-24

scripts
 CPython 1-72
 IronPython 1-16
 overview 1-1
 pausing 1-81
 recording 1-82
 running 1-80
 stopping 1-81
 VBScript 1-2

Scripts and Locked Layers 1-89

Section 10-212

Select Case statement 1-8

SelectScheduler 3-12, 3-56

sensitivity commands
 EditSetup 16-104
 InsertSetup 16-111

Sensitivity Script Commands 16-104

SeparateBody 10-214

SetActiveDefinitionEditor 5-63

SetActiveDesign 5-64

SetActiveEditor 9-70

SetActiveProject 3-58

SetActiveProjectByPath 3-59

SetAllowMaterialOverride	9-70	simple and composite names	1-3
SetBackgroundMaterial	9-71	Simplify	10-218
SetDefaultBaseName	13-10	SimulateAll	5-10, 5-68
SetFieldPlotSettings	18-88	SimValueContext	11-9
SetLengthSettings	9-72	Sleep	3-68
SetLibraryDirectory	3-62	Solution module commands	
SetLinkOutputTraces	12-119	GetSourceContexts	17-13
SetModelUnits	10-215	SetSourceContexts	9-78, 17-15
SetPlotFolderSettings	18-91	Solutions module commands	
SetProjectDirectory	3-62	ConstructVariationString	9-8
SetPropertyValue	5-65, 7-21, 9-74, 23-115	DeleteFieldVariation	9-25
SetPropValue Design	9-73	DeleteFullVariation	9-26
SetPropValue Modeler	10-317	DeleteLinkedDataVariation	9-27
SetPropValue Optimetrics	16-41, 16-78	DeleteSolutionVariation	9-29, 17-1
SetPropValue Project	5-64, 12-120	EditSources	17-2
SetSolutionType	9-75	ExportNMF	17-2
SetSolveInsideThreshold	9-77	ExportTransientData	17-3
SetSourceContexts	9-78, 17-15	GetAdaptiveSettings	17-5
SetTempDirectory	3-66	GetAllSourceMagnitudes	17-6
Setting Numerical Values	1-88	GetAllSourceModes	17-7
SetTopDownViewDirectionForActiveView	10-318	GetAllSourcePhases	17-7
SetTopDownViewDirectionForAllViews	10-319	GetAllSources	17-8
SetVariableValue	5-67, 7-23, 9-79	GetAntennaParameters	17-9
SetWCS	10-216	GetFieldType	17-9
SGetAppDesktop	2-2	GetIncludePortPostProcessing	17-10
ShowWindow	10-217	GetMultipactionBreakdown	17-11
		GetNetworkDataSolution	17-11
		GetNetworkDataSolutionDefinition	17-12

- GetNetworkPostprocSetup 17-13
- GetTerminalExcitationType 17-14
- GetTransientSolveTimes 17-15
- TDROnReport 17-16
- Solutions Module Script Commands 17-1
- Solve 9-80
- SolveAllSetup, Optimetrics module command 16-42, 16-79
- SolveSetup, Optimetrics module command 16-42, 16-79
- Split 10-221
- statistical commands
- EditSetup 16-116
 - InsertSetup 16-119
- Statistical Script Commands 16-116
- Stitch 10-223
- StopSimLink 9-81
- string concatenation operator 1-7
- Sub procedures 1-12, 1-14
- Sub Procedures 1-10
- SubmitJob 3-68
- Subtract 10-225
- SweepAlongPath 10-82
- SweepAlongVector 10-84
- SweepAroundAxis 10-86
- SweepFacesAlongNormal 10-88, 10-88, 10-226, 10-226
- Sweep-
- FacesAlongNormalWithAttributes 10-90
- Symbol Manager Script Commands 23-158
- T**
- TDROnReport 17-16
- ThickenSheet 10-228
- U**
- UncoverFaces 10-230
- underscore (_) character 1-12, 1-14
- Undo
- project-level command 5-68
- Ungroup 10-232
- UnGroupPlotCurvesInGroup 12-121
- Unite 10-231
- UpdateAllFieldsPlots 18-97
- UpdateAllReports 12-122
- UpdateCableHarness 10-339
- UpdateComponentDefinition 10-93
- UpdateDefFromBlock 23-116
- UpdateDefinitions, project-level command 5-69
- UpdatePriorityList 10-319
- UpdateQuantityFieldsPlots 18-97
- UpdateReports 12-122
- UpdateTraces 12-123
- UpdateTracesContextAndSweeps 12-126
- UpgradeVersion 10-320

User defined documents module commands
GetDocumentNames 21-8

User defined solution module commands
CreateUserDefinedSolution 22-1
DeleteUserDefinedSolutions 22-3
EditUserDefinedSolution 22-4
 GetUserDefinedSolutionNames 22-6
 GetUserDefinedSolutionProperties 22-6

User Defined Solutions Commands 22-1

Using a Do Loop 1-9

references 1-12
Sub procedures 1-12, 1-14
VBScript Procedures 1-10

W

WrapSheet 10-233
WriteHistoryTreeLayoutForTest 10-322

V

Validate3DComponent 10-321
ValidateDesign 9-82
Variable Naming Conventions 1-4
variables
array 1-4
assigning information 1-3
declaring 1-3
hierarchy in HFSS 1-76
used as objects 1-12
used in HFSS scripts 1-76

VBScript 1-2
Microsoft user's guide 1-12
operators 1-5