



POWERING INNOVATION THAT DRIVES HUMAN ADVANCEMENT

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

Maxwell® Scripting Guide



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

Release 2025 R2
July 2025

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

Copyright and Trademark Information

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

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

Disclaimer Notice

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

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

U.S. Government Rights

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

Third-Party Software

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

Table of Contents

Table of Contents	Contents-1
1 - Introduction to Scripting	1-1
Scripting Help Conventions	1-1
Variable Types	1-2
Introduction to IronPython	1-2
Scope	1-2
Python Compatibility	1-2
Advantages of IronPython	1-2
Scripting Using Iron Python	1-3
IronPython Script Execution Environment	1-4
Script Argument for IronPython	1-4
Scripting using Embedded VBScript or JavaScript	1-5
Scripting with Python	1-6
Standalone IronPython	1-7
Running Standalone IronPython	1-7
Using a Recorded Script	1-7
Creating an External Script	1-7
Example Script	1-8
IronPython Samples	1-9
Change property	1-10
Create a Cone using IronPython	1-11
Creating User Defined Primitives and User Defined Models in Python Scripts	1-14
Advantages Compared to C++	1-15
Changes compared to C	1-15
Structures	1-15
Return Values for UDM and UDP Functions	1-15
Constants	1-15
Methods	1-15

Output Parameters	1-16
Comparison with C function:	1-16
'List Size' Parameters	1-17
Comparison with C function:	1-17
Added Parameters	1-18
Developing a UDM/UDP	1-19
Creation	1-19
Location	1-19
Organize	1-20
Edit/Reload	1-20
UDPExtension	1-20
IUDPExtension Methods	1-20
Mandatory Methods	1-21
GetLengthParameterUnits()	1-21
Optional Methods	1-21
Example UDP	1-22
UDMExtension	1-23
Import	1-23
Main class: UDMExtension	1-23
IUDMExtension Methods	1-23
Mandatory Methods	1-23
Optional Methods	1-25
Example UDM	1-26
UDMFunctionLibrary	1-26
Functions list:	1-27
UDM/UDP Functions	1-28
Return Values for Each UDM and UDP Function	1-28
UDP/UDM Structures and Constants	1-30
UDP/UDM Structures	1-30
List of Structures	1-32

UDP/UDM Constants	1-38
Enum constants:	1-38
UDP Python Example	1-40
Introduction to CPython	1-44
Creating an External Script	1-45
Start ansysedt as GRPC server	1-45
Connect Functions:	1-46
Example:	1-46
Launching Electronics Desktop	1-46
Connecting with a Running Instance of Electronics Desktop	1-47
Closing Electronics Desktop/Ending the Script	1-47
-grpcsrv Flag	1-47
Ansys Electronics Desktop Scripting	1-48
Overview of Electronics Desktop Scripting Objects	1-48
oAnsoftApp	1-49
oDesktop	1-49
oProject	1-50
oDesign	1-50
oEditor	1-50
oModule	1-51
Example Script Opening	1-51
GetActiveProject and GetActiveDesign for Wider Use	1-51
Running a Script	1-51
Within Electronics Desktop	1-52
From the Command Line	1-52
Recording a Script	1-53
Recording a Script to File	1-53
Recording a Script to a Project	1-54
Working with Project Scripts	1-54
Executing a Script from Within a Script	1-55

Electronics Desktop Scripting Conventions	1-56
Named Arguments	1-56
IronPython Example	1-57
Setting Numerical Values	1-58
PyAEDT (Beta)	1-58
2 - Object-Oriented Property Scripting	2-1
Object-Oriented Scripting	2-1
Material Properties and Examples	2-9
Body Properties and Modification	2-11
Retrieving Variables	2-11
Retrieve Datasets and Values	2-12
GetSolutionData API	2-14
Summary	2-14
Additional Details Specific to AEDT Solvers	2-18
Additional details on Boundaries/Excitations	2-19
3D component encapsulation	2-21
Additional details on Solve setup	2-21
Materials Scripting Support	2-22
Object Oriented Scripting for Materials	2-26
Examples showing change to material property type:	2-27
Examples showing change to a vector component value	2-27
Change choice property value	2-28
Change choice property value	2-29
Project Object Script Commands	2-30
AnalyzeAll [project]	2-33
ClearMessages	2-34
Close	2-35
CopyDesign	2-35
CutDesign	2-36
DeleteDesign	2-37

DeleteToolObject	2-37
GetActiveDesign	2-38
GetDefinitionManager	2-38
GetDependentFiles	2-39
GetDesign	2-40
GetEDBHandle	2-40
GetLegacyName	2-41
GetName [Project]	2-41
GetPath	2-42
GetTopDesignList	2-42
InsertDesign	2-43
InsertDesignWithWorkflow	2-44
InsertToolObject	2-45
Paste (Project Object)	2-45
Redo [Project Level]	2-46
Rename	2-46
RestoreProjectArchive	2-47
Save	2-48
SaveAs	2-48
SaveAsStandAloneProject	2-50
SaveProjectArchive	2-51
SetActiveDefinitionEditor	2-52
SetActiveDesign	2-52
SimulateAll	2-53
Undo [Project]	2-53
UpdateDefinitions	2-54
3 - Application Object Script Commands	3-1
GetAppDesktop	3-2
GetApp	3-2
4 - Desktop Object Script Commands	4-1

AddMessage	4-5
ClearMessages	4-6
CloseAllWindows	4-10
CloseProject	4-10
CloseProjectNoForce	4-11
Count	4-12
DeleteProject	4-12
EnableAutoSave	4-13
ExportOptionsFiles	4-13
GetActiveProject	4-14
GetAutoSaveEnabled	4-15
GetBuildDateTimeString	4-15
GetCustomMenuSet	4-16
GetDefaultUnit	4-16
GetDesigns	4-18
GetDesktopConfiguration	4-19
GetDistributedAnalysisMachines	4-20
GetDistributedAnalysisMachinesForDesignType	4-20
GetExeDir	4-21
GetGDIObjectCount	4-21
GetLibraryDirectory	4-22
GetLocalizationHelper	4-22
GetMessages	4-23
GetName [Desktop]	4-24
GetPersonalLibDirectory	4-25
GetProcessID	4-25
GetProjects	4-26
GetProjectDirectory	4-26
GetProjectList	4-27
GetSchematicEnvironment	4-27

GetScriptingToolsHelper	4-28
GetSysLibDirectory	4-29
GetTempDirectory	4-29
GetUserLibDirectory	4-30
GetVersion	4-31
LaunchJobMonitor	4-31
NewProject	4-32
OpenAndConvertProject	4-32
OpenMultipleProjects	4-33
OpenProject	4-34
OpenProjectWithConversion	4-34
Paste (Project Object)	4-35
PauseRecording	4-35
PauseScript	4-36
Print	4-36
QuitApplication	4-37
RefreshJobMonitor	4-37
ResetLogging	4-38
RestoreProjectArchive	4-39
RestoreWindow	4-40
ResumeRecording	4-40
RunACTWizardScript	4-41
RunProgram	4-41
RunScript	4-42
RunScriptWithArguments	4-43
SelectScheduler	4-45
SetActiveProject	4-46
SetActiveProjectByPath	4-46
SetCustomMenuSet	4-47
SetDesktopConfiguration	4-48

SetLibraryDirectory	4-49
SetProjectDirectory	4-49
SetSchematicEnvironment	4-50
SetTempDirectory	4-51
ShowDockingWindow	4-51
Sleep	4-52
SubmitJob	4-53
TileWindows	4-54
Desktop Commands For Registry Values	4-54
DoesRegistryValueExist	4-55
GetRegistryInt	4-56
GetRegistryString	4-57
SetRegistryFromFile	4-57
SetRegistryInt	4-58
SetRegistryString	4-58
ImportExport Tool Commands	4-59
ImportANF	4-60
ImportANFv2	4-61
ImportAutoCAD	4-61
ImportAWRMicrowaveOffice	4-62
ImportEDB	4-63
ImportExtracta	4-64
ImportGDSII	4-65
ImportGerber	4-65
ImportIDF	4-66
ImportIDFandMerge	4-69
ImportIDX	4-70
ImportIPC	4-72
ImportODB	4-73
ImportXFL	4-73

5 - Running Instances Manager Script Commands	5-1
GetAllRunningInstances	5-1
GetRunningInstanceByProcessID	5-1
GetRunningInstancesMgr	5-2
6 - Project Object Script Commands	6-1
AddDataset	6-4
AnalyzeAll [project]	6-6
ChangeProperty	6-7
ClearMessages	6-14
Close	6-15
CopyDesign	6-15
CutDesign	6-16
DeleteDataset	6-17
DeleteDesign	6-17
DeleteToolObject	6-18
EditDataset	6-18
ExportDataset	6-20
GetActiveDesign	6-21
GetArrayVariables	6-22
GetChildNames [Project]	6-22
GetChildObject [Project]	6-23
GetChildTypes [Project]	6-24
GetDefinitionManager	6-24
GetDependentFiles	6-25
GetDesign	6-25
GetDesigns	6-26
GetEDBHandle	6-27
GetLegacyName	6-27
GetName [Project]	6-28
GetObjPath [Project]	6-29

GetPath	6-29
GetPropEvaluatedValue	6-30
GetPropNames [Project]	6-31
GetPropSIValue	6-31
GetPropValue [Project]	6-32
GetProperties	6-33
GetPropertyValue	6-34
GetTopDesignList	6-35
GetVariableValue	6-36
GetVariables	6-37
ImportDataset	6-37
Note About File Types:	6-38
InsertDesign	6-39
InsertDesignWithWorkflow	6-40
InsertToolObject	6-41
Paste (Project Object)	6-42
Redo [Project Level]	6-42
RemoveAllUnusedDefinitions	6-43
RemoveMaterial	6-43
RemoveUnusedDefinitions	6-44
Rename	6-45
RunToolkit	6-46
Save	6-47
SaveAs	6-47
SaveAsStandAloneProject	6-49
SaveProjectArchive	6-50
SetActiveDefinitionEditor	6-51
SetActiveDesign	6-51
SetPropValue [Project]	6-52
SetPropertyValue	6-53

SetVariableValue	6-54
SimulateAll	6-55
Undo [Project]	6-55
UpdateDefinitions	6-56
7 - Definition Manager Script Commands	7-1
AddMaterial (Maxwell)	7-2
Additional Properties for Nonlinear Property Type	7-6
Additional Properties for Vector Property Type	7-7
Additional Python Examples	7-8
CloneMaterial	7-12
ComputeCoreLossCoefficients	7-13
DoesMaterialExist	7-14
EditMaterial (Maxwell)	7-15
Additional Examples	7-19
GetProjextMaterialNames	7-30
ExportMaterial	7-31
RemoveMaterial	7-32
RemoveUnusedDefinitions	7-33
Component Manager Script Commands	7-34
Add [component manager]	7-35
AddNPortData [component manager]	7-50
AddSolverOnDemandModel	7-54
ClearSolutionCache [component manager]	7-55
Edit [component manager]	7-55
EditSolverOnDemandModel	7-74
EditWithComps [component manager]	7-74
Export [component manager]	7-83
GetNames [component manager]	7-84
GetNPortData [component manager]	7-84
GetSolverOnDemandData	7-85

GetSolverOnDemandModelList	7-85
IsUsed [component manager]	7-85
Remove [component manager]	7-86
RemoveSolverOnDemandModel	7-87
RemoveUnused [component manager]	7-87
Update Dynamic Link [component manager]	7-88
Material Manager Script Commands	7-89
GetNames [material manager]	7-89
GetProperties [material manager]	7-90
IsUsed [material manager]	7-90
RemoveUnused [material manager]	7-91
Model Manager Script Commands	7-91
Add [model manager]	7-92
ConvertToDynamic	7-100
ConvertToParametric	7-101
Edit [deprecated]	7-101
EditWithComps [model manager]	7-101
Export [model manager]	7-110
GetNames [model manager]	7-111
IsUsed [model manager]	7-111
Remove [model manager]	7-112
RemoveUnused [model manager]	7-113
Network Data Explorer Manager Script Commands	7-114
ExportFullWaveSpice	7-115
ExportNetworkData	7-116
ExportNMFData	7-118
Symbol Manager Script Commands	7-118
Add [symbol manager]	7-119
BringToFront [symbol manager]	7-127
Edit [deprecated]	7-127

EditSymbolAndUpdateComps [symbol manager]	7-128
Export [symbol manager]	7-138
GetNames [symbol manager]	7-139
IsUsed [symbol manager]	7-139
Remove [symbol manager]	7-139
RemoveUnused [symbol manager]	7-141
Add [footprint manager]	7-141
Edit [footprint manager]	7-156
EditWithComps [footprint manager]	7-156
Export [footprint manager]	7-171
GetNames [footprint manager]	7-172
IsUsed [footprint manager]	7-172
Remove [footprint manager]	7-172
RemoveUnused [footprint manager]	7-173
Add [padstack manager]	7-174
Edit [padstack manager]	7-179
EditWithComps [padstack manager]	7-184
Export [padstack manager]	7-190
GetNames [padstack manager]	7-191
IsUsed [padstack manager]	7-191
Remove [padstack manager]	7-191
RemoveUnused [padstack manager]	7-192
Script and Library Scripts	7-193
AddScript	7-193
EditScript	7-194
ExportScript	7-195
RemoveScript	7-196
ModifyLibraries	7-196
8 - Property Script Commands	8-1
Object Script Property Function Summary	8-3

Object Path	8-3
Property Object	8-3
Project Object	8-5
Design Object	8-6
3D Modeler Object	8-6
Variable Object	8-7
Optimetrics Module Object:	8-8
Optimetrics Setup Object	8-9
ReportSetup(Results) Module Object:	8-9
ReportSetup(Results) Module Child Objects:	8-10
Radiation Module Object:	8-11
Radiation Module Child Objects:	8-11
Conventions Used in this Chapter	8-12
GetArrayVariables	8-22
GetProperties	8-23
GetPropertyValue	8-24
GetVariables	8-25
GetVariableValue	8-26
SetPropertyValue	8-26
SetVariableValue	8-28
9 - Design Object Script Commands	9-1
ApplyMeshOps	9-4
Analyze	9-4
AnalyzeAll (Maxwell menu)	9-5
ConfigureFluentConductivityCoupling	9-6
ConstructVariationString	9-6
CreateParametricCircuit	9-7
Create3DDesign [Maxwell]	9-8
DeleteFieldVariation	9-9
DeleteFullVariation	9-10

DeleteLinkedDataVariation	9-10
EditNotes	9-11
EnableHarmonicForceCalculation (Transient Only)	9-11
ExportConvergence	9-20
ExportElementBasedHarmonicForce (Transient only)	9-20
ExportHarmonicTransientForce	9-21
ExportMeshStats	9-22
ExportProfile	9-23
GetDesignValidationInfo	9-24
GetGeometryIdsForAllNetLayerCombinations	9-25
GetGeometryIdsForNetLayerCombinations	9-26
GetGeometryMode	9-27
GetManagedFilePath	9-28
GetModule	9-28
GetName	9-29
GetNominalVariation	9-30
GetNoteText	9-30
GetPostProcessingVariables	9-31
GetSelections [Design]	9-31
GetSolutionType	9-31
GetVariationVariableValue	9-32
Is2D	9-32
Is3D	9-33
Redo [Design]	9-33
RenameDesignInstance	9-34
ResetToTimeZero	9-35
SetActiveEditor	9-35
SetConductivityThreshold	9-36
SetDesignSettings [Maxwell]	9-36
SetDesignFlow	9-44

SetFastTransformationForLayoutComponent	9-45
SetObjectAttributesForLayoutComponent	9-45
SetObjectDeformation	9-48
SetObjectTemperature	9-51
SetShowAllLayoutComponents	9-52
SetShowLayoutForLayoutComponent	9-53
SetSolutionType (Maxwell)	9-53
SimulateLink	9-54
Solve	9-55
RunToolkit	9-56
Undo [Design]	9-57
ValidateDesign	9-57
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
Create3DComponent	10-6
CreateBondwire	10-11
CreateBox	10-14
CreateCircle	10-16
CreateCone	10-18
CreateCutplane	10-20
CreateCylinder	10-22
CreateEllipse	10-24
CreateEquationCurve	10-26
CreateEquationSurface	10-29
CreateHelix	10-31
CreatePoint	10-33
CreateUserDefinedPart	10-34

CreatePolyline	10-41
CreateRectangle	10-45
CreateRegion	10-47
CreateRegularPolyhedron	10-50
CreateRegularPolygon	10-52
CreateSphere	10-55
CreateSpiral	10-57
CreateTorus	10-58
CreateUserDefinedModel	10-60
Python Example of Creating a UDM from a Discovery Model	10-66
CreateUserDefinedPart	10-73
Edit3DComponent	10-80
[Beta] EditNativeComponentDefinition [Maxwell]	10-81
EditPolyline	10-84
Get3DComponentParameters	10-88
Get3DComponentDefinitionNames	10-88
Get3DComponentInstanceNames	10-89
Get3DComponentMaterialNames	10-89
Get3DComponentMaterialProperties	10-90
GetUserDefinedPartVersion	10-90
Insert3DComponent	10-91
InsertNativeComponent [Layout Component to Maxwell with CS]	10-93
InsertPolylineSegment	10-97
SweepAlongPath	10-98
SweepAlongVector	10-100
SweepAroundAxis	10-101
SweepFacesAlongNormal	10-103
SweepFacesAlongNormalWithAttributes	10-104
UpdateComponentDefinition	10-106
Edit Menu Commands	10-107

Copy	10-108
DeletePolylinePoint	10-108
DuplicateAlongLine	10-109
DuplicateAroundAxis	10-111
DuplicateMirror	10-113
Mirror	10-115
Move	10-117
OffsetFaces	10-118
Paste (Model Editor)	10-119
Rotate	10-120
Scale	10-121
Modeler Menu Commands	10-122
AssignMaterial	10-125
Chamfer	10-126
Connect	10-128
CoverLines	10-129
CoverSurfaces	10-129
CreateEntityList	10-130
CreateFaceCS	10-131
CreateGroup	10-135
CreateNamedSelection	10-137
CreateObjectCS	10-138
CreateObjectFromEdges	10-144
CreateObjectFromFaces	10-145
CreateRelativeCS	10-146
DeleteEmptyGroups	10-148
DeleteLastOperation	10-149
DetachFaces	10-149
EditEntityList	10-151
EditFaceCS	10-152

EditNamedSelection	10-156
EditObjectCS	10-157
EditRelativeCS	10-163
Export	10-165
ExportModelImageToFile	10-167
Fillet	10-170
FlattenGroup	10-171
GenerateHistory	10-172
HealObject	10-173
GetActiveCoordinateSystem	10-176
GetActiveCoordinateSystemTransform	10-177
GetCoordinateSystems	10-177
Import	10-178
ImportDXF [Modeler]	10-180
ImportGDSII [Modeler]	10-184
Intersect	10-187
MoveCSToEnd	10-188
MoveEntityToGroup	10-188
MoveFaces	10-189
ProjectSheet	10-191
PurgeHistory	10-192
Section	10-193
SeparateBody	10-194
SetModelUnits	10-195
SetWCS	10-196
ShowWindow	10-197
Split	10-197
Subtract	10-199
SweepFacesAlongNormal	10-200
ThickenSheet	10-202

UncoverFaces	10-203
Unite	10-204
Ungroup	10-205
WrapSheet	10-206
Other oEditor Commands	10-207
AddDefinitionFromBlock	10-209
AddDefinitionFromLibFile	10-214
AddViewOrientation	10-218
BreakUDMConnection	10-223
ChangeProperty	10-224
Delete	10-231
GetBodyNamesByPosition	10-232
GetEdgeByPosition	10-233
GetEdgeIDsFromFace	10-234
GetEdgeIDsFromObject	10-235
GetEntityIDsContainedByNamedSelection	10-235
GetEntityListIDByName	10-236
GetFaceArea	10-236
GetFaceCenter	10-237
GetFaceByPosition	10-237
GetFaceIDs	10-239
GetGeometryModelerMode	10-239
GetModelBoundingBox	10-240
GetObjectBoundingBox	10-240
GetObjectIDByName	10-241
GetObjectName	10-241
GetObjectNameByFaceID	10-242
GetObjectsByMaterial	10-242
GetObjectsInGroup	10-243
GetMatchedObjectName	10-244

GetModelUnits	10-244
GetNumObjects	10-245
GetSelections [Model Editor]	10-245
GetUserPosition	10-246
GetVertexIDsFromEdge	10-247
GetVertexIDsFromFace	10-247
GetVertexIDsFromObject	10-248
GetVertexPosition	10-248
PageSetup	10-249
RenamePart	10-250
11 - Output Variable Script Commands	11-1
CreateOutputVariable (Maxwell)	11-1
DeleteOutputVariable	11-2
DoesOutputVariableExist	11-3
EditOutputVariable	11-4
GetOutputVariableValue	11-5
12 - Reporter Editor Script Commands	12-1
AddCartesianLimitLine	12-4
AddCartesianXMarker	12-5
AddCartesianYMarker	12-6
AddDeltaMarker	12-7
AddMarker	12-8
AddNote	12-8
AddTraceCharacteristics	12-10
AddTraces	12-11
ChangeProperty[ReportSetup]	12-13
ClearAllMarkers	12-21
ClearAllTraceCharacteristics	12-22
CopyTracesData	12-23
CopyReportsData	12-23

CopyReportDefinitions	12-24
CopyTraceDefinitions	12-24
CreateReportFromFile	12-25
CreateReport (Maxwell)	12-26
CreateReportFromTemplate	12-32
CreateReportOfAllQuantities	12-33
DeleteMarker	12-34
DeleteAllReports	12-34
DeleteReports	12-35
DeleteTraces	12-35
ExportImageToFile [Reporter]	12-36
ExportPlotImageToFile	12-41
ExportReport	12-42
ExportToFile	12-43
ExportToFile [Reporter]	12-44
ExportMarkerTable	12-45
FFTONReport	12-45
GetAllReportNames	12-46
GetAllCategories	12-47
GetAllQuantities	12-47
GetAvailableDisplayTypes	12-48
GetAvailableReportTypes	12-49
GetAvailableSolutions	12-49
GetDataUnits	12-50
GetDesignVariableNames	12-51
GetDesignVariableUnits	12-51
GetDesignVariableValue	12-52
GetDesignVariationKey	12-53
GetDisplayType (Maxwell)	12-54
GetImagDataValues	12-54

GetPerQuantityPrimarySweepValues	12-55
GetPropertyValue	12-56
GetRealDataValues	12-58
GetReportTraceNames	12-58
GetSolutionContexts	12-59
GetSolutionDataPerVariation	12-60
GetSweepNames	12-61
GetSweepUnits	12-62
GetSweepValues	12-63
GroupPlotCurvesByGroupingStrategy	12-64
ImportIntoReport	12-64
IsDataComplex	12-65
IsPerQuantityPrimarySweep	12-66
MovePlotCurvesToGroup	12-67
MovePlotCurvesToNewGroup	12-67
PasteReports	12-68
PasteTraces	12-69
Release Data	12-69
RenameReport	12-70
RenameTrace	12-70
ResetPlotSettings	12-71
SavePlotSettingsAsDefault	12-72
UpdateAllReports	12-72
UpdateReports	12-73
UpdateTraces	12-73
UpdateTracesContextAndSweeps	12-75
UnGroupPlotCurvesInGroup	12-77
13 - Boundary and Excitation Module Script Commands in Maxwell	13-1
General Commands Recognized by the Boundary/Excitations Module	13-2
AddAssignmentToBoundary	13-3

DeleteAllBoundaries	13-4
DeleteAllExcitations	13-4
DeleteBoundaries	13-5
GetBoundaryAssignment	13-5
GetBoundaries	13-6
GetBoundariesOfType	13-7
GetCoreLossEffect	13-7
GetCoreLossEffectOnField	13-8
GetDisplacementCurrent	13-8
GetEddyEffect	13-8
GetExcitations	13-9
GetExcitationsOfType	13-9
GetNumBoundaries	13-10
GetNumBoundariesOfType	13-11
GetNumExcitations	13-11
GetNumExcitationsOfType	13-12
ReassignBoundary	13-12
RemoveAssignmentFromBoundary	13-13
RenameBoundary	13-14
ReprioritizeBoundaries	13-15
Script Commands for Creating and Modifying Boundaries in Maxwell	13-16
AssignCylindricalHField	13-17
AssignDependent (2D)	13-18
AssignDependent (3D)	13-19
AssignFluxTangential	13-21
AssignImpedance (Maxwell)	13-21
AssignIndependent (Maxwell)	13-22
AssignIntegratedZeroTangentialHField	13-24
AssignInsulating	13-25
AssignRadiation (Maxwell)	13-25

AssignResistiveSheet	13-26
For 3D Magnetostatic, AC Magnetic, and Transient designs	13-27
AssignSymmetry (Maxwell)	13-27
AssignTangentialHField	13-28
AssignThinLayer	13-29
AssignTouching	13-32
AssignVectorPotential	13-33
AssignIntegratedZeroTangentialHField	13-34
EditCylindricalHField	13-34
EditDependent (Maxwell)	13-35
EditFluxTangential	13-35
EditImpedance (Maxwell)	13-36
EditIndependent (Maxwell)	13-36
EditIntegratedZeroTangentialHField	13-37
EditInsulating	13-37
EditRadiation (Maxwell)	13-38
EditResistiveSheet	13-38
EditSymmetry	13-40
EditTangentialHField	13-40
EditThinLayer	13-41
EditTouching	13-43
EditVectorPotential	13-44
EditZeroTangentialHField	13-45
Script Commands for Creating and Modifying Excitations in Maxwell	13-45
AddTerminalsToWinding	13-47
AssignCharge	13-47
AssignCoilGroup	13-48
AssignCoilTerminal	13-49
AssignCoilTerminalGroup	13-49
AssignCurrent	13-50

AssignCurrentDensity	13-52
AssignCurrentDensityGroup	13-52
AssignCurrentDensityTerminal	13-53
AssignCurrentDensityTerminalGroup	13-53
AssignCurrentGroup	13-53
AssignFloating	13-54
AssignSink (Maxwell)	13-54
AssignVoltage (Maxwell)	13-55
AssignVoltageAPhi	13-56
AssignVoltageDrop	13-57
AssignVoltageDropGroup	13-58
AssignVoltageGroup	13-58
AssignVolumeChargeDensity	13-58
AssignWindingGroup	13-59
EditCharge	13-60
EditCoilTerminal	13-61
EditCurrent	13-61
EditCurrentDensity	13-62
EditCurrentDensityTerminal	13-63
EditExternalCircuit	13-63
EditExternalSimulinkCircuit	13-63
EditFloating	13-64
EditSink	13-64
EditVoltage	13-64
EditVoltageAPhi	13-66
EditVoltageDrop	13-67
EditVolumeChargeDensity	13-67
EditWindingGroup	13-67
FFTONMultiExcitationsDataset	13-69
SetCoreLoss	13-70

SetEddyEffect	13-71
SetMinimumTimeStep	13-73
SetupPWMExcitation	13-74
SetupYConnection	13-76
SetPermeabilityComputation	13-78
14 - Mesh Operations Module Script Commands	14-1
General Commands Recognized by the Mesh Operations Module	14-2
DeleteOp	14-2
GetOperationNames	14-2
RenameOp	14-3
Script Commands for Creating and Modifying Mesh Operations	14-4
AssignCylindricalGapOp	14-4
AssignDensityControlOp	14-7
AssignEdgeCutLayerOp	14-9
AssignLengthOp	14-10
AssignModelResolutionOp	14-12
AssignSkinDepthLayerSetting	14-13
AssignSkinDepthOp	14-14
AssignTrueSurfOp	14-16
EditCylindricalGapOp	14-17
EditDensityControlOp	14-19
EditEdgeCutLayerOp	14-21
EditLengthOp	14-22
EditModelResolutionOp	14-24
EditSkinDepthOp	14-25
EditTrueSurfOp	14-26
InitialMeshSettings	14-28
15 - Analysis Setup Module Script Commands	15-1
ClearLinkedData (Module)	15-2
CopySetup	15-2

DeleteSetups	15-3
EditSetup [Maxwell]	15-4
ExportCircuit	15-10
ExportIcepak	15-14
ExportSolnData	15-15
GetSetupCount	15-16
GetSetups	15-17
GetSweepCount	15-17
GetSweeps	15-18
InsertSetup [Maxwell]	15-18
PasteSetup	15-31
PasteSweep	15-31
RenameSetup	15-32
ResetAllToTimeZero	15-32
ResetSetupToTimeZero	15-33
RevertAllToInitial	15-33
RevertAllToInitialTemperature	15-34
RevertSetupToInitial	15-34
RevertSetupToInitialTemperature	15-35
16 - Optimetrics Module Script Commands	16-1
General Commands Recognized by the Optimetrics Module	16-6
CopySetup	16-7
DeleteSetups [Optimetrics]	16-7
DistributedAnalyzeSetup	16-8
ExportDXConfigFile	16-9
ExportOptimetricsProfile	16-9
ExportOptimetricsResult	16-10
ExportParametricResults	16-11
GetSetupNames [Optimetrics]	16-12
GetSetupNamesByType [Optimetrics]	16-12

ImportSetup	16-13
PasteSetup [Optimetrics]	16-14
RenameSetup [Optimetrics]	16-14
SolveSetup [Optimetrics]	16-15
SolveAllSetup	16-16
Parametric Script Commands	16-16
EditSetup [Parametric]	16-17
ExportParametricSetupTable	16-17
GenerateVariationData [Parametric]	16-18
InsertSetup [Parametric]	16-18
Optimization Script Commands	16-24
EditSetup [Optimization]	16-24
InsertSetup [Optimization]	16-29
InsertSetup for System Coupling	16-35
InitializeSystemCoupling	16-38
Sensitivity Script Commands	16-38
EditSetup [Sensitivity]	16-38
InsertSetup [Sensitivity]	16-42
Statistical Script Commands	16-45
EditSetup [Statistical]	16-45
InsertSetup [Statistical]	16-45
17 - Solutions Module Script Commands	17-1
ExportLayoutLayerGridField	17-2
ExportLayoutViaCurrent	17-4
ExportLayoutViaCurrentDensity	17-5
ExportLayoutLayerInducedCurrent	17-6
DeleteSolutionVariation	17-7
ExportNetworkData	17-8
ExportTransientData	17-10
FFTONReport	17-11

GetAdaptiveSettings	17-12
GetAllSourceMagnitudes	17-12
GetAllSourceModes	17-13
GetAllSourcePhases	17-13
GetAllSources	17-14
GetAntennaParameters	17-14
GetAvailableVariations	17-15
GetFieldType	17-16
GetIncludePortPostProcessing	17-16
GetLayerAndNetOfMeshBody	17-17
GetMeshBodiesOnLayerAndNet	17-19
GetMultipactionBreakdown	17-21
GetNetworkDataSolution	17-22
GetNetworkDataSolutionDefinition	17-23
GetNetworkPostprocSetup	17-23
GetTerminalExcitationType	17-24
GetTransientSolveTimes	17-24
GetValidISolutionList	17-25
TDROnReport	17-25
18 - Field Overlays Module Script Commands	18-1
AddMarkerToPlot	18-2
AddMarker[Fields Reporter]	18-3
AddNamedExpression	18-3
CalcOp	18-5
CalcStack	18-5
CalculatorRead	18-6
CalculatorWrite	18-7
ChangeProperty	18-10
ChangeGeomSettings	18-14
ClcEval	18-14

ClcMaterial	18-17
ClcMaterialValue	18-18
ClearAllMarkers[Fields Reporter]	18-18
ClearAllNamedExpr	18-19
CopyNamedExprToStack	18-19
CreateFieldPlot	18-20
Maxwell Field Line Trace Plot Examples	18-35
DeleteFieldPlot	18-37
DeleteMarker[Fields Reporter]	18-37
DeleteNamedExpr	18-38
DeleteUneditablePlot	18-38
DoesNamedExpressionExists	18-39
EditSurfaceMeshSummaryData	18-39
EnterComplex	18-43
EnterComplexVector	18-43
EnterCoord	18-44
EnterEdge	18-45
EnterLine	18-45
EnterOutputVar	18-46
EnterPoint	18-46
EnterQty	18-47
EnterScalar	18-48
EnterScalarFunc	18-48
EnterSurf	18-49
EnterVector	18-49
EnterVectorFunc	18-50
EnterVol	18-50
ExportFieldPlot	18-51
ExportMarkerTable	18-52
ExportOnGrid [Field Overlays]	18-52

ExportPlotImageToFile	18-54
ExportPlotImageWithViewToFile [Reporter]	18-55
ExportSurfaceMeshSummary	18-56
ExportToFile	18-57
GetFieldFolderNames	18-58
GetFieldPlotNames	18-58
GetFieldPlotQuantityName	18-59
GetMeshPlotNames	18-59
ModifyFieldPlot	18-60
GetTopEntryValue	18-62
HidePolarPlot	18-62
HideRadiatedPlotOverlay	18-63
LoadNamedExpressions	18-63
ModifyFieldPlot	18-64
ModifyInceptionParameters	18-66
ReassignFieldPlot	18-67
RenameFieldPlot	18-69
RenamePlotFolder	18-70
SaveFieldsPlots	18-70
SetFieldPlotSettings	18-71
SaveNamedExpressions	18-73
SetFieldPlotSettings	18-73
SetPlotFolderSettings	18-75
SetPlotsViewSolutionContext	18-79
UpdateAllFieldsPlots	18-80
UpdateQuantityFieldsPlots	18-81
19 - Fields Calculator Script Commands	19-1
AddNamedExpression	19-2
Syntax for Algebraic Option	19-4
CalcOp	19-5

CalcRead(deprecated)	19-6
CalculatorRead	19-6
CalcStack	19-7
CalculatorWrite	19-8
ChangeGeomSettings	19-11
ClcEval	19-12
ClcMaterial	19-14
ClearAllNamedExpr	19-15
CopyNamedExprToStack	19-16
DeleteNamedExpr	19-16
EnterComplex	19-17
EnterComplexVector	19-17
EnterLine	19-18
EnterPoint	19-19
EnterQty	19-19
EnterScalar	19-20
EnterScalarFunc	19-20
EnterSurf	19-21
EnterVector	19-21
EnterVectorFunc	19-22
EnterVol	19-23
ExportOnGrid [Fields Calculator]	19-23
ExportToFile [Fields Calculator]	19-25
GetTopEntryValue	19-27
LoadNamedExpressions	19-27
SaveNamedExpressions	19-28
20 - Motion Setup Script Commands	A
DeleteMotionSetup	A
ReassignMoving	B
Commands to Create and Edit the Band	B

AssignBand	B
EditMotionSetup	E
GetMotionSetupNames	F
Other Commands Recognized By the ModelSetup Module	G
GetSymmetryMultiplier	G
SetSymmetryMultiplier	G
21 - Parameter Setup Script Commands	21-1
General Parameter Setup Script Commands	21-1
Commands to Create and Edit Parameters	21-2
AddReduceOp	21-2
AssignForce	21-3
[Beta] AssignLayoutForce	21-4
AssignMatrix	21-6
AssignTorque	21-9
DeleteAllParameters	21-11
DeleteParameters	21-11
DeleteReduceMatrix	21-11
DeleteReduceOp	21-12
EditForce	21-13
[Beta] EditLayoutForce	21-13
EditMatrix	21-15
EditReduceOp	21-15
EditTorque	21-16
ReassignParameter	21-16
RenameParameter	21-17
RenameReduceMatrix	21-17
RenameReduceOp	21-18
22 - User Defined Document Script Commands	22-1
AddDocument	22-1
DeleteAllDocuments	22-2

DeleteDocument	22-3
EditDocument	22-4
GetDocumentDefinitionNames	22-5
GetDocumentNames	22-5
RenameDocument	22-6
SaveHtmlDocumentAs	22-7
SavePdfDocumentAs	22-7
UpdateAllDocuments	22-8
UpdateDocument	22-8
ViewHtmlDocument	22-9
ViewPdfDocument	22-10
Example Python Script: Defining a Document	22-10
23 - User Defined Solutions Commands	23-1
CreateUserDefinedSolution	23-1
DeleteUserDefinedSolutions	23-2
EditUserDefinedSolution	23-3
24 - Network Data Explorer Script Commands	24-1
AddDiffPair	24-3
Cascade (SPISim)	24-4
ClearDiffPairs	24-5
Clone	24-6
Close	24-7
Combine (SPISim)	24-8
Deembed (SPISim)	24-9
DeembedBack (SPISim)	24-10
DeembedFront (SPISim)	24-11
DisableDiffPairs	24-12
EnableDiffPairs	24-13
ExportCitiFile	24-13
ExportMatlab	24-15

ExportSpreadsheet	24-17
ExportTouchstone	24-19
ExportTouchstone2	24-21
Extract (SPISim)	24-23
GetFrequencies	24-24
GetFrequencyCount	24-25
GetName	24-25
GetPortCount	24-26
GetPortNumber	24-27
GetPostProcSettings	24-28
GetVariation	24-29
HasSameData	24-29
LoadSolution	24-31
Open	24-32
Rename (SPISim)	24-33
Renormalize (SPISim)	24-34
Reorder	24-35
Reorder (SPISim)	24-36
Reset	24-37
SetAllPortImpedances	24-38
SetPortDeembedDistance	24-39
SetPortImpedance	24-39
SetPostProcSettings	24-40
Smooth	24-41
Stretch (SPISim)	24-43
Terminate	24-44
25 - ComInstance Script Commands	25-1
Callback Scripting Using ComInstance Object	25-1
ComInstance Functions	25-2
GetComponentName	25-2

GetInstanceID [Component Instance]	25-3
GetInstanceName [Component Instance]	25-3
GetParentDesign	25-4
GetPropHost	25-4
GetPropServerName	25-5
26 - Schematic Scripting	26-1
Method Format	26-1
Editor Scripting IDs	26-4
Format of IDs for different schematic objects:	26-4
Format for Components	26-4
Create Method List	26-5
CreateArc (Schematic Editor)	26-7
CreateCircle (Schematic Editor)	26-9
CreateComponent (Schematic Editor)	26-11
CreateGlobalPort (Schematic Editor)	26-13
CreateGround (Schematic Editor)	26-15
CreateLine (Schematic Editor)	26-17
CreatePagePort (Schematic Editor)	26-18
CreateIPort (Schematic Editor)	26-20
CreatePolygon (Schematic Editor)	26-22
CreateRectangle (Schematic Editor)	26-24
CreateText (Schematic Editor)	26-26
CreateWire (Schematic Editor)	26-29
General Method List	26-31
Activate (Schematic Editor)	26-33
AddPinGrounds (Schematic Editor)	26-34
AddPinIPorts (Schematic Editor)	26-37
AddPinPageConnectors (Schematic Editor)	26-37
AlignHorizontal (Schematic Editor)	26-38
AlignVertical (Schematic Editor)	26-39

BringToFront (Schematic Editor)	26-40
CloseEditor (Schematic Editor)	26-41
Copy (Schematic Editor)	26-42
CopyData [Schematic Editor]	26-43
CopySubdesign [Schematic Editor]	26-43
CreatePage (Schematic Editor)	26-44
Cut (Schematic Editor)	26-45
DeactivateOpen (Schematic Editor)	26-46
DeactivateShort (Schematic Editor)	26-47
Delete (Schematic Editor)	26-47
DeletePage (Schematic Editor)	26-48
ElectricRuleCheck (Schematic Editor)	26-49
ExportImage (Schematic Editor)	26-50
ExportNetlist	26-51
FindElements (Schematic Editor)	26-52
FitToBorder [Schematic Editor]	26-54
GridSetup (Schematic Editor)	26-54
FlipHorizontal (Schematic Editor)	26-56
FlipVertical (Schematic Editor)	26-57
Move (Schematic Editor)	26-58
NameNets (Schematic Editor)	26-59
PageBorders (Schematic Editor)	26-61
Pan (Schematic Editor)	26-61
Paste (Schematic Editor)	26-62
PasteData [Schematic Editor]	26-63
PasteDesign (Schematic Editor)	26-64
PushExcitations	26-65
Rotate (Schematic Editor)	26-67
SelectAll (Schematic Editor)	26-69
SelectPage (Schematic Editor)	26-70

SetPageData [Schematic Editor]	26-70
SendToBack	26-73
ShowVariableBlock (Schematic Editor)	26-73
SortComponents (Schematic Editor)	26-74
Wire (Schematic Editor)	26-75
ZoomArea (Schematic Editor)	26-76
ZoomIn (Schematic Editor)	26-77
ZoomOut (Schematic Editor)	26-77
ZoomPrevious (Schematic Editor)	26-78
ZoomToFit (Schematic Editor)	26-79
Property Method List	26-79
ChangeProperty (Schematic Editor)	26-80
GetEvaluatedPropertyValue (Schematic Editor)	26-80
GetProperties (Schematic Editor)	26-81
GetPropertyAttribute [Schematic Editor]	26-82
GetPropertyValue (Schematic Editor)	26-83
SetPropertyValue (Schematic Editor)	26-84
Information Method List	26-85
GetAllPorts	26-85
GetCompInstanceFromRefDes (Schematic Editor)	26-86
GetComponentInfo (Schematic Editor)	26-87
GetComponentPins (Schematic Editor)	26-88
GetComponentPinInfo (Schematic Editor)	26-89
GetComponentPinLocation [Schematic Editor]	26-90
GetEditorName (Schematic Editor)	26-90
GetNetConnections (Schematic Editor)	26-91
GetNumPages [Schematic Editor]	26-92
GetPortInfo (Schematic Editor)	26-93
GetSelections (Schematic Editor)	26-94
GetSignals (Schematic Editor)	26-94

GetWireConnections (Schematic Editor)	26-96
GetWireInfo (Schematic Editor)	26-97
GetWireSegments (Schematic Editor)	26-99
27 - Definition Editor Script Commands	27-1
Symbol Editor Scripts	27-1
AddLevel (Symbol Editor)	27-3
AlignHorizontal (Symbol Editor)	27-4
AlignVertical (Symbol Editor)	27-5
BringToFront (Symbol Editor)	27-5
ChangeProperty (Symbol Editor)	27-6
CloseEditor (Symbol Editor)	27-16
Copy (Symbol Editor)	27-16
CreateArc (Symbol Editor)	27-17
CreateCircle (Symbol Editor)	27-17
CreateImage (Symbol Editor)	27-18
CreateLine (Symbol Editor)	27-19
CreatePin (Symbol Editor)	27-19
CreatePolygon (Symbol Editor)	27-20
CreateRectangle (Symbol Editor)	27-21
CreateText (Symbol Editor)	27-21
Cut (Symbol Editor)	27-22
Delete (Symbol Editor)	27-22
DisplayPinNames (Symbol Editor)	27-23
ExportFile (Symbol Editor)	27-23
FlipHorizontal (Symbol Editor)	27-24
FlipVertical (Symbol Editor)	27-24
GetProperties (Symbol Editor)	27-25
GetPropertyValue (Symbol Editor)	27-25
GetVisibleLevels (Symbol Editor)	27-25
GridSetup (Symbol Editor)	27-26

Move (Symbol Editor)	27-27
Pan (Symbol Editor)	27-27
Paste (Symbol Editor)	27-28
Redo (Symbol Editor)	27-29
RemoveLevels (Symbol Editor)	27-29
RemovePort (Symbol Editor)	27-29
Rotate (Symbol Editor)	27-30
Save (Symbol Editor)	27-30
SelectAll (Symbol Editor)	27-31
SendToBack (Symbol Editor)	27-31
SetActiveLevel (Symbol Editor)	27-32
SetInitialLevels (Symbol Editor)	27-33
SetPropertyValue (Symbol Editor)	27-33
SetVisibleLevels (Symbol Editor)	27-34
ToggleLevel (Symbol Editor)	27-34
Undo (Symbol Editor)	27-35
ZoomArea (Symbol Editor)	27-35
ZoomIn (Symbol Editor)	27-37
ZoomOut (Symbol Editor)	27-37
ZoomPrevious (Symbol Editor)	27-38
ZoomToFit (Symbol Editor)	27-38
28 - Core Global Script Context Commands	28-1
AddErrorMessage	28-1
AddFatalMessage	28-2
AddInfoMessage	28-2
AddWarningMessage	28-3
LogDebug	28-3
LogError	28-4
29 - Example Scripts	29-1
IronPython Example Scripts	29-1

BH Coordinates Python Script 29-1

 Script Contents 29-2

 Posco_BH_Curve.tab Contents 29-3

Equation Based Curve Python Script 29-6

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 Python, and can run external scripts written in Python. Additionally, it contains a Python 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 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]

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>** – a list contained in square brackets.
- **<value>** – can be an integer, string, or other variable, depending on context.

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.

Python Compatibility

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

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

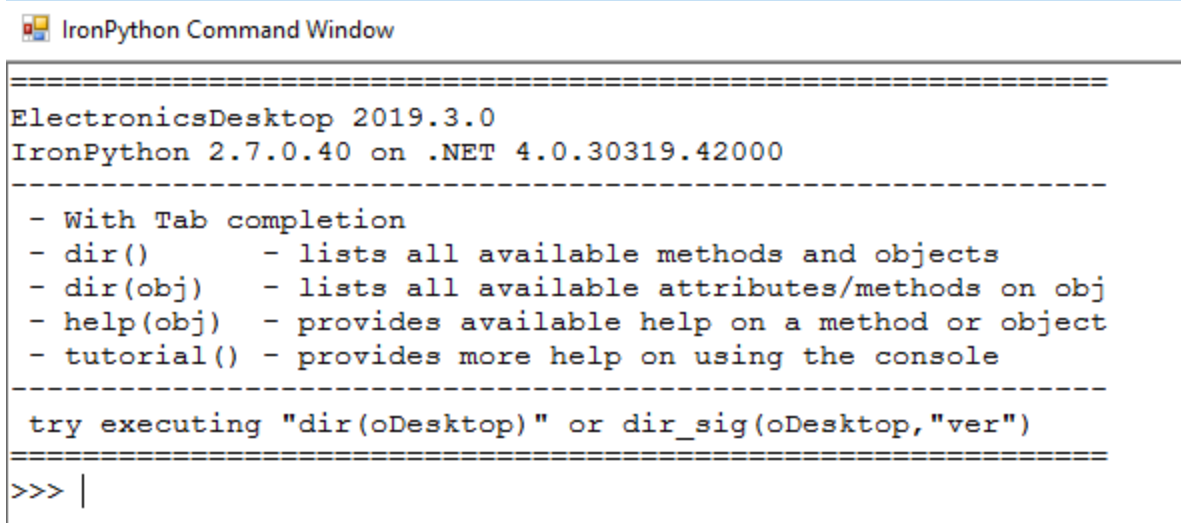
Advantages of IronPython

The advantages that IronPython use provides are significant:

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

- IronPython, in addition, has access to the entire .NET eco system. This allows us, for instance, to create a modern GUI using the **System.Windows.Forms** assembly from IronPython code and call any other .NET assembly for that matter.
- The use of IronPython's technologies enables the ability to interactively script Desktop (feature in development).
- 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")
=====
>>> |
  
```

[Scripting Using Iron Python](#)

[Standalone IronPython and Desktop IronPython](#)

[IronPython Examples](#)

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

Scripting Using Iron Python

The following topics detail scripting using Iron Python:

[IronPython Script Execution Environment](#)

[Scripting with IronPython](#)

[Standalone IronPython and Desktop IronPython](#)

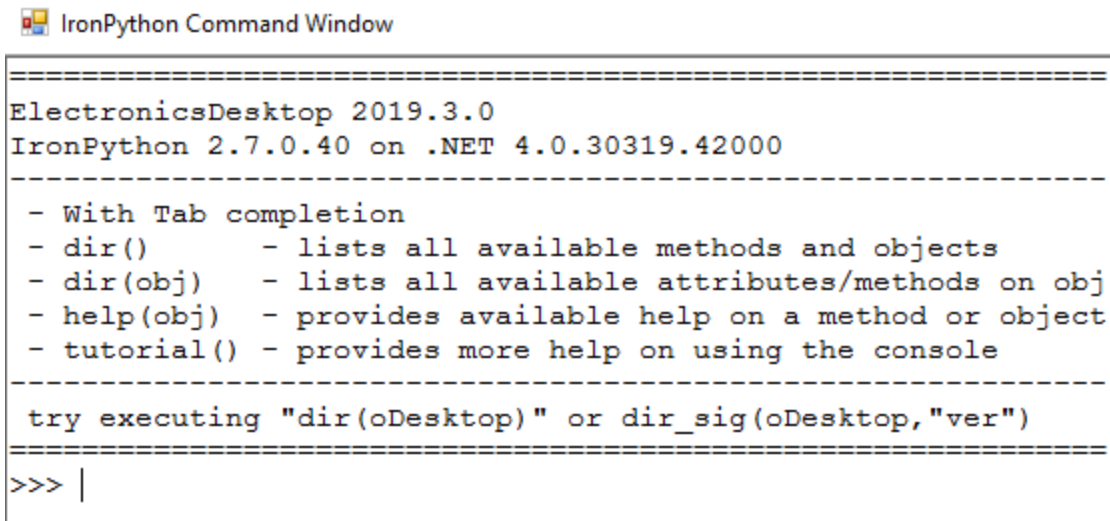
[Introduction to IronPython](#)

[Appendix: IronPython Samples](#)

IronPython Script Execution Environment

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

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



```
IronPython Command Window

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

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

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

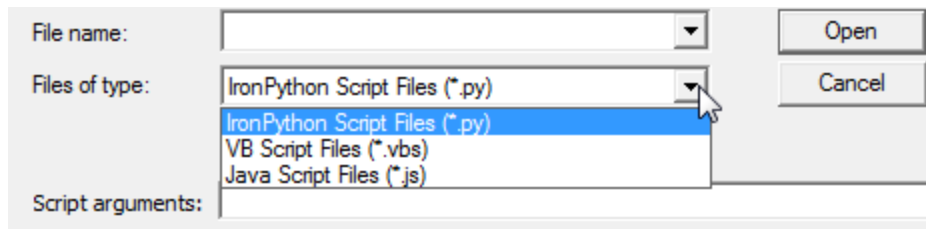


Figure 1: Run Script dialog and script arguments

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

While Ansys Electronics Desktop no longer supports VBScript or JavaScript, users may have a significant collection of VBScript or JavaScript assets. These existing script files can be executed via Python. Various **Run<*>Command** methods have been designed for this purpose.

For example, a user can create a parameterized cone in HFSS by executing the following Python 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, "")
```

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, but 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 Python to directly invoke the script objects.

Related Topics

[IronPython Script Execution Environment](#)

Scripting with Python

Access to application scripting objects is provided via the predefined **oDesktop** object.

Note the following:

- Any argument is supplied via the built in **ScriptArgument** variable.
- The **oDesktop** object is always available.
- Method calls 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 Python array or dictionary syntax.

Related Topics

[IronPython Script Execution Environment](#)

[Standalone IronPython and Desktop IronPython](#)

Standalone IronPython

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

Running Standalone IronPython

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

It is located in:

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

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

```
"C:\Program Files\ANSYS Inc\v252\AnsysEM\common\IronPython\ipy64.exe"  
"<filePath>\myScript.py"
```

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

Using a Recorded Script

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

Creating an External Script

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

- `import sys`

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

 `# Adds the Electronics Desktop installation path to the list of directories Python searches for modules and files.`

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

You must end the script with:

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

Example Script

```
import sys

sys.path.append(r"C:\Program Files\ANSYS Inc\v252\AnsysEM")

sys.path.append(r"C:\Program Files\ANSYS Inc\v252\An-
sysEM\PythonFiles\DesktopPlugin")

import ScriptEnv

ScriptEnv.Initialize("Ansoft.ElectronicsDesktop")

oDesktop.RestoreWindow()

oProject = oDesktop.NewProject()

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

oDesign = oProject.SetActiveDesign("HFSSDesign1")

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

oEditor.CreateRectangle(

[

    "NAME:RectangleParameters",
```

```

    "IsCovered:= ", True,
    "XStart:= ", "-0.2mm",
    "YStart:= ", "-3mm",
    "ZStart:= ", "0mm",
    "Width:= ", "0.8mm",
    "Height:= ", "1.2mm",
    "WhichAxis:= ", "Z"
],
[
    "NAME:Attributes",
    "Name:= ", "Rectangle1",
    "Flags:= ", "",
    "Color:= ", "(132 132 193)",
    "Transparency:= ", 0,
    "PartCoordinateSystem:= ", "Global",
    "UDMId:= ", "",
    "MaterialValue:= ", "\"vacuum\"",
    "SolveInside:= ", True
])
oDesign.SetDesignSettings(['NAME:Design Settings Data', 'Allow Material Override:=', True, 'Calculate Lossy Dielectrics:=', True])
oEditor.SetModelUnits(['NAME:Units Parameter', 'Units:=', 'mil', 'Rescale:=', False ])
ScriptEnv.Shutdown()

```

IronPython Samples

Important:

VBScript is no longer supported in Ansys Electronics Desktop. It is referenced here only to aid users in translating existing VBScript scripts to Python.

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: ChangeProperty command to change color of a cone

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

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

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

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

```
    ]]]  
  ])
```

Sample Script: 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: 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()
```

Related Topics

[Introduction to IronPython](#)

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

Creating User Defined Primitives and User Defined Models in Python Scripts

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

Advantages Compared to C++

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

Changes compared to C

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

Structures

- Structures have the same names as in C implementation.
- Structures fields names are capitalized.
- Arrays in structures become lists in Python (Technically a .NET IList container)
- Structure instances are created using the supplied constructors and members are accessed using the provided access methods.

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

Return Values for UDM and UDP Functions

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

Constants

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

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

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

Methods

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

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

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

Output Parameters

The rule for the output parameters is as follows:

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

one output parameter; return value is ignored

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

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

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

Two output parameter; return value must be preserved

the return tuple is (returnVal, output1, output2)

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

Comparison with C function:

C	Python
<pre>bool getDefinition(UDMDefinition* udmDefinition, void* callbackData); where udmDefinition is an output</pre>	<pre>udmDefinition = udmFunctionLibrary.GetDefinition() (Note: callbackData is omitted in py interface)</pre>

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

'List Size' Parameters

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

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

Example:

input list, input list size

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

output list, output list size

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

output list, input list size

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

Comparison with C function:

C	Python
bool getAllFaces(long partId,	facelDs = udmFunc-

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

Added Parameters

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

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

Example:

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

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

Here cloneIdsSize is a new integer parameter.

Comparison with C function:

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

Developing a UDM/UDP

Creation

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

Location

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

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

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

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

Organize

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

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

For UDM:extension:

```
from UDM import *
```

For UDP:extension:

```
from UDP import *
```

Edit/Reload

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

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

UDPExtension

Import

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

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

```
from UDP import *
```

Main class: UDPExtension

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

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

The class should implement [IUDPExtension methods](#) described in the topic that follows.

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

CreatePrimitive2(funcLib, udpParams)

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

Optional Methods

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

GetPrimitiveParameters()

- returns Python list of strings or NULL

GetRegisteredFaceNames()

- returns Python list of strings or NULL

GetRegisteredEdgeNames()

- returns Python list of strings or NULL

GetRegisteredVertexNames()

- returns Python list of strings or NULL

ProjectParametersOnToValidPlane2(currentUDPParams, projectedUDPParams)

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

MapParametersDefinitionVersions2(oldVersion, oldUDPParams)

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

GetOldPrimitiveParametersDefinition2(version)

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

Example UDP

```
import sys

class UDPExtension(IUDPExtension):

    def GetLengthParameterUnits(self):
        return "mm"

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

```
    return typeInfo  
  
    ...  
  
    ...
```

UDMExtension

Import

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

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

```
from UDM import *
```

Main class: UDMExtension

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

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

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

IUDMExtension Methods

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

Mandatory Methods

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

GetInfo()

- returns UDMInfo object populated with appropriate UDM information.

IsAttachedToExternalEditor()

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

CreateInstance(funcLib)

- funcLib is UDMFunctionLibrary
- returns UDMPParameters.

GetUnits(instanceId)

- instanceId is an integer.
- returns string containing units for the instance.

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

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

- funcLib is UDMFunctionLibrary
- udmlnParams is a list of UDMParameters that comes from desktop
- updatedParams: UDM script can change the UDM parameters it receives. Updated parameters need to be sent back to desktop. If the UDM script is not going to change any of the parameters that it received, it needs to copy udmlnParams to updatedParams.
- refreshModifiedPartsOnly is a Boolean

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

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

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

returns True on success, False on failure.

ReleaseInstance(instanceId)

- instanceId is an integer.
- 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 a 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 open a dialog box 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 open a dialog box. If it is True, application will populate a dialog box with definition, options, parameters that are returned.
 - String returned contains length units for parameters.

DialogForDefinitionAndOptions(self, defData, optData) [Deprecated]

UDM can open a dialog box 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 box with the definitions and options that are returned. As no parameters are returned, no parameters are shown in this dialog box.

GetInstanceSourceInfo(instancelid)

- instancelid is an integer.
- 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 a Python list of strings 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:** vertexlds = **GetAllVertices(int: partId)**
12. **bool:** bret = **SetFaceAttribs(List_of_int: facelds, List_of_string: attribs)**
13. **bool:** bret = **SetEdgeAttribs(List_of_int: edgelds, List_of_string: attribs)**
14. **bool:** bret = **SetVertexAttribs(List_of_int: vertexlds, List_of_string: attribs)**
15. **string:** str = **GetModelerUnit()**
16. **string:** str = **GetCacheFileForUDMResume()**
17. **bool:** bret = **SetPartColor(int: partId, int: nColor)**
18. **bool:** bret = **SetPartFlags(int: partId, int: nFlags)**
19. (**bool:** bret, **UDPPosition:** low, **UDPPosition:** high) = **GetBoundingBox(int: partId, bool: exact)**
20. **bool:** bret = **IsParametricUpdate()**
21. **bool:** bret = **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:** center-Point, **List_of_double:** widthAndHeight, **int:** isCovered)
9. **int:** ID = **CreateArc(CoordinateSystemPlane:** whichPlane, **UDPPosition:** centerPoint, **UDPPosition:** startPoint, **double:** fAngle)
10. **int:** ID = **CreateCircle(CoordinateSystemPlane:** whichPlane, **UDPPosition:** center-Point, **double:** fRadius, **int:** isCovered)
11. **int:** ID = **CreateEllipse(CoordinateSystemPlane:** whichPlane, **UDPPosition:** center-Point, **double:** fMajorRadius, **double:** fRadiusRatio, **int:** isCovered)
12. **int:** ID = **CreateRegularPolygon(CoordinateSystemPlane:** whichPlane, **UDPPosition:** centerPoint, **UDPPosition:** startPoint, **int:** numOfSides, **int:** isCovered)
13. **int:** ID = **CreateEquationBasedCurve(UDPEquationBasedCurveDefinition:** curveDefinition)
14. **int:** ID = **CreateEquationBasedSurface(UDPEquationBasedSurfaceDefinition:** surfaceDefinition)
15. **int:** ID = **CreateSpiral(UDPSpiralDefinition:** spiralDefinition)
16. **int:** ID = **CreateBox(UDPPosition:** startPoint, **List_of_double:** boxXYZsize)
17. **int:** ID = **CreateSphere(UDPPosition:** centerPoint, **double:** fRadius)
18. **int:** ID = **CreateCylinder(CoordinateSystemAxis:** whichAxis, **UDPPosition:** center-Point, **double:** fRadius, **double:** fHeight)
19. **int:** ID = **CreateCone(CoordinateSystemAxis:** whichAxis, **UDPPosition:** centerPoint, **double:** fBottomRadius, **double:** fTopRadius, **double:** fHeight)
20. **int:** ID = **CreateTorus(CoordinateSystemAxis:** whichAxis, **UDPPosition:** centerPoint, **double:** fMajorRadius, **double:** fMinorRadius)

21. **int:** ID = **CreatePolyhedron**(**CoordinateSystemAxis:** whichAxis, **UDPPosition:** center-Point, **UDPPosition:** startPosition, **int:** numofSides, **double:** fHeight)
22. **int:** ID = **CreateHelix**(**UDPHelixDefinition:** helixDefinition)
23. **bool:** SI = **Unite**(**List_of_int:** pObjectIDArray)
24. **bool:** SI = **Subtract**(**List_of_int:** pBlankObjectIDArray, **List_of_int:** pToolObjectIDArray)
25. **bool:** SI = **Intersect**(**List_of_int:** pObjectIDArray)
26. **bool:** SI = **Imprint**(**List_of_int:** pBlankObjectIDArray, **List_of_int:** pToolObjectIDArray)
27. **bool:** SI = **SweepAlongVector**(**int:** profileID, **UDPVector:** sweepVector, **UDPSweepOptions:** sweepOptions)
28. **bool:** SI = **SweepAroundAxis**(**int:** profileID, **CoordinateSystemAxis:** whichAxis, **double:** sweepAngle, **UDPSweepOptions:** sweepOptions)
29. **bool:** SI = **SweepAlongPath**(**int:** profileID, **int:** pathID, **UDPSweepOptions:** sweepOptions)
30. **bool:** SI = **Translate**(**int:** partID, **UDPVector:** translateVector)
31. **bool:** SI = **Rotate**(**int:** partID, **CoordinateSystemAxis:** whichAxis, **double:** rotateAngle)
32. **bool:** SI = **Mirror**(**int:** partID, **UDPPosition:** mirrorPlaneBasePosition, **UDPVector:** mirrorPlaneNormalVector)
33. **bool:** SI = **Transform**(**int:** partID, **List_of_double:** rotationMatrix, **UDPVector:** translateVector)
34. **bool:** SI = **Scale**(**int:** partID, **double:** xScale, **double:** yScale, **double:** zScale)
35. (**bool:** SI, **List_of_int:** cloneIDs) = **DuplicateAlongLine**(**int:** partID, **UDPVector:** translateVector, **int:** numTotalObjs, **int:** cloneIDsListSize)
36. (**bool:** SI, **List_of_int:** cloneIDs) = **DuplicateAroundAxis**(**int:** partID, **CoordinateSystemAxis:** whichAxis, **double:** rotateAngle, **int:** numTotalObjs, **int:** cloneIDsListSize)
37. **int:** ID = **DuplicateAndMirror**(**int:** partID, **UDPPosition:** mirrorPlaneBasePosition, **UDPVector:** mirrorPlaneNormalVector)
38. **bool:** SI = **Connect**(**List_of_int:** objectIDArray)
39. **bool:** SI = **Offset**(**int:** partID, **double:** offsetDistance)
40. **int:** ID? = **Section**(**int:** partID, **CoordinateSystemPlane:** sectionPlane)
41. (**bool:** SI, **int:** ID) = **Split**(**int:** partID, **CoordinateSystemPlane:** splitPlane, **SplitWhichSideToKeep:** whichSideToKeep, **bool:** bSplitCrossingObjectsOnly)
42. (**bool:** SI, **List_of_int:** importedObjectIDs) = **ImportNativeBody2**(**string:** fileNameWithFullPath)
43. (**bool:** SI, **List_of_int:** importedObjectIDs) = **ImportAnsoftGeometry**(**string:** fileNameWithFullPath, **List_of_string:** overridingParamsNameArray, **List_of_UDPPParam:** overridingParamsArray)
44. **int:** ID = **Clone**(**int:** partID)
45. **bool:** SI = **DeletePart**(**int:** partID)

46. **int**: ID = **CreateObjectFromFace**(*int*: faceID)
47. **bool**: SI = **Fillet**(*UDPBLNDElements*: entitiesToFillet, *UDPBLNDFilletOptions*: filletOptions)
48. **bool**: SI = **Chamfer**(*UDPBLNDElements*: entitiesToChamfer, *UDPBLNDChamferOptions*: chamferOptions)
49. (**bool**: SI, *List_of_int*: newPartIDs) = **DetachFaces**(*int*: newPartIDArraySize, *List_of_int*: faceIDs)
50. (**bool**: SI, *List_of_int*: newPartIDs) = **DetachEdges**(*int*: newPartIDArraySize, *List_of_int*: edgeIDs)
51. **int**: ID = **CreateObjectFromEdge**(*int*: edgeID)
52. **bool**: SI = **SheetThicken**(*int*: partID, **double**: fThickness, **bool**: bThickenBothSides)
53. (**bool**: SI, *List_of_int*: newPartIDArray) = **SweepFaceAlongNormal**(*int*: newPartIDArraySize, *List_of_int*: faceIDArray, **double**: sweepLength)
54. **bool**: SI = **CoverLine**(*int*: partID)
55. **bool**: SI = **CoverSurface**(*int*: partID)
56. **bool**: SI = **UncoverFaces**(*List_of_int*: faceIDArray)
57. (**bool**: SI, *int*: numPartsCreated, *List_of_int*: faceIDArray) = **SeparateBodies**(*int*: partID, *int*: numPartsCreated)
58. **bool**: SI = **MoveFaces**(*List_of_int*: faceIDArray, **bool**: bMoveAlongNormal, **double**: fOffsetDistance, *UDPVector*: moveVector)
59. **bool**: SI = **WrapSheet**(*int*: sheetBodyID, *int*: targetBodyID)
60. **bool**: SI = **ImprintProjection**(*int*: blankBodyID, *List_of_int*: toolBodyIDArray, **bool**: bNormalProjection, *UDPVector*: projectDirection, **double**: projectDistance)
61. **string**: path = **GetTempDirPath**()
62. **string**: path = **GetSysLibDirPath**()
63. **string**: path = **GetUserLibDirPath**()
64. **string**: path = **GetPersonalLibDirPath**()
65. **string**: path = **GetInstallDirPath**()
66. **string**: path = **GetProjectPath**()
67. (**bool**: SI, **bool**: abort) = **SetProgress**(*UDPPProgress*: progress)

UDP/UDM Structures and Constants

The following sections describe:

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

UDP/UDM Structures

Differences compared to C API:

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

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

Example (creating UDMParameter list):

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

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

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

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

Example:

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

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

- **UDP3x3Matrix**

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

Example:

```
rotationMatrix =[0,0,1, 1,0,0, 0,0,1]

udpFunctionLib.Transform(partId, rotationMatrix, translationVector)
```

List of Structures

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

Example:

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

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

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

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

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

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

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

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

UDP/UDM Constants

Full names of enum constants must be used in scripts.

Example:

```
unitType = UnitType.LengthUnit
```

```
dataType = ParamDataType.Int
```

Enum constants:

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

enum Constant	Parameters
	ReadOnly MustBeInt MustBeReal Hidden Unknown
CoordinateSystemAxis	XAxis YAxis ZAxis
CoordinateSystemPlane	XYPlane YZPlane ZXPlane
SweepDraftType	ExtendedDraft RoundDraft NaturalDraft MixedDraft
SplitWhichSideToKeep	SplitKeepBoth SplitKeepPositiveOnly SplitKeepNegativeOnly
PolylineSegmentType	LineSegment ArcSegment SplineSegment AngularArcSegment
MessageSeverity	WarningMessage ErrorMessage InfoMessage IncompleteMessage FatalMessage
BLNDFilletRadiusLaw	BLNDConstantRadius BLNDVariableRadius
BLNDFilletType	BLNDRound /* The outward surface of the fillet is curved.*/

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

UDP Python Example

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

```
import sys

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

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

```

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

length_units = "mm"

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

class UDPExtension(IUDPExtension):

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

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

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

        :return: None
        """

        param_dict = self.get_param_dict(param_values)

```

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

# points on the middle of 4 vertical edges
points = [
    [0, 0, param_dict["z_size"]/2],
    [param_dict["x_size"], 0, param_dict["z_size"]/2],
    [param_dict["x_size"], param_dict["y_size"], param_dict["z_
size"]/2],s
    [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, BLNDFil-
letRadiusLaw.BLNDConstantRadius, fillet_rad, 0.0, True, BLNDFil-
letType.BLNDRound, 0.0, 0.0)

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

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

```
# specify attribute ListOfEdges to which edges to apply fillet
operation

blend_element.ListOfEdges = edges[0:2]

func_lib.Fillet(blend_element, fillet_opt)


# redeclare attribute ListOfEdges to which edges to apply chamfer
operation

blend_element = UDPBLNDElements(box)

blend_element.ListOfEdges = edges[2:4]

func_lib.Chamfer(blend_element, chamfer_opt)


# Provide to the user Info message indicating success

func_lib.AddMessage(MessageSeverity.InfoMessage, "Completed!")


def GetPrimitiveTypeInfo(self):
    return primitive_info


def GetLengthParameterUnits(self):
    return length_units


def GetPrimitiveParametersDefinition2(self):
    return primitive_param_definitions


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


# Custom Functions

def get_param_value_by_name(self, param_values, param_name):
    """

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

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

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

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

"""

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

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

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

    :return: dict of parameter name and values

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

Introduction to CPython

CPython can be used to:

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

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

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

Creating an External Script

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

```
import sys

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

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

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

Those lines are followed by:

```
import ScriptEnv

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

Follow that with:

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

Start ansysedt as GRPC server

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

-grpcsrv Flag

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

`ansysedt -grpcsrv`

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

`ansysedt -grpcsrv portNumber`

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

`ansysedt -grpcsrv FirstPortNumber:LastPortNumber`

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

`ansysedt -grpcsrv FirstPortNumber:NumberOfPorts`

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

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

Connect Functions:

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

Example:

```
import sys
sys.path.append(r"<InstallationPath>/PythonFiles/DesktopPlugin")
import ScriptEnv
ScriptEnv.Connect(r"c:\myProjects\Project1.aedt")
oProject = oDesktop.GetActiveProject()
print(oProject.GetName())
```

Launching Electronics Desktop

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

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

Where:

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

Note:

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

Connecting with a Running Instance of Electronics Desktop

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

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

Where:

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

Closing Electronics Desktop/Ending the Script

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

```
ScriptEnv.Shutdown()

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

-grpcsrv Flag

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

On Windows:

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

On Linux:

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

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

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

Related Topics:

[Standalone Scripting in Iron Python](#)

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](#)

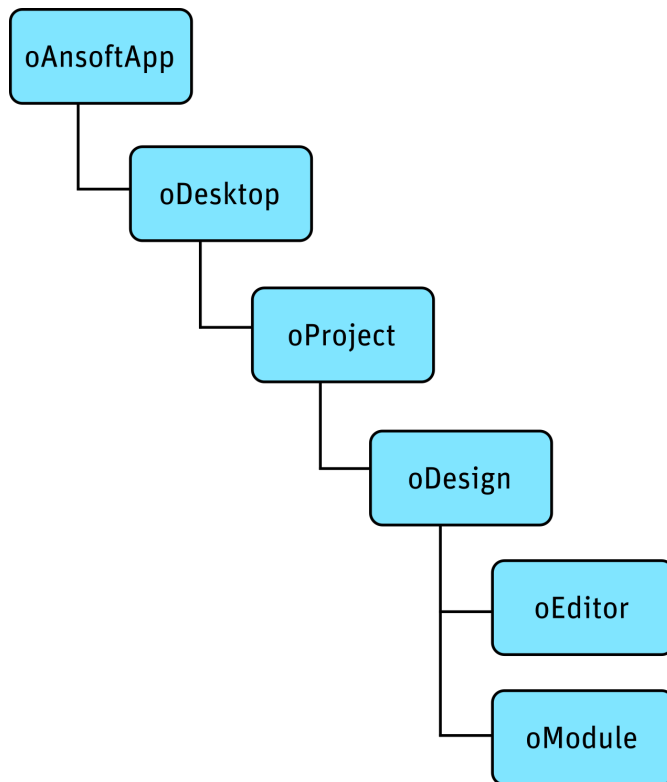
[Working with Project Scripts](#)

[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 to access the `Ansoft.ElectronicsDesktop` product.

In Iron Python, for example:

```
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()
```

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

- [Desktop Object Script Commands](#).

oProject

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

In Iron Python, for example:

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

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

- [Project Object Script Commands](#)

oDesign

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

In Iron Python, for example:

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

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')
```

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

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

Important:

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

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

oModule

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

In IronPython, for example:

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

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")
```

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()
```

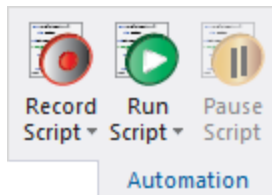
Running a Script

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

Within Electronics Desktop

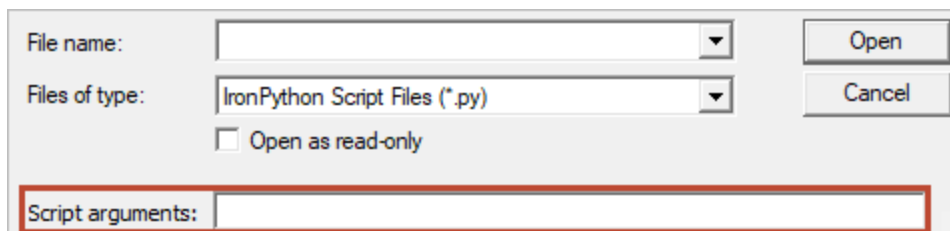
To run scripts in Electronics Desktop:

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 (*.py).
3. If desired, type script arguments in the Script Arguments field:



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"
```

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

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

Recording a Script

Electronics Desktop can record a script based on UI actions and save this script in Python (*.py) format.

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

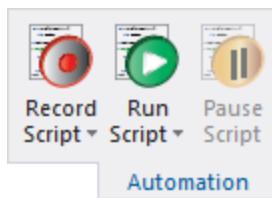
Important:

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:

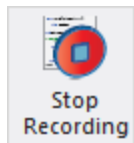
1. Click **Tools > Record Script to File**, or select the **Automation** tab and click the **Record Script** icon:



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. Click **Save**.

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



5. Perform the steps you want to record.

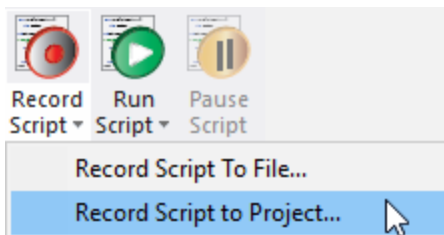
6. When you have finished recording the script, click **Stop Recording**, or select **Tools > Stop Script Recording**.

The recorded script is saved to the folder you specified.

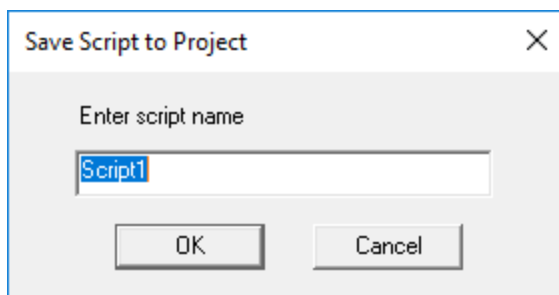
Recording a Script to a Project

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:



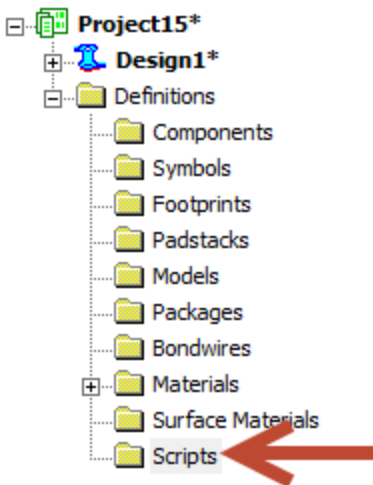
2. Enter a name for the script in the text box, then click **OK**.
3. Perform the steps you want to record.
4. When you have finished, click **Stop Recording**, or select **Tools > Stop Script Recording**.

The recorded script is saved to *scriptname.py* 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 CPython:

```
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](#)

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.

IronPython Example

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

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

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

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

)

Setting Numerical Values

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

- Pass in the number directly.

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

- Pass in a string containing the number with units.

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

- Pass in a variable name.

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

PyAEDT (Beta)

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

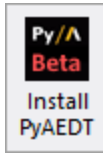
Documentation for PyAEDT can be found online at: <https://aedt-docs.pyansys.com/version/stable/>

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

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

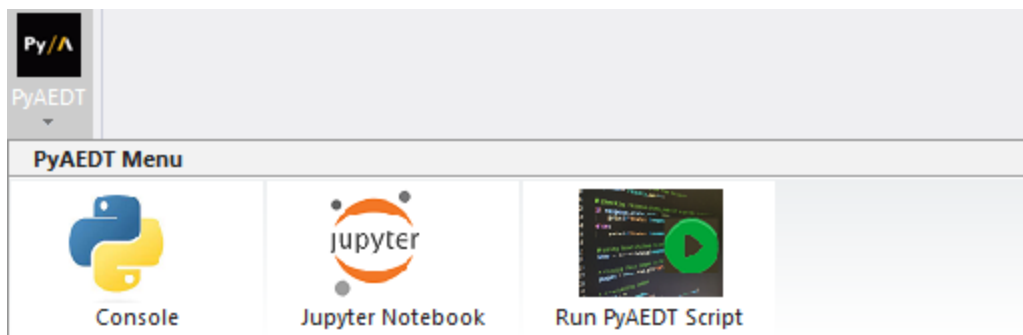
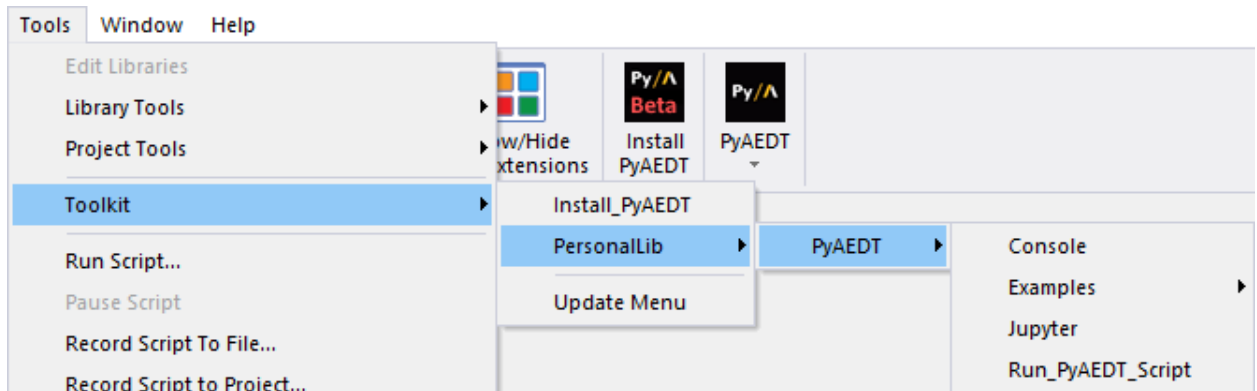
To install PyAEDT:

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



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

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



This page intentionally
left blank.

2 - Object-Oriented Property Scripting

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

This topic covers the following:

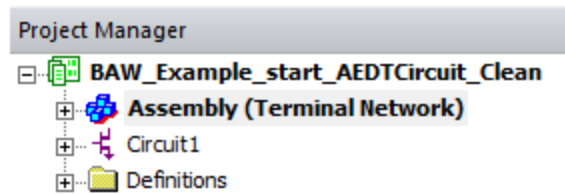
- Logic/syntax
- Basic attributes of project and design properties
- Example Scripts

Object-Oriented Scripting

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

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

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



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

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

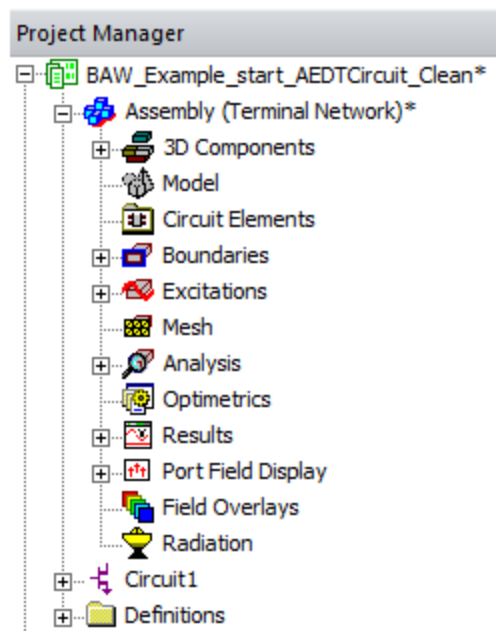
Once the objects have been defined, you can use GetChildNames() to learn what object instances exist for each. As an example, observe the Child Names of oProject, and you see a list of the Designs in the AEDT Project, per the GUI.

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

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

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

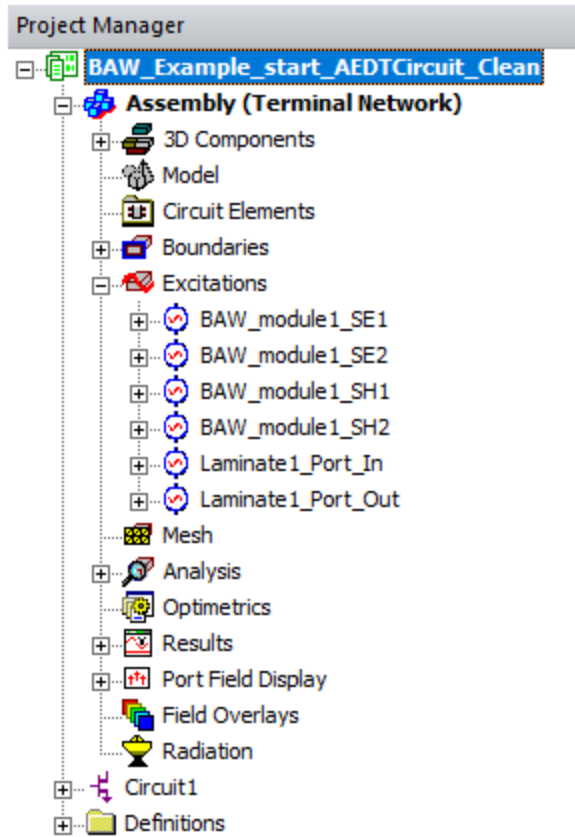
These names are what you would expect based on the Project Manager Layout:



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

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

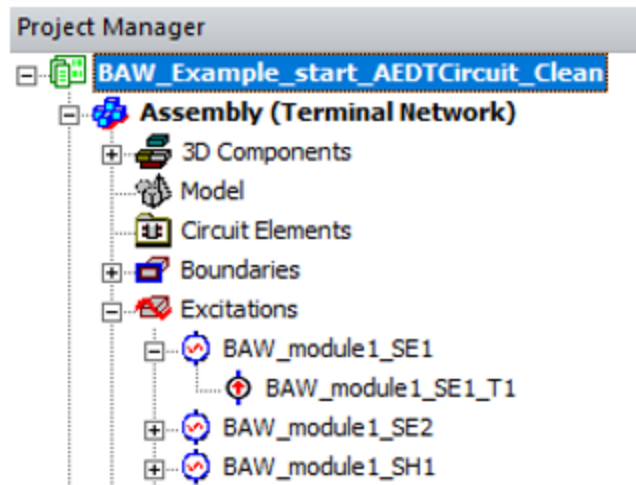
You now have an object, oExcitations defined to be the oDesign Child Object 'Excitations'. What does this mean? If you expand the Excitations dialogue in the Project Manager, you expect that the Child Names of this object would be the names of the Excitations as defined, in this case six ports:



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

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

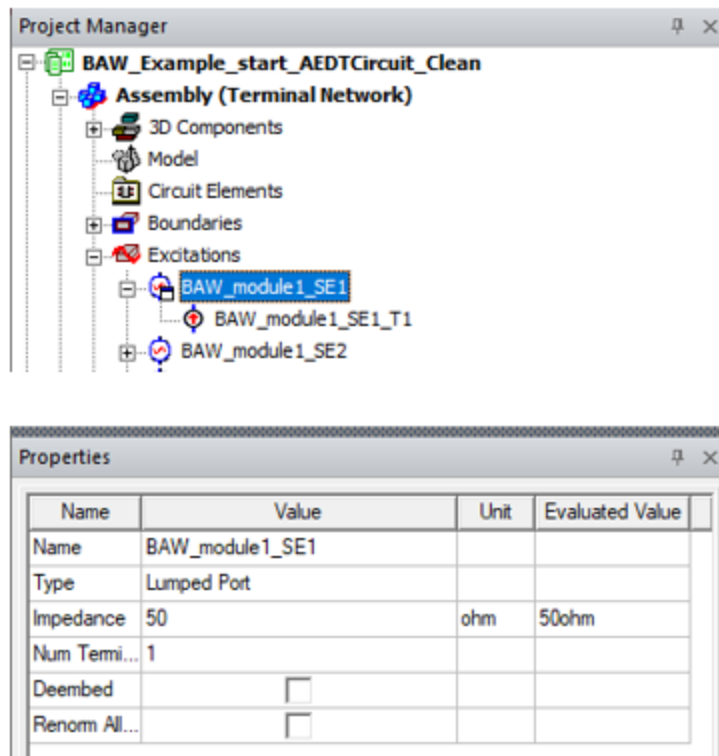
Expand the first port to see its expected Child Object, its Terminal, in the **Project Manager** Window:



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

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

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



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

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

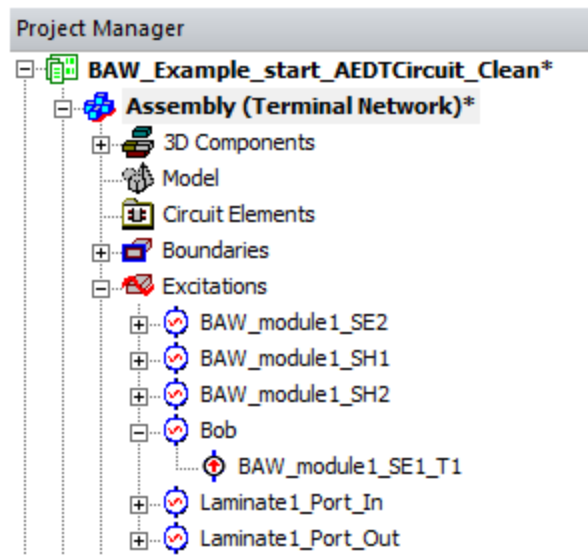
Once you identify the desired object and you know the desired property, you can access the value via `GetPropValue()`. For example, if you want to retrieve the name of the object `oPort`:

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

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

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

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



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

Material Properties and Examples

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

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

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

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

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

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

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

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

Body Properties and Modification

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

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

Retrieving Variables

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

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

Retrieve Datasets and Values

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

```
>>>oDesktop.GetChildTypes()

['Projects']

>>>oDesktop.GetChildNames()

['OptimTee']

>>>arrProjectNames = oDesktop.GetChildNames()

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

>>>tp.GetChildTypes()
```

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

GetSolutionData API

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

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

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

Summary

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

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

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

Each exposed script object supports the following COM functions:

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

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

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

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

For example, a Mechanical design object has these children.

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

Four of the children are of “Module” type

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

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

- **GetChildObject**

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

For example,

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

- **GetPropNames**

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

For example,

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

- **GetPropValue**

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

For example,

```
>>> geom.GetPropValue("material")
'vacuum'
>>> geom.GetPropValue("xsize")
'3mm'
>>> op.GetPropValue("attach to original object")
False
```

- **SetPropValue**

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

For example,

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

- **GetPropEvaluatedValue (<PropName>)**

For example,

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

- **GetPropSIValue (<PropName>)**

For example,

```
oCreateBox = oDesign.GetChildObject("3D Modeler/Box1/CreateBox:1")
oCreateBox.GetPropValue("xSize")
```

```
    return "length / 2"

oCreateBox.GetPropEvaluatedValue("xSize")

    return '0.4mm'

oCreateBox.GetPropSIValue("xSize")

    return 0.0004
```

Additional Details Specific to AEDT Solvers

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

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

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

HFSS

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

HFSS 3D Layout

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

Maxwell 3D/2D

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

RMxpert

- Analysis, Field Overlays, Optimetrics, Results

Q3D

- Boundaries, Nets, Analysis, Optimetrics,

Q2D

- Boundaries, Conductors, Analysis, Field Overlays, Optimetrics,

Icepak

- Thermal, Monitor, Mesh, Analysis, Field Overlays, Optimetrics, Results

Mechanical

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

Circuit

- Optimetrics, Results

Circuit Netlist

- Results

EMIT

- Coupling

Simplorer/Twin Builder

- Analysis, Optimetrics, Results

Additional details on Boundaries/Excitations

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

These module script objects do not have properties

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

```
[]
```

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

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

GetChildNames of these module object returns its immediate children

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

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

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

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

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

You can query/edit these properties

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

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

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

3D component encapsulation

These script interfaces are compliant with encapsulation. For example,

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

Additional details on Solve setup

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

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

This module script object has no property

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

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

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

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

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

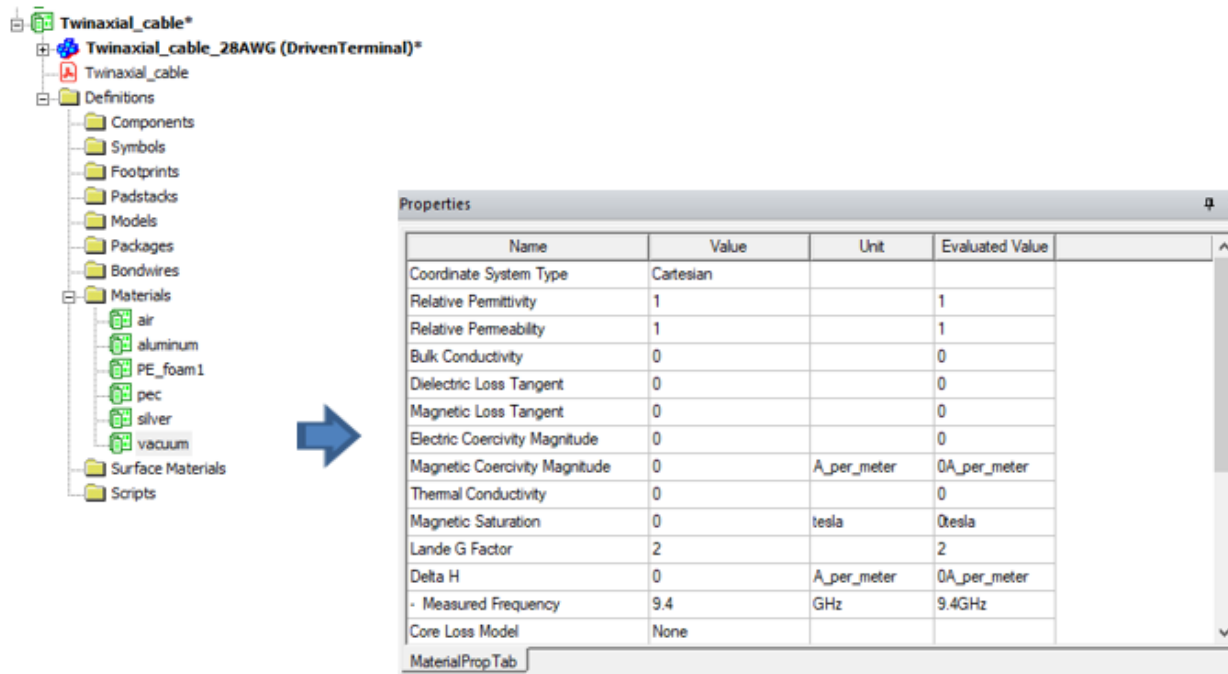
```
>>> setup.GetChildNames()  
  
['Sweep', 'Sweep1', 'Sweep2']  
  
>>> setup.GetChildTypes()  
  
['Discrete', 'Interpolating']  
  
>>> sweep1 = setup.GetChildObject("sweep")
```

Related Topics

Example: GetPropNames and GetPropValues for Layered Impedance Boundary Script

Materials Scripting Support

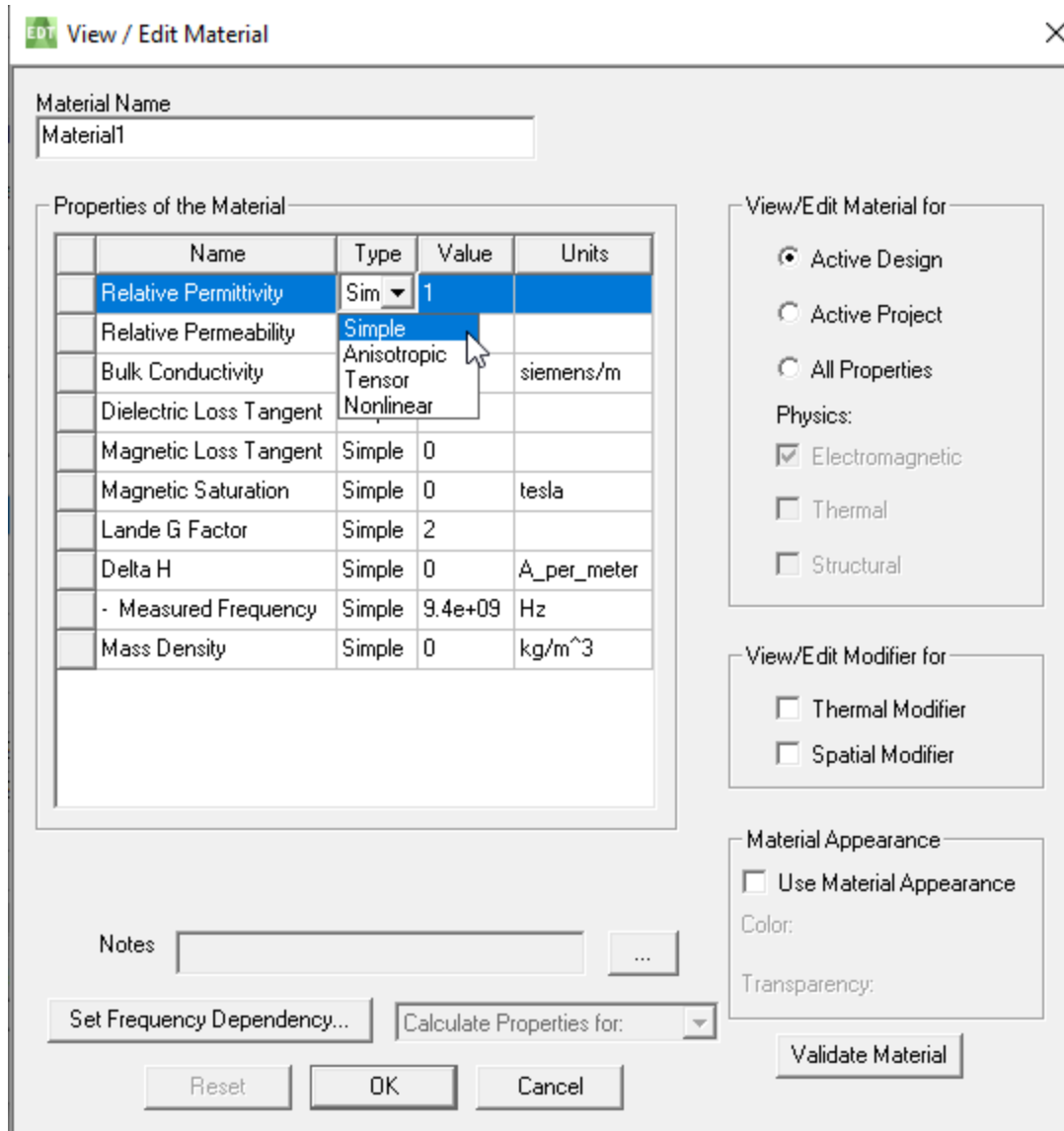
Supported material properties are shown in a Property window for the material item.



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

MaterialPropTab

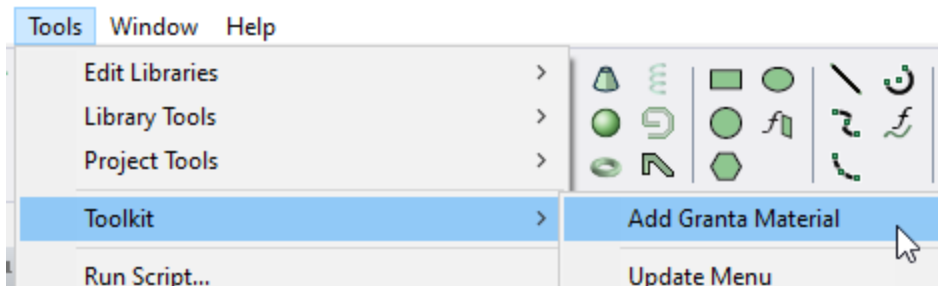
Some Material Properties like Relatively Permittivity may have values assigned as BH Curves or Tensors, as discussed in the Assigning Materials chapter of the online help.

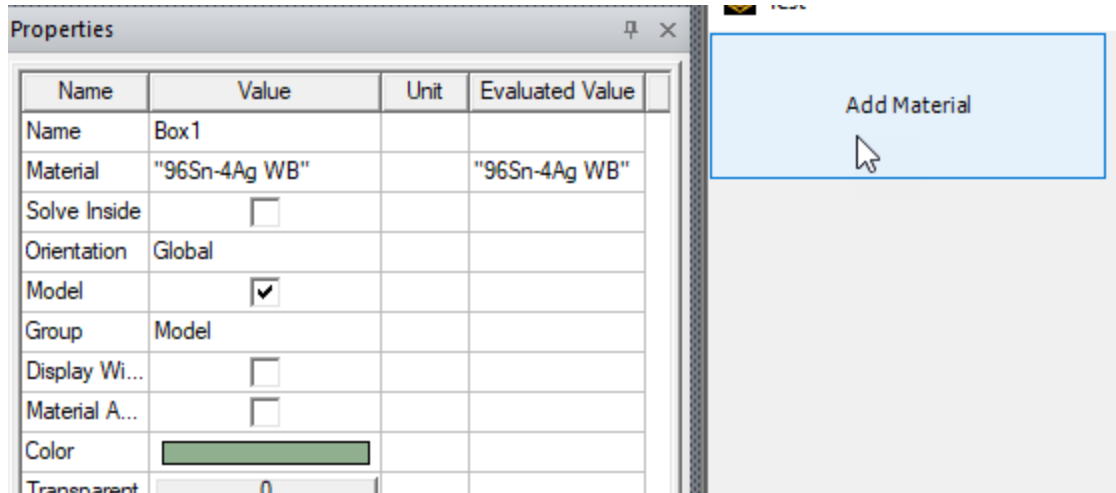


With this feature enabled you can then:

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

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





What is not supported:

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

Object Oriented Scripting for Materials

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

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

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

Examples showing change to material property type:

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

Examples showing change to a vector component value

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

```
True
```

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

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

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

```
True
```

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

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

Change choice property value

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

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

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

```
True
```

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

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

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

```
True
```

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

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

Change choice property value

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

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

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

```
True
```

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

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

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

```
>>>
```

Project Object Script Commands

Project commands should be executed by the oProject object.

One example of accessing this object is:

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

Dataset Script Commands:

[AddDataset](#)

[DeleteDataset](#)

[EditDataset](#)

[ExportDataset](#)

[HasDataset](#)

[ImportDataset](#)

Other Project Object Script Commands:

[AddMaterial \(Maxwell\)](#)

[AnalyzeAll](#)

[ChangeProperty](#)

[ClearMessages](#)

[CloneMaterial](#)

[Close](#)

[CopyDesign](#)

[CutDesign](#)

[DeleteDesign](#)

[DeleteToolObject](#)

[EditMaterial \(Maxwell\)](#)

[ExportMaterial](#)

[GetActiveDesign](#)

[GetArrayVariables](#)

[GetChildNames \[Project\]](#)

[GetChildObject \[Project\]](#)

[GetChildTypes \[Project\]](#)

[GetDefinitionManager](#)

[GetDependentFiles](#)

[GetDesign](#)

[GetDesign](#)

[GetDesigns](#)

[GetEDBHandle](#)

[GetLegacyName](#)

[GetName \[Project\]](#)

[GetObjPath \[Project\]](#)

[GetPath](#)

[GetPropEvaluatedValue](#)

[GetPropNames \[Project\]](#)

[GetPropSIValue](#)

[GetPropValue \[Project\]](#)

[GetProperties](#)

[GetPropertyValue](#)

[GetTopDesignList](#)

[GetVariableValue](#)

[GetVariables](#)

[InsertDesign](#)

[InsertDesignWithWorkflow](#)

[InsertToolObject](#)

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

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

[RemoveAllUnusedDefinitions](#)

[RemoveMaterial](#)

[RemoveUnusedDefinitions](#)

[Rename](#)

[RunToolkit](#)

[Save](#)

[SaveAs](#)

[SaveAsStandAloneProject](#)

[SaveProjectArchive](#)

[SetActiveDefinitionEditor](#)

[SetActiveDesign](#)

[SetPropValue \[Project\]](#)

[SetPropertyValue](#)

[SetVariableValue](#)

[SimulateAll](#)

[Undo \[Project\]](#)

[UpdateDefinitions](#)

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>

ClearMessages

Clears information in the **Messages** window.

Note:

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

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

Python Syntax	ClearMessages()

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

Close

Closes the active project.

Warning:

Unsaved changes will be lost.

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

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

CopyDesign

Copies a specified design.

UI Access	Edit > Copy.		
Parameters	Name	Type	Description
	<DesignName>	String	Name of the design to copy from.

Return Value	None.

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

CutDesign

Cuts a design from the active project. The design is stored in memory and can be pasted.

Warning:

This is a legacy command that is no longer supported and should not be used as it may have unintended effects on solved designs.

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

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

DeleteDesign

Deletes a specified design in the project.

UI Access	Edit > Delete , or Delete in the ribbon.		
Parameters	Name	Type	Description
	<DesignName>	String	Name of the design.
Return Value	None.		

Python Syntax	DeleteDesign (<DesignName>)
Python Example	<code>oProject.DeleteDesign ("MaxwellDesign2")</code>

DeleteToolObject

Note:

This command is for internal Ansys use only.

UI Access	N/A		
Parameters	Name	Type	Description
	<ObjectName>	String	Name of the tool object.

Return Value	None.

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

GetActiveDesign

Returns the design in the active project

Note: GetActiveDesign will return normally if there are no active objects.

UI Access	N/A
Parameters	None.
Return Value	Object of the active design.

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

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>

Note: For more information on commands for the DefinitionManager, see [Definition Manager Script Commands](#).

GetDependentFiles

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

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

Python Syntax	GetDependentFiles()
Python Example	<code>files = oProject.GetDependentFiles()</code>

GetDesign

Use: Returns the specified design.

Command: None

Syntax: GetDesign <DesignName>

Return Value: The specified design.

Parameters: <DesignName>

Type: <string>

Name of the design to return.

GetEDBHandle

Returns the EDB handle for the project.

Important:

This script is for internal Ansys use only.

UI Access	N/A
Parameters	None.
Return Value	String indicating the EDB handle for the project.

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

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>

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>

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>

GetTopDesignList

Returns a list of top-level design names.

UI Access	N/A

Parameters	None.
Return Value	List of strings containing name of top-level designs.

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

InsertDesign

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

UI Access	Project > [Insert Maxwell 3D Design / Insert Maxwell 2D Design / Insert RMXprt Design].		
Parameters	Name	Type	Description
	<DesignType>	String	Design type. "Maxwell 2D", "Maxwell 3D", or "RMxpert".
	<DesignName>	String	Design name.
	<SolutionType>	String	Dependent on design type. For Maxwell 3D, can be: "Magnetostatic", "AC Magnetic", "Transient", "Electrostatic", "DC Conduction", "Electric DC Conduction", or "Electric Transient". For Maxwell 2D, can be: "Magnetostatic", "AC Magnetic", "Transient", "Electrostatic", "AC Conduction", or "DC Conduction". For RMXprt, can be: "Three-Phase Induction Motor", "Single-Phase Induction Motor", "Three-Phase Synchronous Machine", "Brushless Permanent-Magnet DC Motor", "Adjust-Speed Synchronous Machine", "Permanent-Magnet DC Motor", "Switched Reluctance Motor", "Line-Start Permanent-Magnet Synchronous Motor", "Universal Motor", "DC Machine", "Claw-Pole Alternator", "Three-Phase Non-Salient Synchronous Machine", or "Generic Rotating

			Machine".
	""	None	Empty argument.
Return Value	None.		

Python Syntax	InsertDesign (<DesignType>, <DesignName>, <SolutionType>,"")
Python Example	<pre>oProject.InsertDesign('Maxwell 3D', 'MyDesign', 'Electric Transient', '')</pre>

InsertDesignWithWorkflow

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

UI Access	N/A		
Parameters	Name	Type	Description
	<type>	String	Type of design.
	<workflowName>	String	Name of the workflow.
	<specName>	String	Name of the spec.
	<fileName>	String	Name of the file.
	<libLoc>	String	Type of library, such as SysLib.
	<stationaryPath>	String	Path.
Return Value	IDispatch string, such as 'IDispatch(IAltraSimScript)'		

Python Syntax	InsertDesignWithWorkflow(<type>, <workflowName>, <specName>, <fileName>, <libLoc>, <stationaryPath>)

Python Example	<pre>oProject.InsertDesignWithWorkflow ("Circuit Design", "Serial Design", "PCIE3 Stressed", "LongChannel", "SysLib", "C:\\Program Files\\ANSYS Inc\\v252\\AnsysEM\\syslib\\MS" - RT_duroid 6010 (Er=10.2) 0.010 inch, 0.5 oz copper.asty")</pre>

InsertToolObject

Note:

This command is for internal Ansys use only.

Python Syntax	InsertToolObject()
Python Example	oProject.InsertToolObject()

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>

Redo [Project Level]

Reapplies the last project-level command.

UI Access	Edit > Redo.
Parameters	None.
Return Value	None.

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

Rename

Renames the project and saves it. Similar to [SaveAs\(\)](#).

UI Access	Edit > Rename.		
Parameters	Name	Type	Description
	<code><NewName></code>	String	Desired name of the project. The path is optional.
	<code><OverWriteOk></code>	Boolean	• True - overwrite the file on disk if it exists. • False - prevent overwrite.

Return Value	None.

Python Syntax	<code>Rename(<NewName>,<OverWriteOK>)</code>
Python Example	<code>oProject.Rename("c:\projects\MyProject.aedt", True)</code>

RestoreProjectArchive

Restores a previously archived project to a specified path.

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

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

Save

Saves the active project.

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

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

SaveAs

Saves the project under a new name. Requires a full path.

Note:

This script takes two parameters for non-schematic/layout designs and four parameters for schematic/layout designs.

UI Access	File > Save As.		
Parameters	Name	Type	Description
	<NewName>	String	The desired name of the project, with directory and extension.

	<OverWriteOK>	Boolean	True to overwrite the file of the same name, if it exists. False to prevent overwrite.
	<DefaultAction>	String	For Schematic/Layout projects only. Otherwise omit. See note below. Valid actions: ef_overwrite , ef_copy_no_overwrite, ef_make_path_absolute, or empty string.
	<OverwriteActions>	Array	For Schematic/Layout projects only. Otherwise omit. See note below. Structured array: Array("Name: <Action>", <FileName>, <FileName>, ...) Valid actions: ef_overwrite , ef_copy_no_overwrite, ef_make_path_absolute, or empty string.
Return Value	None.		

Python Syntax	For non-Schematic/Layout project: SaveAs (<NewName> <OverWriteOK>) For Schematic/Layout project: SaveAs (<NewName> <OverWriteOK> <DefaultAction> <OverrideActions>)
Python Example	<pre>oProject.SaveAs('D:/projects/project1.aedt', True) ---- oProject.SaveAs('D:/Projects/Project1.aedt', True, 'ef_overwrite', ['NAME:OverrideActions', ['NAME:ef_copy_no_overwrite', ['NAME:Files', '\$PROJECTDIR/circuit_models.inc']], ['NAME:ef_make_path_absolute', ['NAME:Files', '\$PROJECTDIR/SL_6s.sp']]])</pre>

Important:

The DefaultAction and OverrideActions strings correspond to the following actions:

- **ef_overwrite** – Copy file to new project directory and overwrite.
- **ef_copy_no_overwrite** – Copy file to new project directory and don't overwrite.
- **ef_make_path_absolute** – Change reference to point to file in old project directory.
- **Empty String** – Do nothing.

The DefaultAction is applied to all files that are NOT explicitly listed in the OverrideActions array. Those in the OverrideActions array are separate arrays for actions that are different from the default action; those actions are applied to the files listed in the same array:

- If OverrideActions are not specified, DefaultAction is applied to ALL files in project directory.

SaveAsStandAloneProject

Saves the project as a standalone copy.

Note:

This script is not supported when the application is being controlled by Ansys Workbench.

UI Access	N/A		
Parameters	Name	Type	Description
	<projectName>	String	The desired name of the project, with directory and extension.
Return Value	None.		

Python Syntax	<code>SaveAsStandAloneProject(<projectName>)</code>
Python Example	<code>oProject.SaveAsStandAloneProject('D:/projects/project1.aedt')</code>

SaveProjectArchive

Saves the active project as an archive to the specified file path.

UI Access	File > Archive.		
Parameters	Name	Type	Description
	<archiveFilePath>	String	Path to archived file.
	<IncludeExternalFiles>	Boolean	True to include external files; False to exclude.
	<IncludeResultsFiles>	Boolean	True to include simulation files associated with the project; False to exclude.
	<AdditionalFiles>	Array	Additional specified files to include.
	<ArchiveNotes>	String	String describe the archive.
Return Value	None.		

Python Syntax	<code>SaveProjectArchive(<archivefilepath>, <IncludeExternalFiles>, <IncludeResultsFiles>, <AdditionalFiles>, <ArchiveNotes>)</code>
Python Example	<code>oProject.SaveProjectArchive("C:\\Users\\Documents\\Ansoft\\Project27.aedt", True, False, [], "")</code>

SetActiveDefinitionEditor

Obtains a specified definition editor.

UI Access	N/A		
Parameters	Name	Type	Description
	<EditorName>	String	Name of the definition editor to set active, one of "SymbolEditor", "FootprintEditor".
	<DefinitionName>	String	The combination name for the symbol or footprint, <libname>:<def-name>
Return Value	Object for the definition to be edited.		

Python Syntax	SetActiveDefinitionEditor(<EditorName>, <DefinitionName>)
Python Example	<pre>oProject.SetActiveDefinitionEditor("SymbolEditor", "Simplorer Elements\Basic Elements\Circuit\Passive Elements:R")</pre>

SetActiveDesign

Sets a design to be the active design.

UI Access	N/A		
Parameters	Name	Type	Description
	<DesignName>	String	Name of the design to set as the active design.

Return Value	None.

Python Syntax	SetActiveDesign (<DesignName>)
Python Example	<code>oDesign = oProject.SetActiveDesign("SimplorerDesign2")</code>

SimulateAll

Simulates all solution setups and Optimetrics setups for all design instances in the project. Script processing only continues when all analyses are finished.

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

Python Syntax	SimulateAll()
Python 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>

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>

3 - Application Object Script Commands

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

`oAnsoftApp.<CommandName> <args>`

General Application Script Commands

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

- [GetAppDesktop](#)

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

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

GetAppDesktop

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

UI Access	NA		
Parameters	Name	Type	Description
	None		
Return Value	Object		

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

GetApp

Obtains the product application object.

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

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

This page intentionally
left blank.

4 - Desktop Object Script Commands

Desktop commands should be executed by the oDesktop object. Some new commands permit you to query objects when you do not know the names. See: [Object Oriented Property Scripting](#).

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

[AddMessage](#)

[AreThereSimulationsRunning](#)

[ClearMessages](#)

[CloseAllWindows](#)

[CloseProject](#)

[CloseProjectNoForce](#)

[Count](#)

[DeleteProject](#)

[DeleteRegistryEntry](#)

[DoesRegistryValueExist](#)

[DownloadJobResults](#)

[EnableAutoSave](#)

[ExportOptionsFiles](#)

[GetActiveProject](#)

[GetActiveScheduler](#)

[GetActiveSchedulerInfo](#)

[GetAutoSaveEnabled](#)

[GetBuildDateTimeString](#)

[GetCustomMenuSet](#)

[GetDefaultUnit](#)

[GetDesktopConfiguration](#)

[GetDistributedAnalysisMachines](#)

[GetDistributedAnalysisMachinesForDesignType](#)

[GetExeDir](#)

[GetGDIObjectCount](#)

[GetLibraryDirectory](#)

[GetLocalizationHelper](#)

[GetMessages](#)

[GetMonitorData](#)

[GetPersonalLibDirectory](#)

[GetProcessID](#)

[GetProjectDirectory](#)

[GetProjectList](#)

[GetProjects](#)

[GetRegistryInt](#)

[GetRegistryString](#)

[GetRunningInstancesMgr](#)

[GetSchematicEnvironment](#)

[GetScriptingToolsHelper](#)

[GetSysLibDirectory](#)

[GetTempDirectory](#)

[GetUserLibDirectory](#)

[GetVersion](#)

[IsFeatureEnabled](#)

[KeepDesktopResponsive](#)

[LaunchJobMonitor](#)

[NewProject](#)

[OpenAndConvertProject](#)

[OpenMultipleProjects](#)

[OpenProject](#)

[OpenProjectWithConversion](#)

[PageSetup](#)

[PauseRecording](#)

[PauseScript](#)

[Print](#)

[QuitApplication](#)

[RefreshJobMonitor](#)

[ResetLogging](#)

[RestoreProjectArchive](#)

[RestoreWindow](#)

[ResumeRecording](#)

[RunACTWizardScript](#)

[RunProgram](#)

[RunScript](#)

[RunScriptWithArguments](#)

[SelectScheduler](#)

[SetActiveProject](#)

[SetActiveProjectByPath](#)

[SetCustomMenuSet](#)

[SetDesktopConfiguration](#)

[SetLibraryDirectory](#)

[SetProjectDirectory](#)

[SetRegistryFromFile](#)

[SetRegistryInt](#)

[SetRegistryString](#)

[SetSchematicEnvironment](#)

[SetTempDirectory](#)[ShowDockingWindow](#)[Sleep](#)[StopSimulations](#)[SubmitJob](#)[TileWindows](#)**Related Topics:**[Desktop Commands For Registry Values](#)[ImportExport Tool Commands](#)

AddMessage

Add a message with severity and context to message window.

UI Access	N/A		
Parameters	Name	Type	Description
	<projectName>	String	Project name. Passing an empty string adds the message as the desktop global message.
	<designName>	String	Design name. Ignored if project name is empty. Passing an empty string adds the message to project node in the message tree.
	<severity>	Integer	One of "Error", "Warning" or "Info". Anything other than the first two is treated as "Info" 0 = Informational, 1 = Warning, 2 = Error, 3 = Fatal
	<msg>	String	The message for the message window.
	<category>	String	Optional. The category is created with the message under the design tree node if the category does not exist. If the category already exists,

			the new message is added to the end of the existing category. It is ignored if the project or design is empty. If missing or empty, the message is added to the Design node in the message tree.
Return Value	None.		

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

ClearMessages

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

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

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

UI Access	In Message Manager , right-click and select Clear messages for [ProjectName]...		
Parameters	Name	Type	Description
	<projectName>	String	Name of the project from which to clear messages.

	<designName>	String	Name of the design under the <projectName> from which to clear messages.
	<startSeverity>	Integer	User-specified severity level at which messages start getting cleared: • 0 = informational • 1 = warning • 2 = error • 3 = fatal error
	<stopSeverity>	Integer	<i>Optional.</i> User-specified stop severity level until which messages will be cleared. • 0 = informational • 1 = warning • 2 = error • 3 = fatal error If not specified, <stopSeverity> has a default value of 0.
Return Value	None.		

Python Syntax	<code>ClearMessages(<projectName>, <designName>, <startSeverity>, <stopSeverity>)</code>
Python Example	Preserve Current Functionality; stopSeverity = 0 (DEFAULT VALUE) <pre># startSeverity = 0; clears just the Info messages oDesktop.ClearMessages("Ex_EC_1_Asymmetrical_Conductor", "Maxwell13DDesign1", 0) # startSeverity = 1; clears all the Info and Warning messages oDesktop.ClearMessages("Ex_EC_1_Asymmetrical_Conductor", "Maxwell13DDesign1", 1) # startSeverity = 2; clears all the Info, Warning, Error</pre>

```
oDesktop.ClearMessages("Ex_EC_1_Asymmetrical_Conductor", "Maxwell3DDesign1", 2)
# startSeverity = 3; clears all messages, i.e., Info, Warning, Error, Fatal
oDesktop.ClearMessages("Ex_EC_1_Asymmetrical_Conductor", "Maxwell3DDesign1", 3)
Extend the current functionality by enabling range-based deletion
# startSeverity = 0; clears the Info messages;
oDesktop.ClearMessages("Ex_EC_1_Asymmetrical_Conductor", "Maxwell3DDesign1", 0)
# startSeverity = 0 and stopSeverity = 0; clears all the Info messages
oDesktop.ClearMessages("Ex_EC_1_Asymmetrical_Conductor", "Maxwell3DDesign1", 0,
0)
# startSeverity = 0 and stopSeverity = 1; clears all the Info and Warning mes-
sages
oDesktop.ClearMessages("Ex_EC_1_Asymmetrical_Conductor", "Maxwell3DDesign1", 0,
1)
# startSeverity = 0 and stopSeverity = 2; clears all the Info, Warning, and Error
messages
oDesktop.ClearMessages("Ex_EC_1_Asymmetrical_Conductor", "Maxwell3DDesign1", 0,
2)
# startSeverity = 0 and stopSeverity = 3; clears all the Info, Warning, Error,
and Fatal messages
oDesktop.ClearMessages("Ex_EC_1_Asymmetrical_Conductor", "Maxwell3DDesign1", 0,
3)

# startSeverity = 1; clears all the Info and Warning messages;
```

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

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

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

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

CloseAllWindows

Closes all MDI child windows on the desktop.

UI Access	From main menu, Window > CloseAll .
Parameters	None.
Return Value	None.

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

CloseProject

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

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

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

CloseProjectNoForce

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

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

Python Syntax	CloseProjectNoForce (<ProjectName>)		

Python 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>

DeleteProject

Deletes a project from disk.

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

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

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)

ExportOptionsFiles

Copies the options config files to the *DestinationDirectory*.

UI Access	Tools > Options > Export Options Files ...		
Parameters	Name	Type	Description

	<table><tr><td><DestinationDirectory></td><td>String</td><td>The path to the destination directory.</td></tr></table>			<DestinationDirectory>	String	The path to the destination directory.
<DestinationDirectory>	String	The path to the destination directory.				
Return Value	None					

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

GetActiveProject

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

Note:

GetActiveProject returns normally if there are no active objects.

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

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

GetAutoSaveEnabled

Checks whether the autosave feature is enabled.

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

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

GetBuildDateTimeString

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

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

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

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>

GetDefaultUnit

Returns the default unit for a physical quantity.

UI Access	Tools > Options > General Options > Default Units . Note that this menu only displays units that can be

	changed, while the script can be used to view additional default units.		
Parameters	Name	Type	Description
	<type>	String	String containing a type of measurement. Valid strings are (case insensitive): • "Acceleration" • "Angle" • "AngularAcceleration" • "AngularDamping" • "AngularSpeed" • "Capacitance" • "Conductance" • "Current" • "CurrentChangeRate" • "DataRate" • "DeltaH" (Magnetic Field Strength) • "Density" • "Flux" • "Force" • "Frequency" • "Inductance" • "Length" • "MagneticReluctance" • "Mass" • "MassFlowRate" • "MomentInertia"

			• "Power" • "Pressure" • "PressureCoefficient" • "Resistance" • "SaturateMagnetization" (Magnetic Inductance) • "Speed" • "Temperature" • "Time" • "Torque" • "Voltage" • "VoltageChangeRate" • "Volume" • "VolumeFlowPerPressureRoot" • "VolumeFlowRate"
Return Value	String containing the default unit (for example, "mm").		

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

GetDesigns

Obtains all designs in the current project.

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

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

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>

GetDistributedAnalysisMachines

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

GetDistributedAnalysisMachinesForDesignType

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

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

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

GetExeDir

Returns the path where the executable is located.

UI Access	N/A
Parameters	None.
Return Value	String path where executable is located. Example: "C:\Program Files\ANSYS Inc\v252\AnsysEM"

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

GetGDIObjectCount

Note:

This command is for internal Ansys use only.

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

GetLibraryDirectory

Get the path to the SysLib directory.

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

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

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>

GetMessages

Get the messages from a specified project and design.

UI Access	NA		
Parameters	Name	Type	Description
	<ProjectName>	String	Name of the project for which to collect messages. An incorrect project name results in no messages (design is ignored). An empty project name results in all messages (design is ignored)
	<DesignName>	String	Name of the design in the named project for which to collect messages. An incorrect design name results in no messages for the named project. An empty design name results in all messages for the named project
	<Severity>	Integer	Severity is 0-3, and is tied in to info/warning/error/fatal types as follows: 0 – info and above

	<table><tr><td></td><td></td><td> • 1 – warning and above • 2 – error and fatal • 3 – fatal only (rarely used) </td></tr></table>			• 1 – warning and above • 2 – error and fatal • 3 – fatal only (rarely used)
		• 1 – warning and above • 2 – error and fatal • 3 – fatal only (rarely used)		
Return Value	Array of string messages.			

Python Syntax	<code>GetMessages (<ProjectName>, <DesignName>, <Severity>)</code>
Python Example	<code>Messages = oDesktop.GetMessages("MyProject", "Maxwell11", 1)</code>

GetName [Desktop]

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

UI Access	N/A
Parameters	N-
Return Value	0- n-String containing name.

e-
.

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

GetPersonalLibDirectory

Get the path to the `PersonalLib` directory.

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

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

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>

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 scripting 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>

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>

GetProjectList

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

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

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

GetSchematicEnvironment

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

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

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

GetScriptingToolsHelper

Note:

This command is for internal Ansys use only.

Returns the object for the scripting tools helper.

UI Access	NA		
Parameters	Name	Type	Description
	None		

Return Value	Object ScriptingTools helper object

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

GetSysLibDirectory

Get the path to the `SysLib` directory.

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

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

GetTempDirectory

Gets the path to the `Temp` directory.

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

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

GetUserLibDirectory

Gets the path to the UserLib directory.

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

Python Syntax	<code>GetUserLibDirectory()</code>
Python 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>

LaunchJobMonitor

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

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

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

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>

OpenAndConvertProject

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

UI Access	Click File > Open , and choose a legacy project		
Parameters	Name	Type	Description
	<itemPath>	String	full project path of the legacy project
	<legacyChoice>	Integer	0: show conversion dialog box, (same as File > Open of a legacy file) 1: rename (changes extension to .aedt, the original file and results are renamed)

			2: copy (creates new file with .aedt extension, and the original file and results remain available)
Return Value	An object reference to the newly opened project which has the .aedt extension		

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

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

OpenMultipleProjects

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

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

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

OpenProject

Opens a specified project.

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

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

OpenProjectWithConversion

Note:

This command is for internal Ansys use only.

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

PauseRecording

Temporarily stop script recording.

UI Access	NA		
Parameters	Name	Type	Description
	None		
Return Value	None		

Python Syntax	PauseRecording()

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

PauseScript

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

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

Python Syntax	<code>PauseScript (<Message>)</code>
Python 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>

QuitApplication

Exits the desktop.

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

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

RefreshJobMonitor

For use in monitoring a job.

UI Access	Tools > Job Management > Monitor Jobs.

Parameters	None.
Return Value	A string specifying the job state. The result can be any of the following strings: • "Monitor Not Visible" • "Queued" • "Running" • "Shutting Down" • "Unknown" • "Completed" • "Not Monitoring" • "Starting Monitoring"

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

ResetLogging

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

UI Access	N/A		
Parameters	Name	Type	Description
	<logFile>	String	Path to log file.

	<code><logLevel></code>	Integer	Specified log level.
Return Value	None.		

Python Syntax	<code>ResetLogging(<logFile>, <logLevel>)</code>		
Python Example	<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
	<code><ArchiveFilePath></code>	String	Path to archived file
	<code><ProjectFilePath></code>	String	Path to restore location
	<code><OverwriteExistingFiles></code>	Boolean	True to overwrite an existing file of the same name; else False.
	<code><OpenProjectAfterRestore></code>	Boolean	True to open the project after it is restored; else False.
Return Value	None.		

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

RestoreWindow

Restores a minimized Desktop window.

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

Python Syntax	RestoreWindow()
Python 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>

RunACTWizardScript

Note:

This command is for internal Ansys use only.

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

RunProgram

Runs an external program.

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

Return Value	None

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

RunScript

Launches another script from within the script currently being executed.

UI Access	Tools>Run Script		
Parameters	Name	Type	Description
	<ScriptPath>	String	Name or full path of the script to execute. If the full path to the script is not specified, Twin Builder searches for the specified script in the following locations, in this order: 1. Personal library directory. This is the PersonalLib subdirectory in the project directory. The project directory can be specified in the General Options dialog box (click >Tools > Options > General Options to open this dialog box) under the Project Options tab. 2. User library directory.

			This is the userlib subdirectory in the library directory. The library directory can be specified in the General Options dialog in the box (click Tools > Options > General Options to open this dialog box) under the Project Options tab. 3. System library directory. This is the syslib subdirectory in the library directory. The library directory can be specified in the General Options dialog box (click Tools > Options > General Options to open this dialog box) under the Project Options tab. 4. HFSS installation directory.
Return Value	Long the return code for the script method.		

Python Syntax	<code>RunScript (<ScriptPath>)</code>
Python 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	Type	Description
	<code><ScriptPath></code>	String	The name or full path of the script to execute. If the full path to the script is not specified, the software looks for the script in the following locations:

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

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

SelectScheduler

Selects the scheduler used for batch job submission. It tries non-graphical selection of the scheduler, attempting to get version information from the scheduler in order to check for successful selection. If unable to get the information, it displays the **Select Scheduler** window and waits for the user to complete the settings.

UI Access	Tools > Job Management > Select Scheduler.		
Parameters	Name <option>	Type String	Description One of the following options (not case sensitive): • Empty string for remote RSM service • "RSM" for local RSM • "Windows HPC" for Windows HPC • "LSF" for Load-Sharing Facility • "SGE" for Grid Engine (GE, OGE, SGE, UGE, etc.) • "PBS" for Portable Batch Scheduler/System (PBSPRO, Torque, Maui, etc.) • "Ansys Cloud" for Ansys Cloud
	<address (optional)>	String	String specifying the IP address or hostname of the head node or for the remote host running the RSM service.
	<username (optional)>	String	Username string to use for remote RSM service (or blank to use username stored in current submission host user settings). If the (non-blank) username doesn't match the username stored in current submission host user settings, then the Select Scheduler dialog is displayed to allow for password entry prior to job submission.
	<forcePasswordEntry (optional)>	String	Boolean used to force display of the Select Scheduler GUI to allow for password entry prior to job submission.
Return Value	The selected scheduler (if selection was successful, this string should match the input option string, although it could differ in upper/lowercase).		

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

SetActiveProject

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

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

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

SetActiveProjectByPath

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

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

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

SetCustomMenuSet

Sets the custom menu set for Electronics Desktop.

UI Access	Navigate to Tools > Options > General Options > General > Desktop Configuration . Select a configuration using the Custom Menu Set drop-down menu.		
Parameters	Name	Type	Description
	<customMenuSet>	String	Name of desired menu set. Can be one of: • 'Default' • 'EM' • 'RF' • 'RF.0' • 'SI' • 'SI1.0' • 'SI2.0'

	<table><tr><td></td><td></td><td>• 'Twin Builder'</td></tr></table>			• 'Twin Builder'
		• 'Twin Builder'		
Return Value	None			

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

SetDesktopConfiguration

Sets the desktop configuration.

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

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

SetLibraryDirectory

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

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

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

SetProjectDirectory

Sets the project directory path.

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

Return Value	None.

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

SetSchematicEnvironment

Sets the schematic environment for Electronics Desktop.

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

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

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>

ShowDockingWindow

Shows or hides a docking window.

UI Access	Right click docking window > Show/Hide.		
Parameters	Name	Type	Description
	<windowName>	String	The window name (for example, "Message Manager", "Component Libraries", "Properties")
	<show>	Boolean	True to show; False to hide.

Return Value	None.

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

Sleep

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

UI Access	NA		
Parameters	Name	Type	Description
	<TimeInMil- liseconds>	Integer	The time that the execution should be suspended in milliseconds
Return Value	None		

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

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.aredt", "C:\\projects\\basic.aedt") moreIDs = oDesktop.SubmitJob("C:\\hpc-settings\\Submit_ Job_Settings.aredt", "C:\\projects\\basic.aedt", "Design1", "Setup1") </pre>

TileWindows

Arrange all open windows in a tiled format.

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

Python Syntax	TileWindows(< <i>TilingFlag</i> >)
Python 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\<AnsysProductNameversion>\config\<PC_NAME>_user.XML. Most of the Ansys product configuration information is stored in this XML file. These methods allow you to change the product configuration.

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

1. Start Maxwell.
2. Go to the DSO and HPC options and create a setup named "test".
3. Export the setup to a file (for example, c:\temp\test.acf).
4. Copy the exported file to a target PC (for example, f:\temp\test.acf).

5. Run the following script:

```
#import the setup

oDesktop.SetRegistryFromFile("f:\\temp\\test.acf")

# Set Active Setup to "test"

oDesktop.SetRegistryString("Desktop/ActiveDSOConfigurations/Maxwell", "test")
```

See the following subtopics:

[DeleteRegistryEntry](#)

[DoesRegistryValueExist](#)

[GetRegistryInt](#)

[GetRegistryString](#)

[SetRegistryFromFile](#)

[SetRegistryInt](#)

[SetRegistryString](#)

DoesRegistryValueExist

Determines whether a registry value exists.

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

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

GetRegistryInt

Obtains registry key integer value.

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

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

GetRegistryString

Obtains registry key string value.

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

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

SetRegistryFromFile

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

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

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

SetRegistryInt

Sets registry key to an integer value.

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

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

SetRegistryString

Sets registry key to a string value.

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

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

ImportExport Tool Commands

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

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

Scripts run via the ImportExport Tool include:

[ImportANF](#)

[ImportANFV2](#)

[ImportAutoCAD](#)

[ImportAWRMicrowaveOffice](#)

[ImportEDB](#)

[ImportExtracta](#)

[ImportGDSII](#)[ImportGerber](#)[ImportIDF](#)[ImportIDFandMerge](#)[ImportIPC](#)[ImportODB](#)[ImportXFL](#)

ImportANF

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

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

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

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>

ImportAWRMicrowaveOffice

Imports an AWRMicrowaveOffice file into a new project.

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

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

ImportEDB

Imports an EDB file into a new project.

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

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

ImportExtracta

Imports a Cadence Extracta file into a new project.

Note:

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

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

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

ImportGDSII

Imports a GDSII file into a new project.

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

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

ImportGerber

Imports a GERBER file into a new project.

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

Python Syntax	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>

ImportIDF

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

UI Access	placeholder		
Parameters	Name	Type	Description
	<NAME>	string	Settings
	<Board>	bool	Full path of board file

	<Library>	int	Full path of library file
	<Control>	int	Full path of control file
	<Filters>	list	["Cap", "Height", "HeightExclude2D", "Ind", "Power", "Res"]
	<CreateFilteredAsNonModel>	bool	True or False
	<FootPrint>	string	Footprint size and unit
	<NAME>	string	definitionOverridesMap
	<NAME>	string	instanceOverridesMap
	<HighSurfThickness>	string	High side surface thickness and unit
	<LowSurfThickness>	string	Low side surface thickness and unit
	<InternalLayerThickness>	string	Internal layer thickness and unit
	<NumInternalLayer>	int	Number of internal layers
	<HighSurfaceCopper>	int	High side surface coverage percentage
	<LowSurfaceCopper>	int	Low side surface coverage percentage
	<InternalLayerCopper>	int	Internal layer coverage percentage
	<TraceMaterial>	string	Trace material name
	<SubstrateMaterial>	int	Substrate material name
	<CreateBoard>	bool	True or False
	<ModelBoardAsRect>	bool	True or False
	<ModelDeviceAsRect>	bool	True or False
	<Cutoff>	bool	True or False
	<IncludeDrilledHoles>	bool	True or False
	<HoleDiameterCutoff>	string	Hole diameter size and unit
	<ReplaceDevices>	bool	True or False
	<CreatePointsOnBoardForInstances>	bool	True or False
Return Value	None.		

Python Syntax	ImportIDF_1 (<NAME>, <Board>, <Library>, <Control>, <Filters>, <CreateFilteredAsNonModel>, <NAME>,

	<code><NAME>, <HighSurfThickness>, <LowSurfThickness>, <InternalLayerThickness>, <NumInternalLayer>, <HighSurfaceCopper>, <LowSurfaceCopper>, <InternalLayerCopper>, <TraceMaterial>, <SubstrateMaterial>, <CreateBoard>, <ModelBoardAsRect>, <ModelDeviceAsRect>, <Cutoff>, <IncludeDrilledHoles>, <ReplaceDevices>)</code>
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, "FootPrint:=" , "0.1mm2", ["NAME:definitionOverridesMap"], ["NAME:instanceOverridesMap"], "HighSurfThickness:=" , "0.07mm", "LowSurfThickness:=" , "0.07mm", "InternalLayerThickness:=" , "0.07mm", "NumInternalLayer:=" , 2, "HighSurfaceCopper:=" , 30, "LowSurfaceCopper:=" , 30, "InternalLayerCopper:=" , 30, "TraceMaterial:=" , "Cu-Pure", "SubstrateMaterial:=" , "FR-4", "CreateBoard:=" , True, "ModelBoardAsRect:=" , True, </pre>

	<pre> "ModelDeviceAsRect:=" , False, "Cutoff:=" , False, "IncludeDrilledHoles:=" , True, "HoleDiameterCutoff:=" , "0.1mm", "ReplaceDevices:=" , False, "CreatePointsOnBoardForInstances:=", False]) </pre>

ImportIDFandMerge

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

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

Python Syntax	ImportIDFandMerge(<idfFileName>, <libPathName>, <controlFileName>, <project>, <design>)
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')

ImportIDX

Imports and IDX model into a new Icepak project.

UI Access	Icepak > Import IDX		
Parameters	Name	Type	Description
	<NAME>	string	Settings
	<Board>	bool	Full path of .idx file
	<Library>	string	NA
	<Control>	string	NA
	<Filters>	list	HeightExclude2D
	<CreateFilteredAsNonModel>	bool	True or False
	<NAME>	string	definitionOverridesMap
	<NAME>	string	instanceOverridesMap
	<HighSurfThickness>	string	High surface layer thickness value and unit
	<LowSurfThickness>	string	Low surface layer thickness value and unit
	<InternalLayerThickness>	string	Internal surface layer thickness value and unit
	<NumInternalLayer>	int	Number of internal layers value
	<HighSurfaceCopper>	int	High surface percent coverage value
	<LowSurfaceCopper>	int	Low surface percent coverage value
	<InternalLayerCopper>	int	Internal layer percent coverage value
	<TraceMaterial>	string	Trace material name
	<SubstrateMaterial>	int	Substrate material name
	<CreateBoard>	bool	True or False
	<ModelBoardAsRect>	bool	True or False

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

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

	<pre> "TraceMaterial:=" , "Cu-Pure", "SubstrateMaterial:=" , "FR-4", "CreateBoard:=" , True, "ModelBoardAsRect:=" , False, "ModelDeviceAsRect:=" , False, "Cutoff:=" , False, "ReplaceDevices:=" , False]) </pre>

ImportIPC

Imports an IPC2581 file into a new project.

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

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

	<code>'C:/Files/test.xml', 'C:/Files/test.txt')</code>

ImportODB

Imports an ODB++ file into a new project.

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

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

ImportXFL

Imports an XFL file into a new project.

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

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

5 - Running Instances Manager Script Commands

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

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

[GetAllRunningInstances](#)

[GetRunningInstanceByProcessID](#)

[GetRunningInstanceByProject](#)

GetAllRunningInstances

Returns a list of running instances of Ansys Electronics Desktop.

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

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

GetRunningInstanceByProcessID

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

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

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

GetRunningInstancesMgr

Returns the object of the Running Instances Manager.

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

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

This page intentionally
left blank.

6 - Project Object Script Commands

Project commands should be executed by the oProject object.

One example of accessing this object is:

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

Dataset Script Commands:

[AddDataset](#)

[DeleteDataset](#)

[EditDataset](#)

[ExportDataset](#)

[HasDataset](#)

[ImportDataset](#)

Other Project Object Script Commands:

[AddMaterial \(Maxwell\)](#)

[AnalyzeAll](#)

[ChangeProperty](#)

[ClearMessages](#)

[CloneMaterial](#)

[Close](#)

[CopyDesign](#)

[CutDesign](#)

[DeleteDesign](#)

[DeleteToolObject](#)

[EditMaterial \(Maxwell\)](#)

[ExportMaterial](#)

[GetActiveDesign](#)

[GetArrayVariables](#)

[GetChildNames \[Project\]](#)

[GetChildObject \[Project\]](#)

[GetChildTypes \[Project\]](#)

[GetDefinitionManager](#)

[GetDependentFiles](#)

[GetDesign](#)

[GetDesign](#)

[GetDesigns](#)

[GetEDBHandle](#)

[GetLegacyName](#)

[GetName \[Project\]](#)

[GetObjPath \[Project\]](#)

[GetPath](#)

[GetPropEvaluatedValue](#)

[GetPropNames \[Project\]](#)

[GetPropSIValue](#)

[GetPropValue \[Project\]](#)

[GetProperties](#)

[GetPropertyValue](#)

[GetTopDesignList](#)

[GetVariableValue](#)

[GetVariables](#)

[InsertDesign](#)

[InsertDesignWithWorkflow](#)

[InsertToolObject](#)

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

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

[RemoveAllUnusedDefinitions](#)

[RemoveMaterial](#)

[RemoveUnusedDefinitions](#)

[Rename](#)

[RunToolkit](#)

[Save](#)

[SaveAs](#)

[SaveAsStandAloneProject](#)

[SaveProjectArchive](#)

[SetActiveDefinitionEditor](#)

[SetActiveDesign](#)

[SetPropValue \[Project\]](#)

[SetPropertyValue](#)

[SetVariableValue](#)

[SimulateAll](#)

[Undo \[Project\]](#)

[UpdateDefinitions](#)

AddDataset

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

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

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

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

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>

ChangeProperty

Changes the properties of an object in the history tree.

UI Access	Right-click an object in the History Tree and select Properties .		
Parameters	Name	Type	Description
	<code><propertyArgs></code>	Array	Structured array. The properties vary depending on the object. Due to the number of potential configurations, it is recommended that you generate this script using the UI's Automation tab.
Return Value	None.		

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

```
oEditor.ChangeProperty(  
[  
  "NAME:AllTabs",  
  [  
    "NAME:Geometry3DCmdTab",  
    [  
      "NAME:PropServers" ,  
      "Box1:CreateBox:1"  
    ],  
    [  
      "NAME:ChangedProps",  
      [  
        "NAME:Position",  
        "X:="          , "0.35in",  
        "Y:="          , "0.55in",  
        "Z:="          , "0in"  
      ]  
    ]  
  ]  
])
```

Example: Offset the Position of a 3D Component or Layout Component with a Variable

```
oDesign.ChangeProperty(  
[  
  "NAME:AllTabs",  
  [  
    "NAME:LocalVariableTab",  
    [  
      "NAME:PropServers",  
      "LocalVariables"  
    ],  
    [  
      "NAME:NewProps",  
      [  

```



```

        "NAME:zH",
        "PropType:="          , "VariableProp",
        "UserDef:="           , True,
        "Value:="              , "50mm"
    ]
]
]
])
oEditor = oDesign.SetActiveEditor("3D Modeler")
oEditor.ChangeProperty(
[
    "NAME:AllTabs",
    [
        "NAME:General",
        [
            "NAME:PropServers",
            "LC1_1"
        ],
        [
            "NAME:ChangedProps",
            [
                "NAME:Position",
                "X:="          , "0mm",
                "Y:="          , "0mm",
                "Z:="          , "zH"
            ]
        ]
    ]
]
])

```

Example: Changing a Box's Material and Wireframe Display

```

oEditor.ChangeProperty(
[

```

```

"NAME:AllTabs",
[
  "NAME:Geometry3DAttributeTab",
  [
    "NAME:PropServers" ,
    "Box1"
  ],
  [
    "NAME:ChangedProps",
    [
      "NAME:Material",
      "Value:=" , "\"vacuum\""
    ],
    [
      "NAME:Display Wireframe",
      "Value:=" , True
    ]
  ]
]
]
)

```

Example: Change Default Handle (reference coordinate system) for a 3D Component or Layout Component

```

oEditor.ChangeProperty(
[
  "NAME:AllTabs",
  [
    "NAME:General",
    [
      "NAME:PropServers",
      "LC1_1"
    ],
  ],
]
)

```

```

        [
            "NAME:ChangedProps",
            [
                "NAME:Handle",
                "Value:=" , " NotchOutY"
            ]
        ]
    ]
])

```

Example: Changing the Model Depth

```

oEditor.ChangeProperty(
[
    "NAME:AllTabs",
    [
        "NAME:Maxwell2D",
        [
            "NAME:PropServers",
            "Design Settings"
        ],
        [
            "NAME:ChangedProps",
            [
                "NAME:Model settings/Depth",
                "Value:=" , "2.4mm"
            ]
        ]
    ]
])

```

Example: Changing 2D Transient Use of Band Mapping Angle

```

oEditor.ChangeProperty(

```

```
[
  "NAME:AllTabs",
  [
    "NAME:MeshSetupTab",
    [
      "NAME:PropServers",
      "MeshSetup:CylindricalGap1"
    ],
    [
      "NAME:ChangedProps",
      [
        "NAME:Use Band Mapping Angle",
        "Value:="          , True
      ]
    ]
  ]
])
oEditor.ChangeProperty(
[
  "NAME:AllTabs",
  [
    "NAME:MeshSetupTab",
    [
      "NAME:PropServers",
      "MeshSetup:CylindricalGap1"
    ],
    [
      "NAME:ChangedProps",
      [
        "NAME:Band Mapping Angle",
        "Value:="          , "2deg"
      ]
    ]
  ]
]
```

```
    ]
  })

```

Example: Edit VirtualPort Excitation by changing component properties

```
oEditor.ChangeProperty(
[
  "NAME:AllTabs",
  [
    "NAME:Component Data",
    [
      "NAME:PropServers",
      "BoundarySetup:LC1_1_Port1"
    ],
    [
      "NAME:ChangedProps",
      [
        "NAME:Voltage",
        "Value:=" , "10mV"
      ]
    ]
  ]
])
...
```

Example: Change Material Import for a Discovery Model

```
oEditor = oDesign.SetActiveEditor("3D Modeler")
oEditor.ChangeProperty(
[
  "NAME:AllTabs",
  [
    "NAME:Options",
    [
      "NAME:PropServers",
      "Discovery1"
    ],
  ],
  [

```

	<pre> "NAME:ChangedProps", ["NAME:Materials", "Value:=" , "Assignments and Properties"]]]])</pre>

ClearMessages

Clears information in the **Messages** window.

Note:

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

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

Python Syntax	ClearMessages()
Python Example	oProject.ClearMessages()

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>

CopyDesign

Copies a specified design.

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

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

CutDesign

Cuts a design from the active project. The design is stored in memory and can be pasted.

Warning:

This is a legacy command that is no longer supported and should not be used as it may have unintended effects on solved designs.

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

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

DeleteDataset

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

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

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

DeleteDesign

Deletes a specified design in the project.

UI Access	Edit > Delete, or Delete in the ribbon.		
Parameters	Name	Type	Description
	<DesignName>	String	Name of the design.
Return Value	None.		

Python Syntax	DeleteDesign (<DesignName>)
Python Example	<code>oProject.DeleteDesign ("MaxwellDesign2")</code>

DeleteToolObject

Note:

This command is for internal Ansys use only.

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

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

EditDataset

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

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

Python Syntax	EditDataset (<OriginalName> <DatasetDataArray>)
Python Example	<pre>oProject.EditDataset ("ds1" ["NAME:ds2", ["NAME:Coordinates", ["NAME:Coordinate", "X:=", 1, "Y:=", 2], ["NAME:Coordinate", "X:=", 3, "Y:=", 4]]])</pre>

	<pre>oDesign.EditDataset ("ds1" ["NAME:ds2", ["NAME:Coordinates", ["NAME:Coordinate", "X:=", 1, "Y:=", 2], ["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	Name	Type	Description

	<code><datasetFilePath></code>	String	The full path to the file.
Return Value	None.		

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

GetActiveDesign

Returns the design in the active project

Note: `GetActiveDesign` will return normally if there are no active objects.

UI Access	N/A
Parameters	None.
Return Value	Object of the active design.

Python Syntax	<code>GetActiveDesign()</code>
Python Example	<pre>oDesign = oProject.GetActiveDesign()</pre>

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>

GetChildNames [Project]

Returns the names of the project's child objects.

UI Access	N/A		
Parameters	Name	Type	Description
	<type>	String	(Optional) Default is "design"; Any value returned by GetChildTypes() can be used.
Return Value	Array of names of children for the queried object.		

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

GetChildObject [Project]

Returns the project's child objects.

Note:

`GetChildObject` will return normally if there are no active objects.

UI Access	N/A		
Parameters	Name	Type	Description
	<code><path></code>	String	The path may include multiple generations (for example, "designObject/moduleObj/SetupObject"). See Object Path .
Return Value	Object of a found child.		

Python Syntax	<code>GetChildObject(<path>)</code>
Python Example	<pre> oProject = oDesktop.GetActiveProject() oDesign = oProject.GetChildObject("TeeModel") oVariable = oProject.GetChildObject("VariableName") </pre>

	<code>oReport = oProject.GetChildObject("TeeModel/Results/S Parameter Plot 1")</code>

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

GetDefinitionManager

Gets the `DefinitionManager` object.

UI Access	N/A
Parameters	None.
Return Value	<code>DefinitionManager</code> object.

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

Note: For more information on commands for the DefinitionManager, see [Definition Manager Script Commands](#).

GetDependentFiles

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

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

Python Syntax	GetDependentFiles()
Python Example	<code>files = oProject.GetDependentFiles()</code>

GetDesign

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

UI Access	N/A		
Parameters	Name	Type	Description
	< <i>designName</i> >	String	Name of the design.
Return Value	Object of the specified design.		

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

GetDesigns

Obtains all designs in the current project.

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

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

GetEDBHandle

Returns the EDB handle for the project.

Important:

This script is for internal Ansys use only.

UI Access	N/A
Parameters	None.
Return Value	String indicating the EDB handle for the project.

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

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>

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>

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>

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>

GetPropEvaluatedValue

Returns the Evaluated-Value for Value-Property and Variable. Returns the Property-value as text string for other property types

Note:

This command is not supported by the EMIT and Circuit design types.

UI Access	N/A		
Parameters	Name	Type	Description
	<PropName>	String	Name of the property.
Return Value	String value of the evaluated value.		

Python Syntax	GetPropEvaluatedValue (<PropName>)
Python Example	<pre>oVar = oDesign.GetChildObject(" Variables/var") oVar.GetPropEvaluatedValue()</pre>

GetPropNames [Project]

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

UI Access	N/A		
Parameters	Name	Type	Description
	<includeReadOnly>	Boolean	(Optional) • True – Include read only props. • False – Do not include read only props.
Return Value	Empty array.		

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

GetPropSIValue

Returns the SI-Value for Value-Property and Variable. Return NAN for other property type if its value is not able to convert to be a double-floating point value.

Note:

This command is not supported by the EMIT and Circuit design types.

UI Access	N/A

Parameters	Name	Type	Description
	<PropName>	String	Name of the property.
Return Value	Property value as a double floating value, or NAN if the property value cannot be converted to double floating point.		

Python Syntax	GetPropSIValue (<PropName>)
Python Example	<pre>oCreateBox = oDesign.GetChildObject("3D Modeler/Box1/CreateBox:1") oCreateBox.GetPropValue("xSize") return "length / 2" oCreateBox.GetPropEvaluatedValue("xSize") return '0.4mm' oCreateBox.GetPropSIValue("xSize") return 0.0004</pre>

GetPropValue [Project]

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

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

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

GetProperties

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

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

Return Value	Array of strings containing the names of the appropriate properties.

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

GetPropertyValue

Returns the value of a single property belonging to a specific *<PropServer>* and *<PropTab>*. This function is available with the Project, Design or Editor objects, including definition editors.

Tip: Use the script recording feature and edit a property, and then view the resulting script to see the format for that property.

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

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

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

GetTopDesignList

Returns a list of top-level design names.

UI Access	N/A
Parameters	None.
Return Value	List of strings containing name of top-level designs.

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

GetVariableValue

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

UI Access	N/A		
Parameters	Name	Type	Description
	<code><VarName></code>	String	Name of the variable to access.
Return Value	String represents the value of the variable.		

Python Syntax	GetVariableValue(<code><VarName></code>)
Python 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	<pre>oProject.GetVariables() oDesign.GetVariables()</pre>

ImportDataset

Imports a dataset from a named file. This can be executed by the `oProject`, or `oDesign` variables. The name of the dataset is file-name+index number (e.g., `dsdata1`) unless the filename ends with a trailing number. When there is a trailing number at the end, we will remove the number and use first unused index. Alternatively, the name of the dataset can be explicitly defined by providing a string as an optional second argument.

UI Access	Project > Datasets > Import.		
Parameters	Name	Type	Description
	<code><datasetFilePath></code>	String	The full path to the file containing the dataset values. *.tab files recommended (see note below).
	<code><optionalDatasetName></code>	String	<i>Optional.</i> User-defined dataset name.

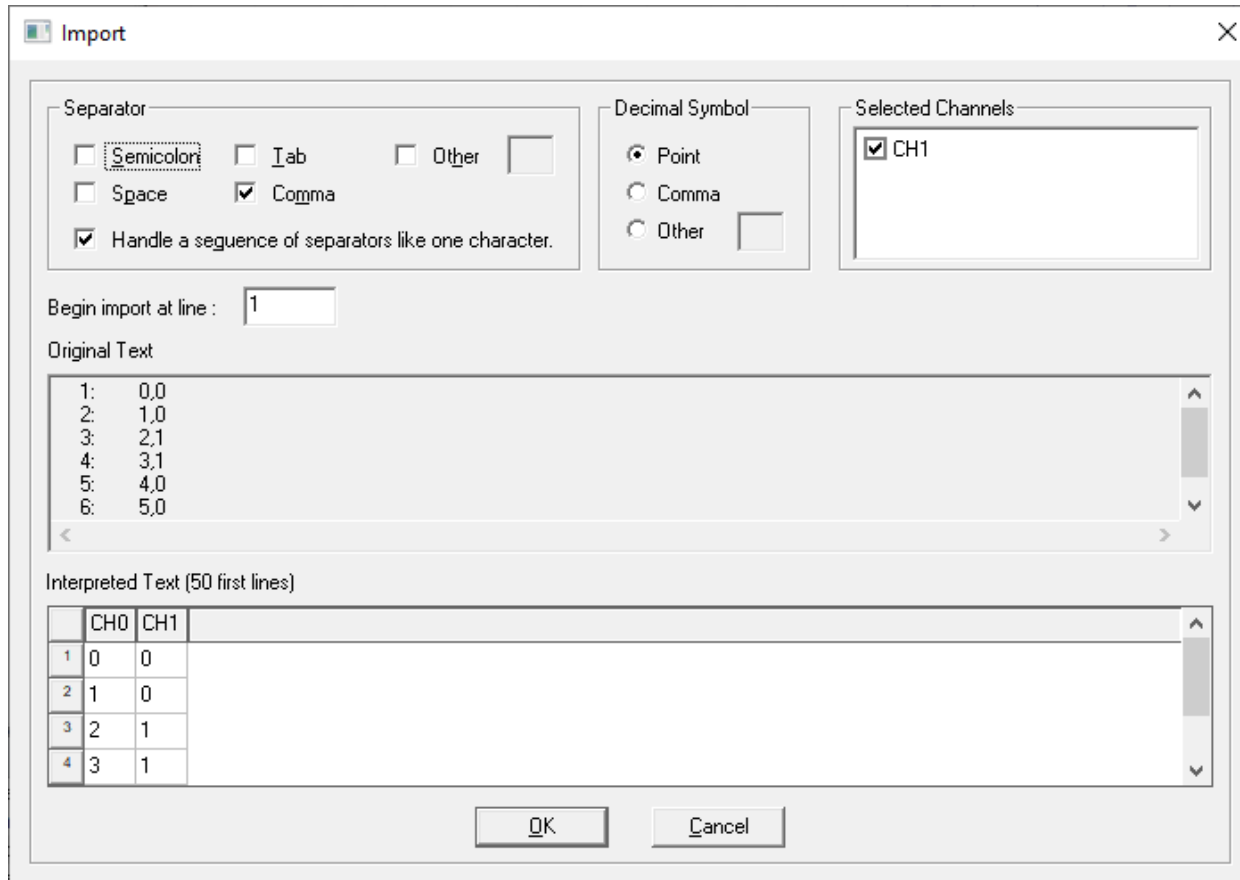
Return Value	None.

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

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 Maxwell scripts, the last argument will always be empty.

UI Access	Project > [Insert Maxwell 3D Design / Insert Maxwell 2D Design / Insert RMXprt Design].		
Parameters	Name	Type	Description
	<DesignType>	String	Design type. "Maxwell 2D", "Maxwell 3D", or "RMXprt".
	<DesignName>	String	Design name.
	<SolutionType>	String	Dependent on design type. For Maxwell 3D, can be: "Magnetostatic", "AC Magnetic", "Transient", "Electrostatic", "DC Conduction", "Electric DC Conduction", or "Electric Transient". For Maxwell 2D, can be: "Magnetostatic", "AC Magnetic", "Transient", "Electrostatic", "AC Conduction", or "DC Conduction". For RMXprt, can be: "Three-Phase Induction Motor", "Single-Phase Induction Motor", "Three-Phase Synchronous Machine", "Brushless Permanent-Magnet DC Motor", "Adjust-Speed Synchronous Machine", "Permanent-Magnet DC Motor", "Switched Reluctance Motor", "Line-Start Permanent-Magnet Synchronous Motor", "Universal Motor", "DC Machine", "Claw-Pole Alternator", "Three-Phase Non-Salient Synchronous Machine", or "Generic Rotating Machine".
	""	None	Empty argument.
Return Value	None.		

Python Syntax	InsertDesign (<DesignType>, <DesignName>, <SolutionType>,"")
Python Example	<pre>oProject.InsertDesign('Maxwell 3D', 'MyDesign', 'Electric Transient', '')</pre>

InsertDesignWithWorkflow

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

UI Access	N/A		
Parameters	Name	Type	Description
	<type>	String	Type of design.
	<workflowName>	String	Name of the workflow.
	<specName>	String	Name of the spec.
	<fileName>	String	Name of the file.
	<libLoc>	String	Type of library, such as SysLib.
	<stationaryPath>	String	Path.
Return Value	IDispatch string, such as 'IDispatch(IAltraSimScript)'		

Python Syntax	InsertDesignWithWorkflow(<type>, <workflowName>, <specName>, <fileName>, <libLoc>, <stationaryPath>)
Python Example	<pre>oProject.InsertDesignWithWorkflow ("Circuit Design", "Serial Design", "PCIE3 Stressed", "LongChannel", "SysLib", "C:\\Program Files\\ANSYS Inc\\v252\\AnsysEM\\syslib\\MS" - RT_duroid 6010 (Er=10.2) 0.010 inch, 0.5 oz copper.asty")</pre>

InsertToolObject

Note:

This command is for internal Ansys use only.

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

Paste (Project Object)

Pastes a design in the active project.

UI Access	Edit > Paste.
Parameters	None.
Return Value	None

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

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>

RemoveAllUnusedDefinitions

Removes all unused project definitions.

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

Python Syntax	RemoveAllUnusedDefinitions()
Python 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, Maxwell assumes the material is a project material. The last two parameters will be ignored. If False, the material is not a project material.

	<LibraryName>	String	Name of the user or personal library where the material resides.
	<LibraryLocation>	String	Location of library. Valid options:"UserLib" or "PersonalLib".
Return Value	None.		

Python Syntax	RemoveMaterial (<MaterialName>, <IsProjectMaterial>, <LibraryName>, <LibraryLocation>)		
Python Example	oDefinitionManager.RemoveMaterial (["Material1", false, "mo0907", "UserLib"])		

RemoveUnusedDefinitions

Removes any unused project definitions.

UI Access	Tools > Project Tools > Remove Unused Definitions.		
Parameters	Name	Type	Description
	<Definitions>	Array	Definitions to be removed, such as materials and surface materials.
Return Value	None.		

Python Syntax	RemoveUnusedDefinitions(<Definitions>)		
Python Example	oProject.RemoveUnusedDefinitions (		

	<pre>[["NAME:Materials", "Al-Extruded"], ["NAME:SurfaceMaterials", "Steel-oxidised-surface"]]</pre>

Rename

Renames the project and saves it. Similar to [SaveAs\(\)](#).

UI Access	Edit > Rename.		
Parameters	Name	Type	Description
	<NewName>	String	Desired name of the project. The path is optional.
	<OverWriteOk>	Boolean	• True - overwrite the file on disk if it exists. • False - prevent overwrite.
Return Value	None.		

Python Syntax	Rename(<NewName>,<OverWriteOK>)
Python Example	<code>oProject.Rename("c:\projects\MyProject.aedt", True)</code>

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	[Maxwell] > Toolkit > [Script Name].		
Parameters	Name	Type	Description
	<LibraryType>	String	Name of the library in which the script is located.
	<ToolkitName>	String	Name of the Python script.
	<ToolkitArg>	Array	Structured array containing arguments required by the toolkit script.
Return Value	User-defined solution(s) and/or output(s).		

Python Syntax	RunToolkit(<LibraryType>, <ToolkitName>, <ToolkitArg>)
Python Example	<code>oDesign.RunToolkit("SysLib", "Cable Modeling/Automotive Cable Bundle",</code>

```
[ ] )

oDesign.RunToolkit("SysLib", "Cable Modeling/Oil-Gas Cable Bundle",
[ ] )
```

Save

Saves the active project.

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

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

SaveAs

Saves the project under a new name. Requires a full path.

Note:

This script takes two parameters for non-schematic/layout designs and four parameters for schematic/layout designs.

UI Access	File > Save As.

Parameters	Name	Type	Description
	<NewName>	String	The desired name of the project, with directory and extension.
	<OverWriteOK>	Boolean	True to overwrite the file of the same name, if it exists. False to prevent overwrite.
	<DefaultAction>	String	For Schematic/Layout projects only. Otherwise omit. See note below. Valid actions: ef_overwrite , ef_copy_no_overwrite, ef_make_path_absolute, or empty string.
	<OverwriteActions>	Array	For Schematic/Layout projects only. Otherwise omit. See note below. Structured array: Array("Name: <Action>", <FileName>, <FileName>, ...) Valid actions: ef_overwrite , ef_copy_no_overwrite, ef_make_path_absolute, or empty string.
Return Value	None.		

Python Syntax	For non-Schematic/Layout project: SaveAs (<NewName> <OverWriteOK>) For Schematic/Layout project: SaveAs (<NewName> <OverWriteOK> <DefaultAction> <OverrideActions>)
Python Example	<pre>oProject.SaveAs('D:/projects/project1.aedt', True) ---- oProject.SaveAs('D:/Projects/Project1.aedt', True, 'ef_overwrite', ['NAME:OverrideActions', ['NAME:ef_copy_no_overwrite', ['NAME:Files', '\$PROJECTDIR/circuit_models.inc']], ['NAME:ef_make_path_absolute', ['NAME:Files', '\$PROJECTDIR/SL_6s.sp']]])</pre>

Important:

The DefaultAction and OverrideActions strings correspond to the following actions:

- **ef_overwrite** – Copy file to new project directory and overwrite.
- **ef_copy_no_overwrite** – Copy file to new project directory and don't overwrite.
- **ef_make_path_absolute** – Change reference to point to file in old project directory.
- **Empty String** – Do nothing.

The DefaultAction is applied to all files that are NOT explicitly listed in the OverrideActions array. Those in the OverrideActions array are separate arrays for actions that are different from the default action; those actions are applied to the files listed in the same array:

- If OverrideActions are not specified, DefaultAction is applied to ALL files in project directory.

SaveAsStandAloneProject

Saves the project as a standalone copy.

Note:

This script is not supported when the application is being controlled by Ansys Workbench.

UI Access	N/A		
Parameters	Name	Type	Description
	<projectName>	String	The desired name of the project, with directory and extension.
Return Value	None.		

Python Syntax	SaveAsStandAloneProject(<projectName>)
Python Example	<code>oProject.SaveAsStandAloneProject('D:/projects/project1.aedt')</code>

SaveProjectArchive

Saves the active project as an archive to the specified file path.

UI Access	File > Archive.		
Parameters	Name	Type	Description
	<archiveFilePath>	String	Path to archived file.
	<IncludeExternalFiles>	Boolean	True to include external files; False to exclude.
	<IncludeResultsFiles>	Boolean	True to include simulation files associated with the project; False to exclude.
	<AdditionalFiles>	Array	Additional specified files to include.
	<ArchiveNotes>	String	String describe the archive.
Return Value	None.		

Python Syntax	SaveProjectArchive(<archivefilepath>, <IncludeExternalFiles>, <IncludeResultsFiles>, <AdditionalFiles>, <ArchiveNotes>)
Python Example	<code>oProject.SaveProjectArchive("C:\\Users\\Documents\\Ansoft\\Project27.aedt", True, False, [], "")</code>

SetActiveDefinitionEditor

Obtains a specified definition editor.

UI Access	N/A		
Parameters	Name	Type	Description
	<EditorName>	String	Name of the definition editor to set active, one of "SymbolEditor", "FootprintEditor".
	<DefinitionName>	String	The combination name for the symbol or footprint, <libname>:<def-name>
Return Value	Object for the definition to be edited.		

Python Syntax	SetActiveDefinitionEditor(<EditorName>, <DefinitionName>)		
Python Example	<pre>oProject.SetActiveDefinitionEditor("SymbolEditor", "Simplorer Elements\Basic Elements\Circuit\Passive Elements:R")</pre>		

SetActiveDesign

Sets a design to be the active design.

UI Access	N/A		
Parameters	Name	Type	Description
	<DesignName>	String	Name of the design to set as the active design.
Return Value	None.		

Python Syntax	SetActiveDesign (<DesignName>)
Python Example	<code>oDesign = oProject.SetActiveDesign("SimplorerDesign2")</code>

SetPropValue [Project]

Sets a property value for an active project's child object.

UI Access	Edit Properties on ProjectTree objects.		
Parameters	Name	Type	Description
	<propPath>	String	A child object's property path. See: Property Function .
	<newValue>	String	New property value.
Return Value	Boolean: • True – property found. • False – property not found.		

Python Syntax	SetPropValue(<propPath>, <newValue>)
Python Example	<code>oProject.SetPropValue("TeeModel/offset", "2mm")</code> <code>oProject.SetPropValue("TeeModel/Results/S Parameter Plot 1/Display Type", "Data Table")</code>

SetPropertyValue

Sets the value of a single property belonging to a specific PropServer and PropTab. This function is available with the Project, Design or Editor objects, including definition editors. This is not supported for properties of the following types: ButtonProp, PointProp, V3DPointProp, and VPointProp. Only the ChangeProperty command can be used to modify these properties.

Use the script recording feature and edit a property, and then view the resulting script entry or use GetPropertyValue for the desired property to see the expected format.

UI Access	N/A		
Parameters	Name	Type	Description
	<propTab>	String	One of the following, where tab titles are shown in parentheses: - PassedParameterTab ("Parameter Values") - DefinitionParameterTab (Parameter Defaults") - LocalVariableTab ("Variables" or "Local Variables") - ProjectVariableTab ("Project variables") - ConstantsTab ("Constants") - BaseElementTab ("Symbol" or "Footprint") - ComponentTab ("General") - Component("Component") - CustomTab ("Intrinsic Variables") - Quantities ("Quantities") - Signals ("Signals")
	<propServer>	String	An object identifier, generally returned from another script method, such as CompInst@R;2;3
	<propName>	String	Name of the property.
	<propValue>	String	The value for the property
Return Value	None.		

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

SetVariableValue

Sets the value of a variable. To set the value of a project variable, execute this command using `oProject`. To set the value of a local variable, use `oDesign`.

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

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

SimulateAll

Simulates all solution setups and Optimetrics setups for all design instances in the project. Script processing only continues when all analyses are finished.

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

Python Syntax	SimulateAll()
Python Example	<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>

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>

7 - Definition Manager Script Commands

The definition manager controls the use of materials and scripts in a project. It also provides access to the managers for symbols, footprints, padstacks, and components in a project.

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

```
Set oDefinitionManager = oProject.GetDefinitionManager()
```

The topics for this section include:

[AddDefinitionFromBlock](#)

[AddMaterial](#)

[CloneMaterial](#)

[DoesMaterialExist](#)

[EditMaterial](#)

[ExportMaterial](#)

[RemoveMaterial](#)

[RemoveUnusedDefinitions](#)

Related Topics:

[Component Manager Script Commands](#)

[Material Manager Script Commands](#)

[Model Manager Script Commands](#)

[Network Data Explorer Manager Script Commands](#)

[Script and Library Scripts](#)

[Symbol Manager Script Commands](#)

AddMaterial (Maxwell)

Adds a local material.

UI Access	Add Material in the material editor.		
Parameters	Name	Type	Description
	<code><MaterialParams></code>	Array	["NAME: <material to be added>", <MatProperty>, <MatProperty>, ...]
	<code><MatProperty></code>	Array	Options will depend on what is specified. Some options include "AnisoProperty", "CustomProperty", and "nonlinear".

			For simple material: <pre>"<PropertyName>:=", <value></pre> For anisotropic material: <pre>["NAME:<PropertyName>", "property_type:=", "AnisoProperty", "unit:=", <Unit>", "component1:=", <value>, "component2:=", <value>, "component3:=", <value>))]</pre> Material properties for other property types are listed further in the topic.
	<PropertyName>	String	Should be one of the following (depending on the material, design, and solution types): <pre>"permittivity", "permeability", "RecoilPermeability", "conductivity", "dielectric_loss_tangent", "magnetic_loss_tangent", "magnetic_coercivity", "core_loss_type", "mass_density", "stacking_type", "magnetostriction", "inverse_magnetostriction", "youngs_modulus", "poissons_ratio"</pre>
	<Unit>	String	Possible values for magnetic properties: delta_H: "Oe", saturation_mag: "Gauss", "uGauss", "Tesla", "uTesla"
	Return Value	None	

Python Syntax	AddMaterial (["NAME:<MaterialName>", <MatProperty>, <MatProperty>, ...])
Python Example	<pre> # Example: Magnetostrictive stress and deformation with full magnetostriction matrix [3x3] oDefinitionManager.AddMaterial(["NAME:Material2", "CoordinateSystemType:=", "Cartesian", "BulkOrSurfaceType:=" , 1, ["NAME:PhysicsTypes", "set:=" , ["Electromagnetic"]], ["NAME:magnetostriction", "property_type:=" , "CustomProperty", "PropertyModelType:=" , 4, # integer value of 1 (Linear), 2 (Nonlinear), 3 (Anisotropic), or 4 (Matrix option) "Dir1ModelType:=" , 0, "Dir2ModelType:=" , 0, "IsoCoeff:=" , "0", "IsoHMax:=" , "0A_per_meter", "IsoHMin:=" , "0A_per_meter", "Dir1Coeff:=" , "0", "Dir1HMax:=" , "0A_per_meter", "Dir1HMin:=" , "0A_per_meter", "Dir1VecX:=" , "1", "Dir1VecY:=" , "0", "Dir1VecZ:=" , "0", "Dir2Coeff:=" , "0", </pre>

```

"Dir2HMax:="          , "0A_per_meter",
"Dir2HMin:="          , "0A_per_meter",
"CoeffMatrixSize:="   , 3, # <int> [0, 3].
# If "PropertyModelType" is "4", size cannot be 0 (should not be empty)).
# CoeffMatrixSize = 3" means 3x3 matrix size, and matrix elements
#(C11, C12, .. C33) are specified in CoeffMatrix block.
"IsCoeffMatrixSymmetric:=", True,

[
  "NAME:CoeffMatrix", # if size is 3, has matrix elements C11, C12, ... C33 ,
  #no element if size is 0
  "C11:="              , "5",
  "C12:="              , "0",
  "C13:="              , "0",
  "C21:="              , "0",
  "C22:="              , "4",
  "C23:="              , "0",
  "C31:="              , "0",
  "C32:="              , "0",
  "C33:="              , "3"
],
"CoeffMatrixHMax:="    , "10A_per_meter", #Value of Hmax
[
  "NAME:Magnetostriction",
  [
    "NAME:Isotropic",
    [
      "NAME:OneVariation",
      "DatasetMapsName:=" , "Lambda-H curves versus Stress"
    ]
  ]
],
[
  "NAME:Direction1",

```

```

        [
            "NAME:OneVariation",
            "DatasetMapsName:=" , "Lambda-H curves versus Stress"
        ]
    ],
    [
        "NAME:Direction2",
        [
            "NAME:OneVariation",
            "DatasetMapsName:=" , "Lambda-H curves versus Stress"
        ]
    ],
    "SelectedDatasetMapsName:=" , "Lambda-H curves versus Stress"
]
]
])

```

Additional Properties for Nonlinear Property Type

<MatProperty> (nonlinear material)

```

Array("NAME:permeability",
    "property_type:=", "nonlinear",
    "BTypeForSingleCurve:=", <string>,
    "Hunit:=", <string>, "Bunit:=", <string>,
    "IsTemperatureDependent:=", <bool>,
Array("NAME:BHCoordinates",
    Array("NAME:Coordinate", "X:=", <value>, "Y:=", <value>, ...

```

BTypeForSingleCurve

Type: <string>

Should be one of the following: "Intrinsic", or "Normal" (defaults to Normal if nothing is specified). An error message displays in the Message Window if the string does not match either "Normal" or "Intrinsic"; advising the user of this and informing the user that this property has been set to "Normal".

Hunit

Type: <string>

Possible values: "Oe", "A_per_meter", "kA_per_meter"

Bunit

Type: <string>

Possible values: "tesla", "kTesla", "uGauss"

IsTemperatureDependent

Type: <bool>

possible values: True or False

Additional Properties for Vector Property Type

<MatProperty> (vector material)

Array("NAME:<PropertyName>",

"property_type:=", "VectorProperty", "Magnitude:=", <value>, "DirComp1:=", <value>, "DirComp2:=", <value>, "DirComp3:=", <value>))

<MatProperty> (choice material)

Array("NAME:<PropertyName>",

"property_type:=", "ChoiceProperty", "Choice:=", <string>")

Additional Python Examples

Python Example (Young's Modulus and Poisson's Ratio):

```
oDefinitionManager.AddMaterial("copper",
[
...
[
    "NAME:youngs_modulus",
    "property_type:" , "AnisoProperty",
    "unit:" , "",
    "component1:" , "1",
    "component2:" , "1.5",
    "component3:" , "3"
],
[
    "NAME:poissons_ratio",
    "property_type:" , "AnisoProperty",
    "unit:" , "",
    "component1:" , "4",
    "component2:" , "5",
    "component3:" , "6"
```



```
],
...
])
```

Note: Young's Modulus and Poisson's Ratio properties are valid only for 2D and 3D Magnetostatic and Transient.

Python Example: Defining Anisotropic Hysteresis Model with Recoil-Line Relative Permeability

```
oDefinitionManager.AddMaterial(
[
    "NAME:Alnico5_HalfLoop_HysteresisModel_Anisotropic",
    "CoordinateSystemType:=", "Cartesian",
    "BulkOrSurfaceType:=", 1,
    [
        "NAME:PhysicsTypes",
        "set:=", ["Electromagnetic", "Thermal"]
    ],
    [
        "NAME:AttachedData",
        [
            "NAME:MatAppearanceData",
            "property_data:=", "appearance_data",
            "Red:=", 218,
```

```
"Green:=" , 222,  
"Blue:=" , 222  
]  
],  
"permittivity:=" , "1",  
[  
  "NAME:permeability",  
  "property_type:=" , "AnisoProperty",  
  "unit:=" , "",  
  [  
    "NAME:component1",  
    "property_type:=" , "nonlinear",  
    "BTypeForSingleCurve:=" , "normal",  
    "HUnit:=" , "A_per_meter",  
    "BUnit:=" , "tesla",  
    "IsTemperatureDependent:=", False,  
    [  
      "NAME:BHCoordinates",  
      [  
        "NAME:DimUnits",
```

```
    "",  
    ""  
  ],  
  [  
    "NAME:Point",  
    -101859.1636,  
    -1.44995289  
  ],  
  ...  
],  
[  
  "NAME:Temperatures"  
],  
"RecoilPermeability:=" , "1.5"  
],  
"component2:=" , "1",  
"component3:=" , "1"  
],  
"conductivity:=" , "2000000",  
"dielectric_loss_tangent:=", "0",  
"magnetic_loss_tangent:=", "0",
```

```
[
  "NAME:magnetic_coercivity",
  "property_type:=" , "VectorProperty",
  "Magnitude:=" , "-50910.9581943227A_per_meter",
  "DirComp1:=" , "0",
  "DirComp2:=" , "0",
  "DirComp3:=" , "0"
],
"thermal_conductivity:=", "50",
"saturation_mag:=" , "0gauss",
"lande_g_factor:=" , "2",
"delta_H:=" , "00e",
"mass_density:=" , "7300",
"specific_heat:=" , "400",
"thermal_expansion_coefficient:=", "1.1e-05"
])
```

CloneMaterial

Clones a local material.

UI Access	N/A

Parameters	Name	Type	Description
	<matName>	String	Name of existing material.
	<newName>	String	Name for newly cloned material.
Return Value	Boolean: 1 - Material is cloned. 0 - Existing material not found or a conflict with the new material name.		

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

ComputeCoreLossCoefficients

Creates a current source.

UI Access	N/A		
Parameters	Name	Type	Description
	<CoreLossModelType>	string	One of: Electrical Steel or Power Ferrite
	<MassDensity>	double	Mass density of the core loss model type
	<AttachedData>	Array	Either <CoreLossData> or <CoreLossMultiCurveData>
Return Value	Array of 3 double values in SI unit: - For Electrical Steel, returns the values for [Kh, Kc, Ke] - For Power Ferrite, returns the values for [Cm, X, Y]		

Python Syntax	ComputeCoreLossCoefficients (<CoreLossModelType>, <MassDensity>, <AttachedData>)
Python Example	<pre>DefinitionManager.ComputeCoreLossCoefficients("Electrical Steel", 7000, ["NAME:CoefficientSetupData", "property_data:=" , "coreloss_data", "coefficient_setup:=" , "w_per_cubic_mete", "Frequency:=" , "60Hz", "Thickness:=" , "0.5mm", "Conductivity:=" , "0", ["NAME:Coordinates", ["NAME:DimUnits", "", ""], ["NAME:Points",0,0,3,6,4,18]]])</pre>

DoesMaterialExist

Checks for the presence of a material in the library by name

UI Access	None.		
Parameters	Name	Type	Description
	<MaterialName>	<String>	Name of the material to search for in the material database

Return Value	Boolean: • True – specified material exists. • False – specified material does not exist.

Python Syntax	DoesMaterialExist(<MaterialName>)
Python Example	oDefinitionManager.DoesMaterialExist("modified_epoxy")

EditMaterial (Maxwell)

Use: Modifies an existing material.

UI Access	View/Edit Materials command in the material editor		
Parameters	Name	Type	Description
	<OriginalName>	String	Name of the material before editing.
	<NewName>	String	New name for the material
	<MatProperties>	Array	Structured array containing material properties. Refer to AddMaterial (Maxwell) for information on material properties.
Return Value	None		

Python Syntax	EditMaterial (<OriginalName>, <NewName>, <MatProperties>)
Python Examp-	Example: Nonlinear Conductivity

le	<pre>oDefinitionManager.EditMaterial("nonlinear_copper", [... ["NAME:conductivity", "property_type:=" , "nonlinear", ["NAME:JECordinates", ["NAME:DimUnits", "V_per_meter", "A_per_m2"], ["NAME:Point", 0, 0], ["NAME:Point", 1, 2], ["NAME:Point", 3, 4]]]])</pre>
	Example using a Thermal Modifier for Electrical Steel Core Loss properties


```

#applicable to Electrical Steel and Power Ferrite core loss models in 3D AC Magnetic and 3D Tran-
sient designs)
oDefinitionManager.EditMaterial("copper",
[
  "NAME:copper",
  "CoordinateSystemType:=", "Cartesian",
  "BulkOrSurfaceType:="    , 1,
  [
    "NAME:PhysicsTypes",
    "set:="    , ["Electromagnetic","Thermal","Structural"]
  ],
  [
    "NAME:AttachedData",
    [
      "NAME:MatAppearanceData",
      "property_data:="    , "appearance_data"
    ]
  ],
  [
    "NAME:ModifierData",
    [
      "NAME:ThermalModifierData",
      "modifier_data:="    , "thermal_modifier_data",
      [
        "NAME:all_thermal_modifiers",
        [
          "NAME:one_thermal_modifier",
          "Property:="    , "core_loss_kh",
          "Index:="    , 0,
          "prop_modifier:="    , "thermal_modifier",
          "use_free_form:="    , True,
          "free_form_value:="    , "if(Temp > 1000cel, 1, if(Temp -273.15cel, 1, 1))"
        ],
        [
          "NAME:one_thermal_modifier",

```

```

        "Property::="      , "core_loss_kc",
        "Index::="        , 0,
        "prop_modifier::=" , "thermal_modifier",
        "use_free_form::=" , True,
        "free_form_value::=" , "if(Temp > 1000cel, 1, if(Temp -273.15cel, 1, 1))"
    ],
    [
        "NAME:one_thermal_modifier",
        "Property::="      , "core_loss_ke",
        "Index::="        , 0,
        "prop_modifier::=" , "thermal_modifier",
        "use_free_form::=" , True,
        "free_form_value::=" , "if(Temp > 1000cel, 1, if(Temp -273.15cel, 1, 1))"
    ],
    [
        "NAME:one_thermal_modifier",
        "Property::="      , "core_loss_kdc",
        "Index::="        , 0,
        "prop_modifier::=" , "thermal_modifier",
        "use_free_form::=" , True,
        "free_form_value::=" , "if(Temp > 1000cel, 1, if(Temp -273.15cel, 1, 1))"
    ],
    [
        "NAME:one_thermal_modifier",
        "Property::="      , "core_loss_equiv_cut_depth",
        "Index::="        , 0,
        "prop_modifier::=" , "thermal_modifier",
        "use_free_form::=" , True,
        "free_form_value::=" , "if(Temp > 1000cel, 1, if(Temp -273.15cel, 1, 1))"
    ]
]
]
]

```

	])

Additional Examples

Note: Young's Modulus and Poisson's Ratio properties are valid only for 2D & 3D Magnetostatic and Transient.

Example of Editing Young's Modulus and Poisson's Ratio:

```
oDefinitionManager.EditMaterial("copper",
[
...
[
  "NAME:youngs_modulus",
  "property_type:" , "AnisoProperty",
  "unit:" , "",
  "component1:" , "1",
  "component2:" , "1.5",
  "component3:" , "3"
],
[
  "NAME:poissons_ratio",
  "property_type:" , "AnisoProperty",
  "unit:" , "",
  "component1:" , "4",
```

```
"component2:=" , "5",  
"component3:=" , "6"  
],  
...  
])
```

Example: Magnetostrictive stress and deformation with full magnetostriction matrix [3x3]

```
oDefinitionManager = oProject.GetDefinitionManager()  
oDefinitionManager.EditMaterial("vacuum",  
[  
    "NAME:vacuum",  
    "CoordinateSystemType:=", "Cartesian",  
    "BulkOrSurfaceType:="      , 1,  
    [  
        "NAME:PhysicsTypes",  
        "set:="                , ["Electromagnetic"]  
    ],  
    [  
        "NAME:AttachedData",  
        [  
            "NAME:MatAppearanceData",  
            "property_data:="      , "appearance_data",  
            "Red:="                , 230,  
            "Green:="              , 230,  
            "Blue:="               , 230,  
            "Transparency:="       , 0.949999988079071  
        ]  
    ],  
    "permittivity:="            , "1",
```

```

[
  "NAME:magnetostriction",
  "property_type:="      , "CustomProperty",
  "PropertyModelType:="  , 4, # Value of 1=Linear, 2=Nonlinear, 3=Anisotropic, 4=Matrix
  "Dir1ModelType:="      , 0,
  "Dir2ModelType:="      , 0,
  "IsoCoeff:="           , "0",
  "IsoHMax:="             , "0A_per_meter",
  "IsoHMin:="             , "0A_per_meter",
  "Dir1Coeff:="           , "0",
  "Dir1HMax:="            , "0A_per_meter",
  "Dir1HMin:="            , "0A_per_meter",
  "Dir1VecX:="            , "1",
  "Dir1VecY:="            , "0",
  "Dir1VecZ:="            , "0",
  "Dir2Coeff:="           , "0",
  "Dir2HMax:="            , "0A_per_meter",
  "Dir2HMin:="            , "0A_per_meter",
  "CoeffMatrixSize:="    , 3, # <int> - [0, 3].
  #If PropertyModelType is 4, (Matrix type), size cannot be 0
  # CoeffMatrixSize = 3 defines a 3x3 matrix size,
  #and matrix elements (C11, C12, .. C33) are specified in CoeffMatrix block
  "IsCoeffMatrixSymmetric:=", False, # takes a Boolean value
  [
    "NAME:CoeffMatrix",
    "C11:="                , "1.05",
    "C12:="                , "2",
    "C13:="                , "3",
    "C21:="                , "4",
    "C22:="                , "5",
    "C23:="                , "6",
    "C31:="                , "7",
    "C32:="                , "8",
    "C33:="                , "9"
  ]
]

```

```
],
"CoeffMatrixHMax:" , "12A_per_meter", #Value of Hmax
[
  "NAME:Magnetostriction",
  [
    "NAME:Isotropic",
    [
      "NAME:OneVariation",
      "DatasetMapsName:" , "Lambda-H curves versus Stress"
    ]
  ],
  [
    "NAME:Direction1",
    [
      "NAME:OneVariation",
      "DatasetMapsName:" , "Lambda-H curves versus Stress"
    ]
  ],
  [
    "NAME:Direction2",
    [
      "NAME:OneVariation",
      "DatasetMapsName:" , "Lambda-H curves versus Stress"
    ]
  ],
  "SelectedDatasetMapsName:" , "Lambda-H curves versus Stress"
]
]
])
```

Note: Core Loss Y Parameter for Electrical Steel is valid only for 2D & 3D AC Magnetic and Transient Designs.

Example: Defining Anisotropic Hysteresis Model with Recoil-Line Relative Permeability

```
oDefinitionManager = oProject.GetDefinitionManager()
oDefinitionManager.EditMaterial("Alnico5_HalfLoop_HysteresisModel_Anisotropic",
[
  "NAME:Alnico5_HalfLoop_HysteresisModel_Anisotropic",
  "CoordinateSystemType:=", "Cartesian",
  "BulkOrSurfaceType:=" , 1,
  [
    "NAME:PhysicsTypes",
    "set:=" , ["Electromagnetic","Thermal"]
  ],
  [
    "NAME:AttachedData",
    [
      "NAME:MatAppearanceData",
      "property_data:=" , "appearance_data",
      "Red:=" , 218,
      "Green:=" , 222,
      "Blue:=" , 222
    ]
  ]
]
```

```
    ]
  ],
  "permittivity:=" , "1",
  [
    "NAME:permeability",
    "property_type:=" , "AnisoProperty",
    "unit:=" , "",
    [
      "NAME:component1",
      "property_type:=" , "nonlinear",
      "BTypeForSingleCurve:=" , "normal",
      "HUnit:=" , "A_per_meter",
      "BUnit:=" , "tesla",
      "IsTemperatureDependent:=", False,
      [
        "NAME:BHCoordinates",
        [
          "NAME:DimUnits",
          "",
          ""
        ]
      ]
    ]
  ]
]
```



```
    ],  
    [  
        "NAME:Point",  
        -101859.1636,  
        -1.44995289  
    ],  
    ...  
],  
[  
    "NAME:Temperatures"  
],  
    "RecoilPermeability:=" , "1.5"  
],  
    "component2:=" , "1",  
    "component3:=" , "1"  
],  
    "conductivity:=" , "2000000",  
    "dielectric_loss_tangent:=" , "0",  
    "magnetic_loss_tangent:=" , "0",  
[  
    "NAME:magnetic_coercivity",
```

```
"property_type:=" , "VectorProperty",  
"Magnitude:=" , "-50910.9581943227A_per_meter",  
"DirComp1:=" , "0",  
"DirComp2:=" , "0",  
"DirComp3:=" , "0"  
],  
"thermal_conductivity:=", "50",  
"saturation_mag:=" , "0gauss",  
"lande_g_factor:=" , "2",  
"delta_H:=" , "00e",  
"mass_density:=" , "7300",  
"specific_heat:=" , "400",  
"thermal_expansion_coefficient:=", "1.1e-05"  
])
```

Example: Core Loss Y Parameter for Electrical Steel (Simple Property)

```
oDefinitionManager = oProject.GetDefinitionManager()  
oDefinitionManager.EditMaterial("steel_1008",  
[  
"NAME:steel_1008",  
"CoordinateSystemType:=", "Cartesian",
```

```
"BulkOrSurfaceType:=" , 1,  
[  
"NAME:PhysicsTypes",  
"set:=" , ["Electromagnetic","Thermal","Structural"]  
],  
...  
...  
[  
"NAME:core_loss_type",  
"property_type:=" , "ChoiceProperty",  
"Choice:=" , "Electrical Steel"  
],  
"core_loss_kh:=" , "105.359604403558",  
"core_loss_kc:=" , "0",  
"core_loss_ke:=" , "2.08295141230884",  
"core_loss_kdc:=" , "0",  
"mass_density:=" , "7872",  
[  
"NAME:stacking_type",  
"property_type:=" , "ChoiceProperty",  
"Choice:=" , "Litz Wire"
```

```
],  
"specific_heat:=" , "481",  
"youngs_modulus:=" , "200000000000",  
"poissons_ratio:=" , "0.25",  
"thermal_expansion_coefficient:=" , "1.26e-05",  
"core_loss_equiv_cut_depth:=" , "0.001meter",  
[  
"NAME:wire_type",  
"property_type:=" , "ChoiceProperty",  
"Choice:=" , "Round"  
],  
"strand_number:=" , "12",  
"wire_diameter:=" , "3mm",  
"core_loss_es_y:=" , "2.3"  
])
```

Note: Core Loss Y Parameter for Electrical Steel is valid only for 2D & 3D AC Magnetic and Transient Designs.

Example: Core Loss Y Parameter for Electrical Steel (Anisotropic Property)

```
oDefinitionManager = oProject.GetDefinitionManager()
```

```
oDefinitionManager.EditMaterial("steel_1008",
[
  "NAME:steel_1008",
  "CoordinateSystemType:=", "Cartesian",
  "BulkOrSurfaceType:=" , 1,

  [
    "NAME:PhysicsTypes",
    "set:=" , ["Electromagnetic","Thermal","Structural"]
  ],
  ...
  ...

  [
    "NAME:core_loss_type",
    "property_type:=" , "ChoiceProperty",
    "Choice:=" , "Electrical Steel"
  ],
  "core_loss_kh:=" , "105.359604403558",
  "core_loss_kc:=" , "0",
  "core_loss_ke:=" , "2.08295141230884",
  "core_loss_kdc:=" , "0",
  "mass_density:=" , "7872",

  [
    "NAME:stacking_type",
    "property_type:=" , "ChoiceProperty",
    "Choice:=" , "Litz Wire"
```

```
],  
  
"specific_heat:=" , "481",  
"youngs_modulus:=" , "200000000000",  
"poissons_ratio:=" , "0.25",  
"thermal_expansion_coefficient:=" , "1.26e-05",  
"core_loss_equiv_cut_depth:=" , "0.001meter",  
  
[  
  "NAME:wire_type",  
  "property_type:=" , "ChoiceProperty",  
  "Choice:=" , "Round"  
],  
  
"strand_number:=" , "12",  
"wire_diameter:=" , "3mm",  
  
[  
  "NAME:core_loss_es_y",  
  "property_type:=" , "AnisoProperty",  
  "unit:=" , "",  
  "component1:=" , "1",  
  "component2:=" , "2",  
  "component3:=" , "3" ]  
])
```

GetProjextMaterialNames

Returns the material names belonging to an active Project.

UI Access	N/A

Parameters	Name	Type	Description
	None		
Return Value	String names of the materials in the active project.		

Python Syntax	GetProjectMaterialNames()
Python Example	<pre>oProject = oDesktop.GetActiveProject() oDefinitionManager = oProject.GetDefinitionManager() materials = oDefinitionManager.GetProjectMaterialNames() AddWarningMessage(str(materials))</pre>

ExportMaterial

Exports a local material to a library.

UI Access	Export to Library command in the material editor.		
Parameters	Name	Type	Description
	<ExportData>	Array	["NAME:<LibraryName>", <MaterialName>, <MaterialName>, ...]
	<LibraryName>	String	Name of the exported library.
	<MaterialName>	String	Name of the material to be exported.
	<LibraryLocation>	String	Location to save the library. Only "PersonalLib" and "UserLib" are allowed.

Return Value	None.

Python Syntax	<code>ExportMaterial (<ExportData>, <LibraryLocation>)</code>
Python Example	<pre>oDefinitionManager.ExportMaterial (["NAME:mylib",_ "Material1", "Material2", "Material3"], "PersonalLib")</pre>

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, Maxwell assumes the material is a project material. The last two parameters will be ignored. If False, the material is not a project material.
	<LibraryName>	String	Name of the user or personal library where the material resides.
	<LibraryLocation>	String	Location of library. Valid options:"UserLib" or "PersonalLib".
Return Value	None.		

Python Syntax	<code>RemoveMaterial (<MaterialName>, <IsProjectMaterial>, <LibraryName>, <LibraryLocation>)</code>
Python Example	<pre>oDefinitionManager.RemoveMaterial (["Material1", false, "mo0907", "UserLib"])</pre>

RemoveUnusedDefinitions

Removes any unused project definitions.

UI Access	Tools > Project Tools > Remove Unused Definitions.		
Parameters	Name	Type	Description
	<Definitions>	Array	Definitions to be removed, such as materials and surface materials.
Return Value	None.		

Python Syntax	<code>RemoveUnusedDefinitions(<Definitions>)</code>
Python Example	<pre>oProject.RemoveUnusedDefinitions (["NAME:Materials", "Al-Extruded"], [</pre>

	<pre>"NAME:SurfaceMaterials", "Steel-oxidised-surface"]])</pre>

Component Manager Script Commands

The component manager provides access to components in a project. The manager object is accessed via the definition manager.

```
Set oDefinitionManager = oProject.GetDefinitionManager()
```

```
Set oComponentManager = oDefinitionManager.GetManager("Component")
```

The topics for this section include:

[Add](#)

[AddNPortData](#)

[AddSolverOnDemandModel](#)

[ClearSolutionCache](#)

[Edit](#)

[EditSolverOnDemandModel](#)

[EditWithComps](#)

[Export](#)

[GetNPortData](#)

[GetSolverOnDemandData](#)

[GetSolverOnDemandModelList](#)

[Remove](#)

[RemoveSolverOnDemandMode](#)

[UpdateDynamicLink](#)

Add [component manager]

*Use:*Add a component

Command: Tools > Edit Configured Libraries > Components > Add Component

*Syntax:*Add Array("NAME:<ComponentName>",

 "Info:=", <ComponentInfo>,

 "RefBase:=", <string>, // reference designator

 "NumParts:=", <int>, // parts per component

 "OriginalComponent:=", <string>

 "Terminal:=", <TerminalInfo>,

 "Terminal:=", <TerminalInfo>, ...

 // The remaining parameters are optional

 Array("NAME:Parameters", // any combo of the following

 "VariableProp:=", <VariableInfo>,

 "CheckboxProp:=", <CheckBoxInfo>,

 "ButtonProp:=", <ButtonInfo>,

 "TextProp:=", <TextInfo>,

 "NumberProp:=", <NumberInfo>,

```
"SeparatorProp:=", <SeparatorInfo>,
"ValueProp:=", <ValueInfo>,
"MenuProp:=", <MenuInfo>),
Array("NAME:Properties", // any combo of the following
"CheckboxProp:=", <CheckBoxInfo>,
"TextProp:=", <TextInfo>,
"NumberProp:=", <NumberInfo>,
"SeparatorProp:=", <SeparatorInfo>,
"ValueProp:=", <ValueInfo>,
"MenuProp:=", <MenuInfo>),
"VPointProp:=", <VPointInfo>,
"PointProp:=", <PointInfo>),
Array("Quantities",
"QuantityProp:=", <QuantityPropInfo>...),
Array("NAME:CosimDefinitions",
<CosimDefInfo>,
<CosimDefInfo>...)
```

Return Value:<string>

// composite name of the component.

// If the name requested conflicts with the name of an existing

// component, the requested name is altered to be unique.
// The name returned reflects any change made to be unique.

Parameters: <ComponentName>:

<string> // simple name of the component

<ComponentInfo>:

Array("Type:=", <TypeInfo>,

"NumTerminals:=", <int>,

"DataSource:=", <string>,

"ModifiedOn:=", <ModifiedOnInfo>,

"Manufacturer:=", "<string>,"

"Symbol:=", <string>,"

"Footprint:=", <string>,"

"Description:=", <string>,"

"InfoTopic:=", <string>,"

"InfoHelpFile:=", <string>,"

"IconFile:=", <string>,"

"LibraryName:=", "",

"OriginalLocation:=", "Project", // Project Location

"Author:=", <string>,"

"OriginalAuthor:=", <string>,

"CreationDate:= ", <int>)

<TypeInfo>:

An integer that is the or-ing of bits for each product listed below. The default setting is 0xffffffff (4294967295) which indicates valid for all products. In the component editing dialog, checking different boxes in the "Specify products for which this component is valid" grid control sets specific flags that correspond to the following hex/decimal settings:

Nexxim -- 100 binary, 4 decimal, 0x4

SlwaveDeNovo -- 1000 binary, 8 decimal, 0x8

Simplorer -- 10000 binary, 16 decimal, 0x10

MaxwellCircuit -- 100000 binary, 32 decimal, 0x20

<ModifiedOnInfo>:

An integer that corresponds to the number of seconds that have elapsed since 00:00 hours, Jan 1, 1970 UTC from the system clock.

<TerminalInfo>:

Array(<string>, // symbol pin

<string> // footprint pin

<string >, // gate name

<bool>, // shared

<int>, // equivalence number

<int>, // what to do if unconnected: flag as error:0, ignore:1

<string> // description

<Nature>)

<Nature>:

<string> // content varies as follows

Nexxim/Circuit:

"Electrical" // the only choice

Simplorer:

// several choices

"Electrical", "Magnetic", "Fluidic", "Translational",

"Translational_V", "Rotational", "Rotational_V",

""Radiant", "Thermal", or <VHDLPackageName>

<VHDLPackageName>:

<string> // in the form <Library>.<Package>

<Library>:

<string> // name of the VHDL library

<Package>:

<string> // name of the VHDL package

<VariableInfo>:

Array(<string>, // name

<FlagLetters>,

<string>, // description

"CB:=", <string>, // optional - script for call back

<string>) // value: number, variable, or expression

<FlagLetters>:

<string> // "D" - has description parameter,

// "RD" - readonly & has description parameter,

// or "RHD" - readonly, hidden, & has description parameter

<CheckBoxInfo>:

Array(<string>, // name

<FlagLetters>,


```
<string>, // description  
"CB:=", <string>, // optional - script for call back  
<bool>) // value: true or false
```

```
<ButtonInfo>:  
Array(<string>, // name  
<FlagLetters>,  
<string>, // description  
<string>, // button title  
<string>, // extra text  
<ClientID>,  
"ButtonPropClientData:= ", <ClientDataArray>)
```

```
<ClientID>:  
<int> // specifies Button Prop Client  
// 0 - unknown, ButtonPropClientData  
// array will be empty  
// 1 - Netlist Prop Client  
// 2 - not used  
// 3 - File Name Prop Client
```

<ClientDataArray>:

varies with <ClientID>

<ClientID> is 0 or 1: empty array

Array()

<ClientID> is 3:

Array("InternalFormatText:=", "<prefix><RelativePath>")

<prefix>:

<string> // "<Project>", "<PersonalLib>", "<UserLib>", or "<SysLib>"

<RelativePath>:

<string> // relative path to file from <prefix>

<TextInfo>:

Array(<string>, // name

<FlagLetters>,

<string>, // description

"CB:=", <string>, // optional - script for call back

<string>) // value: a text string

<NumberInfo>:

Array(<string>, // name

<FlagLetters>,

<string>, // description

"CB:=", <string>, // optional - script for call back

<real>, // value: a number

<string>) // units

<SeparatorInfo>:

Array(<string>, // name

<FlagLetters>,

<string>, // description

"CB:=", <string>, // optional - script for call back

<string>) // value: a text string

<ValueInfo>:

Array(<string>, // name

<FlagLetters>,

<string>, // description

"CB:=", <string>, // optional - script for call back

<string>) // value: a number, variable or expression

<MenuPropInfo>:

Array(<string>, // name

<FlagLetters>,

<string>, // description

<string>, // menu choices - separated by commas

<int>) // 0 based index of current menu choice

<VPointInfo>:

Array(<string>, // name

<FlagLetters>,

<string>, // description

"CB:=", <string>, // optional - script for call back

<string>, // x value: number with length units

<string>) // y value: number with length units

<PointInfo>:

Array(<string>, // name

<FlagLetters>,

<string>, // description

"CB:=", <string>, // optional - script for call back

<real>, // x value

<real>) // y value

<QuantityPropInfo>:

Array(<string>, // name

<FlagLetters>,

<string>, // description

<string>, // value

<TypeString>,

<TypeStringDependentInfo>)

<TypeString>:

<string> // "Across", "Through", or "Free"

<TypeStringDependentInfo>:

<TypeString> is "Free" :

<string>, // direction: "In", "Out", "InOut", or "DontCare"

// Following <string> is not present if direction is "DontCare"

<string> // when to calculate: "BeforeAnalogSolver",

// "BeforeStateGraph", "AfterStateGraph", or "DontCareWhen"

<TypeString> is "Across" or "Through":

<int>, // terminal 1

<int> // terminal 2

<CosimDefInfo>:

Array("NAME:CosimDefinition",

"CosimulatorType:=", <int>,

"CosimDefName:=", <string> // "HFSS 3D Layout", "Circuit",

// "Custom", or "Netlist"

"IsDefinition:=", <bool>,

final array member(s) vary with CosimDefName)

final array members for HFSS 3D Layout:

"CosimStackup:=", <string>,

"CosimDmbedRatio:=", <int>

final array members for Circuit:

"ExportAsNport:=", <int>,

"UsePjt:=", <int>

final array member for Custom:

"DefinitionCompName:=", <string>

final array member for Netlist:

"NetlistString:=", <string>

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>,

	<pre>"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>, "PointProp:=", <PointInfo>], ["Quantities", "QuantityProp:=", <QuantityPropInfo>...], ["NAME:CosimDefinitions", <CosimDefInfo>, <CosimDefInfo>...]]</pre>
Python Example	<pre>oComponentManager.Add(["NAME:Component",</pre>


```
"Info:=", [  
  "Type:=", 0,  
  "NumTerminals:=", 0,  
  "DataSource:=", "",  
  "ModifiedOn:=", 1467910752,  
  "Manufacturer:=", "",  
  "Symbol:=", "Component",  
  "ModelNames:=", "",  
  "Footprint:=", "",  
  "Description:=", "",  
  "InfoTopic:=", "",  
  "InfoHelpFile:=", "",  
  "IconFile:=", "",  
  "Library:=", "",  
  "OriginalLocation:=", "Project",  
  "IEEE:=", "",  
  "Author:=", "",  
  "OriginalAuthor:=", "",  
  "CreationDate:=", 1467910746,  
  "ExampleFile:=", ""],  
  "Refbase:=", "U",
```

	<pre>"NumParts:=", 1, "ModSinceLib:=", True, "CompExtID:=", 2])</pre>

AddNPortData [component manager]

Use: Adds a component using the specified data

Command: Project Menu > Add Model > Add Nport Model

Syntax: AddNPortData Array("NAME:<ComponentDataName>",
"ComponentDataType:=", "NportData",
"name:=", <string>, // Name of the item
"filename:=", <string>, // Path to the file to find the data
"numberofports:=", <int>,
"filelocation:=", <LocationType>,
"domain:=", <string>, // "time" or "frequency"
"datamode:=", <string> // "EnterData", "Import", or "Link"
"devicename:=", <string>,
"ImpedanceTab:=", <bool>,
"NoiseDataTab:=", <bool>,
"DCBehaviorTab:=", <bool>,
"SolutionName:=", <string>,

```
"displayformat:=", <DisplayInfo>,  
"datatype:=", <string>, // "SMatrix", "YMatrix", or "ZMatrix"  
"ShowRefPin:=", <bool>,  
"RefNodeCheckbox:=", <bool>, ...  
<ProductOptionsInfo>)
```

Return Value: <string>

```
// composite name of the component.  
// If the name requested conflicts with the name of an existing  
// component, the requested name is altered to be unique.  
// The name returned reflects any change made to be unique.
```

Parameters: <ComponentDataName>:

```
<string> // simple name of the component
```

<LocationType>:

```
<string> // one of "UsePath", "PersonalLib", "UserLib", "SysLib",  
// or "Project".
```

<dcInfo>:

```
<string> // one of "DCOpen", "DCShort", "DCShShort",  
// "DCNone", or "DCEmpty".
```

<DisplayInfo>:

<string> // one of "MagnitudePhase", "ReallImaginary",
// or "DbPhase".

<ProductOptionsInfo>:

// The remaining parameters differ by product

// HFSS 3D Layout - doesn't support interpolation/DC behavior

"DCOption:=", -1,

"InterpOption:=", -1,

"ExtrapOption:=", -1,

"DataType:=", 0

// Nexxim

"DCOption:=", <NexximDCOption>,

"InterpOption:=", <NexximInterpOption>,

"ExtrapOption:=", <NexximExtrapOption>,

"DataType:=", 2

<NexximDCOption>:

<int> // 0 : Zero Padding

// 1 : Same as last point

// 2 : Linear extrapolation from last 2 points

// 3 : Constant magnitude, linear phase extrapolation

// 4 : Leave all signal lines open circuited

// 5 : Short all signal lines together

// 6 : Short all signal lines to ground

<NexximInterpOption>:

<int> // 0 : Step

// 1 : Linear

<NexximExtrapOption>:

<int> // 0 : Zero padding

// 1 : Same as last point

// 2 : Linear extrapolation from last 2 points

// 3 : Constant magnitude, linear phase extrapolation

<CircuitDCOption>:

<int> // 0 : Leave all signal lines open circuited

// 1 : Short all signal lines together
// 2 : Short all signal lines to ground
// 3 : Extrapolate from data provided (not recommended)

<CircuitInterpOption>:

<int> // 0 : Linear

// 1 : Cubic spline

// 2 : Rational polynomial

<CircuitExtrapOption>:

<int> // 0 : Same as interpolation

// 1 : Zero padding

// 2 : Same as last point

AddSolverOnDemandModel

Use: This method looks for a local component of the name passed in, and to this component it adds an SOD model definition using the information passed in the VARIANT. It returns the name of the SOD model added.

Parameters: BSTR component name.

Parameters: VARIANT which is the SOD model data.

Return Value: Returns the name of the model added.

ClearSolutionCache [component manager]

Use: Clear the solution cache for dynamic link component.

Command: Each of the following commands will clear the solution cache:

- **Dynamic Link Item RCM > Clear Solution Cache**
- **Dynamic Link Component in schematic RCM > Clear Solution Cache**

Syntax: ClearSolutionCache <Component Name >

Return Value: None

Parameters: <component name> is the name of the dynamic link component

Edit [component manager]

Modifies an existing component

Command: Tools > Edit Configured Libraries > Components > Edit Component

Syntax: Edit <ComponentName>,

```
    Array("NAME:<NewComponentName>",  
    "Info:=", <ComponentInfo>,  
    "RefBase:=", <string>, // reference designator  
    "NumParts:=", <int>, // parts per component  
    "OriginalComponent:=", <string>  
    "Terminal:=", <TerminalInfo>,  
    "Terminal:=", <TerminalInfo>, ...  
    // The remaining parameters are optional  
    Array("NAME:Parameters", // any combo of the following
```

```
"VariableProp:=", <VariableInfo>,  
"CheckboxProp:=", <CheckBoxInfo>,  
"ButtonProp:=", <ButtonInfo>,  
"TextProp:=", <TextInfo>,  
"NumberProp:=", <NumberInfo>,  
"SeparatorProp:=", <SeparatorInfo>,  
"ValueProp:=", <ValueInfo>,  
"MenuProp:=", <MenuInfo>),  
Array("NAME:Properties", // any combo of the following  
"CheckboxProp:=", <CheckBoxInfo>,  
"TextProp:=", <TextInfo>,  
"NumberProp:=", <NumberInfo>,  
"SeparatorProp:=", <SeparatorInfo>,  
"ValueProp:=", <ValueInfo>,  
"MenuProp:=", <MenuInfo>),  
"VPointProp:=", <VPointInfo>,  
"PointProp:=", <PointInfo>),  
Array("Quantities",  
"QuantityProp:=", <QuantityPropInfo>...),  
Array("NAME:CosimDefinitions",
```



```
<CosimDefInfo>,  
<CosimDefInfo>...)
```

Return Value: <string>

```
// composite name of the component.  
// If the name requested conflicts with the name of an existing  
// component, the requested name is altered to be unique.  
// The name returned reflects any change made to be unique.
```

Parameters: <ComponentName>:

```
<string> // composite name of the component to edit
```

```
<NewComponentName>:
```

```
<string> // new simple name for the component
```

```
<ComponentInfo>:
```

```
Array("Type:=", <TypeInfo>,  
"NumTerminals:=", <int>,  
"DataSource:=", <string>,  
"ModifiedOn:=", <ModifiedOnInfo>,  
"Manufacturer:=", "<string>,  
"Symbol:=", <string>,
```

```
"Footprint:=", <string>,  
"Description:=", <string>,  
"InfoTopic:=", <string>,  
"InfoHelpFile:=", <string>,  
"IconFile:=", <string>,  
"LibraryName:=", <string>,  
"OriginalLocation:=", <string>, // Project Location  
"Author:=", <string>,  
"OriginalAuthor:=", <string>,  
"CreationDate:= ", <int>)
```

<TypeInfo>:

An integer that is the or-ing of bits for each product listed below. The default setting is 0xffffffff (4294967295) which indicates valid for all products. In the component editing dialog, checking different boxes in the "Specify products for which this component is valid" grid control sets specific flags that correspond to the following hex/decimal settings:

Nexxim -- 100 binary, 4 decimal, 0x4

SIwaveDeNovo -- 1000 binary, 8 decimal, 0x8

Simplorer -- 10000 binary, 16 decimal, 0x10

MaxwellCircuit -- 100000 binary, 32 decimal, 0x20

<ModifiedOnInfo>:

An integer that corresponds to the number of seconds that have elapsed since 00:00 hours, Jan 1, 1970 UTC from the system clock.

<TerminalInfo>:

Array(<string>, // symbol pin

<string> // footprint pin

<string>, // gate name

<bool>, // shared

<int>, // equivalence number

<int>, // what to do if unconnected: flag as error:0, ignore:1

<string>, // description

<Nature>)

<Nature>:

<string> // content varies as follows

Nexxim/Circuit:

"Electrical" // the only choice

Simplorer:

// several choices

"Electrical", "Magnetic", "Fluidic", "Translational",

"Translational_V", "Rotational", "Rotational_V",
""Radiant", "Thermal", or <VHDLPackageName>

<VHDLPackageName>:

<string> // in the form <Library>.<Package>

<Library>:

<string> // name of the VHDL library

<Package>:

<string> // name of the VHDL package

<VariableInfo>:

Array(<string>, // name

<FlagLetters>,

<string>, // description

"CB:=", <string>, // optional - script for call back

<string>) // value: number, variable, or expression

<FlagLetters>:

<string> // "D" - has description parameter,
// "RD" - readonly & has description parameter,
// or "RHD" - readonly, hidden, & has description parameter

<CheckBoxInfo>:

Array(<string>, // name
<FlagLetters>,
<string>, // description
"CB:=", <string>, // optional - script for call back
<bool>) // value: true or false

<ButtonInfo>:

Array(<string>, // name
<FlagLetters>,
<string>, // description
<string>, // button title
<string>, // extra text
<ClientID>,
"ButtonPropClientData:= ", <ClientDataArray>)

<ClientID>:

<int> // specifies Button Prop Client

// 0 - unknown, "ButtonPropClientData

// array will be empty

// 1 - Netlist Prop Client

// 2 - not used

// 3 - File Name Prop Client

<ClientDataArray>:

varies with <ClientID>

<ClientID> is 0 or 1: empty array

Array()

<ClientID> is 3:

Array("InternalFormatText:=", "<prefix><RelativePath>")

<prefix>:

<string> // "<Project>", "<PersonalLib>", "<UserLib>", or "<SysLib>"

<RelativePath>:

<string> // relative path to file from <prefix>

<TextInfo>:

Array(<string>, // name

<FlagLetters>,

<string>, // description

"CB:=", <string>, // optional - script for call back

<string>) // value: a text string

<NumberInfo>:

Array(<string>, // name

<FlagLetters>,

<string>, // description

"CB:=", <string>, // optional - script for call back

<real>, // value: a number

<string>) // units

<SeparatorInfo>:

Array(<string>, // name

<FlagLetters>,

<string>, // description

"CB:=", <string>, // optional - script for call back

<string>) // value: a text string

<ValueInfo>:

Array(<string>, // name

<FlagLetters>,

<string>, // description

"CB:=", <string>, // optional - script for call back

<string>) // value: a number, variable or expression

<MenuPropInfo>:

Array(<string>, // name

<FlagLetters>,

<string>, // description

<string>, // menu choices - separated by commas

<int>) // 0 based index of current menu choice

<VPointInfo>:

Array(<string>, // name

<FlagLetters>,

<string>, // description


```
"CB:=", <string>, // optional - script for call back
<string>, // x value: number with length units
<string>) // y value: number with length units
```

```
<PointInfo>:
Array(<string>, // name
<FlagLetters>,
<string>, // description
"CB:=", <string>, // optional - script for call back
<real>, // x value
<real>) // y value
```

```
<QuantityPropInfo>:
Array(<string>, // name
<FlagLetters>,
<string>, // description
<string>, // value
<TypeString>,
<TypeStringDependentInfo>)
```

```
<TypeString>:
```

<string> // "Across", "Through", or "Free"

<TypeStringDependentInfo>:

"Free" :

<string>, // direction: "In", "Out", "InOut", or "DontCare"

// Following <string> is not present if direction is "DontCare"

<string> // when to calculate: "BeforeAnalogSolver",

// "BeforeStateGraph", "AfterStateGraph", or "DontCareWhen"

"Across" or "Through"

<int>, // terminal 1

<int> // terminal 2

<CosimDefInfo>:

Array("NAME:CosimDefinition",

"CosimulatorType:=", <int>,

"CosimDefName:=", <string> // "HFSS3D", "Circuit",

// "Custom", or "Netlist"

"IsDefinition:=", <bool>,

final array member(s) vary with CosimDefName)

final array members for HFSS 3D Layout:

```
"CosimStackup:=", <string>,  
"CosimDmbedRatio:=", <int>
```

final array members for Circuit:

```
"ExportAsNport:=", <int>,  
"UsePjt:=", <int>
```

final array member for Custom:

```
"DefinitionCompName:=", <string>
```

final array member for Netlist:

```
"NetlistString:=", <string>
```

Python Syntax	<pre>Edit <ComponentName>, ["NAME:<NewComponentName>"; "Info:=", <ComponentInfo>, "RefBase:=", <string>, // reference designator "NumParts:=", <int>, // parts per component "OriginalComponent:=", <string> "Terminal:=", <TerminalInfo></pre>

```
"Terminal:=", <TerminalInfo>, ...  
#The remaining parameters are optional  
["NAME:Parameters", // any combo of the following  
"VariableProp:=", <VariableInfo>,  
"CheckboxProp:=", <CheckBoxInfo>,  
"ButtonProp:=", <ButtonInfo>,  
"TextProp:=", <TextInfo>,  
"NumberProp:=", <NumberInfo>,  
"SeparatorProp:=", <SeparatorInfo>,  
"ValueProp:=", <ValueInfo>,  
"MenuProp:=", <MenuInfo>],  
["NAME:Properties", # any combo of the following  
"CheckboxProp:=", <CheckBoxInfo>,  
"TextProp:=", <TextInfo>,  
"NumberProp:=", <NumberInfo>,  
"SeparatorProp:=", <SeparatorInfo>,  
"ValueProp:=", <ValueInfo>,  
"MenuProp:=", <MenuInfo>),  
"VPointProp:=", <VPointInfo>,  
"PointProp:=", <PointInfo>),
```

	<pre> ["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", _ "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", _ </pre>

```

"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, _

```

	<pre>"ButtonPropClientData:=", [], "TextProp:=", ["ModelName", "HD", "", "Q"]])</pre>
Python Example 2	<pre>name2 = oComponentManager.Edit ("MyComponent",_ (["NAME:MyOtherComponent","Info:=", ["Type:=", 4294901767, _ "NumTerminals:=", 2, "DataSource:=", "", _ "ModifiedOn:=", 1071096503, "Manufacturer:=", "Ansoft", _ "Symbol:=", "bendo", "Footprint:=", "BENDO", _ "Description:=", "", "InfoTopic:=", "", _ "InfoHelpFile:=", "", "IconFile:=", "", _ "LibraryName:=", "", "OriginalLocation:=", "Project", _ "Author:=", "", "OriginalAuthor:=", "", _</pre>

```
"CreationDate:= ", 1147460679], "Refbase:=", "U", _  
  
"NumParts:=", 1, "OriginalComponent:=", "", _  
  
"Terminal:=", ["n1", "n1", "A", false, 0, 0, "", _  
  
"Electrical"], "Terminal:=", ["n2", "n2", "A", _  
  
false, 1, 0, "", Electrical"], ["NAME:Parameters", _  
  
"MenuProp:=", ["CoSimulator", "D", "", _  
  
"Default,Custom,Netlist", 0], "ButtonProp:=", ["CosimDefinition", _  
  
"D", "", "", "Edit", 0, "ButtonPropClientData:=", []], _  
  
["NAME:CosimDefinitions", ["NAME:CosimDefinition", _
```



```
"CosimulatorType:=", 0, "CosimDefName:=", "HFSS3D", _  
  
"IsDefinition:=", true, "CosimStackup:=", "Layout stackup", _  
  
"CosimDmbedRatio:=", 3], ["NAME:CosimDefinition", _  
  
"CosimulatorType:=", 1, "CosimDefName:=", "", _  
  
"IsDefinition:=", true, "ExportAsNport:=", 0, _  
  
"UsePjt:=", 0], ["NAME:CosimDefinition", _  
  
"CosimulatorType:=", 2, "CosimDefName:=", "Custom", _  
  
"IsDefinition:=", true, "DefinitionCompName:=", ""], _  
  
["NAME:CosimDefinition", "CosimulatorType:=", 3, _  
  
"CosimDefName:=", "Netlist", "IsDefinition:=", true, _
```

	<code>"NetlistString:=", ""]]])</code>

EditSolverOnDemandModel

Use: This method looks for a local component of the name passed in, and in this component it looks for an SOD model using the name passed in the second BSTR. It modifies the SOD model using the data in the VARIANT. It returns the name of the SOD model edited.

Return Value: Returns the name of the model edited.

Parameters: BSTR component name.

Parameters: BSTR SOD model name.

Parameters: VARIANT which is the new SOD model data (can include changed name).

EditWithComps [component manager]

Edit an existing component.

Command: None

Syntax: EditWithComps <ComponentName>,
 Array("NAME:<NewComponentName>",
 "ModTime:=", <ModifiedTimeInfo>,
 "Library:=", <string>, // Library name
 "LibLocation:=", <string>, // Project Location
 <PinDefInfo>,
 <PinDefInfo>,... // optional, to define pins

```
<GraphicsDataInfo>, // optional, to define graphics  
<PropDisplayMapInfo>), // optional, to define property displays  
Array(<ListOfComponentNames>) // Component names
```

Return Value: <string>

```
// composite name of the component.  
// If the name requested conflicts with the name of an existing  
// component, the requested name is altered to be unique.  
// The name returned reflects any change made to be unique.
```

Parameters: <ComponentName>:

```
<string> // composite name of the component being edited
```

<NewComponentName>:

```
<string> // new simple name for the component
```

<ModifiedOnInfo>:

An integer that corresponds to the number of seconds that have elapsed
since 00:00 hours, Jan 1, 1970 UTC from the system clock.

<PinDefInfo>:

```
Array("NAME:PinDef",  
"Pin:=", Array (<string>, // pin name  
<real>, // x location  
<real>, // y location  
<real>, // angle in radians  
<PinType>,  
<real>, // line width  
<real>, // line length  
<bool>, // mirrored  
<int>, // color  
<bool>, // true if visible, false if not  
<string>, // hidden net name  
<OptionalPinInfo>, // optional info  
<PropDisplayMapInfo>)) // optional
```

```
<PinType>:  
<string> // "N" : normal pin  
// "I" : input pin  
// "O" : output pin
```

<OptionalPinInfo>:

// Specify both or neither

<bool>, // true if name is to be shown

<bool>, // true if number is to be shown

<PropDisplayMapInfo>:

Array("NAME:PropDisplayMap",

<PropDisplayInfo>,

<PropDisplayInfo>,...)

<PropDisplayInfo>:

<NameString>, Array(<DisplayTypeInfo>,

<DisplayLocationInfo>,

<int>, // optional, level number

<TextInfo>)

<NameString>:

<string> // PropertyName:=, where PropertyName is the name of

// the property to be displayed

<DisplayTypeInfo>:

<int> // 0 : No display
// 1 : Display name only
// 2 : Display value only
// 3 : Display both name and value
// 4: Display evaluated value only
// 5: Display both name and evaluated value

<DisplayLocationInfo>:

<int> // 0 : Left
// 1 : Top
// 2 : Right
// 3 : Bottom
// 4 : Center
// 5 : Custom placement

<GraphicsDataInfo>:

Array("NAME:Graphics",
// one or more of the following
<RectInfo>,
<CircleInfo>,

```
<ArcInfo>,  
<LineInfo>,  
<PolygonInfo>,  
<TextInfo>,  
<ImageInfo>)
```

```
<RectInfo>:  
"Rect:=", Array(<real>, // line width  
<int>, // fill pattern  
<int>, // color  
<real>, // angle, in radians  
<real>, // x position of center  
<real>, // y position of center  
<real>, // width  
< real>) // height
```

```
<CircleInfo>:  
"Circle:=", Array(<real>, // line width  
<int>, // fill pattern  
<int>, // color  
<real>, // x position of center
```

<real>, // y position of center

< real>) // radius

<ArcInfo>:

"Arc:=", Array(<real>, // line width

<int>, // line pattern

<int>, // color

<real>, // x position of center

<real>, // y position of center

< real>, // radius

<real>, // start angle, in radians

<end>) // end angle, in radians

<LineInfo>:

"Line:=", Array(<real>, // line width

<int>, // line pattern

<int>, // color

<PointInfo>, // must specify at least 2 points

<PointInfo>...)

<PointInfo>:

<real>, // x position

<real> // y position

<PolygonInfo>:

"Polygon:=", Array(<real>, // line width

<int>, // fill pattern

<int>, // color

<PointInfo>, // must specify at least 3 points

<PointInfo>...)

<TextInfo>:

"Text:=", Array(<real>, // x position

<real>, // y position

<real>, // angle, in radians

<Justification>,

<bool>, // is plotter font

<string>, // font name

<int>, // color

<string>) // text string

<Justification>:

<int> // 0 : left top
// 1 : left base
// 2 : left bottom
// 3 : center top
// 4 : center base
// 5 : center bottom
// 6 : right top
// 7 : right base
// 8 : right bottom

<ImageInfo>:
"Image:=", Array(<RectInfo>,
<ImageData>,
<bool>) // is mirrored

<ImageData>:
<string>, // file path
<int>, // 0 : use the file path and link to it
// 1 : ignore file path and use next parameter
<string> // text data, only present if preceding int is 1

<ListOfComponentNames>:

<string>,<string> ...

// The list may be empty. When not empty, each string that is listed is a component

// that references the component to be edited. Prior to editing, a clone of the component is

// made, and the components that are listed are modified so that they now refer to

// the clone.

Export [component manager]

Export component(s) to a library

Command: Tools > Edit Configured Libraries > Components > Export to Library

Syntax: Export Array("NAME:<LibraryName>",

<ComponentName>,

<ComponentName>...),

<LibraryLocation>

Return Value: None

Parameters: <LibraryName>:

<string> // name of the library

<ComponentName>:

<string> // composite name of the component to export

<LibraryLocation>:

<string> // location of the library in <LibraryName>

// One of "Project", "PersonalLib", or "UserLib"

Python Syntax	Export(["NAME:<LibraryName>", <ComponentName>, <ComponentName>...], <LibraryLocation>)
Python Example	<pre>oComponentManager.Export (["NAME:mylib", "Simplorer Circuit Elements\BJTs:Level01_NPN"], "PersonalLib")</pre>

GetNames [component manager]

Returns the names of the components (used and unused) in a design. The following script command, **IsUsed**, can then be used to separate used and unused components.

Command: None

Syntax: GetNames()

Return Value: An array of strings

Parameters: None

Python Syntax	GetNames()
Python Example	<pre>componentNames = oComponentManager.GetNames()</pre>

GetNPortData [component manager]

Returns NPort data for the component with the specified name.

Command: None

Return Value: Variant array, whose contents depend on the type of component. The array will be empty if the component does not have NPort data. See the syntax for AddDynamicNPortData and AddNPortData for descriptions of the array contents for components with those types of NPort data.

GetSolverOnDemandData

Use: This method looks for a local component of the name passed in, and in this component it looks for an SOD model of the name passed in and returns the SOD data pertaining to that model.

Parameters: BSTR component name.

Parameters: BSTR SOD model name

Return Value: VARIANT which is the SOD data.

GetSolverOnDemandModelList

Use: This method looks for a local component of the name passed in, and returns a list of SOD model names defined in the component.

Parameters: BSTR component name.

Return Value: VARIANT which is a list of SOD model names.

IsUsed [component manager]

Used to determine if a component is used in the design.

Command: None

Syntax: IsUsed(<ComponentName>)

Return Value: <Boolean> // true if the specified component is used in the design

Parameters: <ComponentName>:

<string>

Python Syntax	IsUsed(<ComponentName>)
Python Example	IsUsed = oComponentManager.IsUsed("MyComponent")

Remove [component manager]

Remove a component from a library

Command: Tools > Edit Configured Libraries > Components > Remove Component

Syntax: Remove <ComponentName>,

 <IsProjectComponent>,

 <LibraryName>,

 <LibraryLocation>

Return Value: None

Parameters: <ComponentName>:

 <string> // composite name of the component to remove

 <IsProjectComponent>:

 <bool>

 <LibraryName>:

 <string> // name of the library

 <LibraryLocation>:

<string> // location of the library in <LibraryName>

// One of "Project", "PersonalLib", or "UserLib"

Python Syntax	Remove (<ComponentName>, <IsProjectComponent>, <LibraryName>, <LibraryLocation>)
Python Example	<pre>oComponentManager.Remove ("Simplorer Circuit Elements\BJTs:Level01_NPN", _ true, "Project")</pre>

RemoveSolverOnDemandModel

Use: This method looks for a local component of the name passed in, and in this component it looks for an SOD model of the name passed in and deletes the SOD model definition from the component.

Parameters: BSTR component name.

Parameters: BSTR SOD model name

Return Value: None.

RemoveUnused [component manager]

Removes components that are not used in the design.

Command: **Project->Remove Unused Definitions** is similar but operates slightly different and does not record script commands.

Syntax: RemoveUnused()

Return Value: <bool> True if one or more components are removed.

Parameters: None

Note:

The order of calls to RemoveUnused is significant. As a result, removing definitions in an unordered fashion may cause other components in dependent definitions to be rendered unusable.

Also, the symbol and footprint of an unused component are not unusable until after the component itself is removed using the Component Manager Remove script.

Python Syntax	RemoveUnused()
Python Example	<code>removedDefs = oComponentManager.RemoveUnused()</code>

Update Dynamic Link [component manager]

Use: Reads data from the linked design and updates the dynamic link component. This will update the following: properties, solutions, ports, geometry.

Command: Each of the following commands will record a non-undoable script command:

- **Dynamic Link Item RCM > Refresh Dynamic Link**
- **Dynamic Link Component in schematic RCM > Refresh Dynamic Link**
- Click the **Refresh Dynamic Link** button in the Footprint tab of the **Properties Window** for selected Dynamic Link components in the Layout Editor.

Syntax: UpdateDynamicLink(<component name>)

Return Value: None

Parameters: <component name> is the name of the dynamic link component

Example:

```
oComponentManager.UpdateDynamicLink("TeeModel_L1")
```

Material Manager Script Commands

The material manager provides access to materials in a project. The manager object is accessed via the definition manager.

```
Set oDefinitionManager = oProject.GetDefinitionManager()
```

```
Set oMaterialManager = oDefinitionManager.GetManager("Material")
```

The topics for this section include:

[GetNames](#)

[GetProperties](#)

[IsUsed](#)

[RemoveUnused](#)

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()
```

Python Syntax	GetNames()
Python Example	<pre>materialnames = oMaterialManager.GetNames()</pre>

GetProperties [material manager]

Get material properties. Differs from GetData in that only material properties available to the user are returned by GetProperties.

UI Access	NA		
Parameters	Name	Type	Description
	<material_name>	string Boolean	Name of the project material True or False. If you use False or only a single argument, then GetProperties returns only the properties that are different from the defaults.
Return Value	Array of material data		

Python Syntax	GetProperties (<materialname>)
Python Example	<pre>oMaterialManager.GetProperties("Gold", True) oMaterialManager.GetProperties("vacuum")</pre>

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.

Python Syntax	IsUsed(<ComboName>)
Python Example	<code>isused = oMaterialManager.IsUsed("mylib:mymaterial")</code>

RemoveUnused [material manager]

Use: Remove all unused materials from the project.

Command: None

Syntax: RemoveUnused

Return Value: None

Parameters: None

Python Syntax	RemoveUnused()
Python Example	<code>thereWereExtras = oMaterialManager.RemoveUnused()</code>

Model Manager Script Commands

The model manager provides access to models in a project. The manager object is accessed via the definition manager.

```
Set oDefinitionManager = oProject.GetDefinitionManager()
```

```
Set oModelManager = oDefinitionManager.GetManager("Model")
```

The model manager script commands are listed below:

[Add](#)

[ConvertToDynamic](#)

[ConvertToParametric](#)

[Edit](#)

[EditWithComps](#)

[Export](#)

[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
```

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

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

Edit [deprecated]

Deprecated command — please use [EditSymbolAndUpdateComps](#).

EditWithComps [model manager]

Use: Edit an existing model.

Command: None

Syntax: EditWithComps <ModelName>,
 Array("NAME:<NewModelName>",
 "ModTime:=", <ModifiedTimeInfo>,
 "Library:=", <string>, // Library name
 "LibLocation:=", <string>, // Project Location
 <PinDefInfo>,
 <PinDefInfo>,... // optional, to define pins
 <GraphicsDataInfo>, // optional, to define graphics

```
<PropDisplayMapInfo>), // optional, to define property displays  
Array(<ListOfComponentNames>) // Component names
```

Return Value: <string>

```
// composite name of the model.  
// If the name requested conflicts with the name of an existing  
// model, the requested name is altered to be unique.  
// The name returned reflects any change made to be unique.
```

Parameters: <ModelName>:

```
<string> // composite name of the model being edited
```

<NewModelName>:

```
<string> // new simple name for the model
```

<ModifiedOnInfo>:

An integer that corresponds to the number of seconds that have elapsed
since 00:00 hours, Jan 1, 1970 UTC from the system clock.

<PinDefInfo>:

```
Array("NAME:PinDef",  
"Pin:=", Array (<string>, // pin name  
<real>, // x location  
<real>, // y location  
<real>, // angle in radians  
<PinType>,  
<real>, // line width  
<real>, // line length  
<bool>, // mirrored  
<int>, // color  
<bool>, // true if visible, false if not  
<string>, // hidden net name  
<OptionalPinInfo>, // optional info  
<PropDisplayMapInfo>)) // optional
```

```
<PinType>:  
<string> // "N" : normal pin  
// "I" : input pin  
// "O" : output pin
```

```
<OptionalPinInfo>:
```

// Specify both or neither

<bool>, // true if name is to be shown

<bool>, // true if number is to be shown

<PropDisplayMapInfo>:

Array("NAME:PropDisplayMap",

<PropDisplayInfo>,

<PropDisplayInfo>,...)

<PropDisplayInfo>:

<NameString>, Array(<DisplayTypeInfo>,

<DisplayLocationInfo>,

<int>, // optional, level number

<TextInfo>)

<NameString>:

<string> // PropertyName:=, where PropertyName is the name of

// the property to be displayed

<DisplayTypeInfo>:

<int> // 0 : No display
// 1 : Display name only
// 2 : Display value only
// 3 : Display both name and value
// 4: Display evaluated value only
// 5: Display both name and evaluated value

<DisplayLocationInfo>:

<int> // 0 : Left
// 1 : Top
// 2 : Right
// 3 : Bottom
// 4 : Center
// 5 : Custom placement

<GraphicsDataInfo>:

Array("NAME:Graphics",
// one or more of the following
<RectInfo>,
<CircleInfo>,
<ArcInfo>,

```
<LineInfo>,  
<PolygonInfo>,  
<TextInfo>,  
<ImageInfo>)
```

```
<RectInfo>:  
"Rect:=", Array(<real>, // line width  
<int>, // fill pattern  
<int>, // color  
<real>, // angle, in radians  
<real>, // x position of center  
<real>, // y position of center  
<real>, // width  
< real>) // height
```

```
<CircleInfo>:  
"Circle:=", Array(<real>, // line width  
<int>, // fill pattern  
<int>, // color  
<real>, // x position of center
```

<real>, // y position of center

< real>) // radius

<ArcInfo>:

"Arc:=", Array(<real>, // line width

<int>, // line pattern

<int>, // color

<real>, // x position of center

<real>, // y position of center

< real>, // radius

<real>, // start angle, in radians

<end>) // end angle, in radians

<LineInfo>:

"Line:=", Array(<real>, // line width

<int>, // line pattern

<int>, // color

<PointInfo>, // must specify at least 2 points

<PointInfo>...)

<PointInfo>:

<real>, // x position

<real> // y position

<PolygonInfo>:

"Polygon:=", Array(<real>, // line width

<int>, // fill pattern

<int>, // color

<PointInfo>, // must specify at least 3 points

<PointInfo>...)

<TextInfo>:

"Text:=", Array(<real>, // x position

<real>, // y position

<real>, // angle, in radians

<Justification>,

<bool>, // is plotter font

<string>, // font name

<int>, // color

<string>) // text string

<Justification>:

<int> // 0 : left top
// 1 : left base
// 2 : left bottom
// 3 : center top
// 4 : center base
// 5 : center bottom
// 6 : right top
// 7 : right base
// 8 : right bottom

<ImageInfo>:
"Image:=", Array(<RectInfo>,
<ImageData>,
<bool>) // is mirrored

<ImageData>:
<string>, // file path
<int>, // 0 : use the file path and link to it
// 1 : ignore file path and use next parameter
<string> // text data, only present if preceding int is 1

<ListOfComponentNames>:

<string>, <string> ...

// The list may be empty. When not empty, each string that is listed is a component

// that references the model to be edited. Prior to editing, a clone of the model is

// made, and the components that are listed are modified so that they now refer to

// the clone.

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"

Python Syntax	Export(["NAME:<LibraryName>", <ComboName>, <ComboName>...], <LibraryLocation>)
Python Example	<code>oModelManager.Export(["NAME:mylib", "model1", "model2"])</code>

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

Python Syntax	GetNames()
Python Example	<code>modelnames = oModelManager.GetNames()</code>

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>

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"

Python Syntax	Remove (<ModelName>, <IsLocal>, <LibraryName>, <LibraryLocation>)
Python Example	<pre>oModelManager.Export(["NAME:mylib", "model1", "model2"])</pre>

RemoveUnused [model manager]

Use: Removes models that are not used in the design.

Command: **Project->Remove Unused Definitions** is similar but operates slightly different and does not record script commands.

Syntax: RemoveUnused()

Return Value: <bool> True if one or more models are removed.

Parameters: None

Note:

The order of calls to RemoveUnused is significant. As a result, removing definitions in an unordered fashion may cause other models in dependent definitions to be rendered unusable.

Also, the model and footprint of an unused component are not unusable until after the component itself is removed using the Component Manager Remove script.

Python Syntax	RemoveUnused()
Python Example	thereWereExtras = oModelManager.RemoveUnused()

Network Data Explorer Manager Script Commands

The network data Explorer (NDE) Manager provides access to certain NDE data.

```
Set oDefinitionManager = oProject.GetDefinitionManager()
```

```
Set oNdExplorerManager = oDefinitionManager.GetManager("NdExplorer")
```

For NDE scripts accessed via the ndExplorer tool, see: [Network Data Explorer Script Commands](#).

The topics for this section include:

[ExportFullWaveSpice](#)

[ExportNetworkData](#)

[ExportNMFData](#)

ExportFullWaveSpice

Use: Export FullWaveSpice data in a format of your choice.

Command: File > Export MacroModel > Broadband (SYZ, FWS....)

Syntax: ExportFullWaveSpice

```
"DesignName", // Design name. Can be left blank, if loading solution from a file.  
true/false, // true - solution loaded from file, false- loaded from design  
"Name", // If loading from design this is the solution name, else this is the  
           // full path of the file from which the solution is loaded  
"variation", // Pick a particular variation. Leave blank if no variation.  
Array("NAME:Frequencies"), // Optional; if none defined all frequencies are used  
Array("NAME:SpiceData", // Spice export options object  
"SpiceType:=", "SSS", // SpiceType can be "PSpice", "HSpice", "Spectre", "SSS",  
           // "Simplorer", "TouchStone1.0", "TouchStone2.0"  
"EnforcePassivity:=", false, // Enforce Passivity true/false  
"EnforceCausality:=", false, // Enforce Causality true/false  
"UseCommonGround:=", false, // Use common ground true/false  
"FittingError:=", 0.5, // Fitting error  
"MaxPoles:=", 400, // Maximum Order  
"PassivityType:=", "ConvexOptimization", // Passivity Type can be "ConvexOptimization",  
           // "PassivityByPerturbation", or "IteratedFittingOfPV"  
"ColumnFittingType:=", "Column", // Column FittingType can be "Column", "Entry", "Matrix"
```

```
"SSFittingType:=", "TWA", // SS Fitting Type can be "TWA", "IterativeRational"
"RelativeErrorToleranc:=", false, // Relative error tolerance true/false
"TouchstoneFormat:=", "MA", // Touchstone Format "MA", "RI", "DB"
"TouchstoneUnits:=", "Hz", // Touchstone Units "Hz", "KHz", "MHz", "MHz"
"TouchStonePrecision:=", 8, // Touchstone precision
"ExportDirectory:=", "C:/Examples/LNA/", // Directory to export to
"ExportSpiceFileName:=", "Linckt_HBTest_2.sss", // Spice export file
"FullwaveSpiceFileName:=", "Linckt_HBTest.sss", // FWS file
"CreateNPortModel:=", true // Create a model based on the exported file true/false
)
```

ExportNetworkData

Exports matrix solution data to a file.

UI Access	N/A		
Parameters	Name	Type	Description
	<DesignVariationKey>	String	Design variation key. Pass empty string for the current nominal variation.
	<SolnSelectionArray>	Array	Array of selected solutions. Array(<SolnSelector>, <SolnSelector>, ...) If more than one array entry, this indicates a combined Interpolating sweep.
	<SolnSelector>	String	Solution setup name and solution name, separated by a colon.

<FileFormat>	Integer	File format value. 2 : Tab delimited spreadsheet format (.tab) 3 : Touchstone (.sNp) 4 : CitiFile (.cit) 7 : Matlab (.m) 8 : Terminal Z0 spreadsheet
<OutFile>	String	Full path to the file to write out.
<FreqsArray>	Array	The frequencies to export. The <FreqsArray> argument contains a vector (e.g. "1GHz", "2GHz", ...) to use, or "all". To export all frequencies, use Array("all"). If no frequencies are specified, all frequencies are used.
<DoRenorm>	Boolean	For Touchstone format only. Specifies whether to renormalize the data before export.
<RenormImped>	Double	For Touchstone format only. Real impedance value in ohms, for renormalization. Required in syntax, but ignored if DoRenorm is false.
<DataType>	Array	Optional. Type: "S", "Y", or "Z". The matrix to export.
<pass>	Integer	Optional. The pass to export. This is ignored if the sourceName is a frequency sweep. Leaving out this value or specifying -1 gets all passes.
<ComplexFormat>	Integer	Optional. Type: "0", "1", or "2" The format to use for the exported data. 0 = Magnitude/Phase. 1 = Real/Imaginary. 2 = db/Phase.
<Precision>	Integer	Optional. Touchstone number of digits precision. Default if not specified is 15.
<UseExportFreqs>	Boolean	Specifies whether to use export frequencies.

	<i><IncludeGammaComments></i>	Boolean	Touchstone only. Specifies whether to include Gamma and Impedance comments.
	<i><SupportNonStdExport></i>	Boolean	Specifies whether to support non-standard Touchstone extensions for mixed reference impedance.
Return Value	None.		

Python Syntax	ExportNetworkData(<i><DesignVariationKey></i> , <i><SolnSelectionArray></i> , <i><SolnSelector></i> , <i><FileFormat></i> , <i><OutFile></i> , <i><FreqsArray></i> , <i><DoRenorm></i> , <i><RenormImped></i> , [Optional <i><DataType></i>], [Optional <i><pass></i>], [Optional <i><ComplexFormat></i>], [Optional <i><Precision></i>], [Optional <i><UseExportFreqs></i>], [Optional <i><IncludeGammaComments></i>], [Optional <i><SupportNonStdExport></i>])		
Python Example			

ExportNMFData

Symbol Manager Script Commands

The symbol manager provides access to symbols in a project. The manager object is accessed via the definition manager.

```
oDefinitionManager = oProject.GetDefinitionManager()
oSymbolManager = oDefinitionManager.GetManager("Symbol")
```

The symbol manager script commands are listed below.

[Add](#)

[BringToFront](#)

[Edit](#)

[EditSymbolAndUpdateComps](#)

[Export](#)

[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

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

Edit [deprecated]

Deprecated command — please use [EditSymbolAndUpdateComps](#).

EditSymbolAndUpdateComps [symbol manager]

Use: Edit an existing symbol.

Command: None

Syntax: EditSymbolAndUpdateComps <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
 <EditContext> //optional

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.

<EditContext>:

// Changes that will be made to the component in support of the symbol changes

<RefPinOption>:

// <int>

// 0 = implied reference to ground,

// 1 = single common reference port,

// 2 = individual hidden reference pins for each port,

// 3 = individual reference pin per port

<CompName> // <string>

<TermAttributes>:

//<array>

<Terminal Name> // <string>

<Symbol Pin Name> // <string>

<InOut> // <int>

// 0=in

// 1=out

// 2= inout

<Domain> // <array>

// kConservative=0

```
// kSignal
// kQuantity
// kParameter
// kState
// kFlexible
<ID> // <int>
//-1, not used.
<Description>
```

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"

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

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>

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"

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

Note:

The order of calls to RemoveUnused is significant. As a result, removing definitions in an unordered fashion may cause other symbols in dependent definitions to be rendered unusable.

Also, the symbol and footprint of an unused component are not unusable until after the component itself is removed using the Component Manager Remove script.

Add [footprint manager]

Use: Add a footprint

UI Access	Tools > Edit Configured Libraries > Footprints > Add Footprint		
Parameters	Name	Type	Description
	Name	Type	Description
Return Value	Returns a string:		

	• Composite name of the footprint. • If the name requested conflicts with the name of an existing • Footprint, the requested name is altered to be unique. • The name returned reflects any change made to be unique.

Syntax: Add Array("NAME:<FootprintName>,"
"ModTime:=", <ModifiedOnInfo>,"
"Library:=", "",
"LibLocation:=", "Project",
"OkayToMirror:=", <bool>,"
"DefUnits:=", <UnitType>,"
Array(NAME:Lyrs",
"Layer:=", <LayerArray>,"
"Layer:=", <LayerArray>...,
"SLayer:=", <StackupLayerArray>,"
"SLayer:=", <StackupLayerArray>...),
"ActLyr:=", <string>, // name of active layer
"Tol:=", <ToleranceArray> // optional
<PrimitivesInfo>, // optional
<PinsInfo>, // optional
<ViasInfo>, // optional

```
<EdgeportsInfo>, // optional  
<ComponentPropertyInfo>,  
<ScriptInfo>) // optional, specified for scripted footprints
```

Parameters: <FootprintName>:

<string> // simple name of footprint to create

<ModifiedOnInfo>:

An integer that corresponds to the number of seconds that have elapsed
since 00:00 hours, Jan 1, 1970 UTC from the system clock.

<UnitType>:

<string> // default length units to use if units are not specified in other
// parameters

<LayerArray>:

Array("N:=", <string>, // layer name

"ID:=", <int> ,

"T:=", <LayerTypeInfo>, // layer type

"TB:=", <TopBottomInfo> ,

"Col:=", <int>, // optional - color

"Pat:=", <int>, // optional - fill pattern

"Vis:=", <bool>, // optional - are objects on layer visible

"Sel:=", <bool>, // optional - are objects on layer selectable

"L:=", <bool>) // optional

// are objects on layer locked (can't be edited)

<LayerTypeInfo>:

<string> // one of: signal, dielectric, metalized signal, assembly, silkscreen, soldermask, solderpaste, glue, or user

<TopBottomInfo>:

<string> // one of: top, neither, bottom, or template

<StackupLayerArray>:

Array(<LayerArray>,

"Elev:=", <ElevationInfo>,

"SubL:=", Array("Th:=", <Dimension>,

"LElev:=", <Dimension>,

"R:=", <Dimension>,

"M:=", <MaterialInfo>))

<ElevationInfo>:

<string> // "top" - snap to top

// "mid" - snap to middle

// "bot" - snap to bottom

// "edit" - manual edit

// "none"

<Dimension>:

<string> // real number, may include units

<MaterialInfo>:

<string> // name of the layer material

<ToleranceArray>:

Array(<real>, // distance tolerance

<real>, // angle tolerance (radians)

<real>) // dimensionless tolerance

<PrimitivesInfo>:

Array("NAME:Prims",

// one or more of the following

<RectInfo>,

<CircleInfo>,

<ArcInfo>,
<LineInfo>,
<PolygonInfo>,
<TextInfo>,
<ImageInfo>)

<RectInfo>:

"Rect:=", Array(<real>, // line width
<int>, // fill pattern
<int>, // color
<real>, // angle, in radians
<real>, // x position of center
<real>, // y position of center
<real>, // width
< real>) // height

<CircleInfo>:

"Circle:=", Array(<real>, // line width
<int>, // fill pattern
<int>, // color

<real>, // x position of center

<real>, // y position of center

< real>) // radius

<ArcInfo>:

"Arc:=", Array(<real>, // line width

<int>, // line pattern

<int>, // color

<real>, // x position of center

<real>, // y position of center

< real>, // radius

<real>, // start angle, in radians

<end>) // end angle, in radians

<LineInfo>:

"Line:=", Array(<real>, // line width

<int>, // line pattern

<int>, // color

<PointInfo>, // must specify at least 2 points

<PointInfo>...)

<PointInfo>:

<real>, // x position

<real> // y position

<PolygonInfo>:

"Polygon:=", Array(<real>, // line width

<int>, // fill pattern

<int>, // color

<PointInfo>, // must specify at least 3 points

<PointInfo>...)

<TextInfo>:

"Text:=", Array(<real>, // x position

<real>, // y position

<real>, // angle, in radians

<Justification>,

<bool>, // is plotter font

<string>, // font name

<int>, // color

<string>) // text string

<Justification>: <int>

// 0 : left top
// 1 : left base
// 2 : left bottom
// 3 : center top
// 4 : center base
// 5 : center bottom
// 6 : right top
// 7 : right base
// 8 : right bottom

<ImageInfo>:
"Image:=", Array(<RectInfo>,
<ImageData>,
<bool>) // is mirrored

<ImageData>:
<string>, // file path
<int>, // 0 : use the file path and link to it
// 1 : ignore file path and use next parameter
<string> // text data, only present if preceding int is 1

<PinsInfo>:

```
Array("NAME:Pins",  
"P:=", <PinArray>,  
"P:=", <PinArray>,...)
```

<PinArray>:

```
Array("Port:=", Array("Id:=", <int>,  
"Clr:=", <real>, // optional - clearance  
"N:=", <string>), // pin name  
"Pos:=", Array("x:=", <Location>, // padstack (x,y) position  
"y:=", <Location>),  
"VRt:=", <Angle>, // optional - rotation  
"HD:=", <Size>, // optional - hole diameter  
<PadstackImplementationInfo>)
```

<Location>:

<string> specifying real number and units (may use variables)

<Angle>:

<string> specifying angle with a real number and units

<Size>:

<string> specifying size with a real number and units

<PadstackImplementationInfo>:

If another with the same implementation has already been specified:

"Ref:=", <int> // id of the other

If not:

"Ref:=", <string>, // name

"Frm:=", <int>, // id of highest layer

"To:=", <int>, // id of lowest layer

<LayerPlacementInfo>,

"Man:=", <int>, // optional, 1 if manually placed, 0 if not (default)

"Use:=", Array(<PadUseInfo>, <PadUseInfo>...) // array may be empty

<LayerPlacementInfo>:

"Lyr:=", Array("Mrg:=", <int>, // optional,

// 1 if all layers have been merged (default)

// 0 if layer are not merged

"Flp:=", <int>, // optional,

// 1 if placed bottom up

// 0 if not (default)

"Map:=", <ParentToLocalLayerInfo>)

<ParentToLocalLayerInfo>:

Array("U:=", <DirectionOfUniqueness>,

"F:=", Array(<int>, <int>, ...), // forward mapping

// -1 is not mapped

"B:=", Array(<int>, <int>, ...)) // backward mapping

// -1 is not mapped

<PadUseInfo>:

"Pad:=", Array("Lid:=", <int>, // layer id

"T:=", <string>, // type : "connected", "thermal",

// "no_pad", "not_connected",

// or "not_connected_thermal"

"Man:=", <int>) // optional, 1 if manually placed

// 0 if not (default)

<DirectionOfUniqueness>:

<string> // one of "forward", "backward", or "two ways"

<ViasInfo>:

Array("NAME:Vias", <VialInfo>, <VialInfo>...)

<VialInfo>:

"V:=", Array("Id:=", <int>,

"N:=", <string>, // name

"Pos:=", Array("x:=", <Location>, // via (x,y) position

"y:=", <Location>),

"VRt:=", <Angle>, // optional - rotation

<ImplementationInfo>

<EdgeportsInfo>:

Array("NAME:EPorts", <EdgePortArray>, <EdgePortArray>...)

<EdgePortArray>:

Array("NAME:EP",

"LP:=", Array("Id:=", <int>, // port id

"N:=", <string>), // port name

"Eo:=", Array(<edge description>, <edge description>,...))

<edge description> for primitive edges

"et:=", "pe", "pr:=", <id>, "ei:=", <edge#>

<id>: integer that is the primitive id

<edge#>: integer that is the edge number on the primitive

<edge description> for via edges

"et:=", "pse", "layer:=", <layer id>, "se:=", <via id>,

"sx:=", <start X location>, "sy:=", <start Y location>, "ex:=", <end X location>,

"ey:=", <end Y location>, "h:=", <arc height>, "rad:=", <radians>

<via id>: an integer that is the id of the via to use

<layer id>: an integer that is the id of the layer of the pad of the via to use

<start X location>

<start Y Location>:

doubles that are the X, Y location of the start point of the edge arc <end X location>

<end Y Location>:

doubles that are the X, Y location of the end point of the edge arc

<arc height>: double giving the height of the edge arc (0 for a straight edge)

<radians>: double giving the arc size in radians (0 for a straight edge)

<ComponentPropertyInfo>:

```
Array("NAME:CProps",  
"VariableProp:=", <VariableInfo>,  
"VariableProp:=", <VariableInfo>,  
...)
```

<VariableInfo>:

```
Array(<string>, // name  
<FlagLetters>,  
<string>, // description  
"CB:=", <string>, // optional - script for call back  
<string>) // value: number, variable, or expression
```

<ScriptInfo>:

```
Array("NAME:script",
```

```
"language:=", <string>, // script language
"UsesScript:=", true,
"script:=", <string>) // contents of script
```

Edit [footprint manager]

Deprecated command — please use [EditWithComps](#).

EditWithComps [footprint manager]

Use: Edit an existing footprint.

Command: None

Syntax: EditWithComps <FootprintName>,
 Array("NAME:<NewFootprintName>,"
 "ModTime:=", <ModifiedOnInfo>,"
 "Library:=", "",
 "LibLocation:=", "Project",
 "OkayToMirror:=", <bool>,"
 "DefUnits:=", <UnitType>,"
 Array(NAME:Lyrs",
 "Layer:=", <LayerArray>,"
 "Layer:=", <LayerArray>...,
 "SLayer:=", <StackupLayerArray>,"

```

"SLayer:=", <StackupLayerArray>...),
"ActLyr:=", <string>, // name of active layer
"Tol:=", <ToleranceArray> // optional
<PrimitivesInfo>, // optional
<PinsInfo>, // optional
<ViasInfo>, // optional
<EdgeportsInfo>, // optional
<ComponentPropertyInfo>,
<ScriptInfo>, // optional, specified for scripted footprints
Array(<ListOfComponentNames>) // Component names

```

Return Value: <string>

```

// composite name of the footprint.
// If the name requested conflicts with the name of an existing
// footprint, the requested name is altered to be unique.
// The name returned reflects any change made to be unique.

```

Parameters: <FootprintName>:

```

<string> // composite name of the footprint being edited

```

<NewFootprintName>:

```

<string> // new simple name for the footprint

```

<ModifiedOnInfo>:

An integer that corresponds to the number of seconds that have elapsed since 00:00 hours, Jan 1, 1970 UTC from the system clock.

<UnitType>:

<string> // default length units to use if units are not specified in other
// parameters

<LayerArray>:

Array("N:=", <string>, // layer name
"ID:=", <int> ,
"T:=", <LayerTypeInfo>, // layer type
"TB:=", <TopBottomInfo>,
"Col:=", <int>, // optional - color
"Pat:=", <int>, // optional - fill pattern
"Vis:=", <bool>, // optional - are objects on layer visible
"Sel:=", <bool>, // optional - are objects on layer selectable
"L:=", <bool>) // optional
// are objects on layer locked (can't be edited)

<LayerTypeInfo>:

<string> // one of: signal, dielectric, metalized signal, assembly, silkscreen, soldermask, solderpaste, glue, or user

<TopBottomInfo>:

<string> // one of: top, neither, bottom, or template

<StackupLayerArray>:

```
Array(<LayerArray>,  
"Elev:=", <ElevationInfo>,  
"SubL:=", Array("Th:=", <Dimension>,  
"LElev:=", <Dimension>,  
"R:=", <Dimension>,  
"M:=", <MaterialInfo>))
```

<ElevationInfo>:

```
<string> // "top" - snap to top  
// "mid" - snap to middle  
// "bot" - snap to bottom  
// "edit" - manual edit  
// "none"
```

<Dimension>:

<string> // real number, may include units

<MaterialInfo>:

<string> // name of the layer material

<ToleranceArray>:

Array(<real>, // distance tolerance

<real>, // angle tolerance (radians)

<real>) // dimensionless tolerance

<PrimitivesInfo>:

Array("NAME:Prims",

// one or more of the following

<RectInfo>,

<CircleInfo>,

<ArcInfo>,

<LineInfo>,

<PolygonInfo>,

<TextInfo>,
<ImageInfo>)

<RectInfo>:

"Rect:=", Array(<real>, // line width
<int>, // fill pattern
<int>, // color
<real>, // angle, in radians
<real>, // x position of center
<real>, // y position of center
<real>, // width
< real>) // height

<CircleInfo>:

"Circle:=", Array(<real>, // line width
<int>, // fill pattern
<int>, // color
<real>, // x position of center
<real>, // y position of center
< real>) // radius

<ArcInfo>:

```
"Arc:=", Array(<real>, // line width
<int>, // line pattern
<int>, // color
<real>, // x position of center
<real>, // y position of center
< real>, // radius
<real>, // start angle, in radians
<end>) // end angle, in radians
```

<LineInfo>:

```
"Line:=", Array(<real>, // line width
<int>, // line pattern
<int>, // color
<PointInfo>, // must specify at least 2 points
<PointInfo>...)
<PointInfo>:
<real>, // x position
<real> // y position
```

<PolygonInfo>:

```
"Polygon:=", Array(<real>, // line width
<int>, // fill pattern
<int>, // color
<PointInfo>, // must specify at least 3 points
<PointInfo>...)
```

<TextInfo>:

```
"Text:=", Array(<real>, // x position
<real>, // y position
<real>, // angle, in radians
<Justification>,
<bool>, // is plotter font
<string>, // font name
<int>, // color
<string>) // text string
<Justification>: <int>
// 0 : left top
// 1 : left base
// 2 : left bottom
// 3 : center top
```

```
// 4 : center base  
// 5 : center bottom  
// 6 : right top  
// 7 : right base  
// 8 : right bottom
```

<ImageInfo>:

```
"Image:=", Array(<RectInfo>,  
<ImageData>,  
<bool>) // is mirrored
```

<ImageData>:

```
<string>, // file path  
<int>, // 0 : use the file path and link to it  
// 1 : ignore file path and use next parameter  
<string> // text data, only present if preceding int is 1
```

<PinsInfo>:

```
Array("NAME:Pins",  
"P:=", <PinArray>,
```

"P:=", <PinArray>,...)

<PinArray>:

Array("Port:=", Array("Id:=", <int>,
"Clr:=", <real>, // optional - clearance
"N:=", <string>), // pin name
"Pos:=", Array("x:=", <Location>, // padstack (x,y) position
"y:=", <Location>),
"VRt:=", <Angle>, // optional - rotation
"HD:=", <Size>, // optional - hole diameter
<PadstackImplementationInfo>)

<Location>:

<string> specifying real number and units (may use variables)

<Angle>:

<string> specifying angle with a real number and units

<Size>:

<string> specifying size with a real number and units

<PadstackImplementationInfo>:

If another with the same implementation has already been specified:

"Ref:=", <int> // id of the other

If not:

"Ref:=", <string>, // name

"Frm:=", <int>, // id of highest layer

"To:=", <int>, // id of lowest layer

<LayerPlacementInfo>,

"Man:=", <int>, // optional, 1 if manually placed, 0 if not (default)

"Use:=", Array(<PadUseInfo>, <PadUseInfo>...) // array may be empty

<LayerPlacementInfo>:

"Lyr:=", Array("Mrg:=", <int>, // optional,

// 1 if all layers have been merged (default)

// 0 if layer are not merged

"Flp:=", <int>, // optional,

// 1 if placed bottom up

// 0 if not (default)

"Map:=", <ParentToLocalLayerInfo>)

<ParentToLocalLayerInfo>:

```
Array("U:=", <DirectionOfUniqueness>,  
"F:=", Array(<int>, <int>, ...), // forward mapping  
// -1 is not mapped  
"B:=", Array(<int>, <int>, ...)) // backward mapping  
// -1 is not mapped
```

<PadUseInfo>:

```
"Pad:=", Array("Lid:=", <int>, // layer id  
"T:=", <string>, // type : "connected", "thermal",  
// "no_pad", "not_connected",  
// or "not_connected_thermal"  
"Man:=", <int>) // optional, 1 if manually placed  
// 0 if not (default)
```

<DirectionOfUniqueness>:

```
<string> // one of "forward", "backward", or "two ways"
```

<ViasInfo>:

```
Array("NAME:Vias", <VialInfo>, <VialInfo>...)
```

<VialInfo>:

```
"V:=", Array("Id:=", <int>,
"N:=", <string>, // name
"Pos:=", Array("x:=", <Location>, // via (x,y) position
"y:=", <Location>),
"VRt:=", <Angle>, // optional - rotation
<ImplementationInfo>
```

<EdgeportsInfo>:

```
Array("NAME:EPorts", <EdgePortArray>, <EdgePortArray>...)
```

<EdgePortArray>

```
Array("NAME:EP",
"LP:=", Array("Id:=", <int>, // port id
"N:=", <string>), // port name
"Eo:=", Array(<edge description>, <edge description>,...))
```

<edgedescription> for primitive edges

```
"et:=", "pe", "pr:=", <id>, "ei:=", <edge#>
```


<id>: integer that is the primitive id

<edge#>: integer that is the edge number on the primitive

<edge description> for via edges

"et:=", "pse", "sel:=", <"via">, "layer:=", <layer id>,

"sx:=", <start X location>, "sy:=", <start Y location>, "ex:=", <end X location>, "ey:=", <end Y location>, "h:=", <arc height>,
"rad:=", <radians>

<"via">: text that is the name of the via to use

<layer id>: an integer that is the id of the layer of the pad of the via to use

<start X location>, <start Y Location>:

doubles that are the X, Y location of the start point of the edge arc

<end X location>, <end Y Location>:

doubles that are the X, Y location of the end point of the edge arc

<arc height>: double giving the height of the edge arc (0 for a straight edge)

<radians>: double giving the arc size in radians (0 for a straight edge)

<ComponentPropertyInfo>:

```
Array("NAME:CProps",  
"VariableProp:=", <VariableInfo>,  
"VariableProp:=", <VariableInfo>,  
...)
```

<VariableInfo>:

```
Array(<string>, // name  
<FlagLetters>,  
<string>, // description  
"CB:=", <string>, // optional - script for call back  
<string>) // value: number, variable, or expression
```

<ScriptInfo>:

```
Array("NAME:script",  
"language:=", <string>, // script language  
"UsesScript:=", true,  
"script:=", <string>) // contents of script
```

<ListOfComponentNames>:

<string>,<string> ...

// The list may be empty. When not empty, each string that is listed is a component

// that references the footprint to be edited. Prior to editing, a clone of the footprint is

// made, and the components that are listed are modified so that they now refer to

// the clone.

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"

GetNames [footprint manager]

Use: Returns the names of the footprints (used and unused) in a design. The following script command, **IsUsed**, can then be used to separate used and unused footprints.

Command: None

Syntax: GetNames()

Return Value: An array of strings

Parameters: None

IsUsed [footprint manager]

Use: Used to determine if a footprint is used in the design.

Command: None

Syntax: IsUsed(<FootprintName>)

Return Value: <Boolean> // true if the specified footprint is used in the design

Parameters: <FootprintName>:

<string>

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"

RemoveUnused [footprint manager]

Use: Removes footprints that are not used in the design.

Command: **Project->Remove Unused Definitions** is similar but operates slightly different and does not record script commands.

Syntax: RemoveUnused()

Return Value: <bool> True if one or more footprints are removed.

Parameters: None

Note:

The order of calls to RemoveUnused is significant. As a result, removing definitions in an unordered fashion may cause other footprints in dependent definitions to be rendered unusable.

Also, the symbol and footprint of an unused component are not unusable until after the component itself is removed using the Component ManagerRemove script.

Add [padstack manager]

Use: Add a padstack

Command: Tools > Edit Configured Libraries > Padstacks > Add Padstack

Syntax: Add Array("NAME:<PadstackName>",
 "ModTime:=", <ModifiedOnInfo>,
 "Library:=", "", // name of the library
 "LibLocation:=", "Project", // location of the named library
 Array("NAME:psd",
 "nam:=", <PadstackName>,
 "lib:=", "", // name of the library
 "mat:=", "", // hole plating material
 "plt:=", "0", // percent of hole's radius filled by plating

```

Array("NAME:pds",
<LayerGeometryArray>,
<LayerGeometryArray....>,
"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

Edit [padstack manager]

Use: Edit an existing padstack.

Command: Tools > Edit Configured Libraries > Padstacks > Edit Padstack

Syntax: Edit <PadstackName>,

Array("NAME:<NewPadstackName>",

"ModTime:=", <ModifiedOnInfo>,

"Library:=", "", // name of the library

"LibLocation:=", "Project", // location of the named library

Array("NAME:psd",

"nam:=", <PadstackName>,

"lib:=", "", // name of the library

"mat:=", "", // hole plating material

"plt:=", "0", // percent of hole's radius filled by plating

Array("NAME:pds",

<LayerGeometryArray>,

```
<LayerGeometryArray....),  
"hle:=", <PadInfo>  
"hRg:=", <HoleRange>,  
"sbsh:=", <SolderballShape>,  
"sbpl:=", <SolderballPlacement>,  
"sbr:=", <string>, // solderball diameter, real with units  
"sb2:=", <string>, // solderball mid diameter, real with units  
"sbn:=", <string>), // name of solderball material  
"ppl:=", <PadPortLayerArray>)
```

Return Value: <string> // composite name of the padstack

// If the name requested conflicts with the name of an existing
// padstack, the requested name is altered to be unique.
// The name returned reflects any change made to be unique.

Parameters: <PadstackName>:

<string> // composite name of padstack to edit

<NewPadstackName>:

<string> // new simple name for padstack

<ModifiedOnInfo>:

An integer that corresponds to the number of seconds that have elapsed since 00:00 hours, Jan 1, 1970 UTC from the system clock.

<LayerGeometryArray>:

```
Array("Name:lgm",  
"lay:=", <string>, // definition layer name  
"id:=", <int>, // definition layer id  
"pad:=", <PadInfo>, // pad  
"ant:=", <PadInfo>, // antipad  
"thm:=", <PadInfo>, // thermal pad  
"X:=", <string>, // pad x connection, real with units  
"Y:=", <string>, // pad y connection, real with units  
"dir:=", <DirectionString>) // pad connection direction
```

<PadInfo>:

```
Array("shp:=", <PadShape>,  
"Szs:=", <DimensionArray>,  
"X:=", <string>, // x offset, real with units  
"Y:=", <string>, // y offset, real with units  
"R:=", <string>) // rotation, real with units
```

<PadShape>:

<string> one of these choices

"No" // no pad

"Cir" // Circle

"Sq" // Square

"Rct" // Rectangle

"Ov" // Oval

"Blt" // Bullet

"Ply" // Polygons

"R45" // Round 45 thermal

"R90" // Round 90 thermal

"S45" // Square 45 thermal

"S90" // Square 90 thermal

<DimensionArray>:

Array(<string>, ...) // each string is a real with units for one of the

// dimensions of the shape

<DirectionString>:

<string> one of these choices

"No" // no direction

"Any" // any direction

"0" // 0 degrees

"45" // 45 degrees

"90" // 90 degrees

"135" // 135 degrees

"180" // 180 degrees

"225" // 225 degrees

"270" // 270 degrees

"315" // 315 degrees

<HoleRange>:

<string> one of these choices

"Thr" // through all layout layers

"Beg" // from upper pad layer to lowest layout layer

"End" // from upper layout layer to lowest pad layer

"UTL" // from upper pad layer to lowest pad layer

<SolderballShape>:

<string> one of these choices

"None" // no solderball

"Cyl" // cylinder solderball

"Sph" // spheroid solderball

<SolderballPlacement>:

<string> one of these choices

"abv" // above padstack

"blw" // below padstack

<PadPortLayerArray>:

Array(<int>, <int>,....) where each int is a layer id

EditWithComps [padstack manager]

Use: Edit an existing padstack.

Command: None

Syntax: EditWithComps <PadstackName>,
 Array("NAME:<NewPadstackName>",
 "ModTime:=", <ModifiedOnInfo>,
 "Library:=", "", // name of the library
 "LibLocation:=", "Project", // location of the named library
 Array("NAME:psd",
 "nam:=", <PadstackName>,


```
"lib:=", "", // name of the library
"mat:=", "", // hole plating material
"plt:=", "0", // percent of hole's radius filled by plating
Array("NAME:pds",
<LayerGeometryArray>,
<LayerGeometryArray....>,
"hle:=", <PadInfo>
"hRg:=", <HoleRange>,
"sbsh:=", <SolderballShape>,
"sbpl:=", <SolderballPlacement>,
"sbr:=", <string>, // solderball diameter, real with units
"sb2:=", <string>, // solderball mid diameter, real with units
"sbn:=", <string>), // name of solderball material
"ppl:=", <PadPortLayerArray>,
Array(<ListOfComponentNames>) // Component names
```

Return Value: <string>

```
// composite name of the padstack.
// If the name requested conflicts with the name of an existing
// padstack, the requested name is altered to be unique.
// The name returned reflects any change made to be unique.
```

Parameters: <PadstackName>:

<string> // composite name of the padstack being edited

<NewPadstackName>:

<string> // new simple name for the padstack

<ModifiedOnInfo>:

An integer that corresponds to the number of seconds that have elapsed since 00:00 hours, Jan 1, 1970 UTC from the system clock.

<LayerGeometryArray>:

Array("Name:lgm",

"lay:=", <string>, // definition layer name

"id:=", <int>, // definition layer id

"pad:=", <PadInfo>, // pad

"ant:=", <PadInfo>, // antipad

"thm:=", <PadInfo>, // thermal pad

"X:=", <string>, // pad x connection, real with units

"Y:=", <string>, // pad y connection, real with units

"dir:=", <DirectionString>) // pad connection direction

<PadInfo>:

Array("shp:=", <PadShape>,

"Szs:=", <DimensionArray>,

"X:=", <string>, // x offset, real with units

"Y:=", <string>, // y offset, real with units

"R:=", <string>) // rotation, real with units

<PadShape>:

<string> one of these choices

"No" // no pad

"Cir" // Circle

"Sq" // Square

"Rct" // Rectangle

"Ov" // Oval

"Blt" // Bullet

"Ply" // Polygons

"R45" // Round 45 thermal

"R90" // Round 90 thermal

"S45" // Square 45 thermal

"S90" // Square 90 thermal

<DimensionArray>:

Array(<string>, ...) // each string is a real with units for one of the

// dimensions of the shape

<DirectionString>:

<string> one of these choices

"No" // no direction

"Any" // any direction

"0" // 0 degrees

"45" // 45 degrees

"90" // 90 degrees

"135" // 135 degrees

"180" // 180 degrees

"225" // 225 degrees

"270" // 270 degrees

"315" // 315 degrees

<HoleRange>:

<string> one of these choices

"Thr" // through all layout layers

"Beg" // from upper pad layer to lowest layout layer

"End" // from upper layout layer to lowest pad layer

"UTL" // from upper pad layer to lowest pad layer

<SolderballShape>:

<string> one of these choices

"None" // no solderball

"Cyl" // cylinder solderball

"Sph" // spheroid solderball

<SolderballPlacement>:

<string> one of these choices

"abv" // above padstack

"blw" // below padstack

<PadPortLayerArray>:

Array(<int>, <int>,....) where each int is a layer id

<ListOfComponentNames>:

<string>,<string> ...

// The list may be empty. When not empty, each string that is listed is a component

```
// that references the padstack to be edited. Prior to editing, a clone of the padstack is  
// made, and the components that are listed are modified so that they now refer to  
// the clone.
```

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"
```

GetNames [padstack manager]

Use: Returns the names of the padstack (used and unused) in a design. The following script command, **IsUsed**, can then be used to separate used and unused padstacks.

Command: None

Syntax: GetNames()

Return Value: An array of strings

Parameters: None

IsUsed [padstack manager]

Use: Used to determine if a component is used in the design.

Command: None

Syntax: IsUsed(<PadstackName>)

Return Value: <Boolean> // true if the specified padstack is used in the design

Parameters: <PadstackName>:

 <string>

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"

RemoveUnused [padstack manager]

Use: Removes padstacks that are not used in the design.

Command: **Project->Remove Unused Definitions** is similar but operates slightly different and does not record script commands.

Syntax: RemoveUnused()

Return Value: <bool> True if one or more padstacks are removed.

Parameters: None

Note:

The order of calls to RemoveUnused is significant. As a result, removing definitions in an unordered fashion may cause other padstacks in dependent definitions to be rendered unusable.

Script and Library Scripts

The definition manager provides access to materials in a project. The manager object is accessed via the definition manager.

```
oDefinitionManager = oProject.GetDefinitionManager()
```

The script and library script commands are listed below.

[AddScript](#)

[EditScript](#)

[ExportScript](#)

[RemoveScript](#)

[ModifyLibraries](#)

AddScript

Adds a script to the definition manager.

UI Access	N/A		
Parameters	Name	Type	Description
	<AddScriptArray>	Array	Structured array. Array("NAME:<string script name>", "ScriptLang:=", <string script language> "ScriptText:=", <string text of script>)

Return Value	None.

Python Syntax	AddScript(<AddScriptArray>)
Python Example	<pre>oDefinitionManager.AddScript(["NAME:MyScript", "ScriptLang:=", "language", "ScriptText:=", "MsgBox(\"HelloWorld\") "])</pre>

EditScript

Edits a script in the definition manager.

UI Access	N/A		
Parameters	Name	Type	Description
	<OriginalName>	String	Name of the script to be edited.
	<EditScriptArray>	Array	Structured array. Array("NAME:<string script name>", "ScriptLang:=", <string script language> "ScriptText:=, <string text of script>)

Return Value	None.

Python Syntax	<code>EditScript(<OriginalName>, <EditScriptArray>)</code>
Python Example	<pre>oDefinitionManager.EditScript("MyScript", ["NAME:MyNewScript", "ScriptLang:=", "language", "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>,...)

Python Syntax	<code>ExportScript(<ExportData>,<Library location>)</code>
Python Example	<pre>oProject.ExportComponent (["NAME:mylib", "myscript"], "PersonalLib")</pre>

RemoveScript

Use: Remove Script in the script definition manager

Command: None

Syntax: RemoveScript <ScriptName>,<IsProjectScript>, <LibraryName>,<LibraryLocation>

Return Value: None

Parameters: <ScriptName>

Type: <string>

<IsProjectScript>

Type: <bool>

<LibraryName>

Type: <string>

<LibraryLocation>

Type: <string>

Python Syntax	RemoveScript (<ScriptName>,<IsProjectScript>, <LibraryName>,<LibraryLocation>)
Python Example	oDefinitionManager.RemoveScript ("myscript", true, "Local", "Project")

ModifyLibraries

Use: Configure Libraries on the Tools menu

Command: None

Syntax: ModifyLibraries <DesignName>,Array(<ConfigLibArray>)

Return Value: None

Parameters: <DesignName>
Type: <string>
<ConfigLibArray>
Array("NAME:<LibraryType>,<ConfiguredLib>,<ConfiguredLib>,...),...
<ConfiguredLib> // blank to leave unchanged
<DefinitionType>
Array("<libraryname >","<libraryname>",<libraryname>,...)

Python Syntax	ModifyLibraries (<DesignName>,[<ConfigLibArray>])
Python Example	<pre>oDefinitionManager.ModifyLibraries("MyCircuit", _ ["NAME:PersonalLib"], _ ["NAME:UserLib"], _ ["NAME:SystemLib", _ "Symbols:=", ["Circuit Elements", "Symbols",_ "ParamExtraElements\PE_Symbols", _ "Vendor Elements\Nonlinear"]])</pre>

This page intentionally
left blank.

8 - 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.

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

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

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

[Object Script Property Function Summary](#)

[Conventions Used in this Chapter](#)

[ChangeProperty](#)

[GetArrayVariables](#)

[GetProperties](#)

[GetPropertyValue](#)

[GetVariables](#)[GetVariableValue](#)[SetPropertyValue](#)[SetVariableValue](#)[Example Use of Record Script and Edit Properties](#)

Object Script Property Function Summary

Object Path

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

- An Object path consisted of one or multiple Object-ID-Nodes separated by "/" .
- Object-ID-Node; may exist in the following forms:
 - A simple object name or property name.
 - Type[Name] for object; Tab[name] for property.
 - Name[attr1="v1", attr2 = "v2", ...]. When more than one child object have the same name, use attributes to specify the difference.
 - ArrayName[index]. For example, in an Optimetric setup with multiple calculations, "Calculation[0]" could be used to identify the first calculation.
 - Name beginning with '@' character denoted as a property name, when an object has a child and property with the same name.

Property Object

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

- GetName()
 - Returns the name of the object.

- GetChildTypes()
 - An object may have different type of children. For example, a design may have variables, modules, and editors.
 - Returns array of text strings; may be empty if the children are *not* divided to different types.
- GetChildNames(<type>)
 - <type> – children type name; default returns all children names for all types.
 - Returns an array of immediate children names, belonging to the type if specified.
- GetChildObject(<objPath>)
 - <objPath> – A child object path. The path may include multiple generations, such as (designObject/moduleObj/SetupObject).
 - Returns a child property object if the object found.
- GetPropEvaluatedValue(<propName>)
 - Return the Evaluated-Value for Value-Property and Variable.
 - Return the Property-value as text string for other property types.
- GetPropSIValue(<propName>)
 - Return the SI-Value for Value-Property and Variable.
 - Return NAN for other property type if its value is cannot be converted to a double-floating point value.
- GetPropNames(<blIncludeReadOnly>);
 - <blIncludeReadOnly> – optional, default to true; True will include read-only properties, False will exclude read-only properties.
 - Returns an array of the object's property names.
- GetPropValue(<propertyPath>)
 - <propertyPath> – the property's full path. A property name or child object's path appended with a property name, like "TeeModel/Offset/SIValue"
 - Returns the property value if the property is found; otherwise causes script error.

- `SetPropValue(<propertyPath>, <data>)`
 - `<propertyPath>` – the property's full path. A property name or child object's path appended with a property name, like "TeeModel/Offset/Value"
 - `<data>` – new data, type is dependent on property type.
 - Returns True if property data is updated successfully; False if the new data is invalid.

Project Object

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

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

`oDesign = oProject("TeeModel")`

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

`oReport = oProject.GetChildObject("TeeModel/Results/S Parameter Plot 1")`
- `GetPropNames(bIncludeReadOnly)` always return empty array since the project has no property.
- `GetPropValue(propertyPath)`

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

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

Design Object

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

- GetChildTypes() always return ['Module', 'Editor', 'Variable'].
- GetChildNames(type)
GetChildNames() will return modules & editor child names.
GetChildNames("Variable") will return all variable names
GetChildNames("Module") will return all module names that support property-object-script like ['Optimetrics', 'RadField', 'Results']
GetChildNames("Editor") will return a 3D editor name for all 3D Designs
- GetChildObject()
oVariable = oDesign.GetChildObject("VariableName")
oReport = oDesign.GetChildObject("Results/S Parameter Plot 1")
oRptModule = oDesign.GetChildObject("ReportSetup")
- GetPropNames(bIncludeReadOnly) always return empty array since the design has no property.
- GetPropValue()
oDesign.GetPropValue("offset/SIValue") //get the offset variable SI value in the Design
oDesign.GetPropValue("offset/SIValue") //get the offset variable SI value in the Design
oDesign.GetPropValue("Results/S Parameter Plot 1/Display Type") // Get the report display type
- SetPropValue()
oDesign.SetPropValue("offset", "2mm") //Set the offset variable value to "2mm" in the Design
oDesign.SetPropValue("Results/S Parameter Plot 1/Display Type", "Data Table") // Set the report display type to data table.

3D Modeler Object

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

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

Variable Object

Is a Property Object that has no child. It also provides quick function call to get/set its 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 be accessed by the project or design object's GetChildObject(VariableName) function.

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

- GetChildTypes() always return empty array.
- GetChildNames() always return empty array, since variable has no child.
- GetChildObject(objPath) it has no child.
- GetPropNames(bIncludeReadOnly) ['EvaluatedValue', 'SIValue'] are read-only properties
 - Independent variable :['Value', 'EvaluatedValue', 'SIValue', 'Description', 'ReadOnly', 'Hidden', 'Sweep', 'Optimization/Included', 'Optimization/Min', 'Optimization/Max', 'Sensitivity/Included', 'Sensitivity/Min', 'Sensitivity/Max', 'Sensitivity/IDisp', 'Statistical', 'Statistical/Included', 'Tuning/Included', 'Tuning/Step', 'Tuning/Min', 'Tuning/Max'].
 - Dependent variable ['Value', 'EvaluatedValue', 'SIValue', 'Description', 'ReadOnly', 'Hidden', 'Sweep']
- GetPropValue(propName)
 - oVar.GetPropValue() return the variable value as text string.
 - oVar.GetPropValue("Value") return the variable value as text string.
 - oVar.GetPropValue("SIValue") return the SI-value of variable as number.
 - oVar.Get_SIValue() also return the SI value.

- SetPropValue(propName, newValue)
oVar.SetPropValue("Value", 888)
oVar.SetPropValue("Sensitivity/Included", True)
oVar.SetPropValue("Sensitivity/Max", '1.8pF')
oVar.Set_Sensitivity_Max('1.8pF') also works as last call.
oVar.SetPropValue("Sensitivity", ['Min:=' , '0.8pF', 'Max:=' , '1.8pF'])
//set multiple attributes at one call:
oVar.SetPropValue("@", ['Value:=' , 288, 'Sensitivity', ['Included', True, 'Min', '0.0']])
oVar.SetPropValue("", ['Value:=' , 288, 'Sensitivity', ['Included', True, 'Min', '0.0']])

Optimetrics Module Object:

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

- GetChildTypes() there are six type of children, they are ['OptiParametric', 'OptiOptimization', 'OptiSensitivity', 'OptiStatistical', 'OptiDesignExplorer', 'OptiDXDOE']. But the return array only included those that have setup defined, so it may be an empty array if no optimetrics setup is defined. The GetChildNames(type) function also recognized the type name without the prefix "Opti".
- GetChildNames(type)
GetChildNames() will return all setup for all types.
GetChildNames("OptiOptimization") & GetChildNames("Optimization") will return all Optimization setup.
- GetChildObject()
oParamSetup = oOptModule.GetChildObject('ParametricSetup1') get the
oOptSetup = oOptModule.GetChildObject('OptimizationSetup1')
- GetPropNames(bIncludeReadOnly) always return empty array since the it has no property.
- GetPropValue(propPath) may be used to get its child's property value
oOptModule.GetPropValue("OptimizationSetup1\Optimizer") get the optimizer name for OptimizationSetup1
- SetPropValue(propPath, newValue) may be used to set its child's property value
oOptModule.SetPropValue(ParametricSetup1\Enabled", False) //disable ParametricSetup1

Optimetrics Setup Object

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

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

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

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

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

ReportSetup(Results) Module Object:

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

- GetChildTypes() always empty array.
- GetChildNames(type)
GetChildNames() return all report names

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

ReportSetup(Results) Module Child Objects:

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

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

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

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

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

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

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

- `SetPropValue(propName, newValue)`
`oTrace.SetPropValue("Primary sweep", "Freq")`

Radiation Module Object:

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

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

Radiation Module Child Objects:

These are Property Objects. It also provides quick function call to get/set its properties by adding functions with property name appended to `Get_` & `Set_` prefix. To find what functions it provides enter `dir(oVar)` the command window.

Those child objects can be access by call all levels of parent object's `GetChildObject(path)` function.

`oRadSetup = oRadModule.GetChildObject(setupName)`

`oRadSetup = oDesign.GetChildObject(RadField/setupName)`

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

- GetPropNames(bIncludeReadOnly) will return the property names listed in the property window when the setup is selected.
- GetPropValue(propName)
 - oRadSetup.GetPropValue("Num Points") return the line setup's "Num Points" property value.
 - oRadSetup.Get_NumPoints() will also get the same value.
- SetPropValue(propName, newValue)
 - oRadSetup.SetPropValue('Num Points', 888)
 - oRadSetup.Set_NumPoints(888)

Conventions Used in this Chapter

General Definitions:

Property	A single item that can be modified in the Properties window or in the modal Properties pop-up window.
<PropServer>	The item whose properties are being modified. This is usually a compound name, giving all information needed by the editor, design, or project in order to locate the item.
<PropTab>	Corresponds to one tab in the Properties window, the one under which properties are being edited.
<PropName>	The name of a single property.

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

For Project Variables:

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

For Local Variables:

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

For Passed Parameters:

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

For Definition Parameters:

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

For Modules and Editors:

<PropServer>	<ModuleName>:<ItemName> where <ItemName> is the boundary name, solution setup name, etc. For example, "BoundarySetup:PerfE1"
<PropTab>	Boundary Module: "HfssTab" Mesh Operations Module: "MeshSetupTab" Analysis Module: "HfssTab" Optimetrics Module: "OptimetricsTab" Solutions Module: <i>Does not support properties.</i> Field Overlays Module: "FieldsPostProcessorTab" Radiation Module: "RadFieldSetupTab" Circuit Module: "CCircuitTab" System Module: "SystemTab" HFSS 3D Layout Module: "HFSS 3D LayoutTab" Nexxim Module: "NexximTab" Layout elements: "BaseElementTab" Schematic elements: "ComponentTab" Optimetrics Module: "OptimetricsTab"

For 3D Model Editor objects:

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

For 3D Model Editor operations:

<PropServer>	<ObjName>:<OperationName>:<int> where <int> is the operation's history index. For example, "Box2:CreateBox:2" refers to the second "CreateBox" operation in Box2's history.
<PropTab>	"Geometry3DCmdTab"

For Reporter operations on Report properties:

<PropServer>	<ReportSetup>
<ChangeProperty>	Array. For example, to set the company name in a plot header to "My Company": <pre>Set oModule = oDesign.GetModule("ReportSetup") oModule.ChangeProperty Array("NAME:AllTabs",_ Array("NAME:Header",_ Array ("NAME:PropServers",_ "XY Plot1:Header"), Array("NAME:ChangedProps",_ Array("NAME:Company Name", "Value:=", "My Company"))))</pre>

For Maxwell Modules:

<PropServer>	<ModuleName>:<ItemName> where <ItemName> is the solution setup name.
<PropTab>	Boundary module: "MaxwellTab" Mesh Operations module: "MeshSetupTab" Analysis module: "MaxwellTab" Optimetrics module: "OptimetricsTab"

	Solutions module: <i>Does not support properties.</i> Field Overlays module: "FieldsPostProcessorTab" Radiation module: "RadFieldSetupTab"

Note:

For scripted property changes in the various modules and editors, refer to the chapters on the System, HFSS 3D Layout, and Nexxim tools, as well as the Layout and Schematic editors.

ChangeProperty (HFSS and Maxwell)

Use: Changes to properties are scripted using the ChangeProperty command. This command can be executed by the oEditor to change editor properties, by the oDesign to change design level properties, and by the oProject to change project level properties. The command can be used to create, edit, and/or remove properties. In HFSS and Maxwell, only Variable and Separator properties can be deleted.

Use the script recording feature and edit a property, and then view the resulting script entry or use GetPropertyValue for the desired property to see the expected format.

Command: None

Syntax: ChangeProperty Array("Name:AllTabs", <PropTabArray>, <PropTabArray>, ...)

ChangeProperty(<modulename>:<setup name>:<sweep name>)

Return Value: None

Parameters: <PropTabArray>

```
Array("Name:<PropTab>",
      <PropServersArray>,
      <NewPropArray>,
      <ChangedPropsArray>,
```

```
<DeletedPropsArray>)
```

```
<PropServersArray>
```

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

```
<NewPropArray>
```

```
Array("Name:NewProps", <PropDataArray>,  
<PropDataArray>, ...)
```

```
<ChangedPropsArray>
```

```
Array("Name:ChangedProps", <PropDataArray>,  
<PropDataArray>, ...)
```

```
<DeletedPropsArray>
```

```
Array("Name:DeletedProps", <PropName>,  
<PropName>, ...)
```

```
<PropDataArray>
```

```
Array("NAME:<PropName>",
```

```
"PropType:=", <PropType>,  
"NewName:=", <string>,  
"Description:=", <string>,  
"NewRowPosition:=", <int>,  
"ReadOnly:=", <bool>,  
"Hidden:=", <bool>,  
<PropTypeSpecificArgs>
```

<PropType>

Type: string

Identifies the type of property when a new property is added. In HFSS and Maxwell, only separator properties and variable properties can be added.

```
"SeparatorProp"  
"VariableProp"  
"TextProp"  
"NumberProp"  
"ValueProp"  
"CheckboxProp"  
"MenuProp"  
"PointProp"  
"VPointProp"  
"V3DPointProp"
```

"ButtonProp"

NewName

Specify the new name of a property if the property's name is being edited. In HFSS and Maxwell, the name can only be changed for separators and variables.

Description

Specify a description of the property. In HFSS and Maxwell, the description can only be changed for separators and variables.

NewRowPosition

Used to reorder rows in the **Property** dialog box.

In HFSS, this only applies to the **Project>Project Variables** panel and the **Hfss>DesignProperties** panel.

In Maxwell, this only applies to the **Project>Project Variables** panel and the **Maxwell3D>DesignProperties**, **Maxwell2D>Design Properties**, or **RMxpert>Design Properties** panels.

Specify the new zero-based row index of the variable or separator.

ReadOnly

Used to mark a property as "read only" so it can not be modified. In HFSS and Maxwell, this flag can only be set for variables and separators.

Hidden

Used to hide a property so it can not be viewed outside of the **Property** dialog box. In HFSS and Maxwell, this flag can only be set for variables and separators.

<PropTypeSpecificArgs>

SeparatorProp: no arguments

TextProp: "Value:=", <string>

NumberProp: "Value:=", <double>

ValueProp: "Value:=", <value>

CheckboxProp: "Value:=", <bool>

MenuProp: "Value:=", <string>

PointProp "X:=", <double>, "Y:=", <double>

VPointProp: "X:=", <value>, "Y:=", <value>

V3DPointProp: "X:=", <value>, "Y:=", <value>, "Z:=", <value>

Material Button: "Material:=", <string>

Color Button: "R:=", <int>, "G:=", <int>, "B:=", <int>

Transparency Button: "Value:=", <double>

<PropTypeSpecificArgs> for VariableProps

Syntax:

"Value:=", <value>, <OptimizationFlagsArray>,

<TuningFlagsArray>, <SensitivityFlagsArray>,

<StatisticsFlagsArray>

Parameters:

<OptimizationFlagsArray>

Array("NAME:Optimization",

"Included:=", <bool>,

"Min:=", <value>,

"Max:=", <value>)

<Tuning flagsArray>

Array("NAME:Tuning",

"Included:=", <bool>,

"Step:=", <value>,

"Min:=", <value>,

"Max:=", <value>)

<SensitivityFlagsArray>

Array("NAME:Sensitivity",

"Included:=", <bool>,

"Min:=", <value>,

"Max:=", <value>,

"IDisp:=", <value>)

<StatisticsFlagsArray>

Array("NAME:Statistical",

"Included:=", <bool>,

"Dist:=", <Distribution>,

"StdD:=", <value>,

"Min:=", <value>,

"Max:=", <value>,

"Tol:=", <string>)

<Distribution>

Type: string

Value should be "Gaussian" or "Uniform"

StdD

Standard deviation.

Min

Low cut-off for the distribution.

Max

High cut-off for the distribution.

Tol

Tolerance for uniform distributions. Format is "<int>%".

Example: "20%".

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>

GetProperties

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

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

Python Syntax	GetProperties(<PropTab>, <PropServer>)
Python 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
	<i><PropTab></i>	String	One of the following, where tab titles are shown in parentheses: - PassedParameterTab ("Parameter Values") - DefinitionParameterTab (Parameter Defaults") - LocalVariableTab ("Variables" or "Local Variables") - ProjectVariableTab ("Project variables") - ConstantsTab ("Constants") - BaseElementTab ("Symbol" or "Footprint") - ComponentTab ("General") - Component("Component") - CustomTab ("Intrinsic Variables") - Quantities ("Quantities") - Signals ("Signals")
	<i><PropServer></i>	String	An object identifier, generally returned from another script method, such as <code>Component@R;2;3</code>
	<i><PropName></i>	String	Name of the property.

Return Value	String value of the property.

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

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	<pre>oProject.GetVariables() oDesign.GetVariables()</pre>

GetVariableValue

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

UI Access	N/A		
Parameters	Name	Type	Description
	<VarName>	String	Name of the variable to access.
Return Value	String represents the value of the variable.		

Python Syntax	GetVariableValue(<VarName>)
Python 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	Type	Description
	<code><propTab></code>	String	One of the following, where tab titles are shown in parentheses: - PassedParameterTab ("Parameter Values") - DefinitionParameterTab (Parameter Defaults") - LocalVariableTab ("Variables" or "Local Variables") - ProjectVariableTab ("Project variables") - ConstantsTab ("Constants") - BaseElementTab ("Symbol" or "Footprint") - ComponentTab ("General") - Component("Component") - CustomTab ("Intrinsic Variables") - Quantities ("Quantities") - Signals ("Signals")
	<code><propServer></code>	String	An object identifier, generally returned from another script method, such as <code>CompInst@R;2;3</code>
	<code><propName></code>	String	Name of the property.
	<code><propValue></code>	String	The value for the property
Return Value	None.		

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

SetVariableValue

Sets the value of a variable. To set the value of a project variable, execute this command using `oProject`. To set the value of a local variable, use `oDesign`.

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

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

9 - Design Object Script Commands

Design object commands should be executed by the oDesign object.

```
oDesign.CommandName <args>
```

For example:

```
Set oDesign = oProject.SetActiveDesign("MaxwellDesign1")
```

Conventions Used in this Chapter

<ModuleName> is a placeholder for any one of the following modules:

- [Analysis Module](#) – "AnalysisSetup"
- [Boundary Module](#) – "BoundarySetup"
- [Field Overlays Module](#) – "FieldsReporter"
- [Mesh Module](#) – "MeshSetup"
- [Optimetrics Module](#) – "Optimetrics"
- Radiation Module – "RadField"
- Reduce Matrix Module – "ReduceMatrix"
- [Reporter Module](#) – "ReportSetup"
- Simulation Setup Module – "SimSetup"
- [Solutions Module](#) – "Solutions"

[AnalyzeAll \(Maxwell\)](#)

[ApplyMeshOps](#)

[ConfigureFluentConductivityCoupling](#)

[ConstructVariationString](#)

[CopyArray](#)

[Create3DDesign](#)

[DeleteFieldVariation](#)

[DeleteFullVariation](#)

[DeleteLinkedDataVariation](#)

[EditNotes](#)

[EnableHarmonicForceCalculation](#)

[ExportConvergence](#)

[ExportElementBasedHarmonicForce](#)

[ExportHarmonicTransientForce](#)

[ExportLayoutLayerGridField](#)

[ExportLayoutViaCurrent](#)

[ExportLayoutViaCurrentDensity](#)

[ExportMatrixData](#)

[ExportMeshStats](#)

[ExportProfile](#)

[ExportReport](#)

[GenerateMesh](#)

[GetAllPorts](#)

[GetChildNames \[Design\]](#)

[GetChildObject \[Design\]](#)

[GetChildTypes \[Design\]](#)

[GetDesignType](#)

[GetDesignValidationInfo](#)

[GetGeometryMode](#)

[GetManagedFilePath](#)

[GetModule](#)

[GetName](#)

[GetNominalVariation](#)

[GetNoteText](#)

[GetProject](#)

[GetPostProcessingVariables](#)

[GetPropNames \[Design\]](#)

[GetPropValue \[Design\]](#)

[GetSelections](#)

[GetSolutionType](#)

[InitializeSystemCoupling](#)

[Is2D](#)

[Is3D](#)

[PasteDesign](#)

[Redo](#)

[RenameDesignInstance](#)

[RenameSource](#)

[RunToolkit](#)

[SetActiveEditor](#)

[SetConductivityThreshold](#)

[SetDesignSettings \(Maxwell\)](#)

[SetFastTransformationForLayoutComponent](#)

[SetObjectDeformation](#)

[SetObjectTemperature](#)

[SetPropValue \[Design\]](#)

[SetPropertyValue](#)

[SetShowLayoutForLayoutComponent](#)

[SetSolutionType \(Maxwell\)](#)

[Solve](#)

[Undo](#)

[ValidateDesign](#)

ApplyMeshOps

This script has been replaced by [GenerateMesh](#).

Analyze

Solves a single solution setup and all of its frequency sweeps.

UI Access	Right-click a solution setup in the project tree, and select Analyze .		
Parameters	Name	Type	Description
	<setup>	String	Setup name.
	isBlocking	Boolean	An optional arg that defaults to <code>true</code> . If it's <code>false</code> , the command immediately returns while the analysis or analyses run in the background. The status of the analyses can be checked with the <code>AreThereSimulationsRunning</code> command.
Return Value	Integer value: • 0 – Success • Else – Error		

Python Syntax	Analyze (<setup>)
Python Example	<code>oDesign.Analyze("Setup1", isBlocking)</code>

AnalyzeAll (Maxwell menu)

Analyzes the nominal problems and all the Optimetrics problems for all the setups under the selected design.

UI Access	Click Maxwell > Analyze All , or right-click a design instance in the project tree and select Analyze All .		
Parameters	Name	Type	Description
	isBlocking	Boolean	An optional arg that defaults to <code>true</code> . If it's <code>false</code> , the command immediately returns while the analysis or analyses run in the background. The status of the analyses can be checked with the <code>AreThereSimulationsRunning</code> command.
Return Value	None		

Python Syntax	AnalyzeAll ()
Python Example	oDesign.AnalyzeAll(isBlocking)

ConfigureFluentConductivityCoupling

Use: Configures the directory for the conductivity coupling between a Maxwell 3D magnetostatic design and Ansys Fluent.

Command: Right-click on the solution setup under Analysis in the project tree and choose **Configure Fluent Conductivity Coupling**.

Syntax: ConfigureFluentConductivityCoupling <SolveSetupName>, <Pathofcouplingdirectory>

Return Value: None

Parameters: <SolveSetupName>

Type: <string>

Name of the solution setup.

<PathOfCouplingDirectory>

Type: string

Path of the coupling directory.

ConstructVariationString

Lists and orders the variables and values associated with a design variation.

UI Access	N/A

Parameters	Name	Type	Description
	<code><ArrayOfVariableNames></code>	Array of Strings	List of variable names.
	<code><ArrayOfVariableValuesIncludingUnits></code>	Array of Strings	List of variable values, including units, in the same order as the list of names.
Return Value	Returns variation string with the variables ordered to correspond to the order of variables in design variations. The values for the variables are inserted into the variation string. For an example of how <code>ConstructVariationString</code> can be used, see the Python example.		

Python Syntax	<code>ConstructVariationString(<ArrayOfVariableNames>, <ArrayOfVariableValuesIncludingUnits>)</code>
Python Example	<pre>varStr = oDesign.ConstructVariationString(["xx", "yy"], ["2mm", "1mm"]) oDesign.ExportProfile("Setup1", varStr, "C:\profile.prof")</pre>

CreateParametricCircuit

Note:

This command is for internal Ansys use only.

Python Syntax	<code>CreateParametricCircuit()</code>
Python Example	<code>oDesign.CreateParametricCircuit()</code>

Create3DDesign [Maxwell]

Creates a Maxwell 3D design from a Maxwell 2D design.

UI Access	Maxwell 2D > Create 3D Design, or right-click a Maxwell 2D design instance in the project tree and select Create 3D Design.		
Parameters	Name	Type	Description
	<DesignSettings>	Array	Structured array. For a 2D design in "Cartesian, XY" geometry mode: ["NAME:Parameters", "ZLength:=", <value>, "PararmetrizeZLength:=", <bool>] For a 2D design in "Cylindrical about Z" geometry mode: ["NAME:Parameters", "SweepAngle:=", <value>, "PararmetrizeSweepAngle:=", <bool>, "DraftAngle:=", <value>, "DraftType:=", <string>, %one of: "Extended", "Round", "Natural" "NumberOfSegments:=", <integer>]
Return Value	None		

Python Syntax	Create3DDesign (<DesignSettings>)
Python Example	For a 2D design in "Cartesian, XY" geometry mode:

```
oDesign.Create3DDesign(
[
    "NAME:Parameters",
    "ZLength:=", "12mm",
    "ParametrizeZLength:=", true
])
```

For a 2D design in "Cylindrical about Z" geometry mode:

```
oDesign.Create3DDesign(
[
    "NAME:Parameters",
    "SweepAngle:=", "120deg",
    "ParametrizeSweepAngle:=", true, "DraftAngle:=", "30deg", "DraftType:=",
    "Round",
    "NumberOfSegments:=", "4"
])
```

DeleteFieldVariation

Deletes field variations, fields and meshes, or fields and meshes for specified variations.

UI Access	[Solver] > Results > Clean Up Solutions > [Fields Only / Fields and Meshes].		
Parameters	Name	Type	Description
	<variationKeys>	Array	Array containing either "All" string to select all variations, or strings of variations to include.
	<deleteMesh>	Boolean	When true, deletes mesh data.
	<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>

DeleteFullVariation

Deletes either all solution data, or selected variation data.

UI Access	[Maxwell] > Results > Clean Up Solutions.		
Parameters	Name	Type	Description
	<variationKeys>	Array	Array containing either "All" string to select all variations, or strings of variations to include.
	<deleteLinkedData>	Boolean	When true, deletes linked data as well.
Return Value	None.		

Python Syntax	DeleteFullVariation(<variationKeys>, <deleteLinkedData>)
Python Example	oDesign.DeleteFullVariation(['All'], false)

DeleteLinkedDataVariation

Deletes the linked data of specified variations.

UI Access	[Maxwell] > Results > Clean Up Solutions.

Parameters	Name	Type	Description
	<DesignVariationKeys>	Array	Array containing strings of the variations keys whose linked data are going to be deleted.
Return Value	None.		

Python Syntax	DeleteLinkedDataVariation(<DesignVariationKeys>)
Python Example	<code>oDesign.DeleteLinkedDataVariation(["current=\'0.9mA\'", "current=\'1.0mA\'"])</code>

EditNotes

Updates the notes for the design.

UI Access	Maxwell > Edit Notes.		
Parameters	Name	Type	Description
	<DesignNotes>	String	New design notes that are applied, replacing any current notes.
Return Value	None.		

Python Syntax	EditNotes(<DesignNotes>)
Python Example	<code>oDesign.EditNotes("My design notes.")</code>

EnableHarmonicForceCalculation (Transient Only)

This command sets the objects for harmonic force calculation for a transient solution.

UI Access	Maxwell 2D or 3D > Enable Harmonic Force Calculation, or right-click a design instance in the project tree and select Enable Harmonic Force Calculation		
Parameters	Name	Type	Description
	<ObjectArray>	String	List of object names for force calculations; for example, "EnabledObjects:=", ["Magnet", "Steel"]
	<ForceType>	Integer	0, ObjectBased; 1, Element Based (Surface); 2, Element Based (Volumetric)
	<NumberOfRepeatedWindows>	Integer	The number of repeated windows; it can also be a variable that evaluates to an integer.
	<UseNumberOfLastCycles>	Boolean	Use number of last cycles for force calculations
	<NumberOfLastCycles>	Integer	Number of last cycles
	<StartTime>	Double	The start time of data collection
	<UseNumberOfCyclesForStopTime>	Boolean	Use number of cycles or time to define the stop time. True if user wants to use the number of cycles for the stop time; otherwise, False.
	<NumberOfCyclesForStopTime>	String	Number of cycles after the start time to make the stop time.
	<StopTime>	Double	Sets specific stop time. Should be greater than the <StartTime>
	<CouplingSteps>	String	Specify "All", "Steps from end", or "Range of steps"
	<ThirdParty>	String	Third-party output file type. Specify "LMS", "Romax", or "UNV"
	<OutputDir>	String	Specify the path to the output directory for third-party files
	<WindowFunctionType>	String	Specify "Rectangular", "Tri", "Van Hann", "Hamming", "Blackman", "Lanczos", or "Welch"
	<OutputFreqRangeType>	String	Specify "Use Range", "Use Number", or "Use All"
	<OutputFreqRangeStop>	Double	Specify the output range stop frequency (with unit)
	<OutputFreqRangeStart>	Double	Specify the output range start frequency (with unit)
	<OutputFreqRangeNum>	Integer	Number count of frequencies. The number count starts from the start frequency from FFT results.
	<CalculateForceType>	String	"Harmonic Force", "Transient Force", or "Harmonic and Tran-

			sient Force"
	<EnableInverterFeedback>	Boolean	Specify switching and maximum frequency to mimic inverter-fed machine
	<SwitchingFrequency>	String	Input, must be greater than zero
	<MaximumFrequency>	String	Input, must be less that switching frequency
Return Value	None		

Python Syntax	EnableHarmonicForceCalculation (<ObjectArray>, <ForceType>, <NumberOfRepeatedWindows>, <UseNumberOfLastCycles>, <NumberOfLastCycles>, <StartTime>, <UseNumberOfCyclesForStopTime>, <NumberOfCyclesForStopTime>, <StopTime>, <CouplingSteps>, <ThirdParty>, <OutputDir>, <WindowFunctionType>, <OutputFreqRangeType>, <OutputFreqRangeStop>, <OutputFreqRangeStart>, <OutputFreqRangeNum>, <CalculateForceType>, <EnableInverterFeedback>, <MaximumFrequency>)
Python Example	Example 1: Setting WindowFunctionType options for a transient design <pre>oDesign.EnableHarmonicForceCalculation(["EnabledObjects:=", ["Cylinder1","Cylinder2","Cylinder3"], "ForceType:=", 0, "NumberOfRepeatedSamples:=", "1", "UseNumberOfLastCycles:=", False, "NumberOfLastCycles:=", 1, "StartTime:=", "0s", "UseNumberOfCyclesForStopTime:=", False, "NumberOfCyclesForStopTime:=", 1,</pre>

```
"StopTime:=", "0.01s",  
"WindowFunctionType:=", "Rectangular",  
])
```

Example 2: Using inverter feedback

```
oDesign.EnableHarmonicForceCalculation(  
[  
"EnabledObjects:=", ["Cylinder1"],  
"ForceType:=", 0,  
"WindowFunctionType:=", "Rectangular",  
"NumberOfRepeatedSamples:=", "1",  
"UseNumberOfLastCycles:=", True,  
"NumberOfLastCycles:=", 1,  
"StartTime:=", "0s",  
"UseNumberOfCyclesForStopTime:=", True,  
"NumberOfCyclesForStopTime:=", 1,  
"StopTime:=", "0.01s",  
"OutputFreqRangeType:=", "Use All",  
"CalculateForceType:=", "Harmonic Force",  
"EnableInverterFeedback:=", True,  
"SwitchingFrequency:=", "4000Hz",
```



```
"MaximumFrequency:="      , "8000Hz"
```

```
])
```

Example 3: Setting OutputFreqRangeType to "Use Range"

```
oDesign.EnableHarmonicForceCalculation(
[
  "EnabledObjects:="      , ["Cylinder1","Cylinder2"],
  "ForceType:="           , 0,
  "WindowFunctionType:="  , "Rectangular",
  "NumberOfRepeatedSamples:=", "1",
  "UseNumberOfLastCycles:=", False,
  "NumberOfLastCycles:="  , 1,
  "StartTime:="           , "0s",
  "UseNumberOfCyclesForStopTime:=", False,
  "NumberOfCyclesForStopTime:=", 1,
  "StopTime:="            , "0.01s",
  "OutputFreqRangeStart:="      , "1Hz",
  "OutputFreqRangeStop:="      , "3Hz",
  "OutputFreqRangeType:="      , "Use Range"
])
```

Example 4: Setting OutputFreqRangeType to "Use Number"

```
oDesign.EnableHarmonicForceCalculation(
```

```
[
  "EnabledObjects:="      , ["Cylinder1","Cylinder2"],
  "ForceType:="           , 0,
  "WindowFunctionType:="  , "Rectangular",
  "NumberOfRepeatedSamples:=", "1",
  "UseNumberOfLastCycles:=", False,
  "NumberOfLastCycles:="  , 1,
  "StartTime:="           , "0s",
  "UseNumberOfCyclesForStopTime:=", False,
  "NumberOfCyclesForStopTime:=", 1,
  "StopTime:="            , "0.01s",
  "OutputFreqRangeStart:="      , "1Hz",
  "OutputFreqRangeNum:="       , "43",
  "OutputFreqRangeType:="      , "Use Number"
])
```

Example 5: Setting OutputFreqRangeType to "Use All"

```
oDesign.EnableHarmonicForceCalculation(
```

```
[
  "EnabledObjects:="      , ["Cylinder1","Cylinder2"],
  "ForceType:="           , 0,
```

```

"WindowFunctionType:=" , "Rectangular",
"NumberOfRepeatedSamples:=", "1",
"UseNumberOfLastCycles:=", False,
"NumberOfLastCycles:=" , 1,
"StartTime:=" , "0s",
"UseNumberOfCyclesForStopTime:=", False,
"NumberOfCyclesForStopTime:=", 1,
"StopTime:=" , "0.01s",
" OutputFreqRangeType:=" , "Use All"
])

```

Example 6: Setting layout force nets and layers sets

```

oDesign.EnableHarmonicForceCalculation(
[
"ForceType:=" , 0,
"WindowFunctionType:=" , "Rectangular",
"NumberOfRepeatedSamples:=", "1",
"UseNumberOfLastCycles:=", True,
"NumberOfLastCycles:=" , 1,
"StartTime:=" , "0s",
"UseNumberOfCyclesForStopTime:=", True,
"NumberOfCyclesForStopTime:=", 1,

```

```
"StopTime:="      , "0.01s",
"OutputFreqRangeType:=" , "Use All",
"CaculateForceType:=" , "Harmonic Force",
[
  "NAME:NetsAndLayersChoices",
  [
    "NAME:LC1_1",
    [
      "NAME:NetLayerSetMap",
      [
        "NAME:net1",
        "LayerSet:="      , ["Top"]
      ],
      [
        "NAME:net2",
        "LayerSet:="      , ["Top"]
      ]
    ]
  ]
]
```

```
])
```

```
...
```

Example 7: Calculating Harmonic AND Transient force

```
oDesign. EnableHarmonicForceCalculation (  
  [  
    "EnabledObjects:=" , ["Cylinder1"],  
    "ForceType:=" , 1,  
    "WindowFunctionType:=" , "Rectangular",  
    "NumberOfRepeatedSamples:=" , "1",  
    "UseNumberOfLastCycles:=" , True,  
    "NumberOfLastCycles:=" , 1,  
    "StartTime:=" , "0s",  
    "UseNumberOfCyclesForStopTime:=" , True,  
    "NumberOfCyclesForStopTime:=" , 1,  
    "StopTime:=" , "0.01s",  
    "OutputFreqRangeType:=" , "Use All",  
    "CalculateForceType:=" , "Harmonic and Transient Force",  
    "EnableInverterFeedback:=" , False  
  ]  
)
```

ExportConvergence

For a given variation, exports convergence data (max mag delta S, E, freq) to *.conv file.

UI Access	Results > Solution Data. Select Convergence tab and click Export .		
Parameters	Name	Type	Description
	<Setup>	String	Setup name.
	<VariationKeys>	String	The variation. Pass empty string for the current nominal variation.
	<FilePath>	String	Full path to desired *.conv file location.
Return Value	None.		

Python Syntax	ExportConvergence(<Setup>, <VariationKeys>, <FilePath> <OverwriteIfExists>)
Python Example	oDesign.ExportConvergence('Setup1', 'x_size = 2mm', 'c:\convergence.conv')

ExportElementBasedHarmonicForce (Transient only)

Exports a element-based harmonic force data to a .csv file for a transient solution.

UI Access	Maxwell 3D > Export Transient / Harmonic Force		
Parameters	Name	Type	Description
	<outputDirectory>	String	The path for the output directory.
	<solveSetupName>	String	The name of the solution setup.
	<number1> ,<number2>		Possible values for <i>number1</i> and <i>number2</i> :

			- When <i>frequencyOption</i> is 1, <i>number1</i> is -1, <i>number2</i> is -1 - When <i>frequencyOption</i> is 2, <i>number1</i> is <i>startFrequency</i><double>, <i>number2</i> is <i>endFrequency</i><double> - When <i>frequencyOption</i> is 3, <i>number1</i> is <i>number of frequencies</i><int>, <i>number2</i> is -1
Return Value	None.		

Python Syntax	ExportElementBasedHarmonicForce(<outputDirectory>, <solveSetupName>, <frequencyOption>, <number1>, <number2>)		
Python Example	<pre>ExportElementBasedHarmonicForce("E:\\tmp", "Setup1", 1, -1, -1) ExportElementBasedHarmonicForce("E:\\tmp", "Setup1", 2, 2.3, 10.5) ExportElementBasedHarmonicForce("E:\\tmp", "Setup1", 3, 4, -1)</pre>		

ExportHarmonicTransientForce

Exports harmonic force solution data from transient designs for use by Ansys Motion and third-party software.

UI Access	Maxwell 2D (or 3D) > Export Harmonic/Transient Force...		
Parameters	Name	Type	Description
	<ExportType>	string	One of: • UNV • Romax • LMS

			• Ansys Motion • Ansys Mechanical
	<ParametricSetupName>	string	The name of the parametric setup whose solution data is to be exported
	<ExportDirectory>	string	The path for the exported files
	<SolverSetupName>	string	The name of the design setup whose solution data is to be exported. If no solution setup is specified, all solve setups under the specified parametric setup will be exported.
	<ExportTimeOption>	integer	Export time steps: 0 for All steps, 1 for Steps from end, 2 for Range of steps.
	<ExportBeginStep>	integer	Integer value for begin step, only valid for <i>ExportTimeOption</i> : Range of steps
	<ExportEndStep>	integer	Integer value for end step, only valid for <i>ExportTimeOption</i> : Range of steps
Return Value	None		

Python Syntax	ExportHarmonicTransientForce (<ExportType>, <ParametricSetupName>, <ExportDirectory>, <SolverSetupName>, <ExportTimeOption>, <ExportBeginStep>, <ExportEndStep>)
Python Example	<pre>oDesign.ExportHarmonicTransientForce("Ansys Motion", " ParametricSetup1", "\$PROJECTDIR", "Setup1", 0, -1, -1)</pre>

ExportMeshStats

Exports mesh statistics to a file.

UI Access	N/A

Parameters	Name	Type	Description
	<setupName>	String	Name of the solution setup.
	<variationString>	String	Name of the variation. An empty string is interpreted as the current nominal variation.
	<filePath>	String	Full file path, including extension *.ms.
	<overwriteIfExists>	Boolean	(Optional). Defaults to True. If True, script overwrites any existing file of the same name. If False, no overwrite.
Return Value	None.		

Python Syntax	ExportMeshStats(<setupName>, <variationString>, <filePath>, <overwriteIfExists>)
Python Example	oDesign.ExportMeshStats ("Setup1", "radius=3mm", "C:\mydir\meshstats.ms", True)

ExportProfile

Exports a solution profile to file.

UI Access	N/A		
Parameters	Name	Type	Description
	<SetupName>	String	Text of the design's notes.
	<VariationString>	String	Variation name. Pass empty string for the current nominal variation.
	<filePath>	String	Full file path, including extension *.prof.
Return Value	None.		

Python Syntax	ExportProfile(<SetupName>, <VariationString>, <filePath>, <overwriteIfExists>)

Python Example	<code>oDesign.ExportProfile('Setup1', 'radius=3mm', 'C:\Files\Profile.prof', True)</code>

GetDesignValidationInfo

Copies a validation cache file that is generated by `oDesign.ValidateDesign()` or doing design validation from UI to a file. This file contains more information besides messages. It also has validation results of some validation check items.

If there is no validation cache file, a file won't be generated. This happens if `GetDesignValidationInfo` is executed and no `ValidateDesign` was performed.

UI Access	NA		
Parameters	Name	Type	Description
	None		
Return Value	Integer 0 if invalid 1 if valid		

Python Syntax	<code>GetDesignValidationInfo()</code>
Python Example	<code>oProject = oDesktop.GetActiveProject() oDesign = oProject.GetActiveDesign() oDesign.GetDesignValidationInfo()</code>

GetGeometryIdsForAllNetLayerCombinations

Returns ID numbers of all faces and edges of the active design for all possible combinations of nets and layers the user can choose.

Note: Intended to support [CreateFieldPlot](#).

UI Access	N/A		
Parameters	Name	Type	Description
	<setupName>	String	Name of the setup.
Return Value	Array of strings indicating face and edge ID numbers.		

Python Syntax	GetGeometryIdsForAllNetLayerCombinations ()
Python Example	<pre>oDesign = oProject.GetActiveDesign() oDesign.GetGeometryIdsForAllNetLayerCombinations("HFSS Setup : Last Adaptive")</pre>
Example with Variables	<pre>oProject = oDesktop.SetActiveProject("Spiral_Inductor_Microstrip") oDesign = oProject.SetActiveDesign("Spiral") res = oDesign.GetGeometryIdsForAllNetLayerCombinations("HFSS Setup : Last Adaptive") res = ['PlotGeomInfo for <no-net>/<no-layer> (net/layer combination):', 'Surface', 'FacesList', '30', '31', '32', '33', '34', '35', '36', '37', '38', '39', '40', '41', '42', '43', '44', '45', '46', '47', '48', '49', '50', '51', '52', '53', '54', '55', '56', '57', 'PlotGeomInfo for <no-net>/AirbridgeMetal</pre>

	(net/layer combination):', 'Surface', 'FacesList', '14', 'PlotGeomInfo for <no-net>/BottomGround (net/layer combination):', 'Surface', 'FacesList', '29', 'PlotGeomInfo for <no-net>/Trace (net/layer combination):', 'Surface', 'FacesList', '21', '22']

GetGeometryIdsForNetLayerCombinations

Returns ID numbers of all faces and edges of the active design related to the current target combination of nets and layers.

Note: Intended to support [CreateFieldPlot](#).

UI Access	N/A		
Parameters	Name	Type	Description
	<netName>	String	Net's name.
	<layerName>	String	Layer's name.
	<setupName>	String	Name of the setup.
Return Value	Array of strings indicating face and edge ID numbers and net/layer combination they are assigned to.		

Python Syntax	GetGeometryIdsForNetLayerCombinations ()
Python Example	<pre>oDesign = oProject.GetActiveDesign() oDesign.GetGeometryIdsForAllNetLayerCombinations ("<no-net>","Trace","HFSS Setup : Last Adaptive")</pre>

Example with Variables	<pre> oProject = oDesktop.SetActiveProject("Spiral_Inductor_Microstrip") oDesign = oProject.SetActiveDesign("Spiral") res = oDesign.GetGeometryIdsForNetLayerCombination("<no-net>","Trace","HFSS Setup : Last Adaptive") res = ['Surface', 'FacesList', '21', '22'] </pre>

GetGeometryMode

Returns the geometry mode for the design. For Maxwell 2D only.

UI Access	N/A		
Parameters	Name	Type	Description
	None		
Return Value	String "XY" if design is "Cartesian, XY" "about Z" if design is "Cylindrical about Z"		

Python Syntax	GetGeometryMode()
Python Example	<pre> oProject = oDesktop.GetActiveProject() oDesign = oProject.GetActiveDesign() oDesign.GetGeometryMode() </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	<code>oDesign.GetManagedFilePath()</code>
Python Example	<code>oDesign.GetManagedFilePath()</code>

GetModule

Returns the IDispatch for the specified module.

UI Access	N/A		
Parameters	Name	Type	Description
	<modulename>	String	One of the following: • Analysis Module – "AnalysisSetup" • Boundary Module – "BoundarySetup" • Field Overlays Module – "FieldsReporter" • Mesh Module – "MeshSetup"

			• Optimetrics Module – "Optimetrics" • Radiation Module – "RadField" • Reduce Matrix Module – "ReduceMatrix" • Reporter Module – "ReportSetup" • Simulation Setup Module – "SimSetup" • Solutions Module – "Solutions"
Return Value	Module IDispatch		

Python Syntax	GetModule (<modulename>)
Python Example	<code>oModule = oDesign.GetModule("SimSetup")</code>

GetName

Returns the design name of the active design, in that order separated by a semicolon.

UI Access	N/A
Parameters	None.
Return Value	String indicating the name of the active design.

Python Syntax	GetName()
Python Example	<code>design_name = oDesign.GetName()</code>

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>

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>

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>

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 Com-

	posite]", "Eigenmode", or "Characteristic".

Python Syntax	GetSolutionType()
Python Example	<code>oDesign.GetSolutionType()</code>

GetVariationVariableValue

Returns the value for a specified variation's variable.

UI Access	N/A		
Parameters	Name	Type	Description
	<VariationString>	String	The name of the design variation.
	<VariableName>	String	The name of the variable.
Return Value	Returns a double precision value in SI units, interpreted to mean the value of the variable contained in the variation string.		

Python Syntax	GetVariationVariableValue(<VariationString>, <VariableName>)
Python Example	<code>oDesign.GetVariationVariableValue('x_size = 2mm y_size = 1mm', 'y_size')</code>

Is2D

Determines whether the current design is 2D.

UI Access	No corresponding user interface command
Parameters	None.
Return Value	Boolean; <code>True</code> if the design is 2D.

Python Syntax	<code>Is2D()</code>
Python Example	<code>oDesign.Is2D()</code>

Is3D

Determines whether the current design is 3D.

UI Access	No corresponding user interface command
Parameters	None.
Return Value	Boolean; <code>True</code> if the design is 3D.

Python Syntax	<code>Is3D()</code>
Python Example	<code>oDesign.Is3D()</code>

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>

RenameDesignInstance

Renames a design instance.

UI Access	Right-click a design instance in the project tree, and then click Rename on the shortcut menu.		
Parameters	Name	Type	Description
	<OldName>	String	The current name of the design, which must be the design on which this command is invoked.
	<NewName>	String	The new name for the design.
Return Value	None.		

Python Syntax	<code>RenameDesignInstance (<OldName>, <NewName>)</code>
Python Example	<code>oDesign.RenameDesignInstance ("Design1", "Design2")</code>

ResetToTimeZero

Resets a simulation to time zero.

UI Access	CleanStop when running Electronics Desktop in Batchmode.		
Parameters	Name	Type	Description
	<setupName>	String	Name of the simulation setup to be reset.
Return Value	None.		

Python Syntax	ResetToTimeZero(<setupName>)
Python Example	<code>oModule.ResetToTimeZero("Setup1")</code>

SetActiveEditor

Sets the active editor.

UI Access	N/A.		
Parameters	Name	Type	Description
	<EditorName>	String	Text of the design's notes.
Return Value	Editor object.		

Python Syntax	SetActiveEditor(<EditorName>)
Python Example	<code>oDesign.SetActiveEditor('3D Modeler')</code>

SetConductivityThreshold

Sets the thresholds for a perfect conductor and an insulator/conductor.

UI Access	Maxwell > Set Material Thresholds or right click on design instance in project tree and choose Set Material Thresholds		
Parameters	Name	Type	Description
	<PerfectConductorThreshold>	double	Perfect conductor threshold
	<Insulator/ConductorThreshold>	double	Insulator/conductor threshold
Return Value	None		

Python Syntax	SetConductivityThreshold(<PerfectConductorThreshold> <Insulator/ConductorThreshold>)
Python Example	<code>oDesign.SetConductivityThreshold(1E+030, 1.00)</code>

SetDesignSettings [Maxwell]

Sets design settings for the active design. Settings vary with solution type.

UI Access	Click Maxwell 3D/ Maxwell 2D /RMxprt > Design Settings , or right-click a design instance in the project tree and select Design Settings .		
Parameters	Name	Type	Description
	<DesignSettingsData>	Array	Input will depend on the type of design. See examples below.
	<ModelValidationSettings>	Array	Applicable to all solution types: "EntityCheckLevel:=",

			Possible values are "Strict", "Basic", "Warning Only", "None" "IgnoreUnclassifiedObjects:=" Input is a Boolean. "SkipIntersectionChecks:=" Input is a Boolean. "Design Validation Settings:=" Possible values are "Perform full validations", "Perform minimal val- idations"
Return Value	None.		

Python Syntax	SetDesignSettings (<DesignSettingsArray>,<ModelValidationSettings>)
Python Example	For 3D Transient solution type: <pre>oDesign.SetDesignSettings(["NAME:Design Settings Data", "Allow Material Override:=", True, "Perform Minimal validation:=", False, "EnabledObjects:=" , [], "PreserveTranSolnAfterDatasetEdit:=", True, "ComputeTransientInductance:=", True, "ComputeIncrementalMatrix:=", False, "PerfectConductorThreshold:=", 1E+30, "InsulatorThreshold:=" , 2500000, "UseSkewModel:=" , False, "EnableTranTranLinkWithSimplorer:=", False,</pre>

```

        "SolveFraction:=" , True, % True enables solve using circumferential fraction. Set to
        False by default.
        "FractionNumber:=" , "4", % Fraction number to generate the partial model. (Used when
        SolveFraction is True.) Set to 1 by default.
        "HalfPeriodicField:=" , True, % True enables half-periodic field. (Used when
        SolveFraction is True.) Set to False by default.
        "HalfAxialGeometry:=" , True % True cuts the original geometry in half in the axial di-
        rection. (Used when SolveFraction is True.) Set to False by default.
        "Multiplier:=" , 1, % Symmetry multiplier value used when SolveFraction is False.
        "SkipMeshChecks:=" , False

    ],
    [

        "NAME:Model Validation Settings",
        "EntityCheckLevel:=" , "Strict",
        "IgnoreUnclassifiedObjects:=" , True,
        "SkipIntersectionChecks:=" , False

    ]
)

```

Design Settings Data for 3D Magnetostatic solution type:

```

["NAME:Design Settings Data",
"Allow Material Override:=", <bool>,
"ComputeIncrementalMatrix:=", <bool>,
"PerfectConductorThreshold:=", <real>,
"InsulatorThreshold:=", <real>,
"SkipMeshChecks:=" , <bool>]

```

Design Settings Data for 3D AC Magnetic solution type:

```

["NAME:Design Settings Data",
"Allow Material Override:=", <bool>,
"PerfectConductorThreshold:=", <real>,

```



```
"InsulatorThreshold:=", <real>,  
"SymmetryMultiplier=", <real>,  
"SkipMeshChecks:=", <bool>]
```

Design Settings Data for 3D Electrostatic solution type:

```
["NAME:Design Settings Data",  
"Allow Material Override:=", <bool>,  
"PerfectConductorThreshold:=", <real>,  
"InsulatorThreshold:=", <real>,  
"Multiplier:=", <int>,  
"SkipMeshChecks:=", <bool>]
```

Design Settings Data for 3D DC Conduction solution type:

```
["NAME:Design Settings Data",  
"Allow Material Override:=", <bool>,  
"PerfectConductorThreshold:=", <real>,  
"InsulatorThreshold:=", <real>,  
"SkipMeshChecks:=", <bool>]
```

Design Settings Data for 3D Electric Transient solution type:

```
["NAME:Design Settings Data",  
"Allow Material Override:=", <bool>,  
"PreservTranSolnAfterDatasetEdit:=", <bool>,  
"SkipMeshChecks:=", <bool>]
```

Design Settings Data for 3D AC Conduction solution type:

```
["NAME:Design Settings Data",  
"Allow Material Override:=", <bool>,  
"PerfectConductorThreshold:=", <real>,  
"InsulatorThreshold:=", <real>,  
"SkipMeshChecks:=", <bool>]
```

Design Settings Data for 2D Transient solution type:

```
["NAME:Design Settings Data",
```

```
"PreserveTranSolnAfterDatasetEdit:=", <bool>,
"ComputeTransientInductance:=", <bool>,
"PerfectConductorThreshold:=", <real>,
"InsulatorThreshold:=", <real>,
"ModelDepth:=", <real>,
"UseSkewModel:=", <bool>,
"SkewModelType:=", <string>
% possible values are: "Continuous", "Step", "V-Shape", "User Defined"
"SkewPart:=", <string>
% possible values are: "Stator", "Rotor"
"SkewAngle:=", <real>,
%for "Continuous", "Step", "V-Shape" model types only
"SkewAngleUnit:=", <string>,
% for "User Defined" model type only; possible values are: "rad", "degsec", "degmin", "deg"
"NumberOfSlices:=", <int>,
% SkewSliceTable array below is for skew model type "User Defined" only.
["NAME:SkewSliceTable",
["NAME:OneSliceInfo", "SkewAngle:=", <real>, "SliceLength:=", <real>,
...
["NAME:OneSliceInfo", "SkewAngle:=", <real>, "SliceLength:=", <real>,]]
"EnableTranTranLinkWithSimplorer:=", <bool>,
"BackgroundMaterialName:=", <string>
"Multiplier:=", <int>]
```

Design Settings Data for 2D Magnetostatic solution type:

```
["NAME:Design Settings Data",
"PerfectConductorThreshold:=", <real>,
"InsulatorThreshold:=", <real>,
"ModelDepth:=", <real>,
"ComputeIncrementalMatrix:=", <bool>,
"BackgroundMaterialName:=", <string>]
```

Design Settings Data for 2D AC Magnetic, Electrostatic, DC Conduction, and AC Conduction solution types:

```
[ "NAME:Design Settings Data",
  "PerfectConductorThreshold:=", <real>,
  "InsulatorThreshold:=", <real>,
  "SymmetryMultiplier:=", <int>,
  "ModelDepth:=", <real>,
  "BackgroundMaterialName:=", <string>]
```

Example:

```
oDesign.SetDesignSettings(
[
  "NAME:Design Settings Data",
  "Perform Minimal validation:=", False,
  "EnabledObjects:=", [],
  "PreserveTranSolnAfterDatasetEdit:=", False,
  "ComputeTransientInductance:=", False,
  "ComputeIncrementalMatrix:=", False,
  "PerfectConductorThreshold:=", 1E+30,
  "InsulatorThreshold:=", 1,
  "ModelDepth:=", "1mm",
  "UseSkewModel:=", True,
  "SkewModelType:=", 0,
  "SkewAngle:=", "6deg",
  "NumberOfSlices:=", "5",
  "EnableTranTranLinkWithSimplorer:=", False,
  "BackgroundMaterialName:=", "vacuum",
  "Multiplier:=", "fractions"
],
[
  "NAME:Model Validation Settings",
  "EntityCheckLevel:=", "Strict",
  "IgnoreUnclassifiedObjects:=", True,
  "SkipIntersectionChecks:=", False
])
```



```

[
  ...
  "SkewType:=" , "V-Shape",
  "SkewPart:=" , "Rotor",
  "SkewAngle:=" , "5deg",
  "NumberOfSlices:=" , "7",
  ...
]
oDesign.SetDesignSettings(
[
  ..
  "SkewType:=" , "User Defined",
  "SkewPart:=" , "Rotor",
  "SkewAngleUnit:=" , "deg",
  "NumberOfSlices:=" , "4",
  [
    "NAME:SkewSliceTable",
    [
      "NAME:OneSliceInfo",
      "SkewAngle:=" , 0.6,
      "SliceLength:=" , 0.1
    ],
    [
      "NAME:OneSliceInfo",
      "SkewAngle:=" , -0.9,
      "SliceLength:=" , 0.2
    ],
    [
      "NAME:OneSliceInfo",
      "SkewAngle:=" , 4,
      "SliceLength:=" , 0.3
    ],
    [
      "NAME:OneSliceInfo",
      "SkewAngle:=" , 18,

```

```
    "SliceLength:=" , 0.4
  ]
],
...
]
```

SetDesignFlow

Modifies an existing machine type.

UI Access	RMxpert > Machine Type or right-click on a machine model in the Project Manager and select Machine Type on the context menu.		
Parameters	Name	Type	Description
	<MachineType>	String	The desired machine type. Possible values are: "ASSM", "BLDC", "CPSM", "DCM", "GRM", "LSSM", "PMDC", "SPIM", "SRM", "TPIM", "NSSM", "TPSM", "UNIM" representing respectively: Adjust-Speed Synchronous Machine, Brushless Permanent-Magnet DC Motor, Claw-pole Synchronous Machine, DC Machine, Generic Rotating Machine, Line-Start Permanent-Magnet Synchronous Motor, Permanent-Magnet Synchronous Generator, Single-Phase Induction Motor, Switched Reluctance Motor, Three-Phase Induction Motor", Non-Salient Synchronous Machine, Three-Phase Synchronous Machine, Universal Motor
Return Value	None.		

Python Syntax	SetDesignFlow("RMxpertSolution", <MachineType>)
Python Example	oProject = oDesktop.SetActiveProject("Project2")

```
oDesign = oProject.SetActiveDesign("RMxpertDesign1")
oDesign.SetDesignFlow("RMxpertSolution", "TPSM")
```

SetFastTransformationForLayoutComponent

Toggles whether layout components in Maxwell show outlines of objects across layers during view changes. This improves the visualization performance.

UI Access	Fast Transformation		
Parameters	Name	Type	Description
	<Layout ComponentName>	String	Name of the Layout Component.
	Boolean	String	True or False
Return Value	None.		

Python Syntax	SetFastTransformationForLayoutComponent ("<LayoutComponentName", <boolean>)
Python Example	oDesign.SetFastTransformationForLayoutComponent ("LC1_1", False)

SetObjectAttributesForLayoutComponent

Sets visualization attributes for layout components in Maxwell for a named component for layers, nets, and objects.

UI Access	Object Attributes		
Parameters	Name	Type	Description
	<Layout ComponentName>	String	Name of the Layout Component.
	<ShowDielectric>	Boolean	True to show dielectrics; false to hide dielectrics

	<i><DisplayMode></i>	Integer	0 (for layout mode), 1 (for net mode), and 2 (for object mode)
	<i><ObjectAttributesinLayerMode></i>	Array	List of layers. The three arguments for each layer represent the “Show”, “Wire Frame”, “and “Transparency” options for that layer in the “Object Attributes” dialog box.
	<i><ObjectAttributesinNetMode></i>	Array	List of nets. The three arguments for each net represent the “Show”, “Wire Frame”, “and “Transparency” options for that net in the “Object Attributes” dialog box.
	<i><ObjectAttributesinObjectMode></i>	Array	List of objects. The three arguments for each object represent the “Show”, “Wire Frame”, “and “Transparency” options for that object in the “Object Attributes” dialog box.
Return Value	None		

Python Syntax	SetObjectAttributesForLayoutComponent (<i><LayoutComponentName></i> , <i><ShowDielectric></i> , <i><DisplayMode></i> , <i><ObjectAttributesinLayerMode></i> <i><ObjectAttributesinNetMode></i> , <i><ObjectAttributesinObjectMode></i>)
Python Example	<pre> oDesign.SetObjectAttributesForLayoutComponent (["Name:=" , "LC1_1", "ShowDielectric:=" , False, "DisplayMode:=" , 2, ["NAME:ObjectAttributesInLayerMode", "GND_1_L2:=" , [True,True,0], "GND_2_L4:=" , [True,True,0], </pre>


```

"GND_3_L6:="          , [True,True,0],
"GND_4_L9:="          , [True,True,0],
"GND_5_L11:="         , [True,True,0],
"Inner_Layer_3_L10:=" , [True,True,0],
"Top_L1:="            , [True,True,100],
"VCC1_L7:="           , [True,True,0],
"VCC2_L8:="           , [True,True,0]
],
[
  "NAME:ObjectAttributesInNetMode",
  "GND:="              , [True,True,29],
  "VCC:="              , [True,True,0],
  "neg:="              , [True,True,67],
  "pos:="              , [True,True,67]
],
[
  "NAME:ObjectAttributesInObjectMode",
  "line_1:="           , [True,True,0],
  "line_2:="           , [True,True,0],
  "line_3:="           , [True,True,100],
  "line_4:="           , [True,True,100],

```

	<pre>"rect_6:=" , [True,True,0], "rect_17:=" , [True,True,0], "rect_18:=" , [True,True,0], "rect_19:=" , [True,True,0], "rect_20:=" , [True,True,0], "rect_21:=" , [True,True,0], "rect_22:=" , [True,True,0], "line_24:=" , [True,True,100], "via_0:=" , [True,True,100], "via_1:=" , [True,True,100], "via_2:=" , [True,True,100], "via_3:=" , [True,True,100]]])</pre>

SetObjectDeformation

Sets the deformation of objects when calling Ansys Electronics Desktop from Ansys Workbench. with

For Maxwell magnetostatic, transient, electrostatic, and AC Magnetic designs, sets the deformation of objects when calling Ansys Electronics Desktop from Ansys Workbench. (Can also be used for setting deformation of objects in Maxwell 2D/3D magnetostatic and transient standalone designs.)

UI Access	N/A		
Parameters	Name	Type	Description
	<ObjectArray>	Array	Array containing the following parameters: • "EnabledStressObjectsName:= "The names of objects that are enabled for deformation. • Optionally, for Maxwell 2D magnetostatic and transient designs: " Enabled2DXYStress<double array>:=" Every 3 numbers are a group of initial stress data (real with unit) for one object in the EnabledStressObjectsName list. (Applies only to 2D magnetostatic and transient XY designs.) " Enabled2DRZStress<double array>:=" Every 3 numbers are a group of initial stress data (real with unit) for one object in the EnabledStressObjectsName list. (Applies only to 2D magnetostatic and transient RZ designs.) • Optionally, for Maxwell 3D magnetostatic and transient designs: "Enabled3DStress<double array>:=" Every 6 real numbers with unit are a group of initial stress data for one object in the EnabledStressObjectsName list.
Return Value	None.		

Python Syntax	SetObjectDeformation(<ObjectArray>)

Python Example	<code>oDesign.SetObjectDeformation(["EnabledStressObjectsName:=" , ["Cylinder1", "Sphere1"]])</code>

For Maxwell 3D Magnetostatic and Transient designs with initial stress

Python Syntax	SetObjectDeformation(<ObjectArray>)
Python Example	<pre>oDesign.SetObjectDeformation(["EnabledStressObjectsName:=" , ["Cylinder1", "Sphere1"], "Enabled3DStressXXObjects:=", ["12.35psi","30.64psi"], "Enabled3DStressYYObjects:=", ["5.72psi","16.93psi"], "Enabled3DStressZZObjects:=", ["14.75psi","12.22psi"], "Enabled3DStressXYObjects:=", ["64.3psi","28.13psi"], "Enabled3DStressYZObjects:=", ["19.43psi","58.67psi"], "Enabled3DStressXZObjects:=", ["27.63psi","40.31psi"]])</pre>

For Maxwell 2D Magnetostatic and Transient XY designs with initial stress

Python Syntax	SetObjectDeformation(<ObjectArray>)
Python Example	<pre>oDesign.SetObjectDeformation(["EnabledStressObjectsName:=" , ["Cylinder1", "Sphere1"], "Enabled2DXYStressXXObjects:=", ["12.35psi","30.64psi"], "Enabled2DXYStressYYObjects:=", ["5.72psi","16.93psi"],</pre>

	<pre>"Enabled2DXStressXYObjects:=", ["64.3psi","28.13psi"],])</pre>

For Maxwell 2D Magnetostatic and Transient RZ designs with initial stress

Python Syntax	SetObjectDeformation(<ObjectArray>)
Python Example	<pre>oDesign.SetObjectDeformation(["EnabledStressObjectsName:=", ["Cylinder1", "Sphere1"], "Enabled2DRZStressRRObjets:=", ["12.35psi","30.64psi"], "Enabled2DRZStressZZObjects:=", ["5.72psi","16.93psi"], "Enabled2DRZStressRZObjects:=", ["64.3psi","28.13psi"],])</pre>

SetObjectTemperature

Sets the temperature of objects.

UI Access	Maxwell > Set Object Temperature		
Parameters	Name	Type	Description
	<TemperatureSettings>	Array	<pre>["NAME": <SettingName>, "IncludeTemperatureDependence:=", <boolean, True to include temperature dependence. False to not include.>, "EnableFeedback:=", <boolean, True to enable feed- back. False to not enable feedback.>, "Temperatures:=", <Temperatures>]</pre>

	<i><Temperatures></i>	Array	Objects and temperatures.
Return Value	None.		

Python Syntax	<code>oDesign.SetObjectTemperature(<TemperatureSettings>)</code>
Python Example	<pre>oDesign.SetObjectTemperature(["NAME:TemperatureSettings", "IncludeTemperatureDependence:=", True, "EnableFeedback:=" , False, "Temperatures:=" , ["RegularPolyhedron1", "22cel", "Polygon1", "22cel"])</pre>

SetShowAllLayoutComponents

This command will show/hide all layout components in a Maxwell design.

UI Access	Edit Layout... View > Visibility > Show All or Hide All		
Parameters	Name	Type	Description
	Boolean	String	True or False
Return Value	None		

Python Syntax	<code>SetShowAllLayoutComponents (<boolean>)</code>
Python Example	<pre>oDesign = oProject.GetActiveDesign() oDesign.SetShowAllComponentLayouts (True)</pre>

SetShowLayoutForLayoutComponent

Layout visualization, rather than bounding box, for a Layout Component in a Maxwell design.

UI Access	Edit Layout...		
Parameters	Name	Type	Description
	<Layout ComponentName>	String	Name of the Layout Component.
	Boolean	String	True or False
Return Value	None.		

Python Syntax	<code>SetShowLayoutForLayoutComponent ("<LayoutComponentName", <boolean>)</code>
Python Example	<code>oDesign.SetShowLayoutForLayoutComponent ("LC1_1", True)</code>

SetSolutionType (Maxwell)

Sets the solution type for the design.

UI Access	Maxwell3D or Maxwell2D > Solution Type , or right-click a design instance in the Project Manager tree and

	select Solution Type .		
Parameters	Name	Type	Description
	<SolutionType>	String	Possible values for 3D designs are "Magnetostatic", "AC Magnetic", "AC Magnetic with DC", "AC Magnetic APhi", "Transient", "Transient APhi", "Electrostatic", "DC Conduction", "Electric DC Conduction", "Electric Transient", "AC Conduction" Possible values for 2D designs are "Magnetostatic", "AC Magnetic", "Transient", "Electrostatic", "DC Conduction", "AC Conduction"
	<GeomMode>	String	Possible values for 2D geometry mode: "XY", "about Z"
Return Value	None		

Python Syntax	SetSolutionType(<SolutionType>, <Geomode>)
Python Example	oDesign.SetSolutionType("AC Magnetic APhi")

SimulateLink

Runs linked simulation.

UI Access	N/A		
Parameters	Name	Type	Description
	<setup>	String	Setup name.
	<linkedDesign>	Object	Optional. Linked design.
	<linkType>	String	Optional. Type of linked design.

	<code><msg></code>	String	Optional. Messages to add.
Return Value	None.		

Python Syntax	<code>SimulateLink(<setup>, <linkedDesign>, <linkType>, <msg>)</code>		
Python Example	<code>oDesign.SimulateLink("1_4GHz")</code>		

Solve

Performs one or more simulation. The next script command will not be executed until the simulation(s) are complete.

UI Access	Select solution setup(s). Right-click and select Analyze .		
Parameters	Name	Type	Description
	<code><SimulationNames></code>	Array	Array containing string simulation names.
Return Value	Integer: • 0 – Simulation(s) completed. • 1 – Simulation error. • -1 – Command execution error.		

Python Syntax	<code>Solve <SimulationNames></code>		
Python Example	<code>oDesign.Solve(['Setup1', 'Setup2', 'Setup3'])</code>		

RunToolkit

Runs a Python toolkit script, applying it to the active design. The toolkit script itself may have prerequisites, such as sweeps defined, or specific model characteristics, such as port definitions.

Important:

Full scripting for cable modeling is not supported and no arguments are allowed in the script. The **RunToolkit** command will not automatically create a cable bundle, but will simply open the **Cable Modeling** dialog box. You will have to manually input your parameters.

UI Access	[Maxwell] > Toolkit > [Script Name].		
Parameters	Name	Type	Description
	<LibraryType>	String	Name of the library in which the script is located.
	<ToolkitName>	String	Name of the Python script.
	<ToolkitArg>	Array	Structured array containing arguments required by the toolkit script.
Return Value	User-defined solution(s) and/or output(s).		

Python Syntax	RunToolkit(<LibraryType>, <ToolkitName>, <ToolkitArg>)
Python Example	<pre>oDesign.RunToolkit("SysLib", "Cable Modeling/Automotive Cable Bundle", []) oDesign.RunToolkit("SysLib", "Cable Modeling/Oil-Gas Cable Bundle", [])</pre>

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>

ValidateDesign

Returns whether a design is valid.

UI Access	Maxwell > Validation Check.
Parameters	None.
Return Value	Integer: • 1 – Validation passed. • 0 – Validation failed.

Python Syntax	ValidateDesign()

Python Example	<pre>oDesign.ValidateDesign()</pre>

10 - 3D Modeler Editor Script Commands

3D Modeler commands should be executed by the "3D Modeler" editor:

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

```
oEditor.<CommandName>
```

Conventions Used in this Chapter:

<AttributesArray>

<AttributesArray> takes the following structure:

```
Array("NAME:Attributes",  
      "Name:=", <string>,  
      "Flags:=", <string>,  
      "Color:=", <string>,  
      "Transparency:=", <value>,  
      "PartCoordinateSystem:=", <string>,  
      "UDMId:=", <string>,  
      "MaterialValue:=", <string>,  
      "SurfaceMaterialValue:=", <string>,  
      "Solveinside:=", <boolean>,  
      "ShellElement:=", <boolean>,  
      "ShellElementThickness:=", <string>,  
      "IsMaterialEditable:=", <boolean>,
```

```
"UseMaterialAppearance:=", <boolean>,  
"IsLightweight:=", <boolean>)
```

Where:

- **Flags** – Takes a string containing "NonModel" and/or "Wireframe", separated by the # character. For example, "NonModel#Wireframe".
- **Color** – Takes a string containing an RGB triple, formatted as "(*<RGB>*)". For example, "(255 255 255)".
- **Transparency** – Takes a value between 0 and 1.
- **PartCoordinateSystem** – Orientation of the primitive. The name of one of the defined coordinate systems should be specified.
- **UDMId** – Takes a string containing an ID.
- **MaterialValue** – Takes a string of the material name.
- **SurfaceMaterialValue** – Takes a string of the surface material name.
- **Solveinside** – Takes a boolean value.
- **ShellElement** – Takes a boolean value specifying whether or not a shell element is present.
- **ShellElementThickness** – Takes a string containing the shell element thickness. If element is not present, pass empty string.
- **IsMaterialEditable** – Takes a boolean value.
- **IsLightweight** – Takes a boolean value.

<SelectionsArray>

<SelectionsArray> typically takes the following structure:

```
Array("NAME:Selections",  
"Selections:=", <string>)
```

Where:

- **Selections** – Takes a comma-separated list of parts on which to perform the action. For example, "Rect1, Rect2, Rect3".

In some cases, <SelectionsArray> takes additional parameters:

```
Array("NAME:Selections",
    "AllowRegionDependentPartSelectionForPMLCreation:=", <boolean>,
    "AllowRegionSelectionForPMLCreation:=", <boolean>,
    "Selections:=", <string>,
    "NewPartsModelFlag:=" , <string>,
    "UseCurrentCS:=", <boolean>)
```

Where:

- **AllowRegionDependentPartSelectionForPMLCreation** – Takes a boolean value. See individual script for whether this parameter is required.
- **AllowRegionSelectionForPMLCreation** – Takes a boolean value. See individual script for whether this parameter is required.
- **Selections** – Takes a comma-separated list of parts on which to perform the action. For example, "Rect1, Rect2, Rect3".
- **NewPartsModelFlag** – Takes either string "Model" or string "Nonmodel". See individual script for whether this parameter is required.
- **UseCurrentCS** – Takes a boolean value. See individual script for whether this parameter is required. Use [GetActiveCoordinateSystem](#) to determine the current CS.

Note:

Selections is the only parameter required in *all* 3D Modeler Editor scripts. See individual scripts for additional required parameters.

Organization

3D Modeler editor scripts are organized into the following categories:

[Draw Menu Commands](#)

[Edit Menu Commands](#)

[Modeler Menu Commands](#)

[Cable Modeling Commands](#)

[Other oEditor Commands](#)

Draw Menu Commands

[Create3DComponent](#)

[CreateBondwire](#)

[CreateBox](#)

[CreateCircle](#)

[CreateCone](#)

[CreateCutplane](#)

[CreateCylinder](#)

[CreateEllipse](#)

[CreateEquationCurve](#)

[CreateEquationSurface](#)

[CreateHelix](#)

[CreatePoint](#)

[CreatePolyline](#)

[CreateRectangle](#)

[CreateRegion](#)

[CreateRegularPolygon](#)

[CreateRegularPolyhedron](#)

[CreateSphere](#)

[CreateSpiral](#)

[CreateSubregion](#)

[CreateTorus](#)

[CreateUserDefinedModel](#)

[CreateUserDefinedPart](#)

[Edit3DComponent](#)

[EditPolyline](#)

[Get3DComponentDefinitionNames](#)

[Get3DComponentInstanceNames](#)

[Get3DComponentMaterialNames](#)

[Get3DComponentMaterialProperties](#)

[Get3DComponentParameters](#)

[Insert3DComponent](#)

[InsertPolylineSegment](#)

[SweepAlongPath](#)

[SweepAlongVector](#)

[SweepAroundAxis](#)

[SweepFacesAlongNormal](#)

[SweepFacesAlongNormalWithAttributes](#)

[UpdateComponentDefinition](#)

Create3DComponent

Creates a 3D component.

UI Access	Draw > 3D Component Library > Create 3D Component.		
Parameters	Name	Type	Description
	<CreateData>	Array	Structured array containing geometry data: (Array["NAME:CreateData", "ComponentName:=", "<string>", "Company:=", "<string>", "Company URL:=", "<string>", "Model Number:=", "<string>", "Help URL:=" , "<string>", "Version:=", "<string>", "Notes:=", "<string>", "IconType:=", "<string>", "Owner:=", "<string>", "Email:=", "<string>", "Date:=", "<string>", #Component creation date "HasLabel:=", <boolean>,

			<pre> "IsEncrypted:=", <boolean>, "AllowEdit:=", <boolean>, "SecurityMessage:=", " <string>", "Password:=", " <string>", "EditPassword:=", " <string>", "PasswordType:=", " <string>", "HideContents:=", <boolean>, "ReplaceNames:=", <boolean>, #for objects and materials "ComponentOutline:=", " <string>,"#"None" or "Bounding Box" "IncludedParts:=", [Array of partnames], "HiddenParts:=", [Array of hidden parts], "IncludedCS:=", [Array of included CS], "DefaultHandle:=", " <reference CS>" "IncludedParameters:=", [Array of included parameters], "IncludedDependentParameters:=", [Array of dependent parameters], "ParameterDescription:=", <array> "IsLicensed:=", <boolean> "LicensingDllName:=", " <string>" "VendorComponentIdentifier:=", " <string>" "PublicKeyFile:=", " <string>" </pre>

			]
	<DesignData>	Array	Structured array containing design data: Array["NAME:DesignData", "Boundaries:=", <array>, "Excitations:=", <array>, "MeshOperations:=", <array>]
	<FileName>	String	Full file path to 3D component.
	<ImageFile>	Array	Optional. Structured array containing file path of 3D component image: Array["NAME:ImageFile", "ImageFile:=", <string>])
Return Value	None.		

Python Syntax	Create3DComponent(<CreateData>, <DesignData>, <FileName>, <ImageFile>)
Python Example	<pre>oEditor.Create3DComponent (["NAME:CreateData", "ComponentName:=", "Connector", "Company:=", "", "Company URL:=", "", "Version:=", "1.0", "Notes:=", "", "Owner:=", "", "IconType:=", "", "Email:=", "",</pre>

```

    "Date:=", "11:41:01 AM Aug 28, 2024",
    "HasLabel:=", false,
    "IsEncrypted:=", , False,
    "AllowEdit:=", , False,
    "SecurityMessage:=", , "",
    "Password:=", , "",
    "EditPassword:=", , "",
    "PasswordType:=", , "UnknownPassword",
    "HideContents:=", , True,
    "ReplaceNames:=", , True,
    "ComponentOutline:=", , "None",
    "IncludedParts:=", ["Box1", "Cylinder1", "Cone1"],
    "IncludedCS:=", ["RelativeCS1"],
    "DefaultHandle:=", "Global",
    "IncludedParameters:=", ["htcone", "lr", "htcyl", "zs", "radcyl", "xs", "$rp",
    "$con"],
    "IncludedDependentParameters:=", [],
    "ParameterDescription:=", [],
    "IsLicensed:=", , False,
    "LicensingDllName:=", , "",
    "VendorComponentIdentifier:=", "",
    "PublicKeyFile:=", , ""
  ],
  [
    "NAME:DesignData",
    "Boundaries:=", ["PerfE1", "FiniteCond1"],
    "Excitations:=", ["1"],
    "MeshOperations:=", []
  ],
  "C:/tmp/Connector.a3dcomp",
  [
    "NAME:ImageFile", "ImageFile:=", ""
  ]
)

```

Example with an External Circuit:

```
oDesign = oProject.SetActiveDesign("Maxwell3DDesign1")
oEditor = oDesign.SetActiveEditor("3D Modeler")
oEditor.Create3DComponent(
[
    "NAME:CreateData",
    "ComponentName:=" , "Maxwell3DDesign1",
    "Company:=" , "",
    "Company URL:=" , "",
    "Model Number:=" , "",
    "Help URL:=" , "",
    "Version:=" , "1.0",
    "Notes:=" , "",
    "IconType:=" , "",
    "Owner:=" , "",
    "Email:=" , "",
    "Date:=" , "3:36:11 PM Feb 14, 2025",
    "HasLabel:=" , False,
    "IsEncrypted:=" , False,
    "AllowEdit:=" , False,
    "SecurityMessage:=" , "",
    "Password:=" , "",
    "EditPassword:=" , "",
    "PasswordType:=" , "UnknownPassword",
    "HideContents:=" , True,
    "ReplaceNames:=" , True,
    "ComponentOutline:=" , "None",
    "IncludedParts:=" , ["Box1"],
    "HiddenParts:=" , [],
    "IncludedCS:=" , [],
    "DefaultHandle:=" , "Global",
    "IncludedParameters:=" , [],
    "IncludedDependentParameters:=" , [],
    "ParameterDescription:=" , [],
```

```
"IsLicensed:=" , False,
"LicensingDllName:=" , "",
"VendorComponentIdentifier:=", "",
"PublicKeyFile:=" , ""
],
[
  "NAME:DesignData",
  "Excitations:=" , ["CoilTerminal1","CoilTerminal2","Winding1","External Circuit"]
],
"F:/Designs/ComponentExternalCircuit/test.a3dcomp",
[
  "NAME:ImageFile",
  "ImageFile:=" , ""
])
```

CreateBondwire

Creates a bondwire.

UI Access	Draw > Bondwire.		
Parameters	Name	Type	Description
	<Parameters>	Array	Structured array. Array ("NAME:BondwireParameters", "WireType:=", <string("JEDEC_4Points", "JEDEC_5Points", or "LOW")>, "WireDiameter:=", <string>, "NumSides:=", <value>, "XPadPos:=", <value>, "YPadPos:=", <value>, "ZPadPos:=", <value>,

			<pre>"XDir:=", <value>, "YDir:=", <value>, "ZDir:=", <value>, "Distance:=", <value>, "h1:=", <value>, "h2:=", <value>, "alpha:=", <value>, "beta:=", <value>, "WhichAxis:=", <string("X","Y", or "Z")>, "ReverseDirection:=", <boolean></pre>
	<AttributesArray>	Array	Structured array. See: AttributesArray .
Return Value	None.		

Python Syntax	CreateBondwire(<Parameters>, <Attributes>)		
Python Example	<pre>oEditor.CreateBondwire(["NAME:BondwireParameters", "WireType:=" , "JEDEC_4Points", "WireDiameter:=" , "0.025mm", "NumSides:=" , "6", "XPadPos:=" , "1.6mm",</pre>		


```

"YPadPos:="          , "-0.2mm",
"ZPadPos:="          , "0mm",
"XDir:="              , "-2.2mm",
"YDir:="              , "-1.4mm",
"ZDir:="              , "0mm",
"Distance:="          , "2.60768096208106mm",
"h1:="                , "0.2mm",
"h2:="                , "0mm",
"alpha:="             , "80deg",
"beta:="              , "0",
"WhichAxis:="         , "Z",
"ReverseDirection:="  , True
],
["NAME:Attributes",
  "Name:="             , "Bondwire1",
  "Flags:="            , "",
  "Color:="            , "(143 175 143)",
  "Transparency:="     , 0,
  "PartCoordinateSystem:=", "Global",
  "UDMId:="            , "",
  "MaterialValue:="    , "\"vacuum\"",

```

	<pre>"SurfaceMaterialValue:=", "\"\"\"", " SolveInside:=" , True, "ShellElement:=" , False, "ShellElementThickness:=", "0mm", "IsMaterialEditable:=" , True, "UseMaterialAppearance:=", False, "IsLightweight:=" , False])</pre>

CreateBox

Creates a box.

UI Access	Draw > Box.		
Parameters	Name	Type	Description
	<Parameters>	Array	Structured array. Array ("NAME:BoxParameters", "XPosition:=", <string>, "YPosition:=", <string>, "ZPosition:=", <string>, "XSize:=" , <string>, "YSize:=" , <string>,

			"ZSize:=" , <string>)
	<AttributesArray>	Array	Structured array. See: AttributesArray .
Return Value	None.		

Python Syntax	CreateBox(<Parameters>, <Attributes>)
Python Example	<pre>oEditor.CreateBox(["NAME:BoxParameters", "XPosition:=" , "0.5mm", "YPosition:=" , "-6.5mm", "ZPosition:=" , "0mm", "XSize:=" , "2mm", "YSize:=" , "1.5mm", "ZSize:=" , "1.5mm"], ["NAME:Attributes", "Name:=" , "Box3", "Flags:=" , "", "Color:=" , "(143 175 143)", "Transparency:=" , 0, "PartCoordinateSystem:=" , "Global", "UDMId:=" , "",</pre>

	<pre>"MaterialValue:=" , "\"copper\"", "SurfaceMaterialValue:=" , "\"\"", "SolveInside:=" , False, "ShellElement:=" , False, "ShellElementThickness:=", "0mm", "IsMaterialEditable:=" , True, "UseMaterialAppearance:=", False, "IsLightweight:=" , False])</pre>

CreateCircle

Creates a circle.

UI Access	Draw > Circle.		
Parameters	Name	Type	Description
	<Parameters>	Array	Structured array. Array("NAME:CircleParameters", "IsCovered:=", <boolean>, "XCenter:=", <value>, "YCenter:=", <value>, "ZCenter:=", <value>,

			"Radius:=", <value>, "WhichAxis:=", <string> "NumSegments:=", <string containing integer>)
	<AttributesArray>	Array	Structured array. See: AttributesArray .
Return Value	None.		

Python Syntax	CreateCircle(<Parameters>, <Attributes>)
Python Example	<pre>oEditor.CreateCircle(["NAME:CircleParameters", "IsCovered:=" , True, "XCenter:=" , "5.5mm", "YCenter:=" , "-3mm", "ZCenter:=" , "0mm", "Radius:=" , "0.707106781186548mm", "WhichAxis:=" , "Z", "NumSegments:=" , "0"], ["NAME:Attributes", "Name:=" , "Circle1", "Flags:=" , "", "Color:=" , "(143 175 143)",</pre>

	<pre>"Transparency:=" , 0, "PartCoordinateSystem:=", "Global", "UDMId:=" , "", "MaterialValue:=" , "\"copper\"", "SurfaceMaterialValue:=", "\"\"", "SolveInside:=" , False, "ShellElement:=" , False, "ShellElementThickness:=", "0mm", "IsMaterialEditable:=" , True, "UseMaterialAppearance:=", False, "IsLightweight:=" , False l)</pre>

CreateCone

Creates a cone.

UI Access	Draw > Cone.		
Parameters	Name	Type	Description
	<Parameters>	Array	Structured array. Array("NAME:ConeParameters", "XCenter:=", <string>,

			"YCenter:=", <string>, "ZCenter:=", <string>, "WhichAxis:=", <string>, "Height:=", <string>, "BottomRadius:=", <string>, "TopRadius:=", <string>)
	<AttributesArray>	Array	Structured array. See: AttributesArray .
Return Value	None.		

Python Syntax	CreateCone(<Parameters>, <Attributes>)
Python Example	<pre>oEditor.CreateCone(["NAME:ConeParameters", "XCenter:=" , "3mm", "YCenter:=" , "-4.5mm", "ZCenter:=" , "0mm", "WhichAxis:=" , "Z", "Height:=" , "2.5mm", "BottomRadius:=" , "2.82842712474619mm", "TopRadius:=" , "2.23606797749979mm"], ["NAME:Attributes",</pre>

	<pre>"Name:=" , "Cone1", "Flags:=" , "", "Color:=" , "(143 175 143)", "Transparency:=" , 0, "PartCoordinateSystem:=", "Global", "UDMId:=" , "", "MaterialValue:=" , "\"copper\"", "SurfaceMaterialValue:=", "\"\"", "SolveInside:=" , False, "ShellElement:=" , False, "ShellElementThickness:=", "0mm", "IsMaterialEditable:=" , True, "UseMaterialAppearance:=", False, "IsLightweight:=" , False]</pre>

CreateCutplane

Creates a cutplane.

UI Access	Draw > Plane.

Parameters	Name	Type	Description
	<Parameters>	Array	Structured array. ["NAME:PlaneParameters", "PlaneBaseX=", <string>, "PlaneBaseY=", <string>, "PlaneBaseZ=", <string>, "PlaneNormalX=", <string>, "PlaneNormalY=", <string>, "PlaneNormalZ=", <string>]
	<AttributesArray>	Array	See: AttributesArray . CreateCutplane only takes the Name and Color attributes.
Return Value	None.		

Python Syntax	CreateCutplane(<Parameters>, <Attributes>)
Python Example	<pre>oEditor.CreateCutplane(["NAME:PlaneParameters", "PlaneBaseX=" , "-0.6mm", "PlaneBaseY=" , "-0.8mm", "PlaneBaseZ=" , "0mm", "PlaneNormalX=" , "1.2mm", "PlaneNormalY=" , "0.2mm", "PlaneNormalZ=" , "0mm"])</pre>

	<pre>], ["NAME:Attributes", "Name:=" , "Plane1", "Color:=" , "(143 175 143)"])</pre>

CreateCylinder

Creates a cylinder.

UI Access	Draw > Cylinder.		
Parameters	Name	Type	Description
	<Parameters>	Array	Structured array. Array("NAME:CylinderParameters", "XCenter:=", <string>, "YCenter:=", <string>, "ZCenter:=", <string>, "Radius:=", <string>, "Height:=", <string>, "WhichAxis:=", <string>, "NumSides:=", <string containing integer>)
	<AttributesArray>	Array	Structured array. See: AttributesArray .

Return Value	None.

Python Syntax	CreateCylinder(<Parameters>, <Attributes>)
Python Example	<pre>oEditor.CreateCylinder(["NAME:CylinderParameters", "XCenter:=" , "6mm", "YCenter:=" , "-4.5mm", "ZCenter:=" , "0mm", "Radius:=" , "0.5mm", "Height:=" , "4.5mm", "WhichAxis:=" , "Z", "NumSides:=" , "0"], ["NAME:Attributes", "Name:=" , "Cylinder1", "Flags:=" , "", "Color:=" , "(143 175 143)", "Transparency:=" , 0, "PartCoordinateSystem:=" , "Global", "UDMId:=" , "", "MaterialValue:=" , "\"copper\"",</pre>

	<pre>"SurfaceMaterialValue:=", "\"\\"", "SolveInside:=" , False, "ShellElement:=" , False, "ShellElementThickness:=", "0mm", "IsMaterialEditable:=" , True, "UseMaterialAppearance:=", False, "IsLightweight:=" , False])</pre>

CreateEllipse

Creates an ellipse.

UI Access	Draw > Ellipse.		
Parameters	Name	Type	Description
	<Parameters>	Array	Structured array. Array("NAME:EllipseParameters", "IsCovered:=", <string>, "XCenter:=", <string>, "YCenter:=", <string>, "ZCenter:=", <string>, "MajRadius:=", <string>,

			"Ratio:=", <string>, "WhichAxis:=", <string>, "NumSegments:=", <string>)
	<AttributesArray>	Array	Structured array. See: AttributesArray .
Return Value	None.		

Python Syntax	CreateEllipse(<Parameters>, <Attributes>)
Python Example	<pre>oEditor.CreateEllipse(["NAME:EllipseParameters", "IsCovered:=" , True, "XCenter:=" , "0.6mm", "YCenter:=" , "-0.6mm", "ZCenter:=" , "0mm", "MajRadius:=" , "0.2mm", "Ratio:=" , "7", "WhichAxis:=" , "Z", "NumSegments:=" , "0"], ["NAME:Attributes", "Name:=" , "Ellipse1", "Flags:=" , "",</pre>

	<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])</pre>

CreateEquationCurve

Creates an equation-based curve.

UI Access	Draw > Equation-Based Curve.		
Parameters	Name	Type	Description
	<Parameters>	Array	Structured array. Array("NAME:EquationBasedCurveParameters",

			"XtFunction:=", <string>, "YtFunction:=", <string>, "ZtFunction:=", <string>, "tStart:=", <string>, "tEnd:=", <string>, "NumOfPointsOnCurve:=", <string>, "Version:=", <integer>), <polylineArray(optional)>
	<AttributesArray>	Array	Structured array. See: AttributesArray .
Return Value	None.		

Python Syntax	CreateEquationCurve(<Parameters>, <Attributes>)
Python Example	<pre>oEditor.CreateEquationCurve(["NAME:EquationBasedCurveParameters", "XtFunction:=" , "1", "YtFunction:=" , "3", "ZtFunction:=" , "32", "tStart:=" , "1", "tEnd:=" , "3", "NumOfPointsOnCurve:=" , "0", "Version:=" , 1,</pre>

```
[ "NAME:PolylineXSection",  
  "XSectionType:="          , "None",  
  "XSectionOrient:="        , "Auto",  
  "XSectionWidth:="         , "0",  
  "XSectionTopWidth:="      , "0",  
  "XSectionHeight:="        , "0",  
  "XSectionNumSegments:="   , "0",  
  "XSectionBendType:="      , "Corner"  
]  
],  
[ "NAME:Attributes",  
  "Name:="                  , "EquationCurve1",  
  "Flags:="                 , "",  
  "Color:="                 , "(143 175 143)",  
  "Transparency:="          , 0,  
  "PartCoordinateSystem:="  , "Global",  
  "UDMId:="                 , "",  
  "MaterialValue:="         , "\"copper\"",  
  "SurfaceMaterialValue:="  , "\"\"",  
  "SolveInside:="           , False,
```


	<pre>"ShellElement:=" , False, "ShellElementThickness:=", "0mm", "IsMaterialEditable:=" , True, "UseMaterialAppearance:=", False, "IsLightweight:=" , False])</pre>

CreateEquationSurface

Creates an equation-based surface.

UI Access	Draw > Equation-Based Surface.		
Parameters	Name	Type	Description
	<Parameters>	Array	Structured array. Array("NAME:EquationBasedSurfaceParameters", "XuvFunction:=" , <string equation containing Function, Operators and/or quantities _u, _v, or PI>, "YuvFunction:=" , <string equation containing Function, Operators and/or quantities _u, _v, or PI>, "ZuvFunction:=" , <string equation containing Function, Operators and/or quantities _u, _v, or PI>, "uStart:=" , <string>, "uEnd:=" , <string>, "vStart:=" , <string>, "vEnd:=" , <string>,

			"Version:=" , <integer>)
	<AttributesArray>	Array	Structured array. See: AttributesArray .
Return Value	None.		

Python Syntax	CreateEquationSurface(<Parameters>, <Attributes>)
Python Example	<pre>oEditor.CreateEquationSurface(["NAME:EquationBasedSurfaceParameters", "XuvFunction:=" , "_u", "YuvFunction:=" , "_v", "ZuvFunction:=" , "sin(_u)+cos(_v)", "uStart:=" , "1", "uEnd:=" , "10", "vStart:=" , "1", "vEnd:=" , "10", "Version:=" , 1], ["NAME:Attributes", "Name:=" , "EquationSurface1", "Flags:=" , "", "Color:=" , "(143 175 143)",</pre>

	<pre>"Transparency:=" , 0, "PartCoordinateSystem:=", "Global", "UDMId:=" , "", "MaterialValue:=" , "\"copper\"", "SurfaceMaterialValue:=", "\"\"", "SolveInside:=" , False, "ShellElement:=" , False, "ShellElementThickness:=", "0mm", "IsMaterialEditable:=" , True, "UseMaterialAppearance:=", False, "IsLightweight:=" , False l)</pre>

CreateHelix

Creates a helix based on a sweep of specified objects.

UI Access	Draw > Helix.		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: SelectionsArray .
	<Parameters>	Array	Structured array.
			<pre>Array("NAME:HelixParameters", "XCenter:=" , <string>, "YCenter:=" , <string>,</pre>

			<code>"ZCenter:=" , <string>, "XStartDir:=" , <string>, "YStartDir:=" , <string>, "ZStartDir:=" , <string>, "NumThread:=" , <string>, "RightHand:=" , <boolean>, "RadiusIncrement:=" , <string>, "Thread:=" , <string>)</code>
Return Value	None.		

Python Syntax	CreateHelix(<SelectionsArray>, <Parameters>)		
Python Example	<pre>oEditor.CreateHelix(["NAME:Selections", "Selections:=" , "EquationSurface2", "NewPartsModelFlag:=" , "Model"], ["NAME:HelixParameters", "XCenter:=" , "10000mm", "YCenter:=" , "40000mm", "ZCenter:=" , "0mm",</pre>		

	<pre> "XStartDir:=" , "0mm", "YStartDir:=" , "10000mm", "ZStartDir:=" , "0mm", "NumThread:=" , "1", "RightHand:=" , True, "RadiusIncrement:=" , "0mm", "Thread:=" , "1mm" l) </pre>

CreatePoint

Creates a point.

UI Access	Draw > Point.		
Parameters	Name	Type	Description
	<Parameters>	Array	Structured array. Array("NAME:PointParameters", "PointX:=", <value>, "PointY:=", <value>, "PointZ:=", <value>)
	<AttributesArray>	Array	Structured array. See: AttributesArray . CreatePoint takes only the Name and Color attributes.
Return Value	None.		

Python Syntax	CreatePoint(<Parameters>, <Attributes>)
Python Example	<pre>oEditor.CreatePoint(["NAME:PointParameters", "PointX:=" , "0.2mm", "PointY:=" , "-0.2mm", "PointZ:=" , "0mm"], ["NAME:Attributes", "Name:=" , "Point1", "Color:=" , "(143 175 143) "])</pre>

CreateUserDefinedPart

Creates a user-defined part.

UI Access	Draw > User-Defined Primitive > [Part]		
Parameters	Name	Type	Description

	<i><Parameters></i>	Array	Structured array. <pre>["NAME:UserDefinedPrimitiveParameters", "DllName:=" , <string>, "Version:=" , <string>, "NoOfParameters:=" , <integer>, "Library:=" , <string>, <paramVectorArray>]</pre>
	<i><paramVectorArray></i>	Array	Structured array containing arrays for each pair: <pre>["NAME:ParamVector", <pair>, <pair>, <pair>, ...] <pair>: ["NAME:Pair", "Name:=" , <string>, "Value:=" , <string>]</pre>
	<i><Attributes></i>	Array	Structured array. See: AttributesArray .
Return Value	None.		

Python Syntax	CreateUserDefinedPart(<i><Parameters></i> , <i><paramVectorArray></i> , <i><Attributes></i>)
Python Example	<pre>oEditor.CreateUserDefinedPart(["NAME:UserDefinedPrimitiveParameters",</pre>

```
"DllName:=" , "RMxpert/LapCoil.dll",  
"Version:=" , "16.0",  
"NoOfParameters:=" , 22,  
"Library:=" , "syslib",  
  ["NAME:ParamVector",  
    ["NAME:Pair",  
      "Name:=" , "DiaGap",  
      "Value:=" , "100mm"  
    ],  
    ["NAME:Pair",  
      "Name:=" , "DiaYoke",  
      "Value:=" , "20mm"  
    ],  
    ["NAME:Pair",  
      "Name:=" , "Length",  
      "Value:=" , "100mm"  
    ],  
    ["NAME:Pair",  
      "Name:=" , "Skew",  
      "Value:=" , "0deg"
```



```
],  
["NAME:Pair",  
"Name:=" , "Slots",  
"Value:=" , "18"  
],  
["NAME:Pair",  
"Name:=" , "SlotType",  
"Value:=" , "1"  
],  
["NAME:Pair",  
"Name:=" , "Hs0",  
"Value:=" , "1mm"  
],  
["NAME:Pair",  
"Name:=" , "Hs1",  
"Value:=" , "1mm"  
],  
["NAME:Pair",  
"Name:=" , "Hs2",  
"Value:=" , "10mm"  
],
```

```
[ "NAME:Pair",  
  "Name:=" , "Bs0",  
  "Value:=" , "2.5mm"  
],  
[ "NAME:Pair",  
  "Name:=" , "Bs1",  
  "Value:=" , "8mm"  
],  
[ "NAME:Pair",  
  "Name:=" , "Bs2",  
  "Value:=" , "5mm"  
],  
[ "NAME:Pair",  
  "Name:=" , "Rs",  
  "Value:=" , "0mm"  
],  
[ "NAME:Pair",  
  "Name:=" , "FilletType",  
  "Value:=" , "0"  
],
```

```
[ "NAME:Pair",  
  "Name:=" , "Layers",  
  "Value:=" , "2"  
],  
[ "NAME:Pair",  
  "Name:=" , "CoilPitch",  
  "Value:=" , "4"  
],  
[ "NAME:Pair",  
  "Name:=" , "EndExt",  
  "Value:=" , "5mm"  
],  
[ "NAME:Pair",  
  "Name:=" , "SpanExt",  
  "Value:=" , "25mm"  
],  
[ "NAME:Pair",  
  "Name:=" , "BendAngle",  
  "Value:=" , "0deg"  
],  
[ "NAME:Pair",
```

```
    "Name:=" , "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:=" , "",
```

	<pre>"MaterialValue:=" , "\"copper\"", "SurfaceMaterialValue:=", "\"\"", "SolveInside:=" , False, "ShellElement:=" , False, "ShellElementThickness:=", "0mm", "IsMaterialEditable:=" , True, "UseMaterialAppearance:=", False, "IsLightweight:=" , False])</pre>

CreatePolyline

Creates a polyline.

UI Access	Draw > Line.		
Parameters	Name	Type	Description
	<Parameters>	Array	Structured array. Array("NAME:PolylineParameters", "IsPolylineCovered:=", <bool>, "IsPolylineClosed:=", <bool>, <PolylinePointsArray>, <PolylineSegmentsArray>)
	<PolylinePointsArray>	Array	Array("NAME:PolylinePoints", <OnePointArray>,

			<OnePointArray>, ...)
	<OnePointArray>	Array	Array("NAME:PLPoint", "X:=", <value>, "Y:=", <value>, "Z:=", <value>))
	<PolylineSegmentsArray>	Array	<PolylineSegmentsArray> Array("NAME:PolylineSegments", <OneSegmentArray>, <OneSegmentArray>, ...)
	<OneSegmentArray>	Array	Array("NAME:PLSegment", "SegmentType:=", <"Line", "Arc", "Spline", or "AngularArc">, "StartIndex:=", <value>, "NoOfPoints:=", <value>)
	<AttributesArray>	Array	Structured array. See: AttributesArray .
Return Value	None.		

Python Syntax	CreatePolyline(<Parameters>, <Attributes>)
Python Example	<pre>oEditor.CreatePolyline(["NAME:PolylineParameters", "IsPolylineCovered:=" , True, "IsPolylineClosed:=" , False,</pre>

```
[ "NAME:PolylinePoints",  
  [ "NAME:PLPoint",  
    "X:=" , "20000mm",  
    "Y:=" , "-20000mm",  
    "Z:=" , "0mm"  
  ],  
  [ "NAME:PLPoint",  
    "X:=" , "-90000mm",  
    "Y:=" , "20000mm",  
    "Z:=" , "0mm"  
  ],  
  [ "NAME:PLPoint",  
    "X:=" , "10000mm",  
    "Y:=" , "-140000mm",  
    "Z:=" , "0mm"  
  ]  
],  
[ "NAME:PolylineSegments",  
  [ "NAME:PLSegment",  
    "SegmentType:=" , "Line",  
    "StartIndex:=" , 0,
```

```
"NoOfPoints:="          , 2
],
["NAME:PLSegment",
"SegmentType:="        , "Line",
"StartIndex:="         , 1,
"NoOfPoints:="         , 2
]
],
["NAME:PolylineXSection",
"XSectionType:="       , "None",
"XSectionOrient:="     , "Auto",
"XSectionWidth:="      , "0mm",
"XSectionTopWidth:="   , "0mm",
"XSectionHeight:="     , "0mm",
"XSectionNumSegments:=" , "0",
"XSectionBendType:="    , "Corner"
]],
["NAME:Attributes",
"Name:="               , "Polyline1",
"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])</pre>

CreateRectangle

Creates a rectangle.

UI Access	Draw > Rectangle.		
Parameters	Name	Type	Description
	<Parameters>	Array	Structured array. Array("NAME:RectangleParameters", "IsCovered:=" , <boolean> ,

			"XStart:=" , <string>, "YStart:=" , <string>, "ZStart:=" , <string>, "Width:=" , <string>, "Height:=" , <string>, "WhichAxis:=" , <string "X", "Y", or "Z">)
	<AttributesArray>	Array	Structured array. See: AttributesArray .
Return Value	None.		

Python Syntax	CreateRectangle(<Parameters>, <Attributes>)
Python Example	<pre>oEditor.CreateRectangle(["NAME:RectangleParameters", "IsCovered:=" , True, "XStart:=" , "-80000mm", "YStart:=" , "-90000mm", "ZStart:=" , "0mm", "Width:=" , "20000mm", "Height:=" , "30000mm", "WhichAxis:=" , "Z"],</pre>

	<pre>["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]</pre>

CreateRegion

Creates a region containing the design.

UI Access	Draw > Region.		
Parameters	Name	Type	Description

	<Parameters>	Array	Structured array. <pre>Array("NAME:RegionParameters", "+XPaddingType=", <string "Percentage Offset", "Absolute Offset", or "Absolute Position">, "+XPadding=", <string X value>, "-XPaddingType=", <string "Percentage Offset", "Absolute Offset", or "Absolute Position">, "-XPadding=", <string -X value>, "+YPaddingType=", <string "Percentage Offset", "Absolute Offset", or "Absolute Position">, "+YPadding=", <string Y value>, "-YPaddingType=", <string "Percentage Offset", "Absolute Offset", or "Absolute Position">, "-YPadding=", <string -Y value>, "+ZPaddingType=", <string "Percentage Offset", "Absolute Offset", or "Absolute Position">, "+ZPadding=", <string Z value>, "-ZPaddingType=", <string "Percentage Offset", "Absolute Offset", or "Absolute Position">, "-ZPadding=", <string -Z value>)</pre>
	<AttributesArray>	Array	Structured array. See: AttributesArray .
Return Value	None.		

Python Syntax	CreateRegion(<Parameters>, <Attributes>)
Python Example	<pre> oEditor.CreateRegion(["NAME:RegionParameters", "+XPaddingType:=" , "Percentage Offset", "+XPadding:=" , "0", "-XPaddingType:=" , "Percentage Offset", "-XPadding:=" , "0", "+YPaddingType:=" , "Percentage Offset", "+YPadding:=" , "0", "-YPaddingType:=" , "Percentage Offset", "-YPadding:=" , "0", "+ZPaddingType:=" , "Percentage Offset", "+ZPadding:=" , "0", "-ZPaddingType:=" , "Percentage Offset", "-ZPadding:=" , "0"], ["NAME:Attributes", "Name:=" , "Region", "Flags:=" , "Wireframe#", "Color:=" , "(143 175 143)", "Transparency:=" , 0, </pre>

	<pre>"PartCoordinateSystem:=", "Global", "UDMId:=" , "", "MaterialValue:=" , "\"vacuum\"", "SurfaceMaterialValue:=", "\"\"", "SolveInside:=" , False, "ShellElement:=" , False, "ShellElementThickness:=", "nan ", "IsMaterialEditable:=" , True, "UseMaterialAppearance:=", False, "IsLightweight:=" , False])</pre>

CreateRegularPolyhedron

Creates a regular polyhedron.

UI Access	Draw > Regular Polyhedron.		
Parameters	Name	Type	Description
	<Parameters>	Array	Structured array. Array("NAME:PolyhedronParameters", "XCenter:=" , <string>, "YCenter:=" , <string>,

			"ZCenter:=" , <string>, "XStart:=" , <string>, "YStart:=" , <string>, "ZStart:=" , <string>, "Height:=" , <string>, "NumSides:=" , <string containing number greater than 2>, "WhichAxis:=" , <string "X", "Y", or "Z">)
	<AttributesArray>	Array	Structured array. See: AttributesArray .
Return Value	None.		

Python Syntax	CreateRegularPolyhedron(<Parameters>, <Attributes>)
Python Example	<pre>oEditor.CreateRegularPolyhedron(["NAME:PolyhedronParameters", "XCenter:=" , "40000mm", "YCenter:=" , "-80000mm", "ZCenter:=" , "0mm", "XStart:=" , "50000mm", "YStart:=" , "-70000mm", "ZStart:=" , "0mm", "Height:=" , "50000mm", "NumSides:=" , "8",</pre>

```
"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
])
```

CreateRegularPolygon

Creates a regular polygon.

UI Access	Draw > Regular Polygon.		
Parameters	Name	Type	Description
	<Parameters>	Array	Structured array. Array ("NAME:RegularPolygonParameters", "IsCovered:=" , <boolean>, "XCenter:=" , <string>, "YCenter:=" , <string>, "ZCenter:=" , <string>, "XStart:=" , <string>, "YStart:=" , <string>, "ZStart:=" , <string>, "NumSides:=" , <string containing number greater than 2>, "WhichAxis:=" , <string "X", "Y", or "Z">)
	<AttributesArray>	Array	Structured array. See: AttributesArray .
Return Value	None.		

Python Syntax	CreateRegularPolygon(<Parameters>, <Attributes>)
Python Example	<pre>oEditor.CreateRegularPolygon(["NAME:RegularPolygonParameters", "IsCovered:=" , True, "XCenter:=" , "-70000mm",</pre>

```
"YCenter:="                , "-100000mm",
"ZCenter:="                , "0mm",
"XStart:="                 , "-50000mm",
"YStart:="                 , "-80000mm",
"ZStart:="                 , "0mm",
"NumSides:="               , "12",
"WhichAxis:="              , "Z"
],
["NAME:Attributes",
  "Name:="                  , "Polygon1",
  "Flags:="                 , "",
  "Color:="                 , "(143 175 143)",
  "Transparency:="          , 0,
  "PartCoordinateSystem:=" , "Global",
  "UDMId:="                 , "",
  "MaterialValue:="         , "\"copper\"",
  "SurfaceMaterialValue:=" , "\"\"",
  "SolveInside:="           , False,
  "ShellElement:="          , False,
  "ShellElementThickness:=" , "0mm",
```

	<pre> "IsMaterialEditable:=" , True, "UseMaterialAppearance:=", False, "IsLightweight:=" , False 1) </pre>

CreateSphere

Creates a sphere.

UI Access	Draw > Sphere.		
Parameters	Name	Type	Description
	<Parameters>	Array	Structured array. <pre> Array("NAME:SphereParameters", "XCenter:=", <string>, "YCenter:=", <string>, "ZCenter:=", <string>, "Radius:=", <string>) </pre>
	<AttributesArray>	Array	Structured array. See: AttributesArray .
Return Value	None.		

Python Syntax	CreateSphere(<Parameters>, <Attributes>)
Python Example	<pre> oEditor.CreateSphere (["NAME:SphereParameters", </pre>

```
"XCenter:="          , "-40000mm",
"YCenter:="          , "-130000mm",
"ZCenter:="          , "0mm",
"Radius:="           , "22360.6797749979mm"
],
["NAME:Attributes",
  "Name:="            , "Sphere1",
  "Flags:="           , "",
  "Color:="           , "(143 175 143)",
  "Transparency:="    , 0,
  "PartCoordinateSystem:=", "Global",
  "UDMId:="           , "",
  "MaterialValue:="   , "\"copper\"",
  "SurfaceMaterialValue:=", "\"\"",
  "SolveInside:="     , False,
  "ShellElement:="    , False,
  "ShellElementThickness:=", "0mm",
  "IsMaterialEditable:=" , True,
  "UseMaterialAppearance:=", False,
  "IsLightweight:="   , False
```

	1)

CreateSpiral

Creates a spiral by sweeping the specified object(s).

UI Access	Draw > Spiral.		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: SelectionsArray .
	<Parameters>	Array	Structured array. Array("NAME:SpiralParameters", "XCenter:=" , <string>, "YCenter:=" , <string>, "ZCenter:=" , <string>, "YStartDir:=" , <string>, "ZStartDir:=" , <string>, "NumThread:=" , <string>, "RightHand:=" , <boolean>, "RadiusIncrement:=" , <string>)
Return Value	None.		

Python Syntax	CreateSpiral(<Parameters>, <Attributes>)
Python Example	oEditor.CreateSpiral (

	<pre>["NAME:Selections", "Selections:=" , "Polygon2", "NewPartsModelFlag:=" , "Model"], ["NAME:SpiralParameters", "XCenter:=" , "-70000mm", "YCenter:=" , "50000mm", "ZCenter:=" , "0mm", "XStartDir:=" , "-60000mm", "YStartDir:=" , "-10000mm", "ZStartDir:=" , "0mm", "NumThread:=" , "1", "RightHand:=" , True, "RadiusIncrement:=" , "1mm"]</pre>

CreateTorus

Creates a torus.

UI Access	Draw > Torus.

Parameters	Name	Type	Description
	<Parameters>	Array	Structured array. <pre>Array("NAME:TorusParameters", "XCenter:=" , <string>, "YCenter:=" , <string>, "ZCenter:=" , <string>, "MajorRadius:=" , <string>, "MinorRadius:=" , <string>, "WhichAxis:=" , <string "X", "Y", or "Z">)</pre>
	<AttributesArray>	Array	Structured array. See: AttributesArray .
Return Value	None.		

Python Syntax	CreateTorus(<Parameters>, <Attributes>)
Python Example	<pre>oEditor.CreateTorus(["NAME:TorusParameters", "XCenter:=" , "0.6mm", "YCenter:=" , "-0.6mm", "ZCenter:=" , "0mm", "MajorRadius:=" , "0.365028153987289mm", "MinorRadius:=" , "0.0821854415126694mm", "WhichAxis:=" , "Z"],</pre>

```
[ "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  
]
```

CreateUserDefinedModel

Creates a user-defined model.

Note: This option can be used to create a UDM from a Discovery model (**Modeler** > [Discovery Link](#)).

UI Access	Draw > User-Defined Model > [Model]		
Parameters	Name	Type	Description
	<Parameters>	Array	Structured array. <pre>["NAME:UserDefinedModelParameters", <definitionArray>, <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	Structured array containing arrays for individual parameters: <pre>["NAME:GeometryParameters", ["NAME:UDMParam", "Name:=" , <string>, "Value:=" , <string>, "PropType2:=" , <integer>, "PropFlag2:=" , <integer>]]</pre> Required UDM parameters depend on the UDM being created. To see which properties apply to a UDM, right-click the UDM in the Project Tree and select Properties. Then select the Parameters tab.

		PropType2 can be any of the following: • 0 – Property takes a string value. • 1 – Property is a menu option. • 2 – Property takes a number (integer or double). • 3 – Property takes a value (numbers, variables, or expressions). • 4 – Property is a file name. • 5 – Property corresponds to a check box. • 6 – Property specifies a 3D position. PropFlag2 can be any of the following: • 0 – No flags • 1 – Read-only • 2 – Must be integer • 4 – Must be real • 8 – Hidden PropFlag2 values can be combined. For example, a read-only property that must be an integer would take the value 3. A hidden property that must be real would take the value 12. These values are further described in the <code>UserDefinedPrimitiveStructures.h</code> file included with the installation under "...\\ANSYS Inc\\v252\\AnsysEM\\UserDefinedPrimitives\\Examples\\Headers"
Return Value	None	

Python Syntax	CreateUserDefinedModel(<Parameters>)
Python Example	<pre> oEditor.CreateUserDefinedModel (["NAME:UserDefinedModelParameters", ["NAME:Definition"], ["NAME:Options"], ["NAME:GeometryParams", ["NAME:UDMParam", "Name:=" , "ILD Thickness (ILD)", "Value:=" , "0.006mm", "PropType2:=" , 3, "PropFlag2:=" , 4], ["NAME:UDMParam", "Name:=" , "Line Spacing (LS)", "Value:=" , "0.004mm", "PropType2:=" , 3, "PropFlag2:=" , 4], ["NAME:UDMParam", "Name:=" , "Line Thickness (LT)", "Value:=" , "0.005mm", </pre>

```
"PropType2:="          , 3,
"PropFlag2:="          , 4
],
["NAME:UDMParam",
"Name:="                , "Line Width (LW)",
"Value:="                , "0.004mm",
"PropType2:="          , 3,
"PropFlag2:="          , 4
],
["NAME:UDMParam",
"Name:="                , "No. of Turns (N)",
"Value:="                , "2",
"PropType2:="          , 3,
"PropFlag2:="          , 2
],
["NAME:UDMParam",
"Name:="                , "Outer Diameter (OD)",
"Value:="                , "0.15mm",
"PropType2:="          , 3,
"PropFlag2:="          , 4
```

```

],
["NAME:UDMParam",
  "Name:=" , "Substrate Thickness",
  "Value:=" , "0.2mm",
  "PropType2:=" , 3,
  "PropFlag2:=" , 4
],
["NAME:UDMParam",
  "Name:=" , "Inductor Type",
  "Value:=" , "\"Square,Square,Octagonal,Circular,Square-Differential,
                Octagonal-Differential,Circular-Differential\"",
  "DataType:=" , "String",
  "PropType2:=" , 1,
  "PropFlag2:=" , 0
],
["NAME:UDMParam",
  "Name:=" , "Underpass Thickness (UT)",
  "Value:=" , "0.001mm",
  "PropType2:=" , 3,
  "PropFlag2:=" , 4
],

```

```
[ "NAME:UDMParam",  
  "Name:=" , "Via Thickness (VT) ",  
  "Value:=" , "0.001mm",  
  "PropType2:=" , 3,  
  "PropFlag2:=" , 4  
]  
],  
"DllName:=" , "Maxwell3D/OnDieSpiralInductor.py",  
"Library:=" , "syslib",  
"Version:=" , "2.0",  
"ConnectionID:=" , ""  
])  
])
```

Python Example of Creating a UDM from a [Discovery](#) Model

```
oEditor.CreateUserDefinedModel(  
  [  
    "NAME:UserDefinedModelParameters",  
    [  
      "NAME:Definition",  
      [  
        "NAME:UDMParam",  
        "Name:=" , "GeometryFilePath",  
        "Value:=" , "\"C:/Users/chenwan/Desktop/restored_files/tee.dsco\"",  

```

```

        "DataType:=" , "String",
        "PropType2:=" , 0,
        "PropFlag2:=" , 1
    ],
    [
        "NAME:UDMParam",
        "Name:=" , "ProcessID",
        "Value:=" , "255988",
        "DataType:=" , "Int",
        "PropType2:=" , 2,
        "PropFlag2:=" , 8
    ]
],
[
    "NAME:Options",
    [
        "NAME:UDMParam",
        "Name:=" , "IsDiscoveryLink",
        "Value:=" , "1",
        "DataType:=" , "Int",
        "PropType2:=" , 5,
        "PropFlag2:=" , 8
    ],
    [
        "NAME:UDMParam",
        "Name:=" , "Solid Bodies",
        "Value:=" , "1",
        "DataType:=" , "Int",
        "PropType2:=" , 5,
        "PropFlag2:=" , 0
    ],
    [
        "NAME:UDMParam",
        "Name:=" , "Surface Bodies",
        "Value:=" , "1",
        "DataType:=" , "Int",
        "PropType2:=" , 5,
        "PropFlag2:=" , 0
    ]
]

```

```
],  
[  
  "NAME:UDMParam",  
  "Name:=" , "Parameters",  
  "Value:=" , "\"eCADImportParameterType_Independent,eCADImportParameterType_None,eCADIm-  
portParameterType_Independent,eCADImportParameterType_All\"",  
  "DataType:=" , "String",  
  "PropType2:=" , 1,  
  "PropFlag2:=" , 0  
],  
[  
  "NAME:UDMParam",  
  "Name:=" , "Parameter Key",  
  "Value:=" , "\"\"",  
  "DataType:=" , "String",  
  "PropType2:=" , 0,  
  "PropFlag2:=" , 0  
],  
[  
  "NAME:UDMParam",  
  "Name:=" , "Named Selections",  
  "Value:=" , "1",  
  "DataType:=" , "Int",  
  "PropType2:=" , 5,  
  "PropFlag2:=" , 8  
],  
[  
  "NAME:UDMParam",  
  "Name:=" , "Materials",  
  "Value:=" , "\"None,None,Assignments,Assignments and Properties\"",  
  "DataType:=" , "String",  
  "PropType2:=" , 1,  
  "PropFlag2:=" , 0  
],  
[  
  "NAME:UDMParam",
```



```

    "Name:=" , "Import As Lightweight",
    "Value:=" , "0",
    "DataType:=" , "Int",
    "PropType2:=" , 5,
    "PropFlag2:=" , 0
  ],
  [
    "NAME:UDMParam",
    "Name:=" , "Facet Level",
    "Value:=" , "\"3,1,2,3,4,5\"",
    "DataType:=" , "String",
    "PropType2:=" , 1,
    "PropFlag2:=" , 0
  ],
  [
    "NAME:UDMParam",
    "Name:=" , "Cleaning",
    "Value:=" , "0",
    "DataType:=" , "Int",
    "PropType2:=" , 5,
    "PropFlag2:=" , 0
  ],
  [
    "NAME:UDMParam",
    "Name:=" , "Use Parasolid For Transfer",
    "Value:=" , "1",
    "DataType:=" , "Int",
    "PropType2:=" , 5,
    "PropFlag2:=" , 8
  ],
  [
    "NAME:UDMParam",
    "Name:=" , "Tolerant Stitching",
    "Value:=" , "1",
    "DataType:=" , "Int",
    "PropType2:=" , 5,
    "PropFlag2:=" , 8
  ],
],

```

```
[
  "NAME:UDMParam",
  "Name:=" , "Stitch Tolerance",
  "Value:=" , "0.1",
  "PropType2:=" , 3,
  "PropFlag2:=" , 8
],
[
  "NAME:UDMParam",
  "Name:=" , "Tighten Gaps",
  "Value:=" , "1",
  "DataType:=" , "Int",
  "PropType2:=" , 5,
  "PropFlag2:=" , 8
],
[
  "NAME:UDMParam",
  "Name:=" , "Tighten Gaps Tolerance",
  "Value:=" , "1e-06",
  "PropType2:=" , 3,
  "PropFlag2:=" , 8
],
[
  "NAME:UDMParam",
  "Name:=" , "Use Associativity",
  "Value:=" , "1",
  "DataType:=" , "Int",
  "PropType2:=" , 5,
  "PropFlag2:=" , 8
],
[
  "NAME:UDMParam",
  "Name:=" , "Attributes",
  "Value:=" , "1",
  "DataType:=" , "Int",
  "PropType2:=" , 5,
```

```

    "PropFlag2:=" , 8
  ],
  [
    "NAME:UDMParam",
    "Name:=" , "Import Coordinate Systems",
    "Value:=" , "1",
    "DataType:=" , "Int",
    "PropType2:=" , 5,
    "PropFlag2:=" , 8
  ],
  [
    "NAME:UDMParam",
    "Name:=" , "Decompose Disjoint Faces",
    "Value:=" , "1",
    "DataType:=" , "Int",
    "PropType2:=" , 5,
    "PropFlag2:=" , 8
  ],
  [
    "NAME:UDMParam",
    "Name:=" , "Import Using Instances",
    "Value:=" , "1",
    "DataType:=" , "Int",
    "PropType2:=" , 5,
    "PropFlag2:=" , 8
  ],
  [
    "NAME:UDMParam",
    "Name:=" , "Line Bodies",
    "Value:=" , "1",
    "DataType:=" , "Int",
    "PropType2:=" , 5,
    "PropFlag2:=" , 8
  ],
  [
    "NAME:UDMParam",
    "Name:=" , "Mixed Import Resolution",
    "Value:=" , "\"eMixedImport_None\"",

```

```
"DataType:=" , "String",
"PropType2:=" , 0,
"PropFlag2:=" , 8
],
[
  "NAME:UDMParam",
  "Name:=" , "Enclosure and Symmetry Processing",
  "Value:=" , "1",
  "DataType:=" , "Int",
  "PropType2:=" , 5,
  "PropFlag2:=" , 8
],
[
  "NAME:UDMParam",
  "Name:=" , "Reader Mode Saves Updated File",
  "Value:=" , "0",
  "DataType:=" , "Int",
  "PropType2:=" , 5,
  "PropFlag2:=" , 8
],
[
  "NAME:UDMParam",
  "Name:=" , "Smart CAD Update",
  "Value:=" , "0",
  "DataType:=" , "Int",
  "PropType2:=" , 5,
  "PropFlag2:=" , 8
],
[
  "NAME:UDMParam",
  "Name:=" , "Import Work Points",
  "Value:=" , "1",
  "DataType:=" , "Int",
  "PropType2:=" , 5,
  "PropFlag2:=" , 8
]
]
```

```
],
[
    "NAME:GeometryParams"
],
"DllName:=" , "SACADIntegUDM",
"Library:=" , "installLib",
"Version:=" , "1.0",
"ConnectionID:=" , ""
])
```

CreateUserDefinedPart

Creates a user-defined part.

UI Access	Draw > User-Defined Primitive > [Part]		
Parameters	Name	Type	Description
	<Parameters>	Array	Structured array. ["NAME:UserDefinedPrimitiveParameters", "DllName:=" , <string>, "Version:=" , <string>, "NoOfParameters:=" , <integer>, "Library:=" , <string>, <paramVectorArray>]
	<paramVectorArray>	Array	Structured array containing arrays for each pair: ["NAME:ParamVector", <pair>, <pair>, <pair>,...] <pair>:

			<code>["NAME:Pair", "Name:=" , <string>, "Value:=" , <string>]</code>
	<code><Attributes></code>	Array	Structured array. See: AttributesArray .
Return Value	None.		

Python Syntax	<code>CreateUserDefinedPart(<Parameters>, <paramVectorArray>, <Attributes>)</code>
Python Example	<pre>oEditor.CreateUserDefinedPart(["NAME:UserDefinedPrimitiveParameters", "DllName:=" , "RMxpert/LapCoil.dll", "Version:=" , "16.0", "NoOfParameters:=" , 22, "Library:=" , "syslib", ["NAME:ParamVector", ["NAME:Pair", "Name:=" , "DiaGap", "Value:=" , "100mm"], ["NAME:Pair",</pre>

```
"Name:=" , "DiaYoke",  
"Value:=" , "20mm"  
],  
["NAME:Pair",  
"Name:=" , "Length",  
"Value:=" , "100mm"  
],  
["NAME:Pair",  
"Name:=" , "Skew",  
"Value:=" , "0deg"  
],  
["NAME:Pair",  
"Name:=" , "Slots",  
"Value:=" , "18"  
],  
["NAME:Pair",  
"Name:=" , "SlotType",  
"Value:=" , "1"  
],  
["NAME:Pair",  
"Name:=" , "Hs0",
```

```
"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
])
```

Edit3DComponent

Edits a specified 3D component by saving it to a new name so that its properties are accessible. To change the properties on an existing component without changing its name, see [ChangeProperty](#).

UI Access	N/A		
Parameters	Name	Type	Description
	<compName>	String	Component name.
	<Parameters>	Array	Structured array. Array("NAME:EditComponentParametersData", "NewComponentName=", <string>, "GeometryParameters=", <string>, "MaterialParameters=", <string>, "DesignParameters=", <string>, <ComponentMeshing>, <Excitations>)
	<ComponentMeshing>	Array	Structured array. Array("NAME:Component Meshing", "MeshAssembly=", <boolean>)
	<Excitations>	Array	Structured array containing array of suppressed excitations. Array("NAME:Excitations", "Suppressed=", <array>)
Return Value	None.		

Python Syntax	Edit3DComponent (<compName>, <Parameters>)
Python Example	<pre>oEditor.Edit3DComponent("Connector1", ["NAME:EditComponentParametersData", "NewComponentName:=", "Connector2", "GeometryParameters:=", "", "MaterialParameters:=", "", "DesignParameters:=", "", "NAME:Component Meshing", "MeshAssembly:=", False], ["NAME:Excitations", "Suppressed:=", []]])</pre>

[Beta] EditNativeComponentDefinition [Maxwell]

Edit a native component definition.

UI Access	In the Project Manager , expand 3D Components , right-click on the component, and select Edit Definition .		
Parameters	Name	Type	Description
	<LinkParameters>	Array	Data of the 3D component.

Return Value	None.

Python Syntax	EditNativeComponentDefinition (<LinkParameters>)
Python Example	<pre>oEditor.EditNativeComponentDefinition(["NAME:EditNativeComponentDefinitionData", "DefinitionName:=" , "LC1", ["NAME:GeometryDefinitionParameters", ["NAME:VariableOrders"]], ["NAME:DesignDefinitionParameters", ["NAME:VariableOrders"]],])</pre>

```
[
  "NAME:MaterialDefinitionParameters",
  [
    "NAME:VariableOrders"
  ]
],
"NextUniqueID:="      , 0,
"MoveBackwards:="     , False,
"DatasetType:="       , "ComponentDatasetType",
[
  "NAME:DatasetDefinitions"
],
[
  "NAME:NativeComponentDefinitionProvider",
  "Type:="              , "Layout Component",
  "Unit:="              , "mm",
  "Version:="          , 1.1,
  "EDBDefinition:="     , "EdgeCircuitPort_Inductor1",
  [
    "NAME:VariableMap"
  ]
],
```

	<pre>"ReferenceCS:=" , "U0", "CSToImport:=" , ["Global","U0"]], "ComponentName:=" , "LC1", "Company:=" , "", "Company URL:=" , "", "Model Number:=" , "", "Help URL:=" , "", "Version:=" , "1.0", "Notes:=" , "", "IconType:=" , "Layout Component"])</pre>

EditPolyline

Modifies a specified polyline. See: [CreatePolyline](#).

UI Access	N/A		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: SelectionsArray .
	<Parameters>	Array	Structured array.
			Array("NAME:PolylineParameters",

			"IsPolylineCovered:=", <bool>, "IsPolylineClosed:=", <bool>, <PolylinePointsArray>, <PolylineSegmentsArray>)
	<PolylinePointsArray>	Array	Array("NAME:PolylinePoints", <OnePointArray>, <OnePointArray>, ...)
	<OnePointArray>	Array	Array("NAME:PLPoint", "X:=", <value>, "Y:=", <value>, "Z:=", <value>))
	<PolylineSegmentsArray>	Array	<PolylineSegmentsArray> Array("NAME:PolylineSegments", <OneSegmentArray>, <OneSegmentArray>, ...)
	<OneSegmentArray>	Array	Array("NAME:PLSegment", "SegmentType:=", <"Line", "Arc", "Spline", or "AngularArc">, "StartIndex:=", <value>, "NoOfPoints:=", <value>)
Return Value	None.		

Python Syntax	EditPolyline(<SelectionsArray>, <Parameters>)
Python Example	<pre>oEditor.EditPolyline(["NAME:Selections",</pre>

```
"Selections:=", "Polyline1"]
["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:=", "-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"

```

	<pre>]])</pre>

Get3DComponentParameters

Returns parameters for a specified 3D component.

UI Access	N/A		
Parameters	Name	Type	Description
	<compName>	String	3D component name.
Return Value	Array containing component parameters.		

Python Syntax	Get3DComponentParameters(<compName>)
Python Example	<code>oEditor.Get3DComponentParameters('Connector')</code>

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>

Get3DComponentInstanceNames

Returns instance names of 3D component definitions.

UI Access	N/A		
Parameters	Name	Type	Description
	<DefinitionName>	String	Definition name.
Return Value	Array containing instance names.		

Python Syntax	Get3DComponentInstanceNames(<DefinitionName>)
Python Example	<code>oEditor.Get3DComponentInstanceNames("Connector")</code>

Get3DComponentMaterialNames

Returns material names for a specified *.a3dcomp format 3D component.

UI Access	N/A		
Parameters	Name	Type	Description
	<InstanceName>	String	Component instance name.
Return Value	Array containing material names.		

Python Syntax	Get3DComponentMaterialNames(<InstanceName>)

Python Example	<code>oEditor.Get3DComponentMaterialNames("Connector1.a3dcomp")</code>

Get3DComponentMaterialProperties

Returns material properties for a specified 3D component.

UI Access	N/A		
Parameters	Name	Type	Description
	<MaterialName>	String	Material name.
Return Value	Array containing material properties.		

Python Syntax	<code>Get3DComponentMaterialProperties(<MaterialName>)</code>
Python Example	<code>oEditor.Get3DComponentMaterialProperties('Connector1:Material01')</code>

GetUserDefinedPartVersion

Get version number for a user-defined part.

UI Access	None		
Parameters	Name	Type	Description
	<Library>	String	Specifies the file location of the user-defined part
	<DLLName>	String	Specifies the .dll or .py file used to create the user-defined part
Return Value	Version number as a string		

Python Syntax	GetUserDefinedPartVersion(<Library>, <DLLName>)
Python Example	Example: <pre>oEditor.GetUserDefinedPartVersion("syslib", "SegmentedHelix/PolygonHelix.dll")</pre> Example: <pre>oEditor.GetUserDefinedPartVersion("syslib", "Examples/geodesic_sphere.py")</pre> Example: <pre>oProject = oDesktop.GetActiveProject() oDesign = oProject.SetActiveDesign("HFSSDesign1") oEditor = oDesign.SetActiveEditor("3D Modeler") print(oEditor.GetUserDefinedPartVersion("syslib", "RMxpert/LapCoil.dll"))</pre>

Insert3DComponent

Inserts a 3D component in specified coordinate system with specified offset.

UI Access	Draw > 3D Component Library > Browse > [Component].		
Parameters	Name	Type	Description
	<ComponentData>	Array	Structured array.
			<pre>Array("NAME:InsertComponentData", "TargetCS:=", <string>, "ComponentFile:=", <string filepath>), "IsLocal:=", <Boolean>,</pre>

			<pre>"InstanceParameters:=", <string>, "XOffset:=" , "float with unit", "YOffset:=" , "float with unit", "ZOffset:=" , "float with unit")</pre>
Return Value	None.		

Python Syntax	Insert3DComponent(<ComponentData>)
Python Example	<pre>oEditor.Insert3DComponent (["NAME:InsertComponentData", "TargetCS:=" , "Global", "ComponentFile:=" , "C:/Program Files/ANSYS Inc/v252/An- sysEM/syslib/3DComponents/HFSS/Rectangular Waveguide/Straight.a3dcomp", "IsLocal:=" , False, "UniqueIdentifier:=" , "", ["NAME:InstanceParameters", "GeometryParameters:=" , "Flange='\0.4in\' FlangeFillet='\0.2in\' FlangeThickness='\0.2in\' GuideHeight='\0.4in\' GuideLength='\4in\' GuideWidth='\0.9in\' WallThickness='\0.1in\'", "MaterialParameters:=" , "",</pre>

	<pre> "DesignParameters:=" , ""], "XOffset:=" , "-70mm", "YOffset:=" , "45mm", "ZOffset:=" , "0mm"])</pre>

InsertNativeComponent [Layout Component to Maxwell with CS]

Inserts a Layout Component to Maxwell 3D with custom coordinate system definitions.

UI Access	Create Layout Component ...		
Parameters	Name	Type	Description
	<LinkParameters>	Array	Structured array containing data of inserted layout component.
Return Value	None.		

Python Syntax	InsertNativeComponent (<LinkedParameters>)
Python Example	<pre>oEditor.InsertNativeComponent (["NAME:InsertNativeComponentData", "TargetCS:=" , "Global", "SubmodelDefinitionName:=" , "LC1",</pre>

```
[
    "NAME:ComponentPriorityLists"
],
"NextUniqueID:="          , 0,
"MoveBackwards:="        , False,
"DatasetType:="           , "ComponentDatasetType",
[
    "NAME:DatasetDefinitions"
],
[
    "NAME:BasicComponentInfo",
    "ComponentName:="      , "LC1",
    "Company:="            , "",
    "Company URL:="        , "",
    "Model Number:="       , "",
    "Help URL:="           , "",
    "Version:="            , "1.0",
    "Notes:="              , "",
    "IconType:="           , "Layout Component"
],
```

```
[
  "NAME:GeometryDefinitionParameters",
  [
    "NAME:VariableOrders"
  ]
],
[
  "NAME:DesignDefinitionParameters",
  [
    "NAME:VariableOrders"
  ]
],
[
  "NAME:MaterialDefinitionParameters",
  [
    "NAME:VariableOrders"
  ]
],
"DefReferenceCSID:=", 1,
"MapInstanceParameters:=", "DesignVariable",
"UniqueDefinitionIdentifier:=", "27e7d8f5-da68-409b-8c4d-bf2bc6d95f69",
```

```
"OriginFilePath:="      , "",
"IsLocal:="             , False,
"ChecksumString:="      , "",
"ChecksumHistory:="     , [],
"VersionHistory:="     , [],
[
  "NAME:NativeComponentDefinitionProvider",
  "Type:="               , "Layout Component",
  "Unit:="               , "mm",
  "EDBDefinition:="     , "EdgeCircuitPort_Inductor1",
  [
    "NAME:VariableMap"
  ],
  "ReferenceCS:="        , "Global",
  "CSToImport:="         , ["Global","U0"]
],
[
  "NAME:InstanceParameters",
  "GeometryParameters:=" , "",
  "MaterialParameters:=" , "",
```

	<pre> "DesignParameters:=" , ""]]) </pre>

InsertPolylineSegment

Inserts a polyline segment before or after a specified existing segment.

UI Access	Draw > Line Segment > [Selection].		
Parameters	Name	Type	Description
	<Parameters>	Array	Structured array. <pre> Array("NAME:Insert Polyline Segment", "Selections:=" , <string>, "Segment Indices:=" , <array containing integers>, "At Start:=" , <boolean>, "SegmentType:=" , <string "Line", "Arc", "Spline", or "AngularArc">, <PolylinePointsArray> </pre>
	<PolylinePointsArray>	Array	Structured array. See: CreatePolyline .
Return Value	None.		

Python Syntax	InsertPolylineSegment(<Parameters>)
Python Example	oEditor.InsertPolylineSegment (

```
[ "NAME:Insert Polyline Segment",  
  "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"  
    ]  
  ]  
]
```

```
] )
```

SweepAlongPath

Sweeps the specified 1D or 2D parts along a path. The last 1D object specified is the path for the sweep.

UI Access	Draw > Sweep > Along Path.		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: SelectionsArray .
	<PathSweepParametersArray>	Array	Array("NAME:PathSweepParameters", "DraftAngle:=", <value>, "DraftType:=", <string>, "CheckFaceFaceIntersection:=", <bool>, "TwistAngle:=", <value>) Possible values for DraftType are "Extended", "Round", and "Natural".
Return Value	None.		

Python Syntax	SweepAlongPath(<SelectionsArray>, <PathSweepParametersArray>)
Python Example	<pre>oEditor.SweepAlongPath(["NAME:Selections", "Selections:=" , "Rectangle1,Polyline1", "NewPartsModelFlag:=" , "Model"], ["NAME:PathSweepParameters",</pre>

	<pre>"DraftAngle:=" , "0deg", "DraftType:=" , "Round", "CheckFaceFaceIntersection:=", False, "TwistAngle:=" , "0deg"])</pre>

SweepAlongVector

Sweeps the specified 1D or 2D parts along a vector.

UI Access	Draw > Sweep > Along Vector.		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: SelectionsArray .
	<VecSweepParametersArray>	Array	Array("NAME:VectorSweepParameters", "DraftAngle:=", <value>, "DraftType:=", <string>, "CheckFaceFaceIntersection:=", <bool>, "SweepVectorX:=", <value> "SweepVectorY:=", <value> "SweepVectorZ:=", <value>) Possible values for DraftType are "Extended", "Round", and "Natural".

Return Value	None.

Python Syntax	SweepAlongVector(<SelectionsArray>, <VecSweepParametersArray>)
Python Example	<pre>oEditor.SweepAlongVector(["NAME:Selections", "Selections:=" , "Rectangle1", "NewPartsModelFlag:=" , "Model"], ["NAME:VectorSweepParameters", "DraftAngle:=" , "0deg", "DraftType:=" , "Round", "CheckFaceFaceIntersection:=", False, "SweepVectorX:=" , "0mm" "SweepVectorY:=" , "0mm" "SweepVectorZ:=" , "12mm"])</pre>

SweepAroundAxis

Sweeps the specified 1D or 2D parts around an axis.

UI Access	Draw > Sweep > Around Axis.		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: SelectionsArray .
	<AxisSweepParametersArray>	Array	<pre>Array("NAME:AxisSweepParameters", "DraftAngle:=", <value>, "DraftType:=", <string>, "CheckFaceFaceIntersection:=", <bool>, "SweepAxis:=", <value> "SweepAngle:=", <value> "NumOfSegments:=", <value>) </pre> Possible values for DraftType are "Extended", "Round", and "Natural". Possible values for SweepAxis are "X", "Y", and "Z".
Return Value	None.		

Python Syntax	SweepAroundAxis(<SelectionsArray>, <AxisSweepParametersArray>)
Python Example	<pre>oEditor.SweepAroundAxis(["NAME:Selections", "Selections:=" , "Rectangle1", </pre>

	<pre>"NewPartsModelFlag:=" , "Model"], ["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 .
	<parameters>	Array	Structured array.
			Array("NAME:Parameters", "NAME:SweepFaceAlongNormalToParameters", "FacesToDetach:=", <faceIDarray>,

	<table><tr><td></td><td></td><td>"LengthOfSweep:=", "<value><units>")</td></tr></table>					"LengthOfSweep:=", "<value><units>")
		"LengthOfSweep:=", "<value><units>")				
Return Value	None					

Python Syntax	SweepFacesAlongNormal(<SelectionsArray> <parameters>)
Python Example	<pre>oEditor.SweepFacesAlongNormal(["NAME:Selections", "Selections:=", "Rectangle1", "NewPartsModelFlag:=", "Model"], ["NAME:Parameters", "NAME:SweepFaceAlongNormalToParameters", "FacesToDetach:=", [183], "LengthOfSweep:=", "0.1mm"])</pre>

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 .

	<table><tr><td><parameters></td><td>Array</td><td>Array("NAME:Parameters", "NAME:SweepFaceAlongNormalToParameters", "FacesToDetach:=", <faceIDarray>, "LengthOfSweep:=", "<value><units>")</td></tr><tr><td><AttributesArray></td><td>Array</td><td>Structured array. See: AttributesArray.</td></tr></table>	<parameters>	Array	Array("NAME:Parameters", "NAME:SweepFaceAlongNormalToParameters", "FacesToDetach:=", <faceIDarray>, "LengthOfSweep:=", "<value><units>")	<AttributesArray>	Array	Structured array. See: AttributesArray .
<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,</pre>

	<pre>"PartCoordinateSystem:=", "Global", "UDMId:=", "", "MaterialValue:=", "\"copper\"", "SurfaceMaterialValue:=", "\"\"", "SolveInside:=", False, "ShellElement:=", False, "ShellElementThickness:=", "0mm", "IsMaterialEditable:=", True, "UseMaterialAppearance:=", False, "IsLightweight:=", False])</pre>

UpdateComponentDefinition

Updates a 3D component's definition.

UI Access	Draw > 3D Component Library > Definitions.		
Parameters	Name	Type	Description
	<data>	Array	Structured array. Array("NAME:UpdateDefinitionData", "DefinitionNames:=", <string>, "Passwords:=", <array of strings>)

Return Value	None.

Python Syntax	UpdateComponentDefinition(<data>)
Python Example	<pre>oEditor.UpdateComponentDefinition(['NAME:UpdateDefinitionData', 'DefinitionNames:=', 'Connector, Magic_Tee', 'Passwords:=', ['', '']])</pre>

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>		

DeletePolylinePoint

Deletes either a start point or an end point from an existing polyline segment.

UI Access	Edit > Delete [Start/End] Point		
Parameters	Name	Type	Description
	<DeletePointArray>	Array	Structured array. Array("NAME:Delete Point", "Selections:=", <string "<PolylineName>:<PolylineAc-

			<pre>tion>:<int>">, "Segment Index:=", <integer>, "At Start:=", <bool True for start point; False for end point>)</pre>
Return Value	None.		

Python Syntax	DeletePolylinePoint (<DeletePointArray>)		
Python Example	<pre>oEditor.DeletePolylinePoint(["NAME:Delete Point", "Selections:=", "Polyline1:CreatePolyline:1", "Segment Index:=", 1, "At Start:=", True])</pre>		

DuplicateAlongLine

Duplicates specified parts along a line.

UI Access	Edit > Duplicate > Along Line.		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. <pre>Array("NAME:Selections", "Selections:=" , <string>, "NewPartsModelFlag:=" , <string>)</pre>
	<ParametersArray>	Array	Structured array. <pre>Array("NAME:DuplicateToAlongLineParameters",</pre>

			"CreateNewObjects:=" , <boolean>, "XComponent:=" , <string>, "YComponent:=" , <string>, "ZComponent:=" , <string>, "NumClones:=" , <string containing number greater than 1>)
	<OptionsArray>	Array	Structured array. Array("NAME:Options", "DuplicateAssignments:=" , <boolean>)
	<CreateGroup>	Array	Optional. Structured array. Array("CreateGroupsForNewObjects:=" , <boolean>)
Return Value	None.		

Python Syntax	DuplicateAlongLine (<SelectionsArray>, <ParametersArray>, <OptionsArray>, <CreateGroup>)
Python Example	<pre>oEditor.DuplicateAlongLine(["NAME:Selections", "Selections:=" , "Box1", "NewPartsModelFlag:=" , "Model"], ["NAME:DuplicateToAlongLineParameters",</pre>

```

"CreateNewObjects:="      , False,
"XComponent:="            , "1mm",
"YComponent:="            , "-0.7mm",
"ZComponent:="            , "0mm",
"NumClones:="             , "2"
],
["NAME:Options",
    "DuplicateAssignments:=", False
],
["CreateGroupsForNewObjects:=", False
])

```

DuplicateAroundAxis

Duplicates specified parts around an axis.

UI Access	Edit > Duplicate > Around Axis.		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. Array("NAME:Selections", "Selections:=" , <string>, "NewPartsModelFlag:=" , <string>)
	<ParametersArray>	Array	Structured array.

			<pre>Array("NAME:DuplicateAroundAxisParameters", "CreateNewObjects:=" , <boolean>, "WhichAxis:=" , <string>, "AngleStr:=" , <string>, "NumClones:=" , <string containing number greater than 1>)</pre>
	<OptionsArray>	Array	Structured array. <pre>Array("NAME:Options", "DuplicateAssignments:=" , <boolean>)</pre>
	<CreateGroup>	Array	Optional. Structured array. <pre>Array("CreateGroupsForNewObjects:=" , <boolean>)</pre>
Return Value	None.		

Python Syntax	DuplicateAroundAxis (<SelectionsArray>, <ParametersArray>, <OptionsArray>, <CreateGroup>)
Python Example	<pre>oEditor.DuplicateAroundAxis(["NAME:Selections", "Selections:=" , "Box1", "NewPartsModelFlag:=" , "Model"], ["NAME:DuplicateAroundAxisParameters",</pre>

	<pre>"CreateNewObjects:=" , True, "WhichAxis:=" , "Z", "AngleStr:=" , "90deg", "NumClones:=" , "2"], ["NAME:Options", "DuplicateAssignments:=", False], ["CreateGroupsForNewObjects:=", False])</pre>

DuplicateMirror

Duplicates specified parts according to a mirror plane.

UI Access	Edit > Duplicate > Mirror.		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. Array("NAME:Selections", "Selections:=" , <string>, "NewPartsModelFlag:=" , <string>)
	<ParametersArray>	Array	Structured array. Array("NAME:DuplicateToMirrorParameters",

			"DuplicateMirrorBaseX:=", <string>, "DuplicateMirrorBaseY:=", <string>, "DuplicateMirrorNormalX:=", <string>, "DuplicateMirrorNormalY:=", <string>, "DuplicateMirrorNormalZ:=", <string>) For Maxwell 2D XY designs, Z parameters should be set to "0". For Maxwell 2D RZ designs, Y parameters should be set to "0".
	<OptionsArray>	Array	Structured array. Array("NAME:Options", "DuplicateAssignments:=", <boolean>)
	<CreateGroup>	Array	Optional. Structured array. Array("CreateGroupsForNewObjects:=", <boolean>)
Return Value	None.		

Python Syntax	DuplicateMirror (<SelectionsArray>, <ParametersArray>, <OptionsArray>, <CreateGroup>)
Python Example	<pre>oEditor.DuplicateMirror(["NAME:Selections", "Selections:=" , "Box1", "NewPartsModelFlag:=" , "Model"],</pre>

	<pre>["NAME:DuplicateToMirrorParameters", "DuplicateMirrorBaseX=", "-0.4mm", "DuplicateMirrorBaseY=", "-1.2mm", "DuplicateMirrorBaseZ=", "0mm", "DuplicateMirrorNormalX=", "0.124034734589208mm", "DuplicateMirrorNormalY=", "0.992277876713668mm", "DuplicateMirrorNormalZ=", "0mm"], ["NAME:Options", "DuplicateAssignments=", False], ["CreateGroupsForNewObjects=", False])</pre>

Mirror

Mirrors specified part(s).

UI Access	Edit > Arrange > Mirror.		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: SelectionsArray .
	<MirrorParameters>	Array	Structured array. Array ("NAME:MirrorParameters",

			<code>"MirrorBaseX:=" , <string>, "MirrorBaseY:=" , <string>, "MirrorBaseZ:=" , <string>, "MirrorNormalX:=" , <string>, "MirrorNormalY:=" , <string>, "MirrorNormalZ:=" , <string>)</code>
Return Value	None.		

Python Syntax	<code>Mirror(<SelectionsArray>, <MirrorParameters>)</code>
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",</pre>

	<code>"MirrorNormalZ:=" , "0mm"</code> <code>l)</code>

Move

Moves specified part(s).

UI Access	Edit > Arrange > Move.		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: SelectionsArray .
	<TranslateParameters>	Array	Structured array. <pre>Array(["NAME:TranslateParameters", "TranslateVectorX:=" , <string>, "TranslateVectorY:=" , <string>, "TranslateVectorZ:=" , <string>])</pre> For Maxwell 2D XY designs, "TranslateVectorZ:=" should be set to "0". For Maxwell 2D RZ designs, "TranslateVectorY:=" should be set to "0".
Return Value	None.		

Python Syntax	Move(<SelectionsArray>, <TranslateParameters>)
Python Example	<pre>oEditor.Move(["NAME:Selections", "Selections:=" , "Box1_1",</pre>

	<pre>"NewPartsModelFlag:=" , "Model"], ["NAME:TranslateParameters", "TranslateVectorX:=" , "-0.5mm", "TranslateVectorY:=" , "0.1mm", "TranslateVectorZ:=" , "0mm"])</pre>

OffsetFaces

Offsets the faces of selected part(s).

UI Access	Edit > Arrange > Offset.		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: SelectionsArray .
	<OffsetParameters>	Array	Structured array. Array("NAME:OffsetParameters", "OffsetDistance:=" , <string>)
Return Value	None.		

Python Syntax	OffsetFaces (<SelectionsArray> , <OffsetParameters>)
Python Example	oEditor.OffsetFaces (

	<pre>["NAME:Selections", "Selections:=", "Box1_1", "NewPartsModelFlag:=", "Model"], ["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>

Rotate

Rotates specified object(s).

UI Access	Edit > Arrange > Rotate.		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: SelectionsArray .
	<RotateParameters>	Array	Structured array. Array("NAME:RotateParameters", "RotateAxis:=" , <string "X", "Y", or "Z">, "RotateAngle:=" , <string>) For Maxwell 2D XY Designs, "RotateAxis:=" should be set to "Z". For Maxwell 2D RZ Designs, "RotateAxis:=" should be set to "Y".
Return Value	None.		

Python Syntax	Rotate(<SelectionsArray>, <RotateParameters>)
Python Example	<pre>oEditor.Rotate(["NAME:Selections", "Selections:=", "Box1_1", "NewPartsModelFlag:=", "Model"], ["NAME:RotateParameters",</pre>

	<pre>"RotateAxis:=", "Z", "RotateAngle:=", "90deg"])</pre>

Scale

Scales specified object(s).

UI Access	Edit > Scale.		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: SelectionsArray .
	<ScaleParameters>	Array	Structured array. <pre>Array("NAME:ScaleParameters", "ScaleX:=" , <string containing scale factor>, "ScaleY:=" , <string containing scale factor>, "ScaleZ:=" , <string containing scale factor>) For Maxwell 2D XY Designs, "ScaleZ:=" should be set to "0". For Maxwell 2D RZ Designs, "ScaleY:=" should be set to "0".</pre>
Return Value	None.		

Python Syntax	Scale (<SelectionsArray>, <ScaleParameters>)
Python Example	<pre>oEditor.Scale(["NAME:Selections",</pre>

	<pre>"Selections:=", "Box1", "NewPartsModelFlag:=", "Model"], ["NAME:ScaleParameters", "ScaleX:=", "2", "ScaleY:=", "2", "ScaleZ:=", "2"])</pre>

Modeler Menu Commands

[AssignMaterial](#)

[Chamfer](#)

[Connect](#)

[CoverLines](#)

[CoverSurfaces](#)

[CreateEntityList](#)

[CreateFaceCS](#)

[CreateGroup](#)

[CreateObjectCS](#)

[CreateObjectFromEdges](#)

[CreateObjectFromFaces](#)

[CreateRelativeCS](#)

[DeleteEmptyGroups](#)

[DeleteLastOperation](#)

[DetachFaces](#)

[EditEntityList](#)

[EditFaceCS](#)

[EditObjectCS](#)

[EditRelativeCS](#)

[Export](#)

[ExportModelImageToFile](#)

[ExportModelMeshToFile](#)

[Fillet](#)

[FlattenGroup](#)

[Generate History](#)

[GetActiveCoordinateSystem](#)

[GetCoordinateSystems](#)

[HealObject](#)

[Import](#)

[ImportDXF](#)

[ImportGDSII \[Modeler\]](#)

[Intersect](#)

[MoveCStoEnd](#)

[MoveEntityToGroup](#)

[MoveFaces](#)

[ProjectSheet](#)

[PurgeHistory](#)

[ReplaceWith3DComponent](#)

[Section](#)

[SeparateBody](#)

[SetModelUnits](#)

[SetWCS](#)

[ShowWindow](#)

[Split](#)

[Subtract](#)

[SweepFacesAlongNormal](#)

[ThickenSheet](#)

[UncoverFaces](#)

[Unite](#)

[Ungroup](#)

[WrapSheet](#)

AssignMaterial

Assigns a material to specified object(s).

UI Access	Modeler > Assign Material.		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: SelectionsArray .
	<AttributesArray>	Array	Structured array. See: AttributesArray . This script supports the following attributes: • MaterialValue • SolveInside • ShellElement • ShellElementThickness • IsMaterialEditable • UseMaterialAppearance • IsLightweight
Return Value	None.		
Python Syntax	AssignMaterial(<SelectionsArray>, <AttributesArray>)		

Python Example	<pre>oEditor.AssignMaterial(["NAME:Selections", "AllowRegionDependentPartSelectionForPMLCreation:=", True, "AllowRegionSelectionForPMLCreation:=", True, "Selections:=" , "Box1"], ["NAME:Attributes", "MaterialValue:=" , "diamond", "SolveInside:=" , False, "ShellElement:=" , False, "ShellElementThickness:=" , "nan", "IsMaterialEditable:=" , True, "UseMaterialAppearance:=" , False, "IsLightweight:=" , False])</pre>

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",</pre>

	<pre>["NAME:ChamferParameters", "Edges:=" , [42], "Vertices:=" , [], "LeftDistance:=" , "0.1mm", "RightDistance:=" , "0.1mm", "ChamferType:=" , "Symmetric"]]</pre>

Connect

Connects two or more 1D polyline objects or 2D sheet objects.

UI Access	Modeler > Surface > Connect.		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: SelectionsArray .
Return Value	None.		

Python Syntax	Connect(<SelectionsArray>)
Python Example	<pre>oEditor.Connect(["NAME:Selections", "Selections:=", "Polyline2,Polyline1"])</pre>

CoverLines

Covers two or more 1D objects to form a sheet.

UI Access	Modeler > Surface > Cover Lines.		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: SelectionsArray .
Return Value	None.		

Python Syntax	CoverLines(<SelectionsArray>)		
Python Example	<pre>oEditor.CoverLines(["NAME:Selections", "Selections:=", "Polyline3,Polyline4", "NewPartsModelFlag:=", "Model"])</pre>		

CoverSurfaces

Covers two or more faces to form a solid object.

UI Access	Modeler > Surface > Cover Faces.		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: SelectionsArray .
Return Value	None.		

Python Syntax	CoverSurfaces (<SelectionsArray>)
Python Example	<pre>oEditor.CoverSurfaces(["NAME:Selections", "Selections:=", "Obj1_Face1,Obj2_Face2", "NewPartsModelFlag:=", "Model"])</pre>

CreateEntityList

Creates a list of entities containing objects or faces (not both).

Warning: As of the Ansys Electronics Desktop25r2 release, this command has been deprecated. It will be removed in a future release. Please use the [CreateNamedSelection](#) command instead.

UI Access	Modeler > Named Selection > Create > [Object / Face] Selection		
Parameters	Name	Type	Description
	<Parameters>	Array	Structured array. Array("NAME:GeometryEntityListParameters", "EntityType:=" , <string "Object" or "Face">, "EntityList:=" , <string of object names or IDs>) See GetObjectIDByName for returning object IDs.
	<AttributesArray>	Array	Structured array. See: AttributesArray . CreateEntityList takes only

	the "Name" parameter.
Return Value	None.

Python Syntax	CreateEntityList(<Parameters>,<AttributesArray>)
Python Example	<pre>oEditor.CreateEntityList(["NAME:GeometryEntityListParameters", "EntityType:=", "Object", "EntityList:=", "Bondwire1,Bondwire2"], ["NAME:Attributes", "Name:=", "Objectlist1"])</pre>

CreateFaceCS

Creates a Face Coordinate System from a selected face.

UI Access	Modeler > Coordinate System > Create > FaceCS.		
Parameters	Name	Type	Description
	<Parameters>	Array	Structured array. Array("NAME:FaceCSParameters", <OriginArray>, "MoveToEnd:=", <boolean>,

			<pre>"FaceID:=", <integer>, <AxisPosnArray>, "WhichAxis:=", <string "X", "Y", or "Z">, "ZRotationAngle:=", <string>, "XOffset:=", <string>, "YOffset:=", <string>, "AutoAxis:=", <boolean></pre>
	<i><OriginArray></i>	Array	<pre>Structured array. Array("NAME:Origin", "IsAttachedToEntity:=", <boolean>, "EntityID:=", <integer>, "FacetedBodyTriangleIndex:=", <integer>, "TriangleVertexIndex:=", <integer>, "PositionType:=", <string "FaceCenter", "EdgeCenter", "OnVertex", "OnEdge", or "OnFace">, "UParam:=", <float between 0 and 1 representing the relative position of the point on the edge or face>, "VParam:=", <float between 0 and 1 representing the relative position of the point on the edge or face>, "XPosition:=", <string>,</pre>

			"YPosition:=" , <string>, "ZPosition:=" , <string>) IsAttachedToEntity specifies whether the point is anchored to a vertex, edge, or face. If True, provide UParam and VParam. If False, provide XPosition, YPosition, and ZPosition to provide fixed position. Pass "0" for unused parameters.
	<AxisPosnArray>	Array	Structured array. Array("NAME:AxisPosn", "IsAttachedToEntity:=" , <boolean>, "EntityID:=" , <integer>, "FacetedBodyTriangleIndex:=" , <integer>, "TriangleVertexIndex:=" , <integer>, "PositionType:=" , <string "FaceCenter", "EdgeCenter", "OnVertex", "OnEdge", or "OnFace">, "UParam:=" , <float>, "VParam:=" , <float>, "XPosition:=" , <string>, "YPosition:=" , <string>, "ZPosition:=" , <string>)
	<AttributesArray>	Array	Structured array. See: AttributesArray .
Return Value	None.		

Python Syntax	CreateFaceCS(<Parameters>,<AttributesArray>)

Python Example

```
oEditor.CreateFaceCS(  
  ["NAME:FaceCSParameters",  
    ["NAME:Origin",  
      "IsAttachedToEntity:=" , True,  
      "EntityID:=" , 46,  
      "FacetedBodyTriangleIndex:=", -1,  
      "TriangleVertexIndex:=" , -1,  
      "PositionType:=" , "FaceCenter",  
      "UParam:=" , 0,  
      "VParam:=" , 0,  
      "XPosition:=" , "0",  
      "YPosition:=" , "0",  
      "ZPosition:=" , "0"  
    ],  
    "MoveToEnd:=" , False,  
    "FaceID:=" , 46,  
    ["NAME:AxisPosn",  
      "IsAttachedToEntity:=" , True,  
      "EntityID:=" , 46,  
      "FacetedBodyTriangleIndex:=", -1,
```

```

    "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"
])

```

CreateGroup

Creates a group from objects specified in the history tree.

UI Access	Modeler > Group > Create.		
Parameters	Name	Type	Description
	<Parameters>	Array	Structured array. Array("NAME:GroupParameter", "ParentGroupID:" , <string>, "Parts:" , <string>, "SubmodelInstances:" , <string>, "Groups:" , <string>)
Return Value	None.		

Python Syntax	CreateGroup(<Parameters>)
Python Example	<pre>oEditor.CreateGroup(["NAME:GroupParameter", "ParentGroupID:" , "Model", "Parts:" , "Box1,Box2,Box3", "SubmodelInstances:" , "", "Groups:" , ""])</pre>

CreateNamedSelection

Creates a list of entities containing objects or faces (not both).

UI Access	Modeler > Named Selection > Create > [Object / Face] Selection		
Parameters	Name	Type	Description
	<Parameters>	Array	Structured array. <code>Array("NAME:NamedSelectionParameters", "Type:=" , <string "Object" or "Face">, "Selection:=" , <string of object names or IDs>)</code> See GetObjectIDByName for returning object IDs.
	<AttributesArray>	Array	Structured array. See AttributesArray . CreateNamedSelection takes only the "Name" parameter and UDM ID.
Return Value	None.		

Python Syntax	CreateNamedSelection(<Parameters>,<AttributesArray>)
Python Example	<pre>namedSelectionName = oEditor.CreateNamedSelection(["NAME:NamedSelectionParameters", "Type:=" , "Face", "Selection:=" , [7,9]], ["NAME:Attributes", "Name:=" , "FaceSelection1", "UDM ID:=" , -1]) </pre>

CreateObjectCS

Creates an object coordinate system from a selected object.

UI Access	Modeler > Coordinate System > Create > Object > [Offset / Rotated / Both].		
Parameters	Name	Type	Description
	<Parameters>	Array	Structured array. Array("NAME:ObjectCSParameters", <OriginArray> "MoveToEnd:=" , <boolean>, "ReverseXAxis:=" , <boolean>, "ReverseYAxis:=" , <boolean>, <xAxisArray / xAxisPosArray> <yAxisArray / yAxisPosArray>) Note: xAxisArray and xAxisPosArray differ. Use xAxisArray for absolute position and xAxisPosArray for relative position. Do the same for yAxisArray and yAxisPosArray.
	<OriginArray>	Array	Structured array. Array("NAME:Origin", "IsAttachedToEntity:=" , <boolean>, "EntityID:=" , <integer>, "FacetedBodyTriangleIndex:=" , <integer>,

		<pre> "TriangleVertexIndex:=" , <integer>, "PositionType:=" , <string "OnVertex", "EdgeCenter", "FaceCenter", "OnEdge", or "AbsolutePosition">, "UParam:=" , <float between 0 and 1 representing the relative position of the point on the edge or face>, "VParam:=" , <float between 0 and 1 representing the relative position of the point on the edge or face>, "XPosition:=" , <string>, "YPosition:=" , <string>, "ZPosition:=" , <string>) </pre> IsAttachedToEntity specifies whether the point is anchored. If True, provide UParam and VParam. If False, provide XPosition, YPosition, and ZPosition to provide fixed position. Pass "0" for unused parameters.
<xAxisArray>	Array	Structured array for absolute position: <pre> Array("NAME:xAxis", "DirectionType:=" , "AbsoluteDirection", "EdgeID:=" , <integer>, "FaceID:=" , <integer>, "xDirection:=" , <string>, "yDirection:=" , <string>, "zDirection:=" , <string>, "UParam:=" , <float>, </pre>

			"VParam:=" , <float>)
<xAxisPosArray>	Array	Structured array for relative position: Array("NAME:xAxisPos", "IsAttachedToEntity:=" , <boolean>, "EntityID:=" , <integer>, "FacetedBodyTriangleIndex:=" , <integer>, "TriangleVertexIndex:=" , <integer>, "PositionType:=" , <string "OnVertex", "EdgeCenter", "FaceCenter", or "OnEdge">, "UParam:=" , <float>, "VParam:=" , <float>, "XPosition:=" , <string>, "YPosition:=" , <string>, "ZPosition:=" , <string>)	
<yAxisArray>	Array	Structured array for absolute position: Array("NAME:yAxis", "DirectionType:=" , "AbsoluteDirection", "EdgeID:=" , <integer>, "FaceID:=" , <integer>, "xDirection:=" , <string>, "yDirection:=" , <string>,	

			"zDirection:=" , <string>, "UParam:=" , <float>, "VParam:=" , <float>)
	<yAxisPosArray>	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>)
	<AttributesArray>	Array	Structured array. See: AttributesArray .
Return Value	None.		

Python Syntax	CreateObjectCS(<Parameters>,<AttributesArray>)
Python Example	<pre>oEditor.CreateObjectCS(["NAME:ObjectCSParameters",</pre>

```
[ "NAME:Origin",  
  "IsAttachedToEntity:=" , True,  
  "EntityID:=" , 59,  
  "FacetedBodyTriangleIndex:=" , -1,  
  "TriangleVertexIndex:=" , -1,  
  "PositionType:=" , "OnVertex",  
  "UParam:=" , 0,  
  "VParam:=" , 0,  
  "XPosition:=" , "0",  
  "YPosition:=" , "0",  
  "ZPosition:=" , "0"  
],  
"MoveToEnd:=" , False,  
"ReverseXAxis:=" , False,  
"ReverseYAxis:=" , False,  
[ "NAME:xAxis",  
  "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"
]
)

```

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. <pre>Array("NAME:Parameters", Array("NAME:BodyFromEdgeToParameters", "Edges:=" , <array containing integer edges>))</pre>
	<CreateGroupsForNewObjects>	Array	Structured array. <pre>Array("CreateGroupsForNewObjects:=", <boolean True to create groups for new objects; else False>)</pre>
Return Value	None.		

Python Syntax	CreateObjectFromEdges(<SelectionsArray>, <ParametersArray>, <CreateGroupsForNewObjects>)
Python Example	<pre>oEditor.CreateObjectFromEdges(["NAME:Selections",</pre>

	<pre>"Selections:=", "Box2", "NewPartsModelFlag:=", "Model"], ["NAME:Parameters", ["NAME:BodyFromEdgeToParameters", "Edges:=", [41]]], ["CreateGroupsForNewObjects:=", False])</pre>

CreateObjectFromFaces

Creates 2D objects from specified face(s).

UI Access	Modeler > Surface > Create Object from Face		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: SelectionsArray .
	<Parameters>	Array	Structured array. Array ("NAME:Parameters", <BodyFromFaceToParameters>)
	<CreateGroupsForNewObjects>	Array	Structured array. Array ("CreateGroupsForNewObjects:=", <boolean>)
Return Value	None.		

Python Syntax	CreateObjectFromFaces (<SelectionsArray>,<Parameters>, <CreateGroupsForNewObjects>)
Python Example	<pre>oEditor.CreateObjectFromFaces (["NAME:Selections", "Selections:=" , "Box3", "NewPartsModelFlag:=" , "Model"], ["NAME:Parameters", ["NAME:BodyFromFaceToParameters", "FacesToDetach:=" , [68]]], ["CreateGroupsForNewObjects:=", False])</pre>

CreateRelativeCS

Creates a Relative Coordinate System.

UI Access	Modeler > Coordinate System > Create > Relative CS > [Offset / Rotated / Both].		
Parameters	Name	Type	Description
	<Parameters>	Array	Structured array.

			<pre>Array("NAME:RelativeCSParameters", "Mode:=" , "Axis/Position", "OriginX:=" , <string>, "OriginY:=" , <string>, "OriginZ:=" , <string>, "XAxisXvec:=" , <string>, "XAxisYvec:=" , <string>, "XAxisZvec:=" , <string>, "YAxisXvec:=" , <string>, "YAxisYvec:=" , <string>, "YAxisZvec:=" , <string>)</pre>
	<AttributesArray>	Array	Structured array. See: AttributesArray . CreateRelativeCS supports only the "Name" parameter.
Return Value	None.		

Python Syntax	CreateRelativeCS(<Parameters>,<AttributesArray>)
Python Example	<pre>oEditor.CreateRelativeCS(["NAME:RelativeCSParameters", "Mode:=" , "Axis/Position", "OriginX:=" , "0.62mm", "OriginY:=" , "-0.7mm", "OriginZ:=" , "0mm",</pre>

	<pre>"XAxisXvec:=" , "1mm", "XAxisYvec:=" , "0mm", "XAxisZvec:=" , "0mm", "YAxisXvec:=" , "0mm", "YAxisYvec:=" , "1mm", "YAxisZvec:=" , "0mm"], ["NAME:Attributes", "Name:=" , "RelativeCS1"])</pre>

DeleteEmptyGroups

Deletes group(s) from the history tree.

UI Access	Modeler > Group > Delete Empty.		
Parameters	Name	Type	Description
	<Parameters>	Array	Structured array. Array("Groups:=" , <array of string group IDs>)
Return Value	None.		
Python Syntax	DeleteEmptyGroups(<Parameters>)		

Python Example	<pre>oEditor.DeleteEmptyGroups(["Groups:=", ["Group1", "Group2", "Group3"]])</pre>

DeleteLastOperation

Deletes the last operation performed on the specified object(s).

UI Access	Modeler > Delete Last Operation.		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: SelectionsArray .
Return Value	None.		

Python Syntax	DeleteLastOperation(<SelectionsArray>)
Python Example	<pre>oEditor.DeleteLastOperation(["NAME:Selections", "Selections:=", "Box3", "NewPartsModelFlag:=", "Model"])</pre>

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>

	]

EditEntityList

Modifies an entity list.

Warning: As of the Ansys Electronics Desktop25r2 release, this command has been deprecated. It will be removed in a future release. Please use the [EditNamedSelection](#) command instead.

UI Access	Modeler > Named Selection > Reassign.		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: SelectionsArray .
	<Parameters>	Array	Structured array. Array ("NAME:GeometryEntityListParameters", "EntityType:=" , <string "Object" or "Face">, "EntityList:=" , <string list>)
Return Value	None.		

Python Syntax	EditEntityList(<SelectionsArray>, <Parameters>)
Python Example	<pre>oEditor.EditEntityList(["NAME:Selections", "Selections:=" , "Objectlist1"</pre>

	<pre>], ["NAME:GeometryEntityListParameters", "EntityType:=" , "Object", "EntityList:=" , "Box1, Box2, Box3"])</pre>

EditFaceCS

Recreates an existing face coordinate system. See: [CreateFaceCS](#).

UI Access	Modeler > Coordinate System > Edit.		
Parameters	Name	Type	Description
	<Parameters>	Array	Structured array. Array("NAME:FaceCSParameters", <OriginArray>, "MoveToEnd:=" , <boolean>, "FaceID:=" , <integer>, <AxisPosnArray>, "WhichAxis:=" , <string "X", "Y", or "Z">, "ZRotationAngle:=" , <string>, "XOffset:=" , <string>, "YOffset:=" , <string>,

		"AutoAxis:=" , <boolean>)
<OriginArray>	Array	Structured array. <pre>Array("NAME:Origin", "IsAttachedToEntity:=" , <boolean>, "EntityID:=" , <integer>, "FacetedBodyTriangleIndex:=", <integer>, "TriangleVertexIndex:=" , <integer>, "PositionType:=" , <string "FaceCenter", "EdgeCenter", "OnVertex", "OnEdge", or "OnFace">, "UParam:=" , <float between 0 and 1 representing the relative position of the point on the edge or face>, "VParam:=" , <float between 0 and 1 representing the relative position of the point on the edge or face>, "XPosition:=" , <string>, "YPosition:=" , <string>, "ZPosition:=" , <string>)</pre> IsAttachedToEntity specifies whether the point is anchored to a vertex, edge, or face. If True, provide UParam and VParam. If False, provide XPosition, YPosition, and ZPosition to provide fixed position. Pass "0" for unused parameters.
<AxisPosnArray>	Array	Structured array. <pre>Array("NAME:AxisPosn", "IsAttachedToEntity:=" , <boolean>,</pre>

			<pre>"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	<pre>oEditor.EditFaceCS (["NAME:FaceCSParameters", ["NAME:Origin", "IsAttachedToEntity:=" , True, "EntityID:=" , 12, "FacetedBodyTriangleIndex:=" , -1,</pre>

```

    "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"
  ]

```

	<pre>], "WhichAxis:=" , "X", "ZRotationAngle:=" , "0deg", "XOffset:=" , "0mm", "YOffset:=" , "0mm", "AutoAxis:=" , False], ["NAME:Attributes", "Name:=" , "FaceCS1", "PartName:=" , "Box1"])</pre>

EditNamedSelection

Modifies an entity list.

UI Access	Modeler > Named Selection > Reassign.		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: SelectionsArray .
	<Parameters>	Array	Structured array. Array ("NAME:NamedSelectionParameters", "Type:=" , <string "Object" or "Face"> ,

			"Selection:=" , <string of object names or IDs>))
Return Value	None.		

Python Syntax	EditNamedSelection(<SelectionsArray>, <Parameters>)
Python Example	<pre>oEditor.EditNamedSelection(["NAME:Selections", "Selections:=" , "FaceSelection1"], ["NAME:NamedSelectionParameters", "Type:=" , "Face", "Selection:=" , [12]]) </pre>

EditObjectCS

Edits an existing object coordinate system. See: [CreateObjectCS](#).

UI Access	Modeler > Coordinate System > Edit.		
Parameters	Name	Type	Description
	<Parameters>	Array	Structured array. Array("NAME:ObjectCSParameters", <OriginArray> "MoveToEnd:=" , <boolean>, "ReverseXAxis:=" , <boolean>,

		<pre>"ReverseYAxis:=" , <boolean>, <xAxisArray / xAxisPosArray> <yAxisArray / yAxisPosArray>)</pre> Note: xAxisArray and xAxisPosArray differ. Use xAxisArray for absolute position and xAxisPosArray for relative position. Do the same for yAxisArray and yAxisPosArray.
	<OriginArray>	Array Structured array. <pre>Array("NAME:Origin", "IsAttachedToEntity:=" , <boolean>, "EntityID:=" , <integer>, "FacetedBodyTriangleIndex:=", <integer>, "TriangleVertexIndex:=" , <integer>, "PositionType:=" , <string "OnVertex", "EdgeCenter", "FaceCenter", "OnEdge", or "AbsolutePosition">, "UParam:=" , <float between 0 and 1 representing the relative position of the point on the edge or face>, "VParam:=" , <float between 0 and 1 representing the relative position of the point on the edge or face>, "XPosition:=" , <string>, "YPosition:=" , <string>,</pre>

		<code>"ZPosition:=" , <string>)</code> <code>IsAttachedToEntity</code> 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.
<code><xAxisArray></code>	Array	Structured array for absolute position: <code>Array("NAME:xAxis",</code> <code>"DirectionType:=" , "AbsoluteDirection",</code> <code>"EdgeID:=" , <integer>,</code> <code>"FaceID:=" , <integer>,</code> <code>"xDirection:=" , <string>,</code> <code>"yDirection:=" , <string>,</code> <code>"zDirection:=" , <string>,</code> <code>"UParam:=" , <float>,</code> <code>"VParam:=" , <float>)</code>
<code><xAxisPosArray></code>	Array	Structured array for relative position: <code>Array("NAME:xAxisPos",</code> <code>"IsAttachedToEntity:=" , <boolean>,</code> <code>"EntityID:=" , <integer>,</code> <code>"FacetedBodyTriangleIndex:=" , <integer>,</code> <code>"TriangleVertexIndex:=" , <integer>,</code> <code>"PositionType:=" , <string "OnVertex",</code> <code>"EdgeCenter", "FaceCenter", or "OnEdge">,</code> <code>"UParam:=" , <float>,</code>

		<pre>"VParam:=" , <float>, "XPosition:=" , <string>, "YPosition:=" , <string>, "ZPosition:=" , <string></pre>
<code><yAxisArray></code>	Array	Structured array for absolute position: <pre>Array("NAME:yAxis", "DirectionType:=" , "AbsoluteDirection", "EdgeID:=" , <integer>, "FaceID:=" , <integer>, "xDirection:=" , <string>, "yDirection:=" , <string>, "zDirection:=" , <string>, "UParam:=" , <float>, "VParam:=" , <float>)</pre>
<code><yAxisPosArray></code>	Array	Structured array for relative position: <pre>Array("NAME:yAxisPos", "IsAttachedToEntity:=" , <boolean>, "EntityID:=" , <integer>, "FacetedBodyTriangleIndex:=" , <integer>, "TriangleVertexIndex:=" , <integer>, "PositionType:=" , <string "OnVertex",</pre>

			"EdgeCenter", "FaceCenter", or "OnEdge">, "UParam:=" , <float>, "VParam:=" , <float>, "XPosition:=" , <string>, "YPosition:=" , <string>, "ZPosition:=" , <string>)
	<AttributesArray>	Array	Structured array. See: AttributesArray . Use to select the coordinate system to edit.
Return Value	None.		

Python Syntax	EditObjectCS(<Parameters>,<AttributesArray>)
Python Example	<pre>oEditor.EditObjectCS(["NAME:ObjectCSParameters", ["NAME:Origin", "IsAttachedToEntity:=" , True, "EntityID:=" , 59, "FacetedBodyTriangleIndex:=", -1, "TriangleVertexIndex:=" , -1, "PositionType:=" , "OnVertex", "UParam:=" , 0, "VParam:=" , 0, "XPosition:=" , "0",</pre>

```
"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,
```

	<pre>"xDirection:=" , "0", "yDirection:=" , "1", "zDirection:=" , "0", "UParam:=" , 0, "VParam:=" , 0]], ["NAME:Attributes", "Name:=" , "ObjectCS1", "PartName:=" , "Box2"])</pre>

EditRelativeCS

Edits an existing Relative Coordinate System. See: [CreateRelativeCS](#).

UI Access	Modeler > Coordinate System > Edit.		
Parameters	Name	Type	Description
	<Parameters>	Array	Structured array. Array("NAME:RelativeCSParameters", "Mode:=" , "Axis/Position", "OriginX:=" , <string>, "OriginY:=" , <string>,

			"OriginZ:=" , <string>, "XAxisXvec:=" , <string>, "XAxisYvec:=" , <string>, "XAxisZvec:=" , <string>, "YAxisXvec:=" , <string>, "YAxisYvec:=" , <string>, "YAxisZvec:=" , <string>)
	<AttributesArray>	Array	Structured array. See: AttributesArray . Use to select the coordinate system to edit.
Return Value	None.		

Python Syntax	EditRelativeCS(<Parameters>,<AttributesArray>)		
Python Example	<pre>oEditor.EditRelativeCS(["NAME:RelativeCSParameters", "Mode:=" , "Axis/Position", "OriginX:=" , "0.62mm", "OriginY:=" , "-0.7mm", "OriginZ:=" , "0mm", "XAxisXvec:=" , "1mm", "XAxisYvec:=" , "0mm",</pre>		

	<pre>"XAxisZvec:=" , "0mm", "YAxisXvec:=" , "0mm", "YAxisYvec:=" , "1mm", "YAxisZvec:=" , "0mm"], ["NAME:Attributes", "Name:=" , "RelativeCS1"])</pre>

Export

Exports the model to a file.

Note: This script does not export image file types or GDSII files. See: [ExportModelImageToFile](#) and `ExportGDSII`.

UI Access	Modeler > Export		
Parameters	Name	Type	Description
	<Parameters>	Array	Structured array. Array("NAME:ExportParameters", "AllowRegionDependentPartSelectionForPMLCreation:=", <boolean>, "AllowRegionSelectionForPMLCreation:=", <boolean>, "Selections:=" , <string list>,

			"File Name:=" , <string filepath>, "Major Version:=" , <integer (-1 if not applicable)>, "Minor Version:=" , <integer (-1 if not applicable)>)
Return Value	None.		

Python Syntax	Export(<Parameters>)
Python Example	<pre>oEditor.Export(["NAME:ExportParameters", "AllowRegionDependentPartSelectionForPMLCreation:=", True, "AllowRegionSelectionForPMLCreation:=", True, "Selections:=" , "Box1,Box2,Box3", "File Name:=" , "C:/Users/jdoe/Desktop/export.sab", "Major Version:=" , 25, "Minor Version:=" , 0])</pre>

ExportModellImageToFile

Exports the model as an image file (*.avz, *.bmp, *.gif, *.jpeg, *.tiff, *.wrl). In Release 23.1, this command is fully supports -ng (non-graphical) mode. To export to Ensign use *.avz. For export to Ensign in -ng mode, the corresponding version of Ensign must be installed. On Linux, it might need manual set environment variable AWP_ROOT212 to its installation path, e.g. AWP_ROOT212-2=/installations/ansys_inc/v212/ for AnsysEDT v21.2 and Ensign 21.2.

ExportModellImageToFile exports a model image with background type and color that respect the AEDT color scheme by default. You can specify the background type and color with following parameters:

Parameter Name	Description	Parameter Value
BackgroundType	Choose one out of four types of background	Default Plain LinearGradient RadialGradient
BackgroundColor	Plain: Background color Lin- earGradient/RadialGradient: Background start color	3 integers with a range of [0,255] that represent red, green, and blue respectively
BackgroundContrastColor	Plain: Ignored LinearGradient/RadialGradient: Back- ground end color	3 integers with a range of [0,255] that represent red, green, and blue respectively

Note: Current scripts will not be affected, the image will be exported with default background color as it always does

If no BackgroundType is specified, or BackgroundType is Default, BackgroundColor and BackgroundContrastColor will be ignored, and the image will be exported with default background color

ExportModellImageToFile supports export overlay of polar plot 3D with existing transformation (scaling, rotation and translation) in -ng (non-graphical) mode.

UI Access	Modeler > Export.		
Parameters	Name	Type	Description
	<path>	String	Full file path including extension.

	<width>	Integer	Width in pixels (use 0 for default).
	<height>	Integer	Height in pixels (use 0 for default).
	<Parameters>	Array	Structured array. Array ("NAME:SaveImageParams", "ShowAxis:=" , <string containing boolean>, "ShowGrid:=" , <string containing boolean>, "ShowRuler:=" , <string containing boolean>, "ShowRegion:=" , <string>, "Selections:=" , <string>, "FieldPlotSelections:=" , <string>' # Comma delimited string. #Use to set which field plot to show. "FitToSelections:=" , "" , Note: "FitToSelections" specify geometry objects for the "Fit" operation. "FitToFieldPlotSelection:=" , "" Note: "FitToFieldPlotSelections" specifies field plots for the "Fit" operation.

			"AutoFit:=" , "True", Note: If FitToSlections or FitToFieldPlotSelections are used , then AutoFit is True, it makes sure color key does not overlap field plot. It is False by default. If neither is used, then "AutoFit" will "Fit" to full model. "Orientation:=" , <string> "ShowOrientationGadget:=" , <False>
Return Value	None.		

Python Syntax	ExportModelImageToFile(<path> <width> <height> <Parameters>)
Python Example	<pre>oEditor.ExportModelImageToFile ("D:/Image.png", 1920, 1080, ["NAME:SaveImageParams", "ShowAxis:=" , "True", "ShowGrid:=" , "True", "ShowRuler:=" , "True", "ShowRegion:=" , "Default", "Selections:=" , "", "FieldPlotSelections:=" , "",</pre>

	<pre>"FitToSelections:=" , "" , "FitToFieldPlotSelections:=" , "" , "AutoFit:=" , "True", "Orientation:=" , ""])</pre>

Fillet

Performs a fillet on specified edge(s).

UI Access	Modeler > Fillet.		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: SelectionsArray .
	<Parameters>	Array	Structured array. Array("NAME:Parameters", <FilletParametersArray>)
	<FilletParametersArray>	Array	Structured array. Array("NAME:FilletParameters", "Edges:=" , <array containing integer edge IDs>, "Vertices:=" , <empty array>, "Radius:=" , <string>, "Setback:=" , <string>)

Return Value	None.

Python Syntax	Fillet(<SelectionsArray>, <Parameters>)
Python Example	<pre> oEditor.Fillet(["NAME:Selections", "Selections:=" , "Box1", "NewPartsModelFlag:=" , "Model"], ["NAME:Parameters", ["NAME:FilletParameters", "Edges:=" , [13], "Vertices:=" , [], "Radius:=" , "1mm", "Setback:=" , "0mm"]]) </pre>

FlattenGroup

Flattens a specified history tree group.

UI Access	Modeler > Group > Flatten.		
Parameters	Name	Type	Description

	<table><tr><td><GroupID></td><td>Array</td><td>Structured array. Array("Groups:=", Array(<string list of group IDs>))</td></tr></table>	<GroupID>	Array	Structured array. Array("Groups:=", Array(<string list of group IDs>))
<GroupID>	Array	Structured array. Array("Groups:=", Array(<string list of group IDs>))		
Return Value	None.			

Python Syntax	FlattenGroup (<GroupID>)
Python Example	<code>oEditor.FlattenGroup(["Groups:=", ["Group1"]])</code>

GenerateHistory

Generates the history for specified 1D object(s).

UI Access	Modeler > Generate History.		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: SelectionsArray .
Return Value	None.		

Python Syntax	GenerateHistory (<SelectionsArray>)
Python Example	<code>oEditor.GenerateHistory(["NAME:Selections",</code>

	<pre> "Selections:=" , "Polyline1", "NewPartsModelFlag:=" , "Model", "UseCurrentCS:=" , True 1) </pre>

HealObject

Heals an imported object.

UI Access	Modeler > Model Preparation > Heal.		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: SelectionsArray .
	<Parameters>	Array	Structured array.
			<pre> Array("NAME:ObjectHealingParameters", "Version:=" , <integer>, "AutoHeal:=" , <boolean>, "TolerantStitch:=" , <boolean>, "SimplifyGeom:=" , <boolean>, "TightenGaps:=" , <boolean>, "HealToSolid:=" , <boolean>, "StopAfterFirstStitchError:=" , <boolean>, "MaxStitchTol:=" , <float>, </pre>

		<pre>"ExplodeAndStitch:=" , <boolean>, "GeomSimplificationTol:=" , <integer>, "MaximumGeneratedRadiusForSimplification:=" , <integer>, "SimplifyType:=" , <integer>, "TightenGapsWidth:=" , <float>, "RemoveSliverFaces:=" , <boolean>, "RemoveSmallEdges:=" , <boolean>, "RemoveSmallFaces:=" , <boolean>, "SliverFaceTol:=" , <integer>, "SmallEdgeTol:=" , <integer>, "SmallFaceAreaTol:=" , <integer>, "BoundingBoxScaleFactor:=" , <integer>, "RemoveHoles:=" , <boolean>, "RemoveChamfers:=" , <boolean>, "RemoveBlends:=" , <boolean>, "HoleRadiusTol:=" , <integer>, "ChamferWidthTol:=" , <integer>, "BlendRadiusTol:=" , <integer>, "AllowableSurfaceAreaChange:=" , <integer>,</pre>

			"AllowableVolumeChange:=", <integer>)
Return Value	None.		

Python Syntax	HealObject(<SelectionsArray>, <Parameters>)
Python Example	<pre>oEditor.HealObject(["NAME:Selections", "Selections:=" , "Box1", "NewPartsModelFlag:=" , "Model"], ["NAME:ObjectHealingParameters", "Version:=" , 1, "AutoHeal:=" , True, "TolerantStitch:=" , True, "SimplifyGeom:=" , True, "TightenGaps:=" , True, "HealToSolid:=" , False, "StopAfterFirstStitchError:=", False, "MaxStitchTol:=" , 0.001, "ExplodeAndStitch:=" , True, "GeomSimplificationTol:=", -1, "MaximumGeneratedRadiusForSimplification:=", -1,</pre>

```
"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  
])
```

GetActiveCoordinateSystem

Returns the active coordinate system.

UI Access	None.
Parameters	None.
Return Value	String name of active coordinate system.

Python Syntax	<code>GetActiveCoordinateSystem()</code>
Python Example	<code>oEditor.GetActiveCoordinateSystem()</code>

GetActiveCoordinateSystemTransform

Returns transformation information for the active coordinate system (with respect to the global coordinate system), which consists of the affine matrix for rotation of the coordinate system and a 3D vector with the translation of the coordinate system's origin with respect to the global coordinate system's origin.

UI Access	None
Parameters	None
Return Value	Array of strings containing transformation information

Python Syntax	<code>GetActiveCoordinateSystemTransform()</code>
Python Example	<code>oEditor.GetActiveCoordinateSystemTransform()</code>

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>

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>,

			<pre> "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> </pre>
Return Value	None.		

Python Syntax	<code>Import(<Parameters>)</code>
Python Example	<pre> oEditor.Import (["NAME:NativeBodyParameters", "HealOption:=" , 0, "Options:=" , "-1", "FileType:=" , "UnRecognized", </pre>

	<pre>"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, "SourceFile:=" , "C:\\Users\\jdoe\\Desktop\\export.model"])</pre>

ImportDXF [Modeler]

Imports a DXF model file.

UI Access	Modeler > Import.

Parameters	Name	Type	Description
	<Parameters>	Array	Structured array. <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>
	<layersArray>	Array	Structured array. <pre>Array("NAME:LayerInfo", <layer>, <layer>, <layer>, ...)</pre>
	<layer>	Array	Structured array. <pre>Array("NAME:<layerName>", "source:=" , <string>, "display_source:=" , <string>,</pre>

			<code>"import:=" , <boolean>, "dest:=" , <string>, "dest_selected:=" , <boolean>, "layer_type:=" , <string>)</code>
Return Value	None.		

Python Syntax	ImportDXF(<Parameters>)
Python Example	<pre>oEditor.ImportDXF(["NAME:options", "FileName:=", "C:/Users/jdoe/Desktop/export.dxf", "Scale:=" , 0.001, "AutoDetectClosed:=" , True, "SelfStitch:=" , True, "DefeatureGeometry:=" , False, "DefeatureDistance:=" , 0, "RoundCoordinates:=" , False, "RoundNumDigits:=" , 4, "WritePolyWithWidthAsFilledPoly:=", False, "ImportMethod:=" , 1, "2DSheetBodies:=" , False,</pre>

```
[ "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,
```

	<pre>"dest:=" , "LAYER_2", "dest_selected:=" , False, "layer_type:=" , "signal"]]])</pre>

ImportGDSII [Modeler]

Imports a GDSII model file.

UI Access	Modeler > Import.		
Parameters	Name	Type	Description
	<Parameters>	Array	Structured array. Array("NAME:options", "FileName:=" , <string>, "FlattenHierarchy:=" , <boolean>, "ImportMethod:=" , <integer>, <layerMapArray>, <layerMapArray>, <layerMapArray>,... "OrderMap:=" , ["entry:=" , <entry>, <entry>, <entry>,...

			]
			])
	<layerMapArray>	Array	Structured array. Array("NAME:LayerMapInfo", "LayerNum:=" , <integer>, "DestLayer:=" , <string>, "layer_type:=" , <string>)
	<entry>	Array	Structured array. Array("order:=" , <integer LayerNum>, "layer:=" , <string DestLayer>)
Return Value	None.		

Python Syntax	ImportGDSII(<Parameters>)
Python Example	<pre>oEditor.ImportGDSII(["NAME:options", "FileName:=", "C:/Users/coil2.gds", "FlattenHierarchy:=" , True, "ImportMethod:=" , 1, ["NAME:LayerMap", ["NAME:LayerMapInfo", "LayerNum:=" , 12, "DestLayer:=" , "Signal12",</pre>

```
"layer_type:="          , "signal"
],
["NAME:LayerMapInfo",
"LayerNum:="            , 13,
"DestLayer:="            , "Signal13",
"layer_type:="            , "signal"
],
["NAME:LayerMapInfo",
"LayerNum:="            , 14,
"DestLayer:="            , "Signal14",
"layer_type:="            , "signal"
]
],
"OrderMap:=",
[
"entry:=",
["order:=", 0, "layer:=", "Signal12"],
"entry:=",
["order:=", 1, "layer:=", "Signal13"],
"entry:=",
```

```

        ["order:=", 2, "layer:=", "Signal14"]
    ]
]
)

```

Intersect

Intersects specified objects.

UI Access	Modeler > Boolean > Intersect.		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: SelectionsArray .
	<Parameters>	Array	Structured array. Array("NAME:IntersectParameters", "KeepOriginals:=" , <boolean True to keep original objects; False to delete>)
Return Value	None.		

Python Syntax	Intersect(<SelectionsArray>, <Parameters>)
Python Example	<pre> oEditor.Intersect(["NAME:Selections", "Selections:=", "Rectangle1,Rectangle2"], ["NAME:IntersectParameters", "KeepOriginals:=", False]) </pre>

	])

MoveCStoEnd

Moves a specified Object Coordinate System to the end of the History tree.

UI Access	Modeler > Coordinate System > Move CS to End.		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: SelectionsArray .
Return Value	None.		

Python Syntax	MoveCStoEnd (<SelectionsArray>)
Python Example	<pre>oEditor.MoveCStoEnd(["NAME:Selections", "Selections:=" , "ObjectCS1"])</pre>

MoveEntityToGroup

Moves a specified entity or entities to a specified group.

UI Access	Drag item into the group in the history tree.

Parameters	Name	Type	Description
	<Objects>	Array	Structured array. Selects the entity/entities to move. Array("Objects:=" , <array containing string object IDs>)
	<MoveEntityToGroup>	Array	Structured array. Array("ParentGroup:=" , <string group name>)
Return Value	None.		

Python Syntax	MoveEntityToGroup (<Objects>, <MoveEntityToGroup>)
Python Example	<pre>oEditor.MoveEntityToGroup(["Objects:=", ["Box3"]], ["ParentGroup:=", "Box_Group"])</pre>

MoveFaces

Moves the specified faces along normal or along a vector.

UI Access	Modeler > Surface > Move Faces > Along [Normal/Vector].		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: SelectionsArray .
	<MoveFacesParametersArray>	Array	Structured array. Array("NAME:Parameters", <FacesOfOneObjToMove>, <FacesOfOneObjToMove>, ...)

	<i><FacesOfOneObjToMove></i>	Array	Structured array. <pre>Array("Name:MoveFacesParameters", "MoveAlongNormalFlag:=", <boolean>, "OffsetDistance:=", <string>, "MoveVectorX:=", <string>, "MoveVectorY:=", <string>, "MoveVectorZ:=", <string>, "FacesToMove:=", <array>)</pre> <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. <code>FacesToMove</code> is an array of integers (the IDs of the faces to move).
Return Value	None		

Python Syntax	MoveFaces (<SelectionsArray>, <MoveFacesParametersArray>)
Python Example	<pre>oEditor.MoveFaces(["NAME:Selections", "Selections:=", , "Rectangle1", "NewPartsModelFlag:=", , "Model"],</pre>

	<pre>["NAME:Parameters", ["NAME:MoveFacesParameters", "MoveAlongNormalFlag:=", True, "OffsetDistance:=", "1mm", "MoveVectorX:=", "0mm", "MoveVectorY:=", "0mm", "MoveVectorZ:=", "0mm", "FacesToMove:=", [183]]]</pre>

ProjectSheet

Project a sheet object, typically for modeling thin conformal deposits. Typically followed by **Thicken Sheet**.

UI Access	Click Modeler > Surface > Project Sheet .		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: SelectionsArray .
	<Parameters>	Array	Array containing string. Array ("NAME:ProjectSheetParameters")
Return Value	None.		

Python Syntax	ProjectSheet (<SelectionsArray>, <Parameters>)
Python Example	<pre>oEditor.ProjectSheet (['NAME:Selections',</pre>

	<pre>'Selections:='', 'Box1,Box2,Polyline1'], ['NAME:ProjectSheetParameters'])</pre>

PurgeHistory

Purges the history of a specified object.

UI Access	Modeler > Purge History.		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: SelectionsArray .
Return Value	None.		

Python Syntax	PurgeHistory (<SelectionsArray>)
Python Example	<pre>oEditor.PurgeHistory(["NAME:Selections", "Selections:=", "Box2", "NewPartsModelFlag:=", "Model"])</pre>

Section

Creates a 2D cross-section of the selection in the specified plane.

UI Access	Modeler > Surface > Section.		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: SelectionsArray .
	<Parameters>	Array	Structured array. <pre>Array("NAME:SectionToParameters", "CreateNewObjects:=" , <boolean>, "SectionPlane:=" , <string "XY", "YZ", or "ZX">, "SectionCrossObject:=" , <boolean>)</pre>
Return Value	None.		

Python Syntax	Section (<SelectionsArray>, <Parameters>)
Python Example	<pre>oEditor.Section(["NAME:Selections", "Selections:=" , "Cone1", "NewPartsModelFlag:=" , "Model"], ["NAME:SectionToParameters", "CreateNewObjects:=" , True,</pre>

	<pre>"SectionPlane:=" , "XY", "SectionCrossObject:=" , False])</pre>

SeparateBody

Separates the bodies of specified multi-lump objects.

UI Access	Modeler > Boolean > Separate Bodies.		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: SelectionsArray .
Return Value	None.		

Python Syntax	SeparateBody (<SelectionsArray>)
Python Example	<pre>oEditor.SeparateBody(["NAME:Selections", "Selections:=", "Rectangle1,Circle1", "NewPartsModelFlag:=", "Model"])</pre>

SetModelUnits

Sets the model units.

UI Access	Modeler > Units.		
Parameters	Name	Type	Description
	<Parameters>	Array	Structured array. <pre>Array("NAME:Units Parameter", "Units:=", <string>, "Rescale:=", <boolean True to rescale model; else False>)</pre> To see valid unit strings, select Modeler > Units .
Return Value	None.		

Python Syntax	SetModelUnits (<Parameters>)
Python Example	<pre>oEditor.SetModelUnits(["NAME:Units Parameter", "Units:=", "km", "Rescale:=", False])</pre>

SetWCS

Sets the working coordinate system.

UI Access	Modeler > Coordinate System > Set Working CS.		
Parameters	Name	Type	Description
	<Parameters>	Array	Structured array. Array("NAME:SetWCS Parameter", "Working Coordinate System:=", <string CS name>, "RegionDepCSOk:=" , <boolean True if region-dependent; else False)
Return Value	None.		

Python Syntax	SetWCS (<Parameters>)
Python Example	<pre>oEditor.SetWCS(["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>

Split

Splits the specified object(s) along a plane.

UI Access	Modeler > Boolean > Split.		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: SelectionsArray .
	<Parameters>	Array	Structured array.
			<pre>Array("NAME:SplitToParameters", "SplitPlane:=" , <string "XY", "YZ", "ZX", or "Dummy">, "WhichSide:=" , <string "PositiveOnly", "Neg- ativeOnly" or "Both">, "ToolType:=" , <string "PlaneTool" or "FaceTool">,</pre>

			<pre>"ToolEntityID:=" , <-1 if using SplitPlane; else FaceID>, "SplitCrossingObjectsOnly:=", <boolean>, >DeleteInvalidObjects:=", <boolean>)</pre> You can split an object either along an existing plane , or by creating a plane using an edge. For existing plane: • SplitPlane is "XY", "YZ", or "ZX" • ToolType is "PlaneTool" • ToolEntityID is -1 Possible values for 2D XY designs are "YZ", "ZX". Possible values for 2D RZ designs are "XY", "YZ". For an edge: • SplitPlane is "Dummy" • ToolType is "EdgeTool" • ToolEntityID is the face ID
Return Value	None.		

Python Syntax	<code>Split(<SelectionsArray>, <Parameters>)</code>
Python Example	<pre>oEditor.Split(["NAME:Selections",</pre>

	<pre>"Selections:=" , "Box1", "NewPartsModelFlag:=" , "Model"], ["NAME:SplitToParameters", "SplitPlane:=" , "XY", "WhichSide:=" , "PositiveOnly", "ToolType:=" , "PlaneTool", "ToolEntityID:=" , -1, "SplitCrossingObjectsOnly:=", False, "DeleteInvalidObjects:=", True])</pre>

Subtract

Subtracts the specified object(s).

UI Access	Modeler > Boolean > Subtract.		
Parameters	Name	Type	Description
	<Selections>	Array	Structured array. Array("NAME:Selections", "Blank Parts:=" , <string object name(s)>, "Tool Parts:=" , <string object name(s)> The Tool Parts object is the object being removed.

			The Blank Parts object has any Tool Parts overlap removed. Either string can contain more than one object.
	<Parameters>	Array	Structured array. Array("NAME:SubtractParameters", "KeepOriginals:=" , <boolean>)
Return Value	None.		

Python Syntax	Subtract(<Selections>, <Parameters>)
Python Example	<pre>oEditor.Subtract(["NAME:Selections", "Blank Parts:=", "Rectangle1", "Tool Parts:=", "Rectangle2"], ["NAME:SubtractParameters", "KeepOriginals:=", False])</pre>

SweepFacesAlongNormal

Sweep the specified face(s) along normal.

UI Access	Modeler > Surface > Sweep Faces Along Normal

Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: SelectionsArray .
	<parameters>	Array	Structured array. Array("NAME:Parameters", "NAME:SweepFaceAlongNormalToParameters", "FacesToDetach:=", <faceIDarray>, "LengthOfSweep:=", "<value><units>")
Return Value	None		

Python Syntax	SweepFacesAlongNormal(<SelectionsArray> <parameters>)
Python Example	<pre>oEditor.SweepFacesAlongNormal(["NAME:Selections", "Selections:=", "Rectangle1", "NewPartsModelFlag:=", "Model"], ["NAME:Parameters", "NAME:SweepFaceAlongNormalToParameters", "FacesToDetach:=", [183], "LengthOfSweep:=", "0.1mm"])</pre>

ThickenSheet

Thickens a sheet object to convert it to a 3D object.

UI Access	Modeler > Surface > Thicken Sheet		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: SelectionsArray .
	<parameters>	Array	Structured array. Array("NAME:SheetThickenParameters", "Thickness:=" , <string>, "BothSides:=" , <boolean>)
Return Value	None.		

Python Syntax	ThickenSheet (<SelectionsArray> <parameters>)
Python Example	<pre>oEditor.ThickenSheet(["NAME:Selections", "Selections:=" , "Rectangle3", "NewPartsModelFlag:=" , "Model"], ["NAME:SheetThickenParameters", "Thickness:=" , "0.01mm",</pre>

	<code>"BothSides:=" , False</code> <code>l)</code>

UncoverFaces

Uncovers the specified face(s).

UI Access	Modeler > Surface > Uncover Faces		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: SelectionsArray .
	<parameters>	Array	Structured array. <pre>Array("NAME:Parameters", Array("NAME:UncoverFacesParameters", "FacesToUncover:=" , <array of face IDs>))</pre>
Return Value	None.		

Python Syntax	UncoverFaces(<SelectionsArray> <parameters>)
Python Example	<pre>oEditor.UncoverFaces(["NAME:Selections", "Selections:=" , "Box1", "NewPartsModelFlag:=" , "Model"],</pre>

	<pre>["NAME:Parameters", ["NAME:UncoverFacesParameters", "FacesToUncover:=" , [12,16,18]]]</pre>

Unite

Unites the specified objects.

UI Access	Modeler > Boolean > Unite		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: SelectionsArray .
	<parameters>	Array	Structured array. Array("NAME:UniteParameters", "KeepOriginals:=" , <boolean>, "TurnOnNBodyBoolean:=", <boolean>)
			Note: TurnOnNBodyBoolean is set to True by default. Enabled by Parasolid, this unites <i>n</i> bodies in a single step, rather than uniting them one by one.

Return Value	None.

Python Syntax	<code>Unite(<SelectionsArray>, <parameters>)</code>
Python Example	<pre>oEditor.Unite(["NAME:Selections", "Selections:=", "Rectangle1,Rectangle2"], ["NAME:UniteParameters", "KeepOriginals:=", False, "TurnOnNBodyBoolean:=", True])</pre>

Ungroup

Ungroups a specified history tree group.

UI Access	Modeler > Group > Ungroup		
Parameters	Name	Type	Description
	<Groups>	Array	Structured array. <code>Array("Groups:=", <array of group names to ungroup>)</code>
Return Value	None		

Python Syntax	Ungroup(<Groups>)
Python Example	<code>oEditor.Ungroup(["Groups:=", ["Group1"]])</code>

WrapSheet

Wraps a sheet object to another object.

UI Access	None.		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: SelectionsArray .
	<Parameters>	Array	Structured array. <code>Array("NAME:WrapSheetParameters", "Imprinted:=", <boolean>)</code>
Return Value	None.		

Python Syntax	WrapSheet (<SelectionsArray>, <Parameters>)
Python Example	<pre>oEditor.WrapSheet(["NAME:Selections", "Selections:=", "Rectangle1, Box1"], ["NAME:WrapSheetParameters",</pre>

	<pre>"Imprinted:=", True])</pre>

Other oEditor Commands

[AddDefinitionFromBlock](#)

[AddDefinitionFromLibFile](#)

[AddViewOrientation](#)

[BreakUDMConnection](#)

[ChangeProperty](#)

[Delete](#)

[FitAll](#)

[GetBodyNamesByPosition](#)

[GetChildNames \[Modeler\]](#)

[GetChildObject \[Modeler\]](#)

[GetChildTypes \[Modeler\]](#)

[GetEdgeByPosition](#)

[GetEdgeIDsFromObject](#)

[GetEdgeIDsFromFace](#)

[GetEntityListIDByName](#)

[GetExtendedDefinitionObject](#)

[GetFaceArea](#)

[GetFaceByPosition](#)

[GetFaceCenter](#)

[GetFaceIDs](#)

[GetGeometryModelerMode](#)

[GetMatchedObjectName](#)

[GetModelBoundingBox](#)

[GetModelUnits](#)

[GetNumObjects](#)

[GetObjectIDByName](#)

[GetObjectName](#)

[GetObjectNameByFaceID](#)

[GetObjectsByMaterial](#)

[GetObjectsInGroup](#)

[GetObjectVolume](#)

[GetPropertyValue](#)

[GetPropEvaluatedValue](#)

[GetPropNames](#)

[GetPropSIValue](#)

[GetPropValue](#)

[GetSelections](#)[GetUserPosition](#)[GetVertexIDsFromEdge](#)[GetVertexIDsFromFace](#)[GetVertexIDsFromObject](#)[GetVertexPosition](#)[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
	<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(["NAME:BoxParameters", "XPosition:=" , "-0.4mm", "YPosition:=" , "-1mm", "ZPosition:=" , "0mm", "XSize:=" , "1.4mm", "YSize:=" , "1.6mm",</pre>

```

        "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") oEditor.CreateBox(["NAME:BoxParameters", "XPosition:=" , "-0.4mm", "YPosition:=" , "-1mm", "ZPosition:=" , "0mm", "XSize:=" , "1.4mm", "YSize:=" , "1.6mm", "ZSize:=" , "0.6mm"], ["NAME:Attributes",</pre>

```
"Name:="                                , "Box1",

"Flags:="                                , "",

"Color:="                                , "(143 175 143)",

"Transparency:="                          , 0,

"PartCoordinateSystem:=", "Global",

"UDMId:="                                , "",

"MaterialValue:="                          , "\"vacuum\"",

"SurfaceMaterialValue:=", "\"\"",

"SolveInside:="                          , True,

"ShellElement:="                          , False,

"ShellElementThickness:=", "0mm",

"IsMaterialEditable:="                    , True,

"UseMaterialAppearance:=", False,
```

```
        "IsLightweight:="          , False

    ])

oDefinitionManager = oProject.GetDefinitionManager()
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
	<orientationName>	String	Name of the new orientation.
	<isGlobalOrientation>	Long	1 to save global; else 0.
	<OrientationVectorComponents>	Array	Structured array. Use <i>either</i> <OrientationVectorComponents> <i>or</i> <OrientationAnglesArray>. <pre>Array("NAME:OrientationVectorComponents", Array("NAME:Direction", <float>, <float>, <float>, Array("NAME:Up", <float>, <float>, <float>))</pre>
	<OrientationAnglesArray>	Array	Structured array. Use <i>either</i> <OrientationVectorComponents> <i>or</i> <OrientationAnglesArray>. <pre>Array("NAME:OrientationAngles", "Psi:=" , <float>,</pre>

		<pre>"Theta:=" , <float>, "Phi:=" , <float>)</pre>
<code><ProjectionParams></code>	Array	<i>Optional.</i> Structured array containing arrays of four optional parameters: Scaling, Translation, Projection Type, and Projection Limits. <pre>Array("NAME:ProjectionParams", Array("NAME:Scaling", <float>, <float>, <float>), Array("NAME:Translation", <float>, <float>, <float>), "ProjectionType:=" , <string "Orthographic" or "Perspective">, Array("NAME:ProjectionLimits", <float>, <float>,</pre>

	<pre><float>, <float>, <float>, <float>))</pre>
Return Value	None.

Python Syntax	AddViewOrientation(<orientationName>, <isGlobalOrientation>, <OrientationVectorComponents or OrientationAnglesArray>, <ProjectionParams>)
Python Example	Example Using Angles and Projection Parameters: <pre>oEditor.AddViewOrientation("orientation3", 0, ["NAME:OrientationAngles", "Psi:=" , -161.565002441406, "Theta:=" , 32.3115005493164, "Phi:=" , 0], ["NAME:ProjectionParams", ["NAME:Scaling", 0.591607987880707,</pre>

```
0.591607987880707,  
  
0.591607928276062  
],  
  
["NAME:Translation",  
  
0,  
2.38418579101563E-07,  
-5.7486834526062  
],  
"ProjectionType:=", "Orthographic",  
  
["NAME:ProjectionLimits",  
  
-2.44362282752991,  
2.44362282752991,  
  
-1,  
  
1,  
0.625207424163818,  
10.8721580505371
```

	<pre>]]) 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]])</pre>

BreakUDMConnection

Breaks a current UDM connection to Discovery.

UI Access	Break Connection.

Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: SelectionsArray .
Return Value	None.		

Python Syntax	BreakUDMConnection (<SelectionsArray>)
Python Example	<pre>oEditor.BreakUDMConnection(["NAME:Selections", "Selections:=" , "Discovery1"])</pre>

ChangeProperty

Changes the properties of an object in the history tree.

UI Access	Right-click an object in the History Tree and select Properties .		
Parameters	Name	Type	Description
	<propertyArgs>	Array	Structured array. The properties vary depending on the object. Due to the number of potential configurations, it is recommended that you generate this script using the UI's Automation tab.
Return Value	None.		

Python Syntax	ChangeProperty(<propertyArgs>)
Python Example	Example: Changing the Position of a Box <pre>oEditor.ChangeProperty(["NAME:AllTabs", ["NAME:Geometry3DCmdTab", ["NAME:PropServers" , "Box1:CreateBox:1"], ["NAME:ChangedProps", ["NAME:Position", "X:=" , "0.35in", "Y:=" , "0.55in", "Z:=" , "0in"]]]])</pre> Example: Offset the Position of a 3D Component or Layout Component with a Variable <pre>oDesign.ChangeProperty(["NAME:AllTabs", ["NAME:LocalVariableTab", ["NAME:PropServers", </pre>

```
        "LocalVariables"
    ],
    [
        "NAME:NewProps",
        [
            "NAME:zH",
            "PropType:="          , "VariableProp",
            "UserDef:="          , True,
            "Value:="            , "50mm"
        ]
    ]
]
])
oEditor = oDesign.SetActiveEditor("3D Modeler")
oEditor.ChangeProperty(
[
    "NAME:AllTabs",
    [
        "NAME:General",
        [
            "NAME:PropServers",
            "LC1_1"
        ],
        [
            "NAME:ChangedProps",
            [
                "NAME:Position",
                "X:="          , "0mm",
                "Y:="          , "0mm",
                "Z:="          , "zH"
            ]
        ]
    ]
]
```

```
])
```

Example: Changing a Box's Material and Wireframe Display

```
oEditor.ChangeProperty(
[
  "NAME:AllTabs",
  [
    "NAME:Geometry3DAttributeTab",
    [
      "NAME:PropServers" ,
      "Box1"
    ],
    [
      "NAME:ChangedProps",
      [
        "NAME:Material",
        "Value:=" , "\"vacuum\""
      ],
      [
        "NAME:Display Wireframe",
        "Value:=" , True
      ]
    ]
  ]
])
```

Example: Change Default Handle (reference coordinate system) for a 3D Component or Layout Component

```
oEditor.ChangeProperty(
[
  "NAME:AllTabs",
  [
    "NAME:General",
    [
```

```
        "NAME:PropServers",
        "LC1_1"
    ],
    [
        "NAME:ChangedProps",
        [
            "NAME:Handle",
            "Value:="          , " NotchOutY"
        ]
    ]
]
])
```

Example: Changing the Model Depth

```
oEditor.ChangeProperty(
[
    "NAME:AllTabs",
    [
        "NAME:Maxwell2D",
        [
            "NAME:PropServers",
            "Design Settings"
        ],
        [
            "NAME:ChangedProps",
            [
                "NAME:Model settings/Depth",
                "Value:="          , "2.4mm"
            ]
        ]
    ]
]
```



```
])
```

Example: Changing 2D Transient Use of Band Mapping Angle

```
oEditor.ChangeProperty(
[
  "NAME:AllTabs",
  [
    "NAME:MeshSetupTab",
    [
      "NAME:PropServers",
      "MeshSetup:CylindricalGap1"
    ],
    [
      "NAME:ChangedProps",
      [
        "NAME:Use Band Mapping Angle",
        "Value:="          , True
      ]
    ]
  ]
])
```

```
oEditor.ChangeProperty(
[
  "NAME:AllTabs",
  [
    "NAME:MeshSetupTab",
    [
      "NAME:PropServers",
      "MeshSetup:CylindricalGap1"
    ],
    [
      "NAME:ChangedProps",
      [
```

```

        "NAME:Band Mapping Angle",
        "Value:="          , "2deg"          ]
    ]
]
])

```

Example: Edit VirtualPort Excitation by changing component properties

```

oEditor.ChangeProperty(
[
    "NAME:AllTabs",
    [
        "NAME:Component Data",
        [
            "NAME:PropServers",
            "BoundarySetup:LC1_1_Port1"
        ],
        [
            "NAME:ChangedProps",
            [
                "NAME:Voltage",
                "Value:="          , "10mV"          ]
            ]
        ]
    ]
]
)
...

```

Example: Change Material Import for a Discovery Model

```

oEditor = oDesign.SetActiveEditor("3D Modeler")
oEditor.ChangeProperty(
[
    "NAME:AllTabs",
    [

```

```

        "NAME:Options",
        [
            "NAME:PropServers",
            "Discovery1"
        ],
        [
            "NAME:ChangedProps",
            [
                "NAME:Materials",
                "Value:=", "Assignments and Properties"
            ]
        ]
    ]
]
)

```

Delete

Deletes the specified object(s).

UI Access	Edit > Delete.		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: SelectionsArray .
Return Value	None.		

Python Syntax	Delete(<SelectionsArray>)
Python Example	<pre> oEditor.Delete(["NAME:Selections", "Selections:=", "Rectangle1,Rectangle2" </pre>

	])

GetBodyNamesByPosition

Returns the names of objects that contact a specified point.

UI Access	N/A		
Parameters	Name	Type	Description
	<positionParameters>	Array	Structured array containing position coordinates for active coordinate system. Array("NAME:Parameters", "XPosition=", <string>, "YPosition=", <string>, "ZPosition=", <string>)
Return Value	Array containing string object names.		

Python Syntax	GetBodyNamesByPosition (<positionParameters>)
Python Example	<pre>oEditor.GetBodyNamesByPosition(["NAME:Parameters", "XPosition=", "0mm", "YPosition=", "15mm",</pre>

	<pre> "ZPosition:=", "0mm"]) </pre>

GetEdgeByPosition

Returns the ID for edge(s) that contact a specified point.

UI Access	N/A		
Parameters	Name	Type	Description
	<positionParameters>	Array	Structured array. <pre> Array("NAME:EdgeParameters", "BodyName:=", <string object name>, "Xposition:=", <string>, "YPosition:=", <string>, "ZPosition:=", <string>) </pre> Note: For 2D XY Designs, ZPosition should be set to "0". For 2D RZ Designs, YPosition should be set to "0".
Return Value	Array containing string edge IDs.		

Python Syntax	GetEdgeByPosition (<positionParameters>)

Python Example	<pre>oEditor.GetEdgeByPosition(["NAME:EdgeParameters", "BodyName:=", "Box1", "Xposition:=", "10mm", "YPosition:=", "0mm", "ZPosition:=", "10mm"])</pre>

GetEdgeIDsFromFace

Returns the edge IDs for a specified Face ID.

UI Access	N/A		
Parameters	Name	Type	Description
	<faceID>	Integer	ID of the specified face.
Return Value	Array containing string edge IDs.		

Python Syntax	GetEdgeIDsFromFace (<faceID>)
Python Example	oEditor.GetEdgeIDsFromFace (20)

GetEdgeIDsFromObject

Returns the edge IDs for a specified object.

UI Access	N/A		
Parameters	Name	Type	Description
	<object>	String	Object name.
Return Value	Array containing string edge IDs.		

Python Syntax	GetEdgeIDsFromObject (<object>)		
Python Example	oEditor.GetEdgeIDsFromObject ("Box1")		

GetEntityIDsContainedByNamedSelection

Returns the list of entity IDs contained in the named selection. See [CreateNamedSelection](#).

UI Access	N/A		
Parameters	Name	Type	Description
	<selectionName>	String	selection name.
Return Value	String containing the list of entity IDs contained in the named selection.		

Python Syntax	GetEntityIDsContainedByNamedSelection (<selectionName>)		
Python Example	entityIDs = oEditor.GetEntityIDsContainedByNamedSelection ("FaceSelection1")		

GetEntityListIDByName

Returns the specified entity list's ID number. See [CreateNamedSelection](#).

UI Access	N/A		
Parameters	Name	Type	Description
	<listName>	String	List name.
Return Value	String containing the entity list ID number.		

Python Syntax	GetEntityListIDByName (<listName>)		
Python Example	oEditor.GetEntityListIDByName ("MyList")		

GetFaceArea

Returns the area of a specified face.

UI Access	N/A		
Parameters	Name	Type	Description
	<faceID>	Integer	ID of specified face.
Return Value	Long integer of face area.		

Python Syntax	GetFaceArea (<faceID>)
Python Example	oEditor.GetFaceArea(19)

GetFaceCenter

Returns the center position of a specified face.

UI Access	N/A		
Parameters	Name	Type	Description
	<planarFaceID>	Long	Face ID
Return Value	Array containing string X, Y, Z coordinates.		

Python Syntax	GetFaceCenter (<planarFaceID>)
Python Example	oEditor.GetFaceCenter(12)

GetFaceByPosition

Returns the face ID located at a specified position.

Note:

The coordinates must point to exactly one face, not a vertex or edge where two or more faces join.

UI Access	N/A

Parameters	Name	Type	Description
	<FaceParameters>	Array	Structured Array. Array("NAME:FaceParameters", "BodyName:=", <string>, "XPosition:=", <string>, "YPosition:=", <string>, "ZPosition:=", <string>) For 2D XY Designs, ZPosition should be set to "0". For 2D RZ Designs, YPosition should be set to "0".
Return Value	Integer Face ID.		

Python Syntax	GetFaceByPosition(<FaceParameters>)
Python Example	<pre>oEditor.GetFaceByPosition(["NAME:FaceParameters", "BodyName:=", "Box1", "XPosition:=", "0.2mm", "YPosition:=", "-0.2mm", "ZPosition:=", "0.4mm"])</pre>

GetFaceIDs

Returns the face IDs associated with a specified object.

UI Access	N/A		
Parameters	Name	Type	Description
	<objectName>	String	Object name.
Return Value	Array containing string face IDs.		

Python Syntax	GetFaceIDs (<objectName>)
Python Example	<code>oEditor.GetFaceIDs('Box1')</code>

GetGeometryModelerMode

Returns the modeler mode (3D or XY) for the current design. This lets you know whether the current model is 2D or 3D.

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>

GetModelBoundingBox

Returns the bounding box of the current model using the global coordinate system as reference. The outputs are in the defined modeler units. Only modeled objects are considered for this computation (non-modeled objects are ignored).

UI Access	N/A
Parameters	None.
Return Value	Array containing string Xmin, Ymin, Zmin, Xmax, Ymax, and Zmax values of the bounding box in the global coordinate system

Python Syntax	GetModelBoundingBox()
Python Example	<code>oEditor.GetModelBoundingBox()</code>

GetObjectBoundingBox

Returns the bounding box of a specified object using the global coordinate system as reference. The outputs are in the defined modeler units.

UI Access	N/A		
Parameters	Name	Type	Description
	<objectName>	String	Name of object to get bounding box
Return Value	Array containing string Xmin, Ymin, Zmin, Xmax, Ymax, and Zmax values of the bounding box in the global coordinate system		

Python Syntax	<code>GetObjectBoundingBox(<objectName>)</code>
Python Example	<code>oEditor.GetObjectBoundingBox('Box1')</code>

GetObjectIDByName

Given an object's name, returns its ID. IDs are used with [CreateEntityList](#).

UI Access	N/A		
Parameters	Name	Type	Description
	<objectName>	String	Object name.
Return Value	Integer object ID.		

Python Syntax	<code>GetObjectIDByName(<objectName>)</code>
Python Example	<code>oEditor.GetObjectIDByName('Box1')</code>

GetObjectName

Returns an object's name from its specified base index (creation order).

UI Access	N/A		
Parameters	Name	Type	Description
	<index>	Integer	Base index (where '0' is the first item created)

Return Value	String containing object name.

Python Syntax	<code>GetObjectName(<index>)</code>
Python Example	<code>oEditor.GetObjectName(3)</code>

GetObjectNameByFaceID

Returns an object name given a face ID.

UI Access	N/A		
Parameters	Name	Type	Description
	<FaceID>	Integer	The face ID.
Return Value	String containing object name.		

Python Syntax	<code>GetObjectNameByFaceID (<FaceID>)</code>
Python Example	<code>oEditor.GetObjectNameByFaceID(88)</code>

GetObjectsByMaterial

Returns a list of objects of a specified material.

UI Access	N/A

Parameters	Name	Type	Description
	<material>	String	Material name.
Return Value	Array containing string object names.		

Python Syntax	GetObjectsByMaterial (<material>)
Python Example	<code>oEditor.GetObjectsByMaterial('copper')</code>

GetObjectsInGroup

Returns a list of objects in a specified group.

UI Access	N/A		
Parameters	Name	Type	Description
	<group>	String	Group name. One of: • "<materialName>" • "<assignmentName>" • "Non Model" • "Solids" • "Unclassified" • "Sheets" • "Lines"
Return Value	Array containing string object names.		

Python Syntax	GetObjectsInGroup (< <i>group</i> >)
Python Example	<code>oEditor.GetObjectsInGroup('Sheets')</code>

GetMatchedObjectName

Returns all object names containing the input text string.

UI Access	N/A		
Parameters	Name	Type	Description
	< <i>wildcardText</i> >	String	Text string present in object name(s).
Return Value	Array containing string object names.		

Python Syntax	GetMatchedObjectName (< <i>wildcardText</i> >)
Python Example	<code>oEditor.GetMatchedObjectName('Box*')</code>
	<code>oEditor.GetMatchedObjectName('?ox?')</code>

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>

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>

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>

GetUserPosition

Returns a user's current coordinates in the 3D Modeler window.

UI Access	N/A		
Parameters	Name	Type	Description
	<i><prompt></i>	String	"Enter a point." Then click a point in the 3D Modeler window.
Return Value	Array containing X, Y, Z coordinates.		

Python Syntax	GetUserPosition(<i><prompt></i>)
Python Example	<code>oEditor.GetUserPosition("Enter a point.")</code>

GetVertexIDsFromEdge

Returns vertex IDs associated with a specified edge.

UI Access	N/A		
Parameters	Name	Type	Description
	<edgeID>	Integer	Edge ID.
Return Value	Array containing string vertex IDs.		

Python Syntax	GetVertexIDsFromEdge(<edgeID>)		
Python Example	<code>oEditor.GetVertexIDsFromEdge(10)</code>		

GetVertexIDsFromFace

Returns vertex IDs associated with a specified face.

UI Access	N/A		
Parameters	Name	Type	Description
	<faceID>	Integer	Face ID.
Return Value	Array containing string vertex IDs.		

Python Syntax	GetVertexIDsFromFace(<faceID>)		
Python Example	<code>oEditor.GetVertexIDsFromFace(10)</code>		

GetVertexIDsFromObject

Returns vertex IDs associated with a specified object.

UI Access	N/A		
Parameters	Name	Type	Description
	<objectName>	String	Object name.
Return Value	Array containing string vertex IDs.		

Python Syntax	GetVertexIDsFromObject(<objectName>)		
Python Example	oEditor.GetVertexIDsFromObject('Box1')		

GetVertexPosition

Returns an array of coordinates for a specified vertex.

UI Access	N/A		
Parameters	Name	Type	Description
	<vertexID>	Integer	Vertex ID.
Return Value	Array containing X, Y, Z coordinates.		

Python Syntax	<code>GetVertexPosition(<vertexID>)</code>
Python Example	<code>oEditor.GetVertexPosition(1)</code>

PageSetup

Specifies page settings for printing.

UI Access	File > Page Setup.		
Parameters	Name	Type	Description
	<Parameters>	Array	Structured array. ["NAME:PageSetupData", "margins:=", <SetupArray>]
	<SetupArray>	Array	Structured Array ["left:=", <string>, "right:=", <string>, "top:=", <string>, "bottom:=", <string>]
Return Value	None.		

Python Syntax	PageSetup(<Parameters>)
Python Example	<pre>oEditor.PageSetup(["NAME:PageSetupData", "margins:=", ["left:=", "10mm", "right:=", "10mm", "top:=", "10mm", "bottom:=", "10mm"]])</pre>

RenamePart

Renames an object.

UI Access	Enter new name in Name field.		
Parameters	Name	Type	Description
	<renameParametersArray>	Array	Structured array. Array("NAME:Rename Data", "Old Name:=", <string>, "New Name:=", <string>)

Return Value	None.

Python Syntax	<code>oEditor.RenamePart(<renameParametersArray>)</code>
Python Example	<pre>oEditor.RenamePart (['NAME:Rename Data', 'Old Name:=', 'partname', 'New Name:=', 'newpartname',])</pre>

This page intentionally
left blank.

11 - Output Variable Script Commands

Output variable commands should be executed by the "OutputVariable" module.

First, obtain the output variable module from oDesign and use it for output variable commands:

```
Set oModule = oDesign.GetModule("OutputVariable")  
  
oModule.<CommandName><args>
```

The commands are:

[CreateOutputVariable](#)

[DeleteOutputVariable](#)

[DoesOutputVariableExist](#)

[EditOutputVariable](#)

[GetOutputVariableValue](#)

[GetOutputVariableValue](#)

[SimValueContext](#)

CreateOutputVariable (Maxwell)

Use: Adds a new output variable to the output variable list. Output variables are associated with a name and an expression. The name of an output variable is not permitted to collide with design variables, Sim values, or other output variable names. It cannot have spaces or any arithmetic or other operators. The definitions cannot be cyclic. For example, $A = 2*B$, $B=3*A$ is not allowed.

Command: Maxwell3D>Results>Output Variables, Maxwell2D>Results>Output Variables, or RMXprt>Results>Output Variables.

Syntax: CreateOutputVariable <OutputVarName>, <Expression>, <SolutionName>, <ReportTypeName>, <simValueContext>

Return Value: None

Parameters: <OutputVarName>

Type: <string>

Name of the output variable.

<Expression>

Type: <value>

Value to assign to the variable.

<SolutionName>

Type: <string>

Name of the solution as listed in the **output variable UI**.

For example: "Setup1 : Last Adaptive"

<ReportTypeName>

Type: <string>

The name of the report type as seen in the output variable UI.

<simValueContext>

Type: <variant>

Context for which the output variable expression is being evaluated

DeleteOutputVariable

Deletes an existing output variable. The variable can only be deleted if it is not in use by any traces.

UI Access	Maxwell > Results > Output Variables. In the Output Variables window, click Delete .

Parameters	Name	Type	Description
	<OutputVarName>	String	Name of the output variable.
Return Value	None.		

Python Syntax	DeleteOutputVariable (<OutputVarName>)
Python Example	<pre>oModule = oDesign.GetModule("OutputVariable") oModule.DeleteOutputVariable ("testNew")</pre>

DoesOutputVariableExist

Verifies whether or not a named output variable exists.

UI Access	N/A		
Parameters	Name	Type	Description
	<outputVariableName>	String	The output variable name.
Return Value	Boolean True if the variable exists; False if it does not.		

Python Syntax	DoesOutputVariableExist(<outputVariableName>)
Python Example	<pre>oProject = oDesktop.GetActiveProject() oDesign = oProject.GetActiveDesign() oModule = oDesign.GetModule("OutputVariable")</pre>

	<code>oModule.DoesOutputVariableExist("MyTestVar")</code>

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 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	<code>EditOutputVariable (<OrigVarName>, <NewExpression>, <NewVarName>, <SolutionName>, <ReportType>, <ContextArray>)</code>

Python Example	<pre>oModule = oDesign.GetModule("OutputVariable") oModule.EditOutputVariable ("test", "normalize(R1_0.V)", "testNew", "TR", "Standard", [])</pre>

GetOutputVariableValue

Use: Gets the double value of an output variable. Only those expressions that return a double value are supported. The expression is evaluated only for a single point.

Syntax: GetOutputVariableValue(<OutputVarName>, <IntrinsicVariation>, <SolutionName>, <ReportTypeName>, <SimValueContext>)

Return Value: Double value of the output variable .

Parameters: <OutputVarName>

Type: <string>

Name of the output variable.

<IntrinsicVariation>

Type: <string>

A set of intrinsic variable value pairs to use when evaluating the output expression. If no variables are present, a null string must be used. The AC Magnetic solution type requires at least the specification of frequency as shown here while the transient solver will require Time as a minimum entry.

For example: "" (Null String)

"freq='60'" (AC Magnetic example)

<SolutionName>

Type: <string>

Name of the solution as listed in the output variables dialog box as shown in SimValueContext.

For example: "Setup1 : Last Adaptive" or "Setup1 : Transient"

<ReportTypeName>

Type: <string>

The name of the report type as seen in the output variable dialog box.

Note:

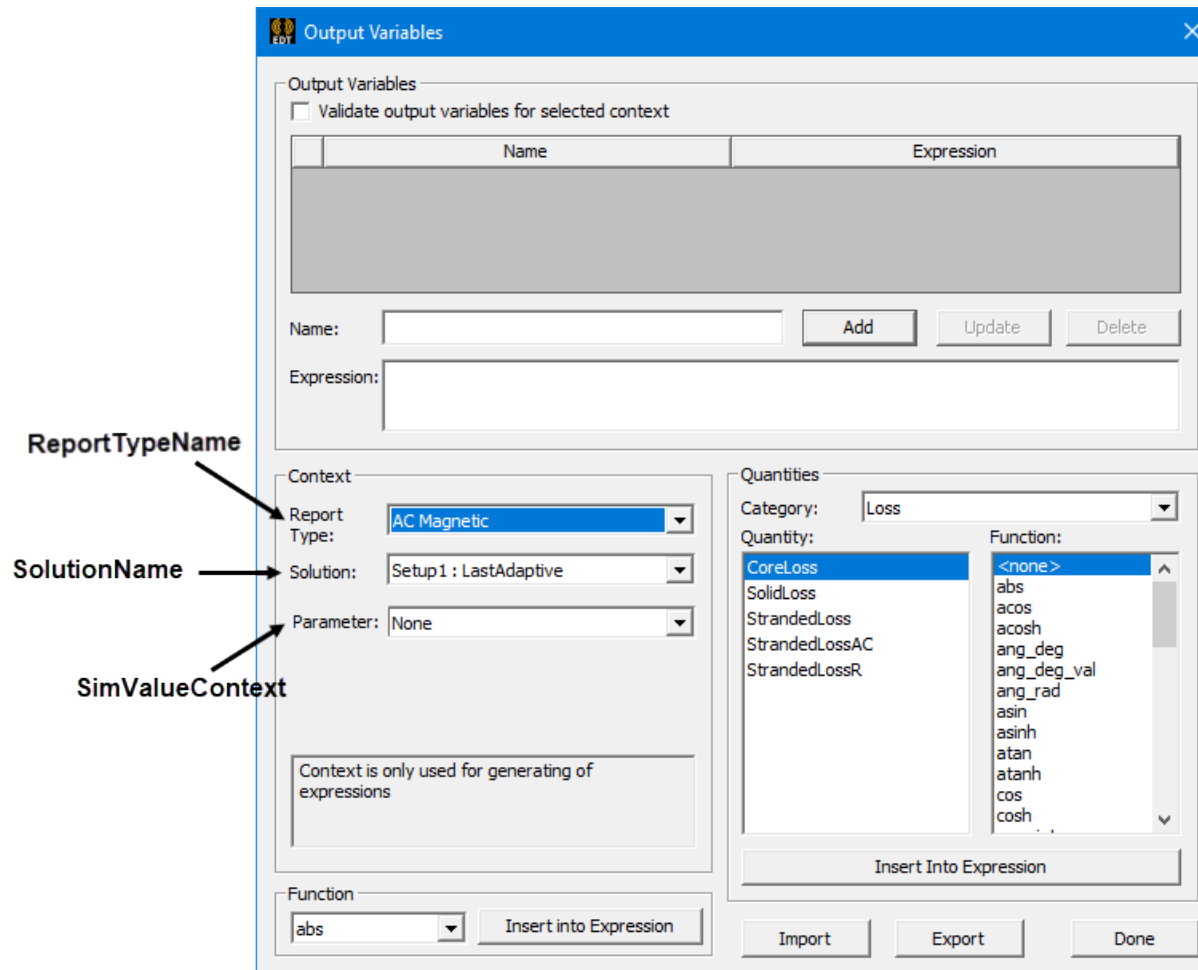
If the Output Variable is based upon a field quantity, then the ReportTypeName must be set to **"Fields"**.

<SimValueContext>

Type: <Array>

Context for which the output variable expression is being evaluated. This section is related to the Parameters or Geometry fields in the UI that specify a context for extracting data quantities. This can be an empty string if there is no context.

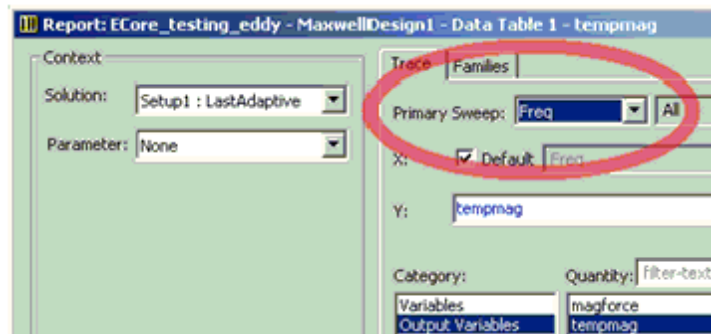
The **Context** section of the Output Variables dialog box contains the information that is required when getting the value of the Output Variable. Note these values when using the dialog box to create an Output Variable.

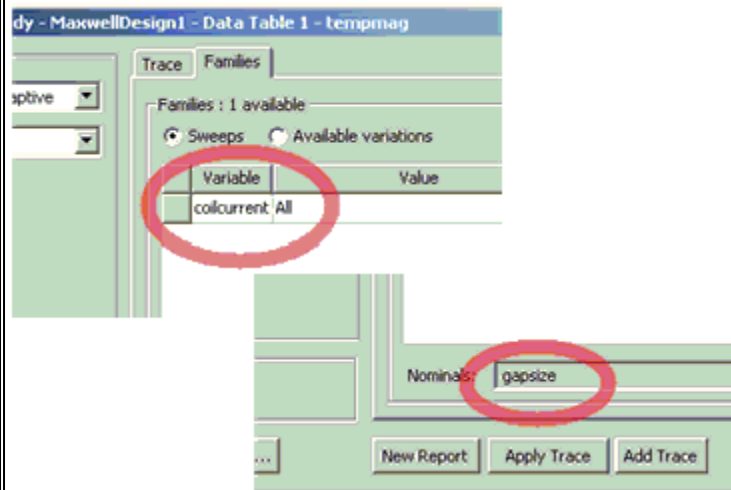


Hint Variation variables can be identified by doing a **Data Table** report and viewing the variable specifications in the table header. For example:

	Freq [Hz]	tempmag coilcurrent='1000' gapsize='-0.001' Setup1 : LastAdaptive
1	60.000000	1.538441

Also, as shown below, the **Trace** and **Families** tabs on the Report dialog box can be used as illustrated to identify variation variables.





This page intentionally
left blank.

12 - Reporter Editor Script Commands

Reporter commands should be executed by the oDesign object.

For example:

```
oDesign = Project.SetActiveDesign("MaxwellDesign1")  
oModule = oDesign.GetModule("ReportSetup")
```

All Report and Trace properties can be edited using the **ChangeProperty** commands. This includes Title properties, General properties, and Background properties such as border color, fonts, X and Y axis scaling, and number display.

Note:

When you execute **Tools > Record Script**, operations performed in the Reporter are automatically recorded.

The list of commands is as follows:

[AddCartesianLimitLine](#)

[AddCartesianXMarker](#)

[AddCartesianYMarker](#)

[AddCartesianYMarkerToStack](#)

[AddDeltaMarker](#)

[AddMarker](#)

[AddNote](#)

[AddTraces](#)

[ClearAllMarkers](#)

[ClearAllTraceCharacteristics](#)

[CopyReportDefinitions](#)

[CopyReportData](#)

[CopyTraceDefinitions](#)

[CopyTracesData](#)

[CreateReport](#)

[CreateReportFromFile](#)

[CreateReportFromTemplate](#)

[CreateReportOfAllQuantities](#)

[DeleteMarker](#)

[DeleteAllReports](#)

[DeleteReports](#)

[DeleteTraces](#)

[FFTONReport](#)

[ExportFieldsToFile](#)

[ExportImageToFile](#)

[ExportModelImageToFile](#)

[ExportModelMeshToFile](#)

[ExportToFile \[Reporter\]](#)

[ExportMarkerTable](#)

[GetAllCategories](#)

[GetAllQuantities](#)

[GetAllReportNames](#)

[GetAvailableDisplayTypes](#)

[GetAvailableReportTypes](#)

[GetAvailableSolutions](#)

[GetChildNames](#)

[GetChildObject](#)

[GetChildTypes](#)

[GetDisplayType](#)

[GetPropertyValue](#)

[GetReportTraceNames](#)

[GetSolutionContexts](#)

[GetSolutionDataPerVariation](#)

[GroupPlotCurvesByGroupingStrategy](#)

[ImportIntoReport](#)

[MovePlotCurvesToGroup](#)

[MovePlotCurvesToNewGroup](#)

[PasteReports](#)

[PasteTraces](#)

[RenameReport](#)

[RenameTrace](#)

[ResetPlotSettings](#)

[SavePlotSettingsAsDefault](#)

[SetPropValue](#)

[UpdateAllFieldsPlots](#)

[UpdateAllReports](#)

[UpdateQuantityFieldsPlots](#)

[UpdateReports](#)

[UpdateTraces](#)

[UpdateTracesContextandSweeps](#)

[UnGroupPlotCurvesInGroup](#)

AddCartesianLimitLine

Adds a limit line to a report on the X axis.

UI Access	Report2D > Add Limit Line> Specify Points...		
Parameters	Name	Type	Description
	<ReportName>	String	Name of the report.
	<Def>	Array	Structured array: Array("NAME:CartesianLimitLine", Array("NAME:XValues", <integer X values>), "XUnits:=", <string unit of measure for X>, Array("NAME:YValues", <integer Y values>),

			<code>"YUnits:=", "<string unit of measure for Y>", "YAxis:=", <string name of associated Y axis>)</code>
Return Value	None.		

Python Syntax	<code>AddCartesianLimitLine (<ReportName>, <Def>)</code>
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>

AddCartesianXMarker

Adds a marker to a report on the X axis.

UI Access	Report2D > Marker > Add X Marker.		
Parameters	Name	Type	Description
	<code><ReportName></code>	String	Name of report.
	<code><MarkerName></code>	String	Marker name, including any trailing number.

	<table><tr><td><XValue></td><td>Double</td><td>X coordinate.</td></tr></table>			<XValue>	Double	X coordinate.
<XValue>	Double	X coordinate.				
Return Value	None					

Python Syntax	AddCartesianXMarker (<ReportName>, <MarkerName>, <XValue>)
Python Example	<code>oModule.AddCartesianXMarker ("XY Plot 1", "MX1", 0)</code>

AddCartesianYMarker

Adds a marker to a report on the Y axis.

UI Access	Report2D > Marker > Add Y Marker.		
Parameters	Name	Type	Description
	<ReportName>	String	Name of report.
	<MarkerName>	String	Marker name, including any trailing number.
	<AxisName>	String	Name of axis.
	<YValue>	Double	Y coordinate.
	<CurveName>	String	Name of curve.
Return Value	None		

Python Syntax	AddCartesianYMarker (<ReportName>, <MarkerName>, <AxisName>, <YValue>, <CurveName>)
Python Example	<code>oModule.AddCartesianYMarker("XY Plot 1", "MY1", "Y1", 0,</code>


```
"dB() : Setup1 : Sweep1")
```

AddDeltaMarker

Add markers to calculate differences between two trace points on a plot.

UI Access	Report2D > Marker > Add Delta Marker.		
Parameters	Name	Type	Description
	<ReportName>	String	Name of report.
	<MarkerName1>	String	Marker name, including any trailing number, for the first marker.
	<CurveName1>	String	Full trace name for the first marker.
	<PrimarySweepValue1>	String	Frequency, including unit, for the first marker's sweep.
	<MarkerName2>	String	Marker name, including any trailing number, for the second marker.
	<CurveName2>	String	Full trace name for the second marker.
	<PrimarySweepValue2>	String	Frequency, including unit, for the second marker's sweep.
Return Value	None		

Python Syntax	AddDeltaMarker(<ReportName>, <MarkerName1>, <CurveName1>, <PrimarySweepValue1>, <MarkerName2>, <CurveName2>, <PrimarySweepValue2>)
Python Example	<pre>oModule.AddDeltaMarker("S Parameter Plot 1", "m1", "dB(S(Port1 Port1)) : Setup1 : Sweep1 : Cartesian", "8.45GHz", "m2", "dB(S(Port1 Port1)) : Setup1 : Sweep1 : Cartesian", "9.5GHz")</pre>

AddMarker

Adds a marker to a trace on a report.

UI Access	Report2D > Marker > Add Marker.		
Parameters	Name	Type	Description
	<ReportName>	String	Name of report.
	<MarkerName>	String	Marker name, including any trailing number.
	<CurveName>	String	Full trace name.
	<PrimarySweepValue>	String	Primary sweep value, including unit.
Return Value	None		

Python Syntax	AddMarker(<ReportName>, <MarkerName>, <CurveName>, <PrimarySweepValue>)
Python Example	<pre>oModule.AddMarker("XY Plot 1", "m5", "GS1.VAL : TR4 : Cartesian", "3.615999999999997s")</pre>

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

	<i><ReportName></i>	String	Name of report
	<i><NoteDataArray></i>	Array	Structured array. Array("NAME:<StringDataName>", <NoteArray>)
	<i><NoteArray></i>	Array	Structured array: Array("NAME:<NoteDataSourceName>", "SourceName:=", <string source name>, "HaveDefaultPos:=", <boolean True for position 0,0; False to specify below>, "DefaultXPos:=", <int X position for note; 0 for default>, "DefaultYPos:=", <int Y position for note; 0 for default>, "String:=", <string note text>)
Return Value	None		

Python Syntax	AddNote (<ReportName>, <NoteDataArray>)
Python Example	<pre>oModule.AddNote("XY Plot1", ["NAME:NoteDataSource", "SourceName:=", "Note1", "HaveDefaultPos:=", False, "DefaultXPos:=", 1996, "DefaultYPos:=", 3177,</pre>

	<pre>"String:=", "This is a note."])</pre>

AddTraceCharacteristics

Adds a trace characteristics field to the legend on a report.

UI Access	Report2D > Trace Characteristics > All. This opens the Add Trace Characteristics dialog box.		
Parameters	Name	Type	Description
	<ReportName>	String	Name of report.
	<FunctionName>	String	The function name. See the Functions column of the Add Trace Characteristics dialog box.
	<FunctionArgs>	Array	Array containing string values for any function arguments. Pass empty array if no arguments exist. To see which argument(s) a function takes, see the bottom of the Add Trace Characteristics dialog box. For function with one argument: <pre>Array(<value>)</pre> For function with multiple arguments: <pre>Array(<value>, <value>, ...)</pre>
	<RangeArgs>	Array	Required. Array containing either string "Full" for a full sweep range, or "Specified" and strings containing the start and end values for the frequency range. For example:

	Array("Specified", "19.5GHz", "24.4GHz")
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>

AddTraces

Creates a new trace and adds it to the specified report.

UI Access	Modify Report > Add Trace.		
Parameters	Name	Type	Description
	<ReportName>	String	Name of Report
	<SolutionName>	String	Name of the solution as listed in the Modify Report dialog box.
	<ContextArray>	Array	Context for which the expression is being evaluated. This can be an empty string if there is no context.
			<pre>Array("Domain:=", <DomainType>) <DomainType> ex. "Sweep" or "Time" Array("Context:=", <GeometryType>)</pre>

			<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 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,</pre>

	<pre>"WindowType:=", 0, "KaiserParameter:=", 1, "MaximumTime:=", 6.2437437437444E-009], ["Time:=", ["All"], "OverridingValues:=", ["0s", "6.24999373748188e-012s", ...]], ["X Component:=", "Time", "Y Component:=", ["TDRZ(WavePort1)"]], [])</pre>

ChangeProperty[ReportSetup]

Change the properties for a Report.

UI Access	Double-click on a report to change its properties.		
Parameters	Name	Type	Description
	<PropertyArray>	Array	Varies, depending on the properties associated with the select object command.
Return Value	None.		

Python Syntax	ChangeProperty(<PropertyArray>)
Python Example	<pre>oModule = oDesign.GetModule("ReportSetup") oModule.ChangeProperty(["NAME:AllTabs", ["NAME:General",</pre>

```
[
    "NAME:PropServers",
    "XY Plot 1:General"
],
[
    "NAME:ChangedProps",
    [
        "NAME:Use Scientific Notation",
        "Value:=", True
    ]
]
])

oModule = oDesign.GetModule("ReportSetup")
oModule.ChangeProperty(
    [
        "NAME:AllTabs",
        [
            "NAME:Legend",
```



```
[
  "NAME:PropServers",
  "S Parameter Plot 1:Legend"
],
[
  "NAME:ChangedProps",
  [
    "NAME:Legend Name",
    "Value:=" , "Ansys"
  ]
]
]
]
)

oModule.ChangeProperty(
[
  "NAME:AllTabs",
  [
    "NAME:General",
    [
      "NAME:PropServers",
      "S Parameter Plot 1:General"
```

```
        ],  
        [  
            "NAME:ChangedProps",  
            [  
                "NAME:Auto Scale Fonts",  
                "Value:=" , False  
            ]  
        ]  
    ]  
])
```

Example For Rectangular Stacked Plot

```
oProject = oDesktop.SetActiveProject("antenna_test")  
oDesign = oProject.SetActiveDesign("Design1")  
oModule = oDesign.GetModule("ReportSetup")  
oModule.ChangeProperty(  
    [  
        "NAME:AllTabs",  
        [  
            ]  
        ]  
    )
```

```

"NAME:Scaling",
[
  "NAME:PropServers",
  "S Parameter Plot 1:AxisY2:Stack1",
  "S Parameter Plot 1:AxisY2:Stack2"
],
[
  "NAME:ChangedProps",
  [
    "NAME:Min",
    "Value:="
    , "-60"
  ]
]
]
]
)

```

Example Change eye rect and eye plot:

```
ScriptEnv.Initialize("Ansoft.ElectronicsDesktop")
oDesktop.RestoreWindow()
oProject = oDesktop.SetActiveProject("test_nport_synth2")
oDesign = oProject.SetActiveDesign("Circuit1")
oModule = oDesign.GetModule("ReportSetup")
```

```
oModule.ChangeProperty(  
  [  
    "NAME:AllTabs",  
    [  
      "NAME:Grid",  
      [  
        "NAME:PropServers",  
        "QuickEye Voltage Plot 1:Grid:Eye"  
      ],  
      [  
        "NAME:ChangedProps",  
        [  
          "NAME:Major grid line style",  
          "Value:=" , "Dot"  
        ],  
        [  
          "NAME:Minor grid line style",  
          "Value:=" , "Dot"  
        ]  
      ]  
    ]  
  ]  
)
```

```
],  
[  
  "NAME:Axis",  
  [  
    "NAME:PropServers",  
    "QuickEye Voltage Plot 1:AxisX:Eye"  
  ],  
  [  
    "NAME:ChangedProps",  
    [  
      "NAME:Axis Color",  
      "R:=" , 255,  
      "G:=" , 128,  
      "B:=" , 128  
    ]  
  ]  
],  
[  
  "NAME:Axis",  
  [  
    "NAME:PropServers",
```

```
        "QuickEye Voltage Plot 1:AxisY1:Rect"
    ],
    [
        "NAME:ChangedProps",
        [
            "NAME:Axis Color",
            "R:=" , 255,
            "G:=" , 128,
            "B:=" , 128
        ]
    ]
]
])
oModule.ChangeProperty(
    [
        "NAME:AllTabs",
        [
            "NAME:Grid",
            [
                "NAME:PropServers",
```

	<pre> "QuickEye Voltage Plot 1:Grid:Rect"], ["NAME:ChangedProps", ["NAME:Major grid line style", "Value:=" , "LongDash"], ["NAME:Minor grid line style", "Value:=" , "LongDash"]]]])</pre>

ClearAllMarkers

Clears all markers from a report.

UI Access	Report2D > Markers > Clear All.

Parameters	Name	Type	Description
	<ReportName>	String	Name of Report
Return Value	None		

Python Syntax	ClearAllMarkers(<ReportName>)
Python Example	<code>oModule.ClearAllMarkers("XY Plot 1")</code>

ClearAllTraceCharacteristics

Clears all trace characteristics from the legend in a report.

UI Access	Report2D > Trace Characteristics > Clear All.		
Parameters	Name	Type	Description
	<PlotName>	String	Name of the plot
Return Value	None		

Python Syntax	ClearAllTraceCharacteristics(<PlotName>)
Python Example	<code>oModule.ClearAllTraceCharacteristics("XY Plot 1")</code>

CopyTracesData

Copies trace data for a paste operation.

UI Access	Select a trace in the Project tree, right-click and select Copy Data .		
Parameters	Name	Type	Description
	<ReportName>	String	Name of Report.
	<TraceArray>	Array	Trace definitions from which to copy corresponding data.
Return Value	None.		

Python Syntax	CopyTracesData(<ReportName>, <TracesArray>)		
Python Example	<pre>oModule.CopyTracesData ("Transmission", ["mag (S (Port1, Port2)) "])</pre>		

CopyReportsData

Copy all data corresponding to the specified reports.

UI Access	Select a report in the Project tree, right-click and select Copy Data		
Parameters	Name	Type	Description
	<ReportsArray>	Array	Names of reports from which to copy data
Return Value	None		

Python Syntax	CopyReportsData (<ReportsArray>)
Python Example	<code>oModule.CopyReportsData (["Transmission", "Reflection"])</code>

CopyReportDefinitions

Copy the definition of a report for paste operations.

UI Access	Select a report in the Project tree, right-click and select Copy Definition		
Parameters	Name	Type	Description
	<ReportsArray>	Array	Names of reports from which to copy the definitions
Return Value	None		

Python Syntax	CopyReportDefinitions(<ReportsArray>)
Python Example	<code>oModule.CopyReportDefinitions (["Transmission", "Reflection"])</code>

CopyTraceDefinitions

Copy trace definitions for a paste operation.

UI Access	Select a trace in the Project tree, right-click and select Copy Definition		
Parameters	Name	Type	Description

	<ReportName>	String	Name of Report
	<TracesArray>	Array	Trace definitions to copy
Return Value	None		

Python Syntax	CopyTraceDefinitions(<ReportName>, <TracesArray>)		
Python Example	<pre>oModule.CopyTraceDefinitions ("Transmission", ["mag (S (Port1, Port2)) "])</pre>		

CreateReportFromFile

Creates a new report from an .rdat file.

UI Access	Right-click on Results > Create Report From File...		
Parameters	Name	Type	Description
	<FilePathName>	String	Path to .rdat file.
Return Value	None.		

Python Syntax	CreateReportFromFile(<FilePathName>)		
Python Example	<pre>oModule.CreateReportFromFile ("C:/Users/MyDir/Documents/Return_Loss.rdat")</pre>		

CreateReport (Maxwell)

Creates a new report with a single trace and adds it to the **Results** branch in the project tree. To add more traces, use the **AddTraces** command. To edit the display properties, use the [ChangeProperty](#) Script command.

UI Access	Maxwell2D or Maxwell3D > Results > Create<type> Report		
Parameters	Name	Type	Description
	<ReportName>	String	Name of report
	<ReportType>	String	Some possible values are "Transient APhi", "Fields", "Noise Vibration", "Time Averaged Fields", "Matrix", "DC R/L Fields", "AC R/L Fields", "C Fields", "Harmonic Force"
	<DisplayType>	String	Possible values are "Rectangular Plot", "Rectangular Stacked Plot", "Data Table", "3D Rectangular Plot", "Rectangular Contour Plot".
	<TraceArray>	Array	<pre>["NAME:Traces", <OneTraceArray>, <OneTraceArray>, ...]</pre> <OneTraceArray> <pre>["NAME:<TraceName>, "SolutionName:=", "string", "Context:=", "string", <DisplayTypeDependentData>]</pre>
	<SolutionName>	String	Name of the solution as listed in the Traces dialog box. For example: "Setup1 : Last Adaptive"
	<ContextArray>	Array	Context for which the output variable expression is being evaluated. <pre>["Domain:=", <DomainType>]</pre> <DomainType>

			ex. "Sweep" or "Time" This can be an empty string if there is no context. Example: "Line1" or "" Field reports usually require a polyline (e.g. "Line1") unless they are integrations.. <DisplayTypeDependentData> This data varies according to the display type. See the examples below.
	<FamiliesArray>	Array	Array of strings that contains sweep definitions for the report. ["<VariableName>:= ", <ValueArray>] <ValueArray> ["All" [or ["Value1", "Value2", ... "Valuen"]] examples of <VariableName> "Freq", "Theta", "Distance"
	<ReportDataArray>	Array	Array of strings that contains the report quantity and X, Y, and (Z) axis definitions. ["X Component:=", <VariableName>, "Y Component:=", <VariableName> <ReportQuantityArray>] <ReportQuantityArray> ex. Array("dB(S(Port1, Port1))")
Return Value	None.		

Python Syntax	CreateReport (<ReportName> <ReportType> <DisplayType> <SolutionName> <ContextArray>

	<FamiliesArray><ReportDataArray>
Python Example	Example for Transient <pre>oModule.CreateReport("Loss Plot 1", "Transient", "Rectangular Plot", "Setup1 : Transient", ["Domain:=" , "Average and RMS", "npoles:=" , "2", "alignment:=" , "0deg", "orientation:=" , "CounterClockwise", "losses:=" , "0W", "speed:=" , "0rpm", "Resistance(PhA):=" , "0ohm", "Resistance(PhB):=" , "0ohm", "Resistance(PhC):=" , "0ohm", "Inductance(PhA):=" , "0H", "Inductance(PhB):=" , "0H", "Inductance(PhC):=" , "0H"], ["Im:=" , ["All"], "Gamma:=" , ["Nominal"], "speed_rpm:=" , ["Nominal"], "poles:=" , ["Nominal"]], ["X Component:=" , "Im", "Y Component:=" , ["AvgCoreLoss"]], [])</pre> Example for Transient A-Phi Formulation:

```
oModule.CreateReport("Loss Plot 1", "Transient APhi", "Rectangular Plot", "Setup1
: Transient",
[
    "Domain:=" , "Sweep"
],
[
    "Time:=" , ["All"],
    "$HcRef:=" , ["Nominal"],
    "$MagnetTemperature:=" , ["Nominal"]
],
[
    "X Component:=" , "Time",
    "Y Component:=" , ["TerminalVoltage(Voltage1)", "TerminalVoltage(Voltage2)"]
])
```

[Beta]Example for Transient A-Phi, LayoutForce plot:

```
oModule.CreateReport("LayoutForce Plot 1", "Transient APhi", "Rectangular Plot",
"Setup1 : Transient",
[
    "Domain:=" , "Sweep"
],
[
    "Time:=" , ["All"]
],
[
    "X Component:=" , "Time",
    "Y Component:=" , ["LayoutForce1.Force_mag"]
])
...`
```

Example for Average_Surface_Loss_Density:

```
oModule.CreateReport("Calculator Expressions Plot 1", "Time Averaged Fields",
"Rectangular Plot", "Setup1 : Transient",
```

```
[
  "Context:=" , "Point1",
  "PointCount:=" , 1001,
  "Time0:=" , "0s"
],
[
  "Time:=" , ["All"]
],
[
  "X Component:=" , "Time",
  "Y Component:=" , ["Average_Surface_Loss_Density"]
])
```

Example for Transient and AC Magnetic Per-Winding Losses:

```
oModule.CreateReport("Loss Plot 4", "Transient", "Rectangular Plot", "Setup1 :
Transient",
[
  "Domain:=" , "Sweep"
],
[
  "Time:=" , ["All"]
],
[
  "X Component:=" , "Time",
  "Y Component:=" , ["PerWindingSolidLoss(WindingV1_Solid)"]
])
oModule.AddTraces("Loss Plot 4", "Setup1 : Transient",
[
  "Domain:=" , "Sweep"
],
[
  "Time:=" , ["All"]
```



```

],
[
    "X Component:=" , "Time",
    "Y Component:=" , ["PerWindingStrandedLoss(WindingI3_Litz)"]
])
oModule.AddTraces("Loss Plot 4", "Setup1 : Transient",
[
    "Domain:=" , "Sweep"
],
[
    "Time:=" , ["All"]
],
[
    "X Component:=" , "Time",
    "Y Component:=" , ["PerWindingStrandedLossAC(WindingI3_Litz)"]
])

```

Example for Harmonic Force Report Type:

```

oModule.CreateReport("Harmonic Force Plot 5", "Harmonic Force", "3D Rectangular
Bar Plot", "Setup1 : Transient",
[
    "Freq:=" , ["All"],
    "SpatialOrder:=" , ["All"],
    "fractions:=" , ["Nominal"],
    "halfAxial:=" , ["Nominal"],
    "endRegion:=" , ["Nominal"],
    "RPM:=" , ["1500rpm"],
    "ppairs:=" , ["Nominal"]
],
[
    "X Component:=" , "Freq",
    "Y Component:=" , "SpatialOrder",
    "Z Component:=" , ["re(HarmonicForce_a)"]
]

```

	<pre>])</pre> Example for Thin Layer (Air Gap) Report Type: <pre>oModule.CreateReport("Magnetic Energy Table 1", "Magnetostatic", "Data Table", "Setup1 : LastAdaptive", [], ["\$Current:=" , ["All"]], ["X Component:=" , "\$Current", "Y Component:=" , ["MagneticEnergy(ThinLayer1)"]]])</pre>

CreateReportFromTemplate

Creates a report from a saved template.

UI Access	[product] > Results > Report Templates > PersonalLib > [TemplateName]		
Parameters	Name	Type	Description
	<TemplatePath>	String	Path to report template
Return Value	None		

Python Syntax	CreateReportFromTemplate(<TemplatePath>)
Python Example	<pre>oModule.CreateReportFromTemplate("C:/MyHFSSProjects/PersonalLib/ReportTemplates/TestTemplate.rpt")</pre>

CreateReportOfAllQuantities

Creates a report including all quantities in a category. Cannot create a report with expressions.

UI Access	NA		
Parameters	Name	Type	Description
	<ReportType>	String	Report type name as input parameter
	<DisplayType>	String	Display type name as input parameter
	<SolutionName>	String	Solution name as input parameter
	<SimValueCtxt>	String	A context name, or array of string that encoded the context(I).
	<CategoryName>	String	Category name as input parameter
	<PointSet>	Array	Array of strings(II)
	<CommonComponentsOfTraces>	Array	Array of strings(III)
	<ExtTraceInfo>	Array	Array of strings(IV)
Return Value	None.		

Python Syntax	CreateReportOfAllQuantities(<ReportNameArg>, <ReportType>, <DisplayType>, <SolutionName>, <SimValueCtxt>, <CategoryName>, <PointSet>, <CommonComponentsOfTraces>, <ExtTraceInfo>)
Python Example	<pre>oModule.CreateReportOfAllQuantities("Smith Chart all", "Modal Solution Data", "Smith Chart", "Setup1 : LastAdaptive", [], "S Parameter", ["Freq:=", ["All"], "offset:=", ["All"], "a:=", ["Nominal"], "b:=", ["Nominal"]], [], [])</pre>

DeleteMarker

*Use:*Deletes the specified marker.

UI Access	[product] > Fields > Fields > Marker > Delete Marker .		
Parameters	Name	Type	Description
	<MarkerName>	String	Name of the marker.
Return Value	None.		

Python Syntax	DeleteMarker(<MarkerName>)
Python Example	<code>oModule.DeleteMarker("m1")</code>

DeleteAllReports

Deletes all existing reports.

UI Access	Right-click the report to delete in the project tree, and then click Delete All Reports on the shortcut menu.
Parameters	None.
Return Value	None.

Python Syntax	DeleteAllReports()
Python Example	<code>oModule.DeleteAllReports()</code>

DeleteReports

Deletes an existing report or reports.

UI Access	Right-click the report to delete in the project tree, and then click Delete on the shortcut menu		
Parameters	Name	Type	Description
	<code><ReportNameArray></code>	Array	Array of report names to be deleted
Return Value	None.		

Python Syntax	DeleteReports(<code><ReportNameArray></code>)
Python Example	<code>oModule.DeleteReports (["Rept2DRectFreq"])</code>

DeleteTraces

Deletes an existing traces or traces.

UI Access	Right-click the Trace to delete in the project tree, and then click Delete on the shortcut menu		
Parameters	Name	Type	Description
	<code><TraceSelection></code>	Array	Structured array define selections. <code>Array("<ReportName>:=", <TracesArray>, <TracesAr-</code>

			<code>ray>, ...)</code>
	<code><ReportName></code>	String	Name of Report
	<code><TracesArray></code>	Array	Contains the traces to delete within a report <code>Array(<Trace>, <Trace>, ...)</code>
	<code><Trace></code>	String	A specific trace that the user wishes to delete
Return Value	None.		

Python Syntax	<code>DeleteTraces(<TraceSelection>)</code>
Python Example	<pre>oModule.DeleteTraces (["XY Plot 1:=", ["dB(S(LumpPort1,LumpPort1))"], "XY Plot 2:=", ["Mag_E"]])</pre>

ExportImageToFile [Reporter]

Exports a report image in a specified format. This command fully supports -ng (non-graphical) mode.

UI Access	N/A		
Parameters	Name	Type	Description
	<code><ReportName></code>	String	Name of report to be exported.
	<code><FileName></code>	String	Full path of the exported image file name; with extension of jpg, gif, tiff, tif, bmp, or wrl.
	<code><Width></code>	Integer	Image width in pixels; if width or height is less or equal to zero, use the report window width, or 500 pixels if report window is closed.

	<Height>	Integer	Image height in pixels; if width or height is less or equal to zero, use the report window height, or 500 pixels report window is closed.
	[<SaveImageParameters2D>]	Boolean	Whether to ShowLegend, ShowMarkerTable, ShowDeltaMarkerTable. Values can be True, False or Default. Default is True.
	[<SaveImageParameters3D>]		Whether to AutoFit [Default is "False"] ShowLegend, ShowMarkerTable, ShowDeltaMarkerTable. Values can be True, False or Default. Default is True.
Return Value	None.		

Python Syntax	ExportImageToFile(<ReportName>, <FileName>, <Width>, <Height>)
Python Example	<pre>##### Script Recorded for Saving 2D oDesktop.OpenProject("E:/Ansoft/Tee2.aedt") oProject = oDesktop.SetActiveProject("Tee2") oDesign = oProject.SetActiveDesign("TeeModel") oModule = oDesign.GetModule("ReportSetup") oModule.ExportImageToFile("S Parameter Plot 1", "E:/Ansoft/S Parameter Plot 1.png", 1920, 1080) oModule.ExportImageToFile("S Parameter Plot 12", "E:/Ansoft/S Parameter Plot 12.png", 1920, 1080) oModule.ExportImageToFile("S Parameter Plot 1", "E:/Ansoft/S Parameter Plot 1.png", 1920, 1080, ["NAME:SaveImageParams",</pre>

```
"ShowLegend:=" , "False",  
"ShowMarkerTable:=" , "False"  
"ShowDeltaMarkerTable:=" , "False"  
])  
oModule.ExportImageToFile("S Parameter Plot 1", "E:/Ansoft/S Parameter Plot 1.png",  
1920, 1080,  
[  
"NAME:SaveImageParams",  
"ShowLegend:=" , "True",  
"ShowMarkerTable:=" , "True"  
"ShowDeltaMarkerTable:=" , "False"  
])  
oModule.ExportImageToFile("S Parameter Plot 12", "E:/Ansoft/S Parameter Plot 12.png",  
1920, 1080,  
[  
"NAME:SaveImageParams",  
"ShowLegend:=" , "False",  
"ShowMarkerTable:=" , "Default"  
"ShowDeltaMarkerTable:=" , "False"  
])  
oModule.ExportImageToFile("S Parameter Plot 12", "E:/Ansoft/S Parameter Plot 12.png",  
1920, 1080,
```



```
[  
    "NAME:SaveImageParams",  
    "ShowLegend:=" , "True",  
    "ShowMarkerTable:=" , "Default"  
    "ShowDeltaMarkerTable:=" , "True"  
])  
oDesktop.CloseProject("Tee2")
```

Script Recorded for saving as 3D:

```
oDesktop.OpenProject("E:/Ansoft/Tee2.aedt")  
oProject = oDesktop.SetActiveProject("Tee2")  
oDesign = oProject.SetActiveDesign("TeeModel")  
oModule = oDesign.GetModule("ReportSetup")  
oModule.ExportImageToFile("S Parameter Plot 6", "E:/Ansoft/S Parameter Plot 11.png",  
1335, 466,  
[  
    "NAME:SaveImageParams",  
    "AutoFit:=" , "False",  
    "Orientation:=" , "",  
    "ShowOrientationGadget:=", "Default"  
])
```

```
oModule.ExportImageToFile("S Parameter Plot 6", "E:/Ansoft/S Parameter Plot 11.png",
1335, 466,

[
    "NAME:SaveImageParams",
    "AutoFit:=" , "False",
    "Orientation:=" , "",
    "ShowOrientationGadget:=", "Default"
    "ShowLegend:=" , "True"
])

oModule.ExportImageToFile("S Parameter Plot 6", "E:/Ansoft/S Parameter Plot 10.png",
1335, 466,

[
    "NAME:SaveImageParams",
    "AutoFit:=" , "False",
    "Orientation:=" , "",
    "ShowOrientationGadget:=", "Default",
    "ShowLegend:=" , "False"
])
```

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>

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

Python Syntax	ExportReport(<ReportName>, <FileName>)
Python Example	oDesign.ExportReport ("Plot1", "c:\report1.dat")

ExportToFile

Note:

The ExportToFile script command has replaced the script command [ExportReport](#). ExportReport remains in order to retain backward compatibility for existing scripts, but it is strongly recommended that you now use ExportToFile.

From a data table or plot, generates text format, comma delimited, tab delimited, or .dat type output files.

UI Access	Right-click on report name in the project tree and select Export Data .		
Parameters	Name	Type	Description
	<ReportName>	String	Name of report to be exported.
	<FileName>	String	Full path of the exported image file name; with extension of .txt - Post processor format file .csv - Comma-delimited data file .tab - Tab-separated file .dat - Ansys plot data file
	<UnitSpec>	String	For example, "kV, Mhz, yd"
	<UseTraceNumberFormat>	Boolean	"True", "False"
Return Value	None.		

Python Syntax	ExportToFile(<ReportName>, <FileName>)

Python Example	<pre>oModule.ExportToFile("rETotal", "C:/Users/Documents/rETotal.csv") oModule.ExportToFile("S Parameter Table 1", "D:/Users/Documents/cfft.csv", False, " kV, MHz, yd ", True)</pre>

ExportToFile [Reporter]

From a data table or plot, generates text format, comma delimited, tab delimited, .dat, or .rdat type output files.

UI Access	Right-click on report name in the Project tree and select Export .			
Parameters	Name	Type	Description	
	<ReportName>	String	Name of the report	
	<FileName>	String	Path and File Name	
			Supported formats	
			.txt	Post processor format file
			.csv	Comma-delimited data file
			.tab	Tab-separated file
			.dat	Ansys plot data file
			.rdat	Ansys report data file
Return Value	None			

Python Syntax	ExportToFile (<ReportName>, <FileName>)
Python Example	<pre>oModule.ExportToFile (</pre>

	"Plot1", "c:\report1.dat")

ExportMarkerTable

Exports the marker table to a .csv or .tab file.

UI Access	[product] > Fields > Plot Fields > Marker > Export Marker Table .		
Parameters	Name	Type	Description
	<FileName>	String	Name of export file include path.
Return Value	None.		

Python Syntax	ExportMarkerTable(<FileName>)
Python Example	oModule.ExportMarkerTable("C:/work/FieldMarkerTable.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	Name	Type	Description
	<ReportName>	String	Name of specified report.
	<WindowName>	String	Name of window to apply for FFT. Possible values are "Rectangular", "Tri", "Van Hann", "Hamming", "Blackman", "Lanczos", "Weber", "Welch".
	<Function>	String	Function to apply on transformation values. Possible values are "none", "ang_

			deg", "ang_rad", "arg", "cang_deg", "cang_rad", "dB", "dB1 normalize", "dB20normalize", "dBc", "im", "mag", "normalize", "re".
Return Value	None.		

Python Syntax	FFTONReport <ReportName>, <WindowName>, <Function>
Python Example	oModule.FFTONReport("XY Plot 1", "Rectangular", "dB")

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	oModule.GetAllReportNames()

GetAllCategories

Get all available category names (not including variable and output-variables) in a solution for a report type and display type, returned as an array of text strings.

UI Access	NA		
Parameters	Name	Type	Description
	<ReportType>	String	Report type name.
	<DisplayType>	String	Name of display type.
	<SolutionName>	String	Name of solution.
	<SimValueCtxt>	Array	A context name, or array of strings that encode the contexts.
Return Value	Array of text strings		

Python Syntax	GetAllCategories(<ReportType>, <DisplayType>, <SolutionName>, <SimValueCtxt>)
Python Example	<pre>oModule.GetAllCategories("Far Fields", "Rectangular Plot", "Setup1 : LastAdaptive", "Infinite Sphere1")</pre>

GetAllQuantities

Gets all available quantity names in category, returned as an array of text strings.

UI Access	NA		
Parameters	Name	Type	Description
	<ReportType>	String	Report type name.
	<DisplayType>	String	Display type name.

	<SolutionName>	String	Name of solution.
	<SimValueCtxt>	Array	A context name, or array of string that encoded the contexts(I).
	<CategoryName>	String	A category name as input parameter. a category name returned in GetAllCategories() or "Variables", or "Output Variables"
Return Value	Array of text strings		

Python Syntax	GetAllQuantities(<ReportType>,<DisplayType>, <SolutionName>, <SimValueCtxt>, <CategoryName>)		
Python Example	<pre>oModule.GetAllQuantities("Far Fields", "Rectangular Plot", "Setup1 : LastAdaptive", "Infinite Sphere1", "Gain")</pre>		

GetAvailableDisplayTypes

Retrieves all supported display types in report type as an array of text strings.

UI Access	N/A		
Parameters	Name	Type	Description
	<ReportType>	String	Report type name.
Return Value	Array of text strings		

Python Syntax	<code>GetAvailableDisplayTypes(<ReportType>)</code>
Python Example	<code>oModule.GetAvailableDisplayTypes("Far Fields")</code>

GetAvailableReportTypes

Retrieves all available report types in the current Design as an array of text string.

UI Access	N/A
Parameters	None.
Return Value	Array of text strings

Python Syntax	<code>GetAvailableReportTypes()</code>
Python Example	<code>oModule.GetAvailableReportTypes()</code>

GetAvailableSolutions

Gets all available solutions in report type as an array of text strings.

UI Access	N/A		
Parameters	Name	Type	Description
	<ReportType>	String	Report type name.
Return Value	Array of text strings		

Python Syntax	<code>GetAvailableSolutions(<ReportType>)</code>
Python Example	<code>oModule.GetAvailableSolutions("Far Fields")</code>

GetDataUnits

Returns text string containing units.

Important:

This is a member function of `ISolutionDataResultComInterface` object, which is the element of the returned array from function [GetSolutionDataPerVariation](#).

UI Access	N/A		
Parameters	Name	Type	Description
	<expressionString>	String	Can be returned from <code>GetDataExpressions()</code>
Return Value	Text string of units; empty if no units		

Python Syntax	<code>GetDataUnits(<expressionString>)</code>
Python Example	<code>oModule.GetDataUnits(expressions)</code>

GetDesignVariableNames

Returns array of strings containing design variable names.

Important:

This is a member function of ISolutionDataResultComInterface object, which is the element of the returned array from function [GetSolutionDataPerVariation](#).

UI Access	NA
Parameters	NA
Return Value	Array of strings

Python Syntax	GetDesignVariableNames()
Python Example	<pre>names = oModule.GetDesignVariableNames()</pre>

GetDesignVariableUnits

Returns array of strings containing design variable units.

Important:

This is a member function of ISolutionDataResultComInterface object, which is the element of the returned array from function [GetSolutionDataPerVariation](#).

UI Access	N/A		
Parameters	Name	Type	Description
	<varName>	String	Can be returned from GetDesignVariableNames()
Return Value	Text string of units; empty if no units.		

Python Syntax	GetDesignVariableUnits(<varName>)
Python Example	<pre>units = oModule.GetDesignVariableUnits('Variable Name')</pre>

GetDesignVariableValue

Returns a design variable's value.

Important:

This is a member function of ISolutionDataResultComInterface object, which is the element of the returned array from function [GetSolutionDataPerVariation](#).

UI Access	N/A		
Parameters	Name	Type	Description
	<varName>	String	Can be returned from GetDesignVariableNames()
	<siValue>	Boolean	True to return SI-value; False to return by GetDesignVariableUnits()
Return Value	Double value		

Python Syntax	GetDesignVariableValue(<varName>, <siValue>)
Python Example	<pre>value = oModule.GetDesignVariableValue('varName',1)</pre>

GetDesignVariationKey

Returns a design's Variation Key.

Important:

This is a member function of ISolutionDataResultComInterface object, which is the element of the returned array from function [GetSolutionDataPerVariation](#).

UI Access	N/A
Parameters	N/A
Return Value	String containing variation key.

Python Syntax	GetDesignVariationKey()
Python Example	<code>oModule.GetDesignVariationKey()</code>

GetDisplayType (Maxwell)

Use: Get Display type of a report.

Command: None

Syntax: GetDisplayType <ReportName>

Return Value: String for display type

Parameters:

<ReportName>

Type: string

The name of report to find the display type for.

Example: Dim displayType

```
displayType = oModule.GetDisplayType("XY Plot 1")
```

GetImagDataValues

Returns array of imaginary data values.

Important:

This is a member function of ISolutionDataResultComInterface object, which is the element of the returned array from function [GetSolutionDataPerVariation](#).

UI Access	N/A		
Parameters	Name	Type	Description
	<expressionString>	String	Can be returned from <code>GetDataExpressions()</code>
	<siValue>	Boolean	True to return SI-value; False to return with units returned in GetSweepUnits() .
Return Value	Array of doubles		

Python Syntax	<code>GetImagDataValues(<expressionString>,<siValue>)</code>
Python Example	<code>imaginaryvalues = oModule.GetImagDataValues('expression',1)</code>

GetPerQuantityPrimarySweepValues

Returns per quantity primary sweep values.

Important:

This is a member function of ISolutionDataResultComInterface object, which is the element of the returned array from function [GetSolutionDataPerVariation](#).

UI Access	N/A		
Parameters	Name	Type	Description
	<expressionString>	String	Can be returned from <code>GetDataExpressions()</code>
	<siValue>	Boolean	True to return SI-value; False to return by GetSweepUnits() .
Return Value	Array of doubles if IsPerQuantityPrimarySweep() returned True; error if returned False		

Python Syntax	<code>GetPerQuantitySweepValues(<expressionString>, <siValue>)</code>
Python Example	<pre>sweepvalues = oModule.GetPerQuantitySweepValues('0.111,0.201,0.345,0.231', 1)</pre>

GetPropertyValue

Returns the value of a single property belonging to a specific *<PropServer>* and *<PropTab>*. This function is available with the Project, Design or Editor objects, including definition editors.

Tip: Use the script recording feature and edit a property, and then view the resulting script to see the format for that property.

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

			• LocalVariableTab ("Variables" or "Local Variables") • ProjectVariableTab ("Project variables") • ConstantsTab ("Constants") • BaseElementTab ("Symbol" or "Footprint") • ComponentTab ("General") • Component("Component") • CustomTab ("Intrinsic Variables") • Quantities ("Quantities") • Signals ("Signals")
	<PropServer>	String	An object identifier, generally returned from another script method, such as <code>CompInst@R;2;3</code>
	<PropName>	String	Name of the property.
Return Value	String value of the property.		

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

GetRealDataValues

Returns array of real data values.

Important:

This is a member function of `ISolutionDataResultComInterface` object, which is the element of the returned array from function [GetSolutionDataPerVariation](#).

UI Access	N/A		
Parameters	Name	Type	Description
	<expressionString>	String	Can be returned from <code>GetDataExpressions()</code>
	<siValue>	Boolean	True to return SI-value; False to return with units returned in GetSweepUnits() .
Return Value	Array of doubles		

Python Syntax	<code>GetRealDataValues(<expressionString>,<siValue>)</code>
Python Example	<code>realvalues = oModule.GetRealDataValues('expression',1)</code>

GetReportTraceNames

Gets the names of existing trace names in a plot.

UI Access	N/A		
Parameters	Name	Type	Description
	<PlotName>	String	Name of specified plot.
Return Value	Array of strings containing trace names.		

Python Syntax	GetReportTraceNames(<PlotName>)
Python Example	<code>oModule.GetReportTraceNames("SParameter Plot 1")</code>

GetSolutionContexts

Gets all available solution context names in a solution as an array of text strings.

UI Access	N/A		
Parameters	Name	Type	Description
	<ReportType>	String	Report type name.
	<DisplayType>	String	Display type name.
	<SolutionName>	String	Name of solution.
Return Value	Array of text strings		

Python Syntax	GetSolutionContexts(<ReportType>, <DisplayType>, <SolutionName>)
Python Example	<code>oModule.GetSolutionContexts("Far Fields", "Rectangular Plot", "Setup1:LastAdaptive")</code>

GetSolutionDataPerVariation

Obtains solution data for a given report type and solution. You must have already run a simulation.

UI Access	N/A		
Parameters	Name	Type	Description
	<reportTypeArg>	String	Report type name as input parameter.
	<solutionNameArg>	String	Solution name as input parameter.
	<simValueCtxtArg>	Structured Array	Same as ContextArray values created in the relevant CreateReport script.
	<familiesArg>	Array of Strings	Same as FamiliesArray values created in the relevant CreateReport script.
	<expressionArg>	String or Array of Strings	Text string or array of text strings; valid expression, may validate it as the data-table Y-component.
Return Value	ARRAY of ISolutionDataResultComInterface objects, containing: • GetSweepNames() • GetSweepUnits() • GetSweepValues() • IsPerQuantityPrimarySweep() • GetPerQuantityPrimarySweepValues() • IsDataComplex() • GetDataUnits() • GetRealDataValues() • GetImagDataValues()		

- [ReleaseData\(\)](#)
- [GetDesignVariableNames\(\)](#)
- [GetDesignVariableUnits\(\)](#)
- [GetDesignVariableValue\(\)](#)
- [GetDesignVariationKey\(\)](#)

Note:

- This command is *not* recordable from the UI, but its parameters are similar to CreateReport, so you may record a CreateReport script to get the parameter values.
- For the returned ISolutionDataResultComInterface object, some of its functions have an optional boolean parameter: SIValue. SIValue defaults to True. When the pass in value is True, return data values will be in Standard International values; when False, return data values will be in the current units.

Example: Freq Sweep with [1GHz, 2GHz,3GHz], GetSweepUnits("Freq") return "GHz"; GetSweepValues("Freq", True) return [1000000,2000000,3000000]; GetSweepValues("Freq", False) return [1,2,3].

Python Syntax	GetSolutionDataPerVariation(reportTypeArg, solutionNameArg, simValueCtxtArg, familiesArg, expressionArg)
Python Example	<pre>oModule = oDesign.GetModule("ReportSetup") arr = oModule.GetSolutionDataPerVariation('Modal Solution Data', 'Setup1 : Sweep', ['Domain:=' , 'Sweep'], ['Freq:=' , ['All']], ['offset:=' , ['All']], ['S (Port1,Port1) ' , 'dB(S(Port1,Port3)) '])</pre>

GetSweepNames

Returns array of text strings containing primary sweep name(s).

Important:

This is a member function of ISolutionDataResultComInterface object, which is the element of the returned array from function [GetSolutionDataPerVariation](#).

UI Access	N/A
Parameters	N/A
Return Value	Array of text strings

Python Syntax	GetSweepNames()
Python Example	<pre>sweepnames = arr[0].GetSweepNames()</pre>

GetSweepUnits

Returns text string containing units.

Important:

This is a member function of ISolutionDataResultComInterface object, which is the element of the returned array from function [GetSolutionDataPerVariation](#).

UI Access	N/A		
Parameters	Name	Type	Description
	<sweepName>	String	Primary sweep name
Return Value	Text string containing units		

Python Syntax	GetSweepUnits(<sweepName>)		
Python Example	<pre>sweepunits = oModule.GetSweepUnits('Sweep 1')</pre>		

GetSweepValues

Returns sweep values.

Important:

This is a member function of ISolutionDataResultComInterface object, which is the element of the returned array from function [GetSolutionDataPerVariation](#).

UI Access	N/A		
Parameters	Name	Type	Description
	<sweepName>	String	Primary sweep name
	<siValue>	Boolean	True to return SI-value; False to return by GetSweepUnits() .
Return Value	Array of doubles		

Python Syntax	GetSweepValues(<sweepName>, <siValue>)
Python Example	sweepvalues = oModule.GetSweepValues('Sweep 1', True)

GroupPlotCurvesByGroupingStrategy

Groups curves in a Stacked Plot automatically based on a curve grouping strategy.

UI Access	N/A		
Parameters	Name	Type	Description
	<ReportName>	String	Name of report.
	<GroupStrategy>	String	Strategy for grouping, "Single", "By Trace" or "By Units".
Return Value	None.		

Python Syntax	GroupPlotCurvesByGroupingStrategy(<ReportName>, <GroupStrategy>)
Python Example	oModule.GroupPlotCurvesByGroupingStrategy("Transient Plot 1", "By Trace")

ImportIntoReport

Imports .tab, .csv, and .dat format files into a report.

UI Access	Right-click on report name in the Project tree and select Import....

Parameters	Name	Type	Description	
	<ReportName>	String	Name of the Report	
	<FileName>	String	Path and File Name	
			.csv	Comma-delimited data file
			.tab	Tab-separated file
			.dat	Ansys plot data file
Return Value	None			

Python Syntax	ImportIntoReport (<ReportName>, <FileName>)
Python Example	oDesign.ImportIntoReport ("Plot1", "c:\report1.dat")

IsDataComplex

Returns whether an expression is complex.

Important:

This is a member function of ISolutionDataResultComInterface object, which is the element of the returned array from function [GetSolutionDataPerVariation](#).

UI Access	NA		
Parameters	Name	Type	Description
	<expressionString>	String	Can be returned from <code>GetDataExpressions()</code> .

Return Value	Boolean (True if expression is Complex data; False if not)

Python Syntax	IsDataComplex(<expressionString>)
Python Example	<code>oModule.IsDataComplex('.001,.234,.455,.434')</code>

IsPerQuantityPrimarySweep

Returns whether data expressions have different primary sweep values.

Important:

This is a member function of ISolutionDataResultComInterface object, which is the element of the returned array from function [GetSolutionDataPerVariation](#).

UI Access	N/A
Parameters	N/A
Return Value	Boolean (True if data expressions have different primary sweep values)

Python Syntax	IsPerQuantityPrimarySweep()
Python Example	<code>var = oModule.IsPerQuantityPrimarySweep()</code>

MovePlotCurvesToGroup

In a Stacked Plot move curve(s) from its stack(s) to an existing stack. Here term 'group' is synonymous to 'stack' in the context of cartesian stacked plot.

UI Access	N/A		
Parameters	Name	Type	Description
	<ReportName>	String	Name of report.
	<CurveArray>	Array	Array of curve names to move.
	<StackName>	String	Name of stack to move to.
Return Value	None.		

Python Syntax	MovePlotCurvesToGroup(<ReportName>, <CurveArray>, <StackName>)
Python Example	<code>oModule.MovePlotCurvesToGroup("XY Stacked Plot 1", ["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	Name	Type	Description
	<ReportName>	String	Name of report.

	<table><tr><td><CurveArray></td><td>Array</td><td>Array of curve names to move.</td></tr></table>	<CurveArray>	Array	Array of curve names to move.
<CurveArray>	Array	Array of curve names to move.		
Return Value	None.			

Python Syntax	MovePlotCurvesToNewGroup (<ReportName>, <CurveArray>)
Python Example	<pre>oModule.MovePlotCurvesToNewGroup ("XY Stacked Plot 1", ["R2.V : TR", "R2.I : TR"])</pre>

PasteReports

Paste copied reports to results in the current project.

UI Access	Edit > Paste
Parameters	None.
Return Value	None.

Python Syntax	PasteReports ()
Python Example	<pre>oModule.PasteReports ()</pre>

PasteTraces

Pastes copied traces to a named plot.

UI Access	Paste		
Parameters	Name	Type	Description
	<ReportName>	String	Name of plot
Return Value	None		

Python Syntax	PasteTraces (<ReportName>)
Python Example	<code>oModule.PasteTraces ("XY Plot1")</code>

Release Data

Releases all cached data. After this function is called, all subsequent function calls to the object will fail.

Important:

This is a member function of ISolutionDataResultComInterface object, which is the element of the returned array from function [GetSolutionDataPerVariation](#).

UI Access	N/A
Parameters	N/A
Return Value	N/A

Python Syntax	ReleaseData()
Python Example	<code>oModule.ReleaseData()</code>

RenameReport

Renames an existing report.

UI Access	Select a report on the Project tree, right-click and select Rename		
Parameters	Name	Type	Description
	<i><OldReportName></i>	String	Old Report Name
	<i><NewReportName></i>	String	New Report Name
Return Value	None.		

Python Syntax	RenameReport (<i><OldReportName></i> , <i><NewReportName></i>)
Python Example	<code>oModule.RenameReport("XY Plot1", "Reflection")</code>

RenameTrace

To rename a trace in a plot

UI Access	N/A		
Parameters	Name	Type	Description
	<ReportName>	String	Name of report.
	<TraceName>	String	Name of Trace
	<NewName>	String	New trace name.
Return Value	None.		

Python Syntax	RenameTrace(<ReportName>, <TraceName>, <NewName>)		
Python Example	<pre>oModule.RenameTrace ("XY Plot1", "dB(S(WavePort1,WavePort1))1", "Port1dbS")</pre>		

ResetPlotSettings

Resets plot settings to defaults.

UI Access	Right-click on a plot, select Edit > Reset Plot Settings		
Parameters	Name	Type	Description
	<PlotName>	String	Name of specified plot.
Return Value	None.		

Python Syntax	ResetPlotSettings(<PlotName>)		
Python Example	oModule.ResetPlotSettings("Differential S-parameters")		

SavePlotSettingsAsDefault

Saves report plot settings as default.

UI Access	Report Templates > Save Settings as Default		
Parameters	Name	Type	Description
	<PlotName>	String	Name of plot to use for plot defaults.
Return Value	None.		

Python Syntax	SavePlotSettingsAsDefault("<PlotName>")
Python Example	oModule.SavePlotSettingsAsDefault ("XY Plot1")

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	<code>oModule.UpdateAllReports()</code>

UpdateReports

Updates specified reports.

UI Access	N/A		
Parameters	Name	Type	Description
	<ReportNames>	Array	Array of strings containing report names.
Return Value	None.		

Python Syntax	UpdateReports(<ReportNames>)
Python Example	<code>oModule.UpdateReports(["XY Plot 1", "XY Plot 4"])</code>

UpdateTraces

Update the traces in a report for which traces are not automatically updated by the Report Traces dialog box, Update Report, Real Time selection.

UI Access	In Report dialog, click Apply Traces button		
Parameters	Name	Type	Description
	<ReportName>	String	Name of Report.
	<TraceNames>	Array	Array of strings containing trace names.

	<SolutionName>	String	Name of the solution.
	<ContextArray>	Array	Context for which the expression is being evaluated. This can be an empty string if there is no context. <code>Array("Domain:=", <DomainType>)</code> <DomainType> ex. "Sweep" or "Time" <code>Array("Context:=", <GeometryType>)</code> <GeometryType> ex. "Infinite Spheren", "Spheren", "Polylinen"
	<FamiliesArray>	Array	Contains sweep definitions for the report. <code>Array("<VariableName>:= ", <ValueArray>)</code> <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. <code>Array("X Component:=", <VariableName>, "Y Component:=", <VariableName> <ReportQuantityArray>)</code> <ReportQuantityArray> ex. Array("dB(S(Port1, Port1))")
	<ExtTraceInfo>	Array	Optional. Array defines extended trace information.
Return Value	None.		

Python Syntax	UpdateTraces(<ReportName>, <SolutionName>, <ContextArray>, <FamiliesArray>, <ReportDataArray>)
Python Example	<pre> oModule.UpdateTraces("XY Plot 1", ["NEG1.VAL"], "TR4", ["NAME:Context", "SimValueContext:=", [2,0,2,0,False,False, -1,1,0,1,1,"",0,0,"CG",False,"0","KP",False,"0","MH",False, "100","TE",False,"100s","TH",False,"40", "TS",False,"0ns","UF",False, "0","WT",False,"0","WW",False,"100"]], ["Spectrum:=", ["All"]], ["X Component:=", "Spectrum", "Y Component:=", ["mag(NEG1.VAL)"]], []) </pre>

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", ["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",</pre>

	<pre>"9.4GHz", "9.45GHz", "9.5GHz", "9.55GHz", "9.6GHz", "9.65GHz", "9.7GHz", "9.9GHz", "9.95GHz", "10GHz"], "offset:=",["All"]])</pre>

UnGroupPlotCurvesInGroup

From a Stacked Plot, ungroups curves in a stack.

UI Access	N/A		
Parameters	Name	Type	Description
	<ReportName>	String	Name of report.
	<GroupName>	String	Stack group name.
Return Value	None.		

Python Syntax	UnGroupPlotCurvesInGroup(<ReportName>, <GroupName>)
Python Example	<code>oModule.UngroupPlotCurvesInGroup("S Parameter Plot 3", "Stack 1")</code>

This page intentionally
left blank.

13 - Boundary and Excitation Module Script Commands in Maxwell

Boundary and excitation commands should be executed by the **BoundarySetup** module.

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

```
oModule.CommandName <args>
```

Conventions Used in this Chapter

<BoundName>

Type: string.

Name of a boundary.

<BoundNameArray>

Type: Array of strings

An array of the names in a boundary/excitation group.

<AssignmentObjects>

Type: Array of strings.

An array of object names.

<AssignmentFaces>

Type: Array of integers.

An array of face IDs. The ID of a face can be determined through the user interface using the **Modeler>Measure>Area** command. The face ID is given in the **Measure Information** dialog box.

<LineEndPoint>

```
    Array(<double>, <double>, <double>)  
<CoordSysArray>  
    Array("NAME:CoordSysVector",  
        "Origin:=", <CoordSysPoint>,  
        "UPos:=", <LineEndPoint>)
```

Related Topics

[General Commands Recognized by the Boundary/Excitations Module](#)

[Script Commands for Creating and Modifying Boundaries in Maxwell](#)

[Script Commands for Creating and Modifying Excitations in Maxwell](#)

General Commands Recognized by the Boundary/Excitations Module

[AddAssignmentToBoundary](#)

[DeleteAllBoundaries](#)

[DeleteAllExcitations](#)

[DeleteBoundaries](#)

[GetBoundaryAssignment](#)

[GetBoundaries](#)

[GetBoundariesOfType](#)

[GetCoreLossEffect](#)

[GetCoreLossEffectOnField](#)

[GetDisplacementCurrent](#)

[GetEddyEffect](#)

[GetExcitations](#)

[GetExcitationsOfType](#)

[GetHybridRegions](#)

[GetHybridRegionsOfType](#)

[GetNumBoundaries](#)

[GetNumBoundariesOfType](#)

[GetNumExcitations](#)

[GetNumExcitationsOfType](#)

[ReassignBoundary](#)

[RemoveAssignmentFromBoundary](#)

[RenameBoundary](#)

[ReprioritizeBoundaries](#)

AddAssignmentToBoundary

Adds a new geometry assignment to a boundary.

UI Access	N/A		
Parameters	Name	Type	Description
	<Assignment>	Array	Structured array. Array("Name:<BoundName>", "Objects:=", <AssignmentObjects>,

	<table><tr><td></td><td></td><td>"Faces:=", <AssignmentFaces>)</td></tr></table>					"Faces:=", <AssignmentFaces>)
		"Faces:=", <AssignmentFaces>)				
Return Value	None.					

Python Syntax	AddAssignmentToBoundary(<Assignment>)
Python Example	<pre>oModule.AddAssignmentToBoundary((["NAME:PerfE1", "Faces:=", [12]])</pre>

DeleteAllBoundaries

Deletes all boundaries.

UI Access	[product] > Boundaries > Delete All
Parameters	None.
Return Value	None.

Python Syntax	DeleteAllBoundaries()
Python Example	<pre>oModule.DeleteAllBoundaries()</pre>

DeleteAllExcitations

Deletes all excitations.

UI Access	[product] > Excitations > Delete All
Parameters	None.
Return Value	None.

Python Syntax	DeleteAllExcitations()
Python Example	<code>oModule.DeleteAllExcitations()</code>

DeleteBoundaries

Deletes the specified boundaries and excitations.

UI Access	Delete command in the List dialog box. Click [product] > List to open the List dialog box.		
Parameters	Name	Type	Description
	<NameArray>	Array	Array of boundary condition names.
Return Value	None.		

Python Syntax	DeleteBoundaries(<NameArray>)
Python Example	<code>oModule.DeleteBoundaries(["PerfE1", "WavePort1"])</code>

GetBoundaryAssignment

Gets a list of face IDs associated with the given boundary or excitation assignment.

UI Access	N/A		
Parameters	Name	Type	Description
	<BoundaryName>	String	Name of the specified boundary or excitation.
Return Value	Array of face IDs or object IDs.		

Python Syntax	GetBoundaryAssignment(<BoundaryName>)
Python Example	<code>oModule.GetBoundaryAssignment("Rad1")</code>

GetBoundaries

Gets boundary names in the current design.

UI Access	N/A
Parameters	None.
Return Value	Array of boundary names.

Python Syntax	GetBoundaries()
Python Example	<code>oModule.GetBoundaries()</code>

GetBoundariesOfType

Gets boundary names of the given type.

UI Access	N/A		
Parameters	Name	Type	Description
	<BoundaryType>	String	Name of boundary type.
Return Value	Array of boundary names of the given type.		

Python Syntax	GetBoundariesOfType(<BoundaryType>)
Python Example	oModule.GetBoundariesOfType("PerfectE")

GetCoreLossEffect

Use: Returns whether the core loss is included in the simulation for the object. Applicable to 2D and 3D Transient and to 3D AC Magnetic solution types.

Command: None

Syntax: GetCoreLossEffect(<ObjectName>)

Return Value: Returns true or false based on the check box selection. True if checked.

Parameters: <ObjectName>

Type:<string>

Object name.

For example: "inner_arm"

Example: `oModule.GetCoreLossEffect("inner_arm")`

GetCoreLossEffectOnField

Use: Returns whether the core loss effect on the field is being considered. Applicable to 2D and 3D Transient solution types.

Command: None

Syntax: `GetCoreLossEffectOnField()`

Return Value: Returns true or false based on the check box selection. True if checked.

Parameters: None

Example: `oModule.GetCoreLossEffectOnField()`

GetDisplacementCurrent

Use: Returns whether the displacement current calculation is activated for the object. Applicable to AC Magnetic solution type.

Command: None

Syntax: `GetDisplacementCurrent(<ObjectName>)`

Return Value: Returns true or false based on the check box selection. True if checked.

Parameters: `<ObjectName>`

Type: `<string>`

Object name.

For example: `"coil1"`

Example: `oModule.GetDisplacementCurrent("coil1")`

GetEddyEffect

Use: Returns whether the eddy effect is activated for the object. Applicable to 2D and 3D Transient and to AC Magnetic solution types.

Command: None

Syntax: GetEddyEffect(<ObjectName>)

Return Value: Returns true or false based on the given object name. True if checked.

Parameters: <ObjectName>

Type:<string>

Object name.

For example: "inner_arm"

Example: oModule.GetEddyEffect("outer_arm")

GetExcitations

Gets excitation port and terminal names for a model.

UI Access	N/A
Parameters	None.
Return Value	Array of excitation name paired with excitation type.

Python Syntax	GetExcitations()
Python Example	oModule.GetExcitations()

GetExcitationsOfType

Gets excitation names of the given type.

UI Access	N/A		
Parameters	Name	Type	Description
	<ExcitationType>	String	Name of excitation type.
Return Value	Array of excitation names of the given type.		

Python Syntax	GetExcitationsOfType(<ExcitationType>)
Python Example	<code>oModule.GetExcitationsOfType("Wave Port")</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>

GetNumBoundariesOfType

Gets the number of boundaries of the given type.

UI Access	N/A		
Parameters	Name	Type	Description
	<BoundaryType>	String	Specified boundary type.
Return Value	Integer number of boundaries.		

Python Syntax	GetNumBoundariesOfType(<BoundaryType>)
Python Example	<code>oModule.GetNumBoundariesOfType("PerfectE")</code>

GetNumExcitations

Gets the number of excitations in a design.

UI Access	N/A
Parameters	None.
Return Value	Integer number of excitations.

Python Syntax	GetNumExcitations()
Python Example	<code>oModule.GetNumExcitations()</code>

GetNumExcitationsOfType

Gets the number of excitations of the given type, including all defined modes and terminals of ports.

UI Access	N/A		
Parameters	Name	Type	Description
	<ExcitationType>	String	Specified type of excitation.
Return Value	Integer number of excitations.		

Python Syntax	GetNumExcitationsOfType(<ExcitationType>)
Python Example	<code>oModule.GetNumExcitationsOfType("Voltage")</code>

ReassignBoundary

Specifies a new geometry assignment for a boundary.

UI Access	Right-click Boundaries > Reassign or Excitations > Reassign		
Parameters	Name	Type	Description
	<AssignmentArray>	Array	Structured array. Array("Name:<BoundName>", "Objects:=", <AssignmentObjects>, "Faces:=", <AssignmentFaces>)

Return Value	None.

Python Syntax	ReassignBoundary(<AssignmentArray>)
Python Example	<pre>oModule.ReassignBoundary(["NAME:PerfH12", "Faces:=", [17]])</pre>

RemoveAssignmentFromBoundary

Removes a geometry assignment from a boundary.

UI Access	Right-click on a boundary or an excitation, select Remove from Assignment .		
Parameters	Name	Type	Description
	<AssignmentArray>	Array	Structured array. Array("Name:<BoundName>", "Objects:=", <AssignmentObjects>, "Faces:=", <AssignmentFaces>)
Return Value	None.		

Python Syntax	<code>RemoveAssignmentFromBoundary(<AssignmentArray>)</code>
Python Example	<pre>oModule.RemoveAssignmentFromBoundary(["NAME:Rad1", "Faces:=", [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	Type	Description
	<OldName>	String	Name of the boundary to be renamed.
	<NewName>	String	New name for the boundary.
Return Value	None.		

Python Syntax	<code>RenameBoundary(<OldName>, <NewName>)</code>
Python Example	<pre>oModule.RenameBoundary("Rad2", "Rad3")</pre>

ReprioritizeBoundaries

Specifies the order in which the boundaries and excitations are recognized by the solver. The first boundary in the list has the highest priority.

Note: this command is only valid if all defined boundaries and excitations appear in the list. All ports must be listed before any other boundary type.

UI Access	[product] > Boundaries > Reprioritize		
Parameters	Name	Type	Description
	<NewOrderArray>	Array	Structured array. <code>Array("NAME:NewOrder", <BoundName>, <BoundName>, ...)</code>
Return Value	None.		

Python Syntax	ReprioritizeBoundaries(<NewOrderArray>)
Python Example	<pre>oModule.ReprioritizeBoundaries(["NAME:NewOrder", "Imped1", "PerfE1", "PerfH1"])</pre>

Script Commands for Creating and Modifying Boundaries in Maxwell

Following are script commands for creating and modifying boundaries that are recognized by the "BoundarySetup" module. In the following commands, all named data can be included/excluded as desired and may appear in any order.

- [AssignZeroTangentialHField](#)
- [EditZeroTangentialHField](#)
- [AssignFluxTangential](#)
- [EditFluxTangential](#)
- [AssignInsulating](#)
- [EditInsulating](#)
- [AssignSymmetry](#)
- [EditSymmetry](#)
- [AssignIndependent](#)
- [AssignDependent \(2D\)](#)
- [AssignDependent \(3D\)](#)
- [EditIndependent](#)
- [EditDependent](#)
- [AssignRadiation](#)
- [EditRadiation](#)
- [AssignImpedance](#)
- [EditImpedance](#)
- [AssignTangentialHField](#)
- [EditTangentialHField](#)
- [AssignVectorPotential](#)
- [EditVectorPotential](#)

- [AssignCylindricalHField](#)
- [EditCylindricalHField](#)
- [AssignResistiveSheet](#)
- [EditResistiveSheet](#)
- [AssignTouching](#)
- [EditTouching](#)
- [AssignThinLayer](#)
- [EditThinLayer](#)

AssignCylindricalHField

Use: Creates an H-Field boundary on a cylindrical face.

Command: Maxwell > Boundaries > Assign > Tangential H Field

Syntax: AssignCylindricalHField <CylindricalHFieldArray>

Return Value: None

Parameters: <CylindricalHFieldArray>

```
Array("NAME:<BoundName>",  
      "ComponentPhiReal:=", <value>,  
      "ComponentPhiImg:=", <value>,  
      "ComponentZReal:=", <value>,  
      "ComponentZImg:=", <value>,  
      "ReverseZ:=", <bool>,  
      "Origin:=", <LineEndPoint>)  
      "Objects:=", <AssignmentObjects>,  
      "Faces:=", <AssignmentObjects>)
```

AssignDependent (2D)

Creates a dependent boundary.

UI Access	Maxwell2D > Boundaries > Assign > Matching > Dependent		
Parameters	Name	Type	Description
	<BoundName>	String	Name of boundary to be designated as dependent
	<Edges>	Array	List of IDs of the edges which the boundary is assigned to
	or		
	<Objects>	Array	List of name(s) of the polylines the boundary is assigned to.
	<ReverseU>	Boolean	Reverse the direction of the vector
	<Independent>	String	Name of Independent boundary
	<SameasMaster>	Boolean	<code>True</code> if dependent and independent boundaries have the same magnitude and oscillate in phase; <code>False</code> if the dependent boundary field has the same magnitude as the independent boundary field and oscillate 180 degrees out of phase.
Return Value	None.		

Python Syntax	AssignDependent ([DependentArray])
Python Example	Example command applied to polyline: <pre>oModule.AssignDependent (["NAME:Dependent1", "Objects:=" , ["Dependent"], "ReverseU:=" , False, "Independent:=" , "Independent1",</pre>

```

    "SameAsMaster:="      , False
  })

Example command applied to edge:

oModule.AssignDependent(
  ["NAME:Dependent1",
   "Edge="              , [8],
   "ReverseU:="         , True,
   "Independent:="      , "Independent1",
   "SameAsMaster:="     , False
  ])

```

AssignDependent (3D)

Creates a dependent boundary for a 3D design.

UI Access	Maxwell 3D > Boundaries > Assign > Matching > Dependent		
Parameters	Name	Type	Description
	<BoundName>	String	Name of boundary to be designated as dependent
	<Faces>	Array	List of IDs of the faces which the boundary is assigned to
	or		
	<Objects>	Array	List of name(s) of the sheets the boundary is assigned to
	<CoordSysArray>	Array	Includes "NAME:CoordSysVector", "Coordinate System=:, string", "Origin:=", <list of coordinates>, "UPos:=", <list of coordinates>
	<ReverseV>	Boolean	Reverse the direction of the vector
	<Independent>	String	Name of Independent boundary
	<RelationIsSame>	Boolean	True if dependent and independent boundaries have the same magnitude and oscillate in phase; False if the dependent boundary field has the same magnitude as the independent boundary field and oscillate 180 degrees out of

	phase.		
Return Value	None		

Python Syntax	AssignDependent ([DependentArray])
Python Example	Example command applied to sheet: <pre>oModule.AssignDependent(["NAME:Dependent1", "Objects:=", ["DependentSheet"], ["NAME:CoordSysVector", "Coordinate System:=", "Global", "Origin:=", ["0mm", "0mm", "0mm"], "UPos:=", ["0mm", "60mm", "0mm"], "ReverseV:=", True, "Independent:=", "Independent1", "RelationIsSame:=", False]])</pre> Example command applied to face: <pre>oModule.AssignDependent(["NAME:Dependent1", "Faces:=", [131], ["NAME:CoordSysVector", "Coordinate System:=", "Global", "Origin:=", ["0mm", "0mm", "0mm"], "UPos:=", ["0mm", "16mm", "0mm"]], "ReverseV:=", False,])</pre>

	<pre> "Independent:=" , "Independent1", "RelationIsSame:=" , False]) </pre>

AssignFluxTangential

Assigns a Flux Tangential boundary condition to the selected face(s).

UI Access	Boundaries > Assign > Flux Tangential		
Parameters	Name	Type	Description
	<NAME>	string	Name of the flux tangential boundary
	<Faces>	list	List of face identifiers
Return Value	None		

Python Syntax	AssignFluxTangential (<NAME>, <Faces>)
Python Example	<pre> oModule.AssignFluxTangential (["NAME:FluxTangential2", "Faces:=", [285,313]]) </pre>

AssignImpedance (Maxwell)

Use: Creates an impedance boundary.

Command: Maxwell3D or Maxwell2D > Boundaries > Assign > Impedance

Syntax: AssignImpedance <ImpedanceArray>

Return Value: None

Parameters: <ImpedanceArray>

```
Array("NAME:<BoundName>",  
      "Faces:=", <AssignmentFaces>  
      "Conductivity:=", <value>,  
      "Permeability:=", <value>,  
      "UseMaterial:=", <true or false>,  
      "MaterialName:=", <Material name>,  
      "NonlinearCoefficient:=", <number>,  
      "IsPermeabilityNonlinear:=", <true or false>,  
      "Objects:=", <AssignmentObjects>,
```

AssignIndependent (Maxwell)

Creates an independent boundary.

UI Access	Maxwell 3D or Maxwell 2D > Boundaries > Assign > Matching > Independent		
Parameters	Name	Type	Description
	<BoundName>	String	Name of boundary to be designated as independent
	<Edge ID list>	Array	For 2D, the IDs of the edges that the boundary is assigned to. The value is a list of edge numbers.
	<AssignmentFaces>		For 3D, a list of faces that the boundary is assigned to
	or		

	<Objects>	Array	For 2D or 3D, a list of name(s) of the objects the boundary is assigned to. For 2D, objects are Polyline. For 3D, objects can be sheets.
	<CoordSysVector>	Array	Includes "NAME:CoordSysVector", "CoordinateSystem:=" <string>, "Origin:=", <list of coordinates>, "UPos:=", <list of coordinates>
	<ReverseV>	Boolean	Reverse the direction of the vector
Return Value	None		

Python Syntax	AssignIndependent (<IndependentArray>)
Python Example	Example AssignIndependent 2D: <pre>oModule.AssignIndependent(["NAME:Independent1", "Edges:=" , [20], "ReverseV:=" , False])</pre> Example command applied to polyline: <pre>oModule.AssignIndependent(["NAME:Independent1", "Objects:=" , ["Independent"], "ReverseV:=" , False])</pre> Example AssignIndependent 3D: <pre>oModule.AssignIndependent(["NAME:Independent1", "Faces:=" , [25], ["NAME:CoordSysVector", "CoordinateSystem:=", "Global", "Origin:=" , ["0mm", "0mm", "82.1090449266692mm"],</pre>

	<pre> "UPos:=" , ["0mm", "60mm", "82.1090449266692mm"]], "ReverseV:=" , False }) Example command applied to sheet: oModule.AssignIndependent(["NAME:Independent1", "Objects:=" , ["IndependentSheet"], ["NAME:CoordSysVector", "CoordinateSystem:=", "Global", "Origin:=" , ["0mm", "0mm", "0mm"], "UPos:=" , ["60mm", "0mm", "0mm"]], "ReverseV:=" , True]) </pre>

AssignIntegratedZeroTangentialHField

Creates an integrated zero tangential H-Field boundary (Maxwell 3D Magnetostatic, Eddy Current, and Transient designs only).

UI Access	Maxwell 3D > Boundaries > Assign > Integrated Zero Tangential H Field		
Parameters	Name	Type	Description
	<IntegratedZeroTangentialHFieldArray>	Array	Structured array. Array("NAME:<BoundName>", "Faces:=", <Array of face IDs>,
Return Value	None.		

Python Syntax	AssignIntegratedZeroTangentialHField <IntegratedZeroTangentialHFieldArray>
Python Example	<pre>oModule.AssignIntegratedZeroTangentialHField(["NAME:IntegratedZeroTangentialHField1", "Faces:=", [7]]) </pre>

AssignInsulating

Use: Creates an insulating boundary.

Command: Maxwell3D > Boundaries > Assign > Insulating

Syntax: AssignInsulating <InsulatingArray>

Return Value: None

Parameters: <InsulatingArray>

```
Array("Name:<BoundName>",
    "Objects:=", <AssignmentObjects>,
    "Faces:=", <AssignmentFaces>
    "Permittivity:=", <double value>
    "Thickness:=", <value><unit>

```

AssignRadiation (Maxwell)

Use: Creates a radiation boundary.

Command: Maxwell3D>Boundaries>Assign>Radiation

Syntax: AssignRadiation <RadiationArray>

Return Value: None

Parameters: <RadiationArray>

```

Array("NAME:<BoundName>",
"Objects:=", <AssignmentObjects>,
"Faces:=", <AssignmentFaces>)

```

Example: oModule.AssignRadiation Array("NAME:Radiation1", _ "Faces:=", Array(12, 11))

AssignResistiveSheet

Use: For Maxwell 3D Magnetostatic, AC Magnetic and Transient designs, creates a resistive sheet boundary between two conductors.

For 3D AC Magnetic and Transient designs, the user specifies a Boundary name, boundary face ID list, and a resistance value with unit.

For 3D Magnetostatic designs, the user specifies a Boundary name, boundary face ID list, and nonlinear anode and cathode coefficients.

Note: For more information on the nonlinear resistive sheet boundary condition, see *Assigning a Resistive Sheet Boundary for the 3D Magnetostatic Solver* in the Maxwell Help.

UI Access	Maxwell 3D > Boundaries > Assign > Resistive Sheet		
Parameters	Name	Type	Description
	<Name>	String	Boundary name
	<Objects>	List	Name of objects or faces to apply boundary condition; it can be a list of names or integers.
	<Nonlinear>	Boolean	If set to True, designates a nonlinear resistive sheet boundary condition, and user must specify A, B, C, and D inputs (double values) for anode and cathode coefficients (see examples).
	<Resistance>	String	Specifies resistance value with unit; not available if Nonlinear set to True.

Return Value	None

For 3D Magnetostatic, AC Magnetic, and Transient designs

Python Syntax	AssignResistiveSheet ([<NAME>, <Objects>, <Nonlinear>,<Resistance>])
Python Example	<pre>oModule.AssignResistiveSheet (["NAME:ResistiveSheet2", "Faces:=", [93], "Nonlinear:=", False, "Resistance:=", "2ohm"]) oModule.AssignResistiveSheet (["NAME:ResistiveSheet1", "Objects:=", ["Circle1"], "Nonlinear:=", True, "AnodeParA:=", "300000000", "AnodeParB:=", "5", "AnodeParC:=", "1100000000000000", "AnodeParD:=", "2", "CathodeParA:=", "300000000", "CathodeParB:=", "10", "CathodeParC:=", "1100000000000000", "CathodeParD:=", "2"])</pre>

AssignSymmetry (Maxwell)

Use: Creates a symmetry boundary.

Command: Maxwell3D or Maxwell2D>Boundaries>Assign>Symmetry

Syntax: AssignSymmetry <SymmetryArray>

Return Value: None

Parameters: <SymmetryArray>

```
Array( "NAME:<BoundName>",  
      "IsOdd:=", <bool>, //true for odd, false for even  
      "Objects:=", <AssignmentObjects>,  
      "Faces:=", <AssignmentFaces>)
```

Example: oModule.AssignSymmetry Array("NAME:Symmetry1", _ "IsOdd:=", true, "Faces:=", Array(35))

AssignTangentialHField

Use: Creates an H-Field boundary on a planar face.

Command: Maxwell3D>Boundaries>Assign>Tangential H Field

Syntax: AssignTangentialHField <TangentialHFieldArray>

Return Value: None

Parameters: <TangentialHFieldArray>

```
Array("NAME:<BoundName>",  
      "ComponentXReal:=", <value>,  
      "ComponentXImg:=", <value>,  
      "ComponentYReal:=", <value>,  
      "ComponentYImg:=", <value>,  
      <CoordSysArray>
```

```
"ReverseV:=", <bool>,
"Objects:=", <AssignmentObjects>,
"Faces:=", <AssignmentObjects>)
```

AssignThinLayer

Use: For Maxwell 3D AC Conduction, 3D DC Conduction, 3D Magnetostatic, 3D AC Magnetic, and 3D Transient designs, creates a thin layer boundary between two conductors.

UI Access	Maxwell 3D > Boundaries > Assign > Thin Layer		
Parameters	Name	Type	Description
	<ThinLayerArray>	Array	- For 3D AC Conduction designs: Array containing the boundary name, boundary face ID array, and permittivity, conductivity, thickness value with unit. - For 3D DC Conduction designs: Array containing the boundary name, boundary face ID array, and conductivity and thickness value with unit. - For 3D Magnetostatic: Create an air gap boundary between two non-conductors or designate a thin ferromagnet. Array containing the boundary name, relative permeability (no air gap) and thickness value with unit. - For 3D Transient (T-Omega only): Create an air gap boundary between two non-conductors. Array containing the boundary name, and thickness value with unit. - For 3D AC Magnetic: Array containing the boundary name, AirGap (bool); If AirGap=False, include boundary face ID array, conductivity, relative permeability, and thickness value with unit. If AirGap=true, include only thickness value with unit.
Return Value	None		

For 3D AC Conduction designs

Python Syntax	AssignThinLayer ([<NAME>, <Faces>, <Permittivity>, <Conductivity>, <Thickness>])
Python Example	<pre>oModule.AssignThinLayer(["NAME:ThinLayer1", "Faces:=", [40], "Permittivity:=", "1" "Conductivity:=", "1S_per_m" "Thickness:=", "1mm"])</pre>

For 3D DC Conduction designs

Python Syntax	AssignThinLayer ([<NAME>, <Faces>, <Conductivity>, <Thickness>])
Python Example	<pre>oModule.AssignThinLayer(["NAME:ThinLayer1", "Faces:=", [40], "Conductivity:=", "1S_per_m" "Thickness:=", "1mm"])</pre>

For 3D Magnetostatic designs

Python Syntax	AssignThinLayer ([<NAME>, <Objects>, <AirGap>, <RelativePermeability>, <Thickness>])
Python Example	With air gap: <pre>oModule.AssignThinLayer([</pre>

	<pre> "NAME:ThinLayer1", "Objects:=", ["Rectangle1"], "AirGap:=", True, "Thickness:=", "1mm"]) Without air gap: oModule.AssignThinLayer(["NAME:ThinLayer1", "Objects:=", ["Rectangle1"], "AirGap:=", False, "RelativePermeability:=", "2", "Thickness:=", "2mm"]) </pre>

For 3D Transient designs

Python Syntax	AssignThinLayer ([<NAME>, <Objects>,<Thickness>])
Python Example	<pre> oModule.AssignThinLayer(["NAME:ThinLayer1", "Objects:=", ["Rectangle1"], "Thickness:=", "1mm"]) </pre>

For 3D AC Magnetic designs

Python Syntax	AssignThinLayer ([<NAME>, <Faces>, <AirGap>, <Conductivity>, <RelativePermeability>, <Thickness>])
Python Example	<pre> oModule.AssignThinLayer(["NAME:ThinLayer1", </pre>

	<pre>"Faces:=", [40], "AirGap:=", False, "Conductivity:=", "50000000000S_per_m" "RelativePermeability:=", "1" "Thickness:=", "0.02mm"])</pre>

AssignTouching

Use: For Maxwell 3D Transient designs with translational motion with a moving part touching stationary part, assigns a touching boundary condition to a face of the moving part that touches a stationary part.

UI Access	Maxwell 3D>Boundaries>Assign>Touching		
Parameters	Name	Type	Description
	<NAME>	string	The name of the boundary
	<Faces>	list	Array of face IDs
Return Value	None		

Python Syntax	AssignTouching (<NAME>, <Faces>)
Python Example	<pre>oModule.AssignTouching(["NAME:Touching3", "Faces:=", [210]])</pre>

AssignVectorPotential

Use: For Maxwell 2D AC Magnetic, Magnetostatic, and Transient designs, assigns a vector potential boundary condition to the specified edges.

UI Access	Maxwell 2D > Boundaries > Assign > Vector Potential		
Parameters	Name	Type	Description
	<NAME>	string	The name of the boundary
	<Edges>	list	Array of edge IDs
	<Phase>	string	Phase angle with unit (for 2D AC Magnetic designs only)
	<Value>	string	Vector potential value in weber/m
	<CoordinateSystem>	string	Name of the reference coordinate system if the vector potential value is a spatial-dependent function such as z+2
Return Value	None		

Python Syntax	AssignVectorPotential (<NAME>, <Edges>, <Phase>, <Value>, <CoordinateSystem>)
Python Example	<pre>oModule.AssignVectorPotential (["NAME:VectorPotential2", "Edges:=", [21], "Phase:=", "60deg", "Value:=", "Z+2", "CoordinateSystem:=", "Global"])</pre>

AssignIntegratedZeroTangentialHField

Creates an integrated zero tangential H-Field boundary (Maxwell 3D Magnetostatic, Eddy Current, and Transient designs only).

UI Access	Maxwell 3D > Boundaries > Assign > Integrated Zero Tangential H Field		
Parameters	Name	Type	Description
	<IntegratedZeroTangentialHFieldArray>	Array	Structured array. Array("NAME:<BoundName>", "Faces:=", <Array of face IDs>,
Return Value	None.		

Python Syntax	AssignIntegratedZeroTangentialHField <IntegratedZeroTangentialHFieldArray>
Python Example	<pre>oModule.AssignIntegratedZeroTangentialHField(["NAME:IntegratedZeroTangentialHField1", "Faces:=", [7]])</pre>

EditCylindricalHField

Use: Edits a cylindrical H-Field boundary.

Command: Double-click the boundary in the project tree to edit it.

Syntax:EditCylindricalHField <BoundName> <CylindricalHFieldArray>

Return Value: None

Script Commands for Creating and Modifying Excitations

EditDependent (Maxwell)

Use: Modifies a dependent boundary.

Command: Double-click the boundary in the project tree to modify its settings.

Syntax: EditDependent <BoundName> <DependentArray>

Return Value: None

Note: Refer to [AssignDependent \(2D\)](#) and [AssignDependent \(3D\)](#) examples for syntax details.

EditFluxTangential

Edits a Flux Tangential boundary condition name.

UI Access	Double-click the Flux Tangential Boundaries item in the Project tree to edit the name.		
Parameters	Name	Type	Description
	<NAME>	string	The edited name
Return Value	None		

Python Syntax	EditFluxTangential_1 (<NAME>)
Python Example	<pre>oModule.EditFluxTangential("FluxTangential1", ["NAME:FluxTangential_New"])</pre>

EditImpedance (Maxwell)

Use: Edits an impedance boundary.

Command: Double-click the boundary in the project tree to edit it.

Syntax: EditImpedance <BoundName> <ImpedanceArray>

Return Value: None

Parameters: <BoundName>, string name of boundary

```
<ImpedanceArray>  
Array("NAME:<BoundName>",  
"Faces:=", <AssignmentFaces>)  
"Conductivity:=", <value>,  
"Permeability:=", <value>,  
"UseMaterial:=", <true or false>,  
"MaterialName:=", <Material name>,  
"NonlinearCoefficient:=", <number>,  
"IsPermeabilityNonlinear:=", <true or false>,  
"Objects:=", <AssignmentObjects>,
```

EditIndependent (Maxwell)

Use: Modifies an independent boundary.

Command: Double-click the boundary in the Project Manager to modify its settings.

Syntax: Edit <BoundName> <PrimaryArray>

Return Value: None

Note: Refer to [AssignIndependent](#) examples for syntax details.

EditIntegratedZeroTangentialHField

Edits an integrated zero tangential H-Field boundary (Maxwell 3D Magnetostatic, Eddy Current, and Transient designs only).

UI Access	Double-click the boundary in the Project Manager tree to edit it		
Parameters	Name	Type	Description
	<IntegratedZeroTangentialHFieldArray>	Array	Structured array. <code>Array ("NAME : <BoundName>")</code>
Return Value	None.		

Python Syntax	EditIntegratedZeroTangentialHField <IntegratedZeroTangentialHFieldArray>
Python Example	<pre>oModule.EditIntegratedZeroTangentialHField(["NAME: IntegratedZeroTangentialHField1"])</pre>

EditInsulating

Use: Edits an insulating boundary.

Command: Double-click the boundary in the project tree to edit it.

Syntax: EditInsulating <BoundName> <InsulatingArray>

Return Value: None

Parameters: <BoundName> , string name of boundary

```
<InsulatingArray>
  Array("Name:<BoundName>",
  "Permittivity:=", <double value>
  "Thickness:=", <value><unit>
```

EditRadiation (Maxwell)

Use: Edits a radiation boundary.

Command: Double-click the boundary in the project tree to edit it.

Syntax:EditRadiation <BoundName> <RadiationArray>

Return Value: None

EditResistiveSheet

For Maxwell 3D Magnetostatic, AC Magnetic, Transient designs, edits a resistive sheet boundary. A nonlinear boundary condition can be used in a Maxwell 3D Magnetostatic design.

Note: For more information on the nonlinear resistive sheet boundary condition, see *Assigning a Resistive Sheet Boundary for the 3D Magnetostatic Solver* in the Maxwell Help.

UI Access	Double-click the boundary in the project tree to edit it.		
Parameters	Name	Type	Description
	<BoundName>	String	The name of the boundary.
	<Res-istiveSheetArray>	Array	For 3D AC Magnetic and Transient designs: Array containing the boundary name, and resistance value with unit. For 3D Magnetostatic designs: Array containing the boundary name, and non-

			linear boundary condition. If the nonlinear boundary condition is selected, the user has to specify inputs (double values) for anode and cathode coefficients.
Return Value	None		

For 3D Magnetostatic, AC Magnetic, and Transient designs

Python Syntax	EditResistiveSheet (<BoundName>, [<NAME>, <Nonlinear>, <Resistance>])
Python Example	<pre>oModule.EditResistiveSheet("ResistiveSheet1", ["NAME:ResistiveSheet1", "Nonlinear:=", False, "Resistance:=", "3ohm"])</pre> With a nonlinear boundary condition: <pre>oModule.EditResistiveSheet("ResistiveSheet1", ["NAME:ResistiveSheet1", "Nonlinear:=", True, "AnodeParA:=", "2000000000", "AnodeParB:=", "7.1", "AnodeParC:=", "12000000000000000", "AnodeParD:=", "6", "CathodeParA:=", "1000000000", "CathodeParB:=", "12", "CathodeParC:=", "12000000000000000", "CathodeParD:=", "4"])</pre>

EditSymmetry

Modifies a symmetry boundary.

UI Access	Double-click the symmetry boundary in the project tree to edit its settings.		
Parameters	Name	Type	Description
	<BoundaryName>	String	Name of the symmetry boundary to be edited.
	<SymmetryArray>	Array	Structured array. <code>Array("NAME:<BoundName>", "IsPerfectE:=", <boolean>)</code>
Return Value	None.		

Python Syntax	EditSymmetry(<BoundaryName>, <SymmetryArray>)
Python Example	<pre>oModule.EditSymmetry("Sym1", ["NAME:Sym1After", "IsPerfectE:=", True])</pre>

EditTangentialHField

Use: Edits a tangential H-Field boundary.

Command: Double-click the boundary in the project tree to edit it.

Syntax: EditTangentialHField <BoundName> <TangentialHFieldArray>

Return Value: None

EditThinLayer

Use: For Maxwell 3D AC Conduction, 3D DC Conduction, 3D Magnetostatic, 3D AC Magnetic, and 3D Transient designs, edit a thin layer boundary.

UI Access	Double-click the boundary in the project tree to edit it.		
Parameters	Name	Type	Description
	<BoundName>	String	The name of the boundary.
	<ThinLayerArray>	Array	- For 3D AC Conduction designs: Array containing the boundary name, boundary face ID array, and permittivity, conductivity, thickness value with unit. - For 3D DC Conduction designs: Array containing the boundary name, boundary face ID array, and conductivity and thickness value with unit. - For 3D Magnetostatic: Create an air gap boundary between two non-conductors or designate a thin ferromagnet. Array containing the boundary name, relative permeability (no air gap) and thickness value with unit. - For 3D Transient (T-Omega only): Create an air gap boundary between two non-conductors. Array containing the boundary name, and thickness value with unit. - For 3D AC Magnetic: Array containing the boundary name, AirGap (bool); If AirGap=False, include conductivity, relative permeability, and thickness value with unit. If AirGap=true, include only thickness value with unit.
Return Value	None		

For 3D AC Conduction and 3D DC Conduction designs

Python Syntax	EditThinLayer (<i>BoundName</i> [<NAME>, <Faces>, <Permittivity(AC Conduction)>, <Conductivity>, <Thickness>])
Python Example	<pre>oModule.EditThinLayer("ThinLayer1" ["NAME:ThinLayer1", "Faces:=", [40], "Permittivity:=", "1" "Conductivity:=", "1S_per_m" "Thickness:=", "1mm"])</pre>

For 3D Magnetostatic designs

Python Syntax	EditThinLayer (<i>BoundName</i> [<NAME>, <AirGap>, <RelativePermeability>, <Thickness>])
Python Example	<pre>oModule.Edit("ThinLayer1", ["NAME:ThinLayer1", "AirGap:=", True, "Thickness:=", "2mm"])</pre>

For 3D Transient (T-Omega) designs

Python Syntax	EditThinLayer (<i>BoundName</i> [<NAME>, <Thickness>])
Python Example	<pre>oModule.EditThinLayer("ThinLayer1" ["NAME:ThinLayer1", "Thickness:=", "1mm"])</pre>

For 3D AC Magnetic designs

Python Syntax	EditThinLayer (<i>BoundName</i> [<NAME>, <AirGap>, <Conductivity>, <RelativePermeability>, <Thickness>])
Python Example	<pre>oModule.EditThinLayer("ThinLayer1" ["NAME:ThinLayer1", "AirGap:=", False, "Conductivity:=", "500000000000S_per_m" "RelativePermeability:=", "1" "Thickness:=", "0.02mm"])</pre>

EditTouching

Use: For Maxwell 3D Transient designs with translational motion with a moving part touching stationary part, edits a touching boundary condition.

UI Access	Double-click a touching boundary condition in the Project Manager to open the Touching Boundary dialog box.		
Parameters	Name	Type	Description
	<NAME>	string	The name of the boundary
Return Value	None		

Python Syntax	EditTouching (<NAME>)
Python Example	<pre>oModule.EditTouching(["NAME:Touching3"])</pre>

EditVectorPotential

Use: For Maxwell 2D AC Magnetic, Magnetostatic, and Transient designs, edits a vector potential boundary condition.

UI Access	Double-click a vector potential boundary condition in the Project Manager to open the Vector Potential Boundary dialog box.		
Parameters	Name	Type	Description
	<NAME>	string	The name of the boundary
	<Edges>	list	Array of edge IDs
	<Phase>	string	Phase angle with unit (for 2D AC Magnetic designs only)
	<Value>	string	Vector potential value in weber/m
	<CoordinateSystem>	string	Name of the reference coordinate system if the vector potential value is a spatial-dependent function such as z+2
Return Value	None		

Python Syntax	EditVectorPotential (<NAME>, <Edges>, <Phase>, <Value>, <CoordinateSystem>)
Python Example	<pre>oModule.EditVectorPotential(["NAME:VectorPotential2", "Edges:=", [21], "Phase:=", "60deg", "Value:=", "z+2", "CoordinateSystem:=", "Global"])</pre>

EditZeroTangentialHField

Use: Edits a zero tangential H-Field boundary.

Command: Double-click the boundary in the project tree to edit it.

Syntax: EditZeroTangentialHField <BoundName>, <PerfectEArray>

Return Value: None

Script Commands for Creating and Modifying Excitations in Maxwell

Following are script commands for creating and modifying excitations that are recognized by the "BoundarySetup" module. In the following commands, all named data can be included/excluded as desired and may appear in any order.

- [AddTerminalsToWinding](#)
- [AssignCharge](#)
- [AssignCoilTerminal](#)
- [AssignCoilTerminalGroup](#)
- [AssignCurrent](#)
- [AssignCurrentDensity](#)
- [AssignCurrentDensityGroup](#)
- [AssignCurrentDensityTerminal](#)
- [AssignCurrentDensityTerminalGroup](#)
- [AssignCurrentGroup](#)
- [AssignFloating](#)
- [AssignSink](#)
- [AssignVoltage](#)
- [AssignVoltageAPhi](#)
- [AssignVolumeChargeDensity](#)
- [AssignVoltageDrop](#)

- [AssignVoltageDropGroup](#)
- [AssignVoltageGroup](#)
- [Assign Winding Group](#)
- [EditCharge](#)
- [EditCoilTerminal](#)
- [EditCurrent](#)
- [EditCurrentDensity](#)
- [EditCurrentDensityTerminal](#)
- [EditExternalCircuit](#)
- [EditFloating](#)
- [EditSink](#)
- [EditVoltage](#)
- [EditVoltageAPhi](#)
- [EditVolumeChargeDensity](#)
- [EditVoltageDrop](#)
- [EditWindingGroup](#)
- [SetCoreLoss](#)
- [SetEddyEffect](#)
- [SetMagnetizationCompute](#)
- [SetMinimumTimeStep](#)
- [SetupPWMExcitation](#)
- [SetupYConnection](#)
- [SetPermeabilityComputation](#)

AddTerminalsToWinding

Use: Adds existing terminal(s) to the selected winding.

Command: Right-click a winding item in the project tree, and then select **Add Terminals**.

Syntax: AddTerminalsToWinding (<AddTerminalsToWindingArray>)

Return Value: None

Parameters: < AddTerminalsToWindingArray >

```
Array(" NAME:AddTerminalsToWinding", <BoundListArray>)  
  
<BoundListArray> Array("NAME:BoundaryList",<CoilTerminalArray>)
```

AssignCharge

Creates a charge excitation.

UI Access	Maxwell > Excitations > Assign > Charge		
Parameters	Name	Type	Description
	<BoundName>	String	Name of charge excitation to be created
	<Value>	Real	Value of charge excitation
	<Objects> or <Faces>	Array	Array of objects or faces to which the charge will be applied
Return Value	None		

Python Syntax	AssignCharge (<BoundName>, <Value>, <Objects> or <Faces>)
Python Example	<pre>oModule.AssignCharge (["NAME:Charge1", "Value:=", "1",</pre>

	<code>"Faces:=", [11]])</code>

AssignCoilGroup

Use: For Maxwell 2D designs, creates a group of coil excitations. The coil excitations can be assigned on 2D objects.

UI Access	Select the desired objects (coils) on the model or in the History tree. Then, select Maxwell 2D> Excitations>Assign>Coil to open the Coil Excitation dialog where you can enter the base Name for the group of excitations being created, the number of conductors, and the polarity to be assigned to each of the new excitations.		
Parameters	Name	Type	Description
	<NAME>	string	The name of the coil group
	<Objects>	list	List of objects to which the excitations are applied
	<Conductor number>	int	The number of conductors
	<Polarity type>	string	Positive or Negative
Return Value	None		

Python Syntax	<code>AssignCoilGroup (CoilGroupArray, <NAME>, <Objects>, <Conductor number>, <Polarity type>)</code>
Python Example	<pre>oModule.AssignCoilGroup(["terminal_A_plus", "terminal_A_plus_1"], ["NAME:terminal_A_plus", "Objects:=", ["terminal_A_plus","terminal_A_plus_1"], "Conductor number:=", "10", "PolarityType:=", Positive])</pre>

AssignCoilTerminal

Use: Assigns 2D terminals

Command: Maxwell>Excitations>Assign>Coil Terminal

Syntax: AssignCoilTerminal <TerminalArray>

Parameters: <TerminalArray>

```
Array("NAME:AssignTerminals",  
      Array("Name:SourceList", Array("Name:<SourceName>",  
    "Excitation:=", <NetObject>, "Objects:=", <Assignment 2D>)),  
      Array("Name:SinkList", Array("Name:<SinkName>",  
    "Excitation:=", <NetObject>, "Objects:=", <Assignment 2D>)),  
      "Name>DeleteList", <Name Array>)
```

Return Value: None

AssignCoilTerminalGroup

Use: For Maxwell 3D designs, creates a group of coil terminal excitations. The coil terminals can be assigned on the faces of 3D objects or directly on sheet objects.

UI Access	Select the desired objects (coils) on the model or in the History tree. Then, select Maxwell 3D> Excitations>Assign>Coil Terminal to open the Coil Terminal Excitation dialog where you can enter the base Name for the group of excitations being created, the number of conductors, and use the Swap Direction button to set the current direction to be assigned to each of the new excitations.		
Parameters	Name	Type	Description
	<NAME>	string	The name of the coil terminal group
	<Objects>	list	List of faces or sheet objects to which the excitations are applied
	<Conductor number>	int	The number of conductors

	<table><tr><td><Point out of terminal></td><td>bool</td><td>True sets the current direction out of the terminal</td></tr></table>	<Point out of terminal>	bool	True sets the current direction out of the terminal
<Point out of terminal>	bool	True sets the current direction out of the terminal		
Return Value	None			

Python Syntax	<code>AssignCoilTerminalGroup (TerminalGroupArray, <NAME>, <Objects>, <Conductor number>, <Point out of terminal>)</code>
Python Example	<pre>oModule.AssignCoilTerminalGroup(["CoilTerminal_1", "CoilTerminal_2"], ["NAME:CoilTerminal_1", "Objects:=" , ["terminal_A_plus","terminal_A_plus_1"], "Conductor number:=" , "10", "Point out of terminal:=", False]) </pre>

AssignCurrent

Creates a current excitation. Current excitations can be applied in [Magnetostatic](#), [AC Conduction](#), [DC Conduction](#), [AC Magnetic T-Omega](#), [AC Magnetic A-Phi](#), [2D Transient](#), [3D Transient A-Phi](#), and [Electric Transient](#) solutions. The current parameters for each type of solution will vary; see the individual help topics for more details on the parameters for current excitation.

UI Access	Maxwell > Excitations > Assign > Current		
Parameters	Name	Type	Description
	<CurrentArray>	Array	Structured array, parameters are determined by the solution type.
Return Value	None.		

Python Syntax	AssignCurrent_1 (<CurrentArray>)
Python Example	AssignCurrent for a 3D Magnetostatic solution; <pre>oModule.AssignCurrent (["NAME:Current6", "Faces:=", "Current:=", "IsSolid:=", "Point out of terminal:=", False %Swap Direction in interface])</pre> AssignCurrent with a AC Magnetic\Include DC Current solution: <pre>oModule.AssignCurrent (["NAME:Current3", "Faces:=", "Phase:=", "Current:=", "IsSolid:=", "Point out of terminal:=", False, %Swap Direction in interface "DC Current:=",])</pre> AssignCurrent with AC Magnetic A-Phi solver: <pre>oModule.AssignCurrent (["NAME:Current2", "Faces:=", "Phase:=", "Current:=", "CurrentExcitationModel:=", "Double Potentials",</pre>

	<pre>"Point out of terminal:=", False %Swap Direction in interface])</pre>

AssignCurrentDensity

Use: Creates a current density excitation.

Command: Maxwell > Excitations > Assign > Current Density

Syntax: AssignCurrentDensity <CurrentDensityArray>

Return Value: None

Parameters: <CurrentDensityArray>

```
Array("NAME:<BoundName>",  
      "CurrentDensityX:=", <value>,  
      "Phase:=", <value>,//for AC Magnetic solution type  
      "CurrentDensityY:=", <value>,  
      "CurrentDensityZ:=", <bool>,  
      "CoordinateSystem Name:=", <string>,  
      "CoordinateSystem Type:=", <string>,  
      "Objects:=", <AssignmentObjects>)
```

AssignCurrentDensityGroup

Use: Creates a group of current density excitations.

Command: Maxwell>Excitations>Assign>Current Density

Syntax: AssignCurrentDensityGroup <BoundNameArray> <CurrentDensityArray>

Return Value: None

AssignCurrentDensityTerminal

Use: Creates a current density terminal excitation.

Command: Maxwell>Excitations>Assign>Current Density Terminal

Syntax: AssignCurrentDensityTerminal <CurrentDensityTerminalArray>

Return Value: None

Parameters: <CurrentDensityTerminalArray>

```
Array("Name:<BoundName>",  
      "Objects:=", <AssignmentObjects>,  
      "Faces:=", <AssignmentFaces>)
```

AssignCurrentDensityTerminalGroup

Use: Creates a group of current density terminal excitations.

Command: Maxwell>Excitations>Assign>Current Density Terminal

Syntax: AssignCurrentDensityTerminalGroup <BoundNameArray> <CurrentDensityTerminalArray>

Return Value: None

AssignCurrentGroup

Use: Creates a group of current excitations.

Command: Maxwell>Excitations>Assign>Current

Syntax: AssignCurrentGroup <BoundNameArray> <CurrentArray>

Return Value: None

AssignFloating

Use: Creates a floating excitation.

Command: Maxwell>Boundaries>Assign>Floating

Syntax: AssignFloating <FloatingArray>

Return Value: None

Parameters: <FloatingArray>

```
Array("NAME:<BoundName>",  
      "Value:=", <value>,  
      "Objects:=", <AssignmentObjects>,  
      "Faces:=", <AssignmentFaces>)
```

Example: oModule.AssignFloating Array("NAME:Floating1", _ "Value:=", "1", "Faces:=", Array(11))

AssignSink (Maxwell)

Use: Creates a sink.

Command: Maxwell>Excitations>Assign>Sink

Syntax: AssignSink <SinkArray>

Return Value: None

Parameters: <SinkArray>

```
Array("NAME:<SinkName>",  
      "Faces:=", <AssignmentFaces>)
```

Example: oModule.AssignSink Array("NAME:Sink1",_
"Faces:=", Array(12))

AssignVoltage (Maxwell)

Creates a voltage excitation. Voltage excitations can be applied in [Electrostatic](#), [Magnetostatic](#), [AC Conduction](#), [DC Conduction](#), [AC Magnetic A-Phi](#), and [Electric Transient](#) solutions. The voltage parameters for each type of solution will vary; see the individual help topics for more details on the parameters for voltage excitation.

Note: The Transient A-Phi solver has its own voltage excitation command; see [AssignVoltageAPhi](#) for more details.

UI Access	Maxwell> Excitations > Assign > Voltage...		
Parameters	Name	Type	Description
	<VoltageArray>	Array	Structured array, parameters are determined by the solution type.
Return Value	None.		

Python Syntax	AssignVoltage (<VoltageArray>)
Python Example	Magnetostatic example: <pre>oModule.AssignVoltage (["NAME:Voltage1", "Faces:=" , [239], "Voltage:=" , "5mV"])</pre> AC Magnetic A-Phi: <pre>oModule.AssignVoltage ([</pre>

	<pre>"NAME:Voltage4", "Faces:=" , [122], "Voltage:=" , "0mV", "Phase:=" , "0deg", "VoltageAphiExcitationModel:=", "Double Potentials", "VoltageAphi_Point_out_of_terminal:=", False])</pre>

AssignVoltageAPhi

Assigns a voltage excitation for 3D transient A-Phi solutions.

UI Access	Excitations > Assign > Voltage		
Parameters	Name	Type	Description
	<NAME>	string	The name of the voltage excitation
	<Objects>	list	List of objects to which the excitation is applied
	<VoltageAPhi>	string	Voltage value with unit
	<VoltageAPhiInitialCurrent>	string	Initial current value with unit
	<VoltageAPhiHasInitialCurrent>	bool	True if Initial Current is used.
	<VoltageAPhiExcitationModel>	string	"Single Potential" "Double Potentials" "Double Potentials with Ground"
	<VoltageAPhi_Point_out_of_terminal>	bool	True sets the voltage direction out of the terminal
Return Value	None		

Python Syntax	AssignVoltageAPhi (<NAME>, <Objects>, <VoltageAPhi>, <VoltageAPhiInitialCurrent>, <VoltageAPhiHasInitialCurrent>, <VoltageAPhiExcitationModel>, <VoltageAPhi_Point_out_of_terminal>)
Python Example	<pre>oModule.AssignVoltageAPhi (["NAME:Voltage4", "Objects:=", ["Cylinder1"], "VoltageAPhi:=", "0mV", "VoltageAPhiInitialCurrent:=", "0mA", "VoltageAPhiHasInitialCurrent:=", False, "VoltageAPhiExcitationModel:=", 0, "VoltageAPhi_Point_out_of_terminal:=", False]) </pre>

AssignVoltageDrop

Use: Creates a voltage drop excitation.

Command: Maxwell > Excitations > Assign > Voltage Drop

Syntax: AssignVoltageDrop <VoltageDropArray>

Return Value: None

Parameters: <VoltageDropArray>

```
Array("NAME:<BoundName>",
    "Voltage Drop:=", <value>,
    "Point out of terminal:=", <bool>,
    "Objects:=", <AssignmentObjects>,
    "Faces:=", <AssignmentFaces>)
```

AssignVoltageDropGroup

Use: Creates a group of voltage drop excitations.

Command: Maxwell > Excitations > Assign > Voltage Drop

Syntax: AssignVoltageDropGroup <BoundNameArray> <VoltageDropArray>

Return Value: None

AssignVoltageGroup

Use: Creates a group of voltage excitations. The size of the bound name array must be identical to the size of assignment. This command is not supported for the **Electric** solution types.

Command: Maxwell > Excitations > Assign > Voltage

Syntax: AssignVoltageGroup <BoundNameArray> <VoltageArray>

Return Value: None

AssignVolumeChargeDensity

Use: Creates a volume charge density excitation.

Command: Maxwell>Excitations>Assign>Volume Charge Density

Syntax: AssignVolumeChargeDensity <VolumeChargeDensityArray>

Return Value: None

Parameters: <VolumeChargeDensityArray>

```
Array("NAME:<BoundName>",  
      "Value:=", <value>,  
      "CoordinateSystem :=", <string>,
```

```
"Objects:=", <AssignmentObjects>)
```

AssignWindingGroup

Use: Creates a winding group.

Command: Maxwell > Excitations > Add Winding

Syntax: AssignWindingGroup <WindingGroupArray >

Return Value: None

Parameters:

<WindingGroupArray>

```
[ "NAME:<BoundName>",
  "Type:=", <WindingType>, // if the solution is "AC Magnetic with DC", the value is either "Current" or "Voltage"
  "IsSolid:=", <bool>, //true for solid, false for stranded
  "Current:=", <value>,
  "Resistance:=", <value>,
  "Inductance:=", <value>,
  "Voltage:=", <value>,
  "DC Current:=", <value> //Available if the solution is "AC Magnetic with DC" and winding Type is set to "Current"
  "DC Voltage:=", <value> //Available if the solution is "AC Magnetic with DC" and winding Type is set to "Voltage"
]
```

Example: AC Magnetic with Include DC Fields, Current Winding

```
oModule.AssignWindingGroup(
```

```
[
  "NAME:Winding1",
  "Type:=" , "Current",
  "IsSolid:=" , True,
  "Current:=" , "0mA",
  "Resistance:=" , "0ohm",
  "Inductance:=" , "0nH",
  "Voltage:=" , "0mV",
  "ParallelBranchesNum:=" , "1",
  "Phase:=" , "0deg",
  "DC Current:=" , "11mA"
])
```

Example: AC Magnetic with Include DC Fields, Voltage Winding

```
oModule.AssignWindingGroup(
[
  "NAME:Winding2",
  "Type:=" , "Voltage",
  "IsSolid:=" , True,
  "Current:=" , "0mA",
  "Resistance:=" , "0ohm",
  "Inductance:=" , "0nH",
  "Voltage:=" , "0mV",
  "ParallelBranchesNum:=" , "1",
  "Phase:=" , "0deg",
  "DC Voltage:=" , "22mV"
])
```

EditCharge

Use: Edits a charge excitation.

Command: Double-click the excitation in the project tree to edit it.

Syntax: EditCharge <BoundName> <ChargeArray>

Return Value: None

EditCoilTerminal

Use: Edits a coil terminal excitation.

Command: Double-click the excitation in the project tree to edit it.

Syntax: EditCoilTerminal <BoundName> <CoilTerminalArray>

Return Value: None

EditCurrent

Modifies a current source.

UI Access	In the Project Manager tree, right click on Excitations > Current or in the Modeler window, right click on a planar entity and select Assign Excitation > Current		
Parameters	Name	Type	Description
	<NAME>	string	The name of the excitation
	<Phase>		
	<Current>	string	Current value with unit
	<IsSolid>	bool	True if type is Solid; False if Stranded. Note that this parameter is not available for a 3D AC Magnetic A-Phi design because only Solid is available.
	<CurrentExcitationModel>	int	0 for Single Potential 1 for Double Potentials 2 for Double Potentials with Ground
	<Point out of terminal>	bool	True sets the current direction out of the terminal
	<DC Current>	string	Setting is available when Include DC Fields is selected for an AC Magnetic simulation. It specifies the DC input as a current.
Return Value	None		

Python Syntax	EditCurrent_1 (<NAME>, <Current>, <IsSolid>, <CurrentExcitationModel>, <Point out of terminal>)
Python Example	<pre>oModule.EditCurrent("Current7", ["NAME:Current7", "Current:=" , "0mA", "IsSolid:=" , True, "CurrentExcitationModel:=", 1, "Point out of terminal:=", False])</pre> Specify DC current input for an AC Magnetic\Include DC Fields simulation: <pre>oModule.EditCurrent("Current1", ["NAME:Current1", "Phase:=" , "0deg", "Current:=" , "a", "IsSolid:=" , True, "Point out of terminal:=", False, "DC Current:=" , "12A"])</pre>

EditCurrentDensity

Use: Edits a current density excitation.

Command: Double-click the excitation in the project tree to edit it.

Syntax:EditCurrentDensity <BoundName> <CurrentDensityArray>

Return Value: None

EditCurrentDensityTerminal

Use: Edits a current density terminal excitation.

Note: Double-click the excitation in the project tree to edit it.

Syntax: EditCurrentDensityTerminal <BoundName> <CurrentDensityTerminalArray>

Return Value: None

EditExternalCircuit

Edits the external circuit for the winding.

UI Access	Maxwell > Excitations > External Circuit > Edit External Circuit		
Parameters	Name	Type	Description
	<fileName>	String	Filepath to external circuit
	<SourceNameArray>	Array	The SourceNameArray and SourceTypeArray parameters must be of the same size and be a one-to-one match.
	<SourceTypeArray>	Array	Array(<srcTypeEnum>) 1- Time dependent. 2-Position dependent, 3- Speed dependent
Return Value	None.		

Python Syntax	EditExternalCircuit (<fileName>, <SourceNameArray>, <SourceTypeArray>)
Python Example	

EditExternalSimulinkCircuit

Edits the external Simulink circuit.

UI Access	Edit Circuit button on the Maxwell-Simulink Link tab of the Setup Co-Simulation with Simulink dialog box.		
Parameters	Name	Type	Description
	<fileName>	String	Simulink model full file name and path
Return Value	None.		

Python Syntax	EditExternalCircuit (<Simulink model full file name and path>)
Python Example	<code>oModule.EditExternalSimulinkCircuit("D:\Work\Projects\test_ckt.mdl")</code>

EditFloating

Use: Edits a floating excitation.

Command: Double-click the excitation in the project tree to edit it.

Syntax: EditFloating <BoundName> <FloatingArray>

Return Value: None

EditSink

Use: Edits a sink excitation.

Command: Double-click the excitation in the project tree to edit it.

Syntax: EditSink <BoundName> <SinkArray>

Return Value: None

EditVoltage

Modifies a voltage source.

UI Access	Double-click the excitation in the project tree to modify its settings.		
Parameters	Name	Type	Description
	<BoundaryName>	String	Name of the voltage source to be edited.
	<VoltageArray>	Array	Structured array. <pre>Array("NAME:<BoundName>", "Voltage:=", <value>, <DirectionArray>)</pre>
	<DirectionArray>	Array	Structured array. <pre>Array("NAME:Direction", "Start:=", <LineEndPoint>, "End:=", <LineEndPoint>)</pre>
Return Value	None.		

Python Syntax	EditVoltage(<BoundaryName>, <VoltageArray>)
Python Example	<pre>oModule.EditVoltage("Voltage1", ["NAME:Voltage1After", "Voltage:=", "1000mV", ["NAME:Direction", "Start:=", [-0.4, -1.2, 0], "End:=", [-1.4, -1.2, 0]]])</pre>

	)

EditVoltageAPhi

Edits a voltage excitation for 3D transient A-Phi solutions.

UI Access	Double-click the Voltage Excitations item in the Project tree to edit.		
Parameters	Name	Type	Description
	<NAME>	string	The name of the voltage excitation
	<VoltageAPhi>	string	Voltage value with unit
	<VoltageAPhiInitialCurrent>	string	Initial current value with unit
	<VoltageAPhiHasInitialCurrent>	bool	True if Initial Current is used.
	<VoltageAPhiExcitationModel>	int	0 for Single Potential 1 for Double Potentials 2 for Double Potentials with Ground
	<VoltageAPhi_Point_out_of_terminal>	bool	True sets the voltage direction out of the terminal
Return Value	None		

Python Syntax	EditVoltageAPhi (<NAME>, <VoltageAPhi>, <VoltageAPhiInitialCurrent>, <VoltageAPhiHasInitialCurrent>, <VoltageAPhiExcitationModel>, <VoltageAPhi_Point_out_of_terminal>)
Python Example	<pre>oModule.EditVoltageAPhi ("Voltage1", [</pre>

	<pre>"NAME:Voltage1", "VoltageAPhi:=" , "0mV", "VoltageAPhiInitialCurrent:=", "0mA", "VoltageAPhiHasInitialCurrent:=", True, "VoltageAPhiExcitationModel:=", 1, "VoltageAPhi_Point_out_of_terminal:=", True])</pre>

EditVoltageDrop

Use: Edits a voltage drop excitation.

Command: Double-click the excitation in the project tree to edit it.

Syntax: EditVoltageDrop <BoundName> <VoltageDropArray>

Return Value: None

EditVolumeChargeDensity

Use: Edits a volume charge density excitation.

Command: Double-click the excitation in the project tree to edit it.

Syntax: EditVolumeChargeDensity <BoundName> <VolumeChargeDensityArray>

Return Value: None

EditWindingGroup

Use: Edits a winding group.

Command: Double-click the winding group item in the project tree to edit it.

Syntax: EditWindingGroup <BoundName> <WindingGroupArray>

Return Value: None

Python Example: AC Magnetic with Include DC Fields, Current Winding

```
oModule.EditWindingGroup("Winding1",
[
    "NAME:Winding1",
    "Type:=" , "Current",
    "IsSolid:=" , True,
    "Current:=" , "0mA",
    "Resistance:=" , "0ohm",
    "Inductance:=" , "0nH",
    "Voltage:=" , "0mV",
    "ParallelBranchesNum:=" , "1",
    "Phase:=" , "0deg",
    "DC Current:=" , "13mA" //Available if the solution is "AC Magnetic with DC" and winding Type
    is set to "Current"
])
```

Example: AC Magnetic with Include DC Fields, Voltage Winding

```
oModule.EditWindingGroup("Winding2",
[
    "NAME:Winding2",
    "Type:=" , "Voltage",
    "IsSolid:=" , True,
    "Current:=" , "0mA",
    "Resistance:=" , "0ohm",
    "Inductance:=" , "0nH",
    "Voltage:=" , "0mV",
    "ParallelBranchesNum:=" , "1",
    "Phase:=" , "0deg",
    "DC Voltage:=" , "23mV" //Available if the solution is "AC Magnetic with DC" and winding
    Type is set to "Voltage"
])
```

FFTONMultiExcitationsDataset

For 2D and 3D T-Omega AC Magnetic simulations, this command creates a transient excitation as a design dataset so Maxwell can use it to reconstruct the signal vs time using a certain amount of harmonics from an FFT plot and applies it to selected [Winding](#) or [Current](#) excitations.

Note: For information on how the FFT can be used in place of a transient simulation, see [Using AC Magnetic to Generate Harmonic Losses](#).

UI Access	Maxwell > Excitations > Set up Datasets from FFT		
Parameters	Name	Type	Description
	<Name>	string	Name of dataset to use for the FFT calculation
	<ExcitationToDataset>	Array	Maps each current or winding excitation to a transient dataset.
	<WindowType>	string	FFT window type; possible values are "Rectangular", "Tri", "Van Hann", "Hamming", "Blackman", "Lanczos", "Weber", "Welch".
	<UseFreqRange>	bool	If set to <code>True</code> , the user can specify a frequency range.
	<UseNoiseThreshold>	bool	If set to <code>True</code> , specify the noiseThreshold; default is 1.
	<StartFreq>	string	Specify start frequency with frequency unit
	<EndFreq>	string	Specify end frequency with frequency unit; use if the frequencyRangeOption is set to 0.
	<FreqRangeOption>	int	0 uses startFreq and endFreq to create a range; 1 uses startFreq and FreqNum to create a range.
	<FreqNum>	int	Specify the number of frequencies to use in the calculation; use if FreqRangeOption is set to 1.
	<NoiseThreshold>	double	Filters out harmonics with small magnitudes; input is a percentage of maximum magnitude value. Data of (A, f) whose magnitudes are smaller than the value will not be added to frequency-domain datasets. The data of (ϕ , f) at that frequency will also be discarded.
	<OverwritePreviousDatasets>	bool	If set to <code>True</code> , function will overwrite any existing datasets with the same name. If set to <code>False</code> , the function will append the new datasets to the existing list of datasets.

Return Value	Array of dataset names: The first name is the dataset magnitude vs. Frequency, and the second name is the dataset phase vs. frequency.

Python Syntax	<code>FFTOnMultiExcitationsDataset (<Name>, <ExcitationToDataset>, <WindowType>, <UseFreqRange>, <UseNoiseThreshold>, <StartFreq>,<EndFreq>, <FreqRangeOption>, <FreqNum>, <NoiseThreshold>, <OverwritePreviousDatasets>)</code>
Python Example	<pre>oModule = oDesign.GetModule("BoundarySetup") oModule.FFTOnMultiExcitationsDataset(["NAME:FFTOnMultiExcitations", "ExcitationToDataset:=" , ["Wind- ing1","ds1","Winding2","ds2","Winding3","ds3"], "WindowType:=" , 0, "UseFreqRange:=" , False, "UseNoiseThreshold:=" , True, "StartFreq:=" , "0", "EndFreq:=" , "0", "FreqRangeOption:=" , 0, "FreqNum:=" , 5, "NoiseThreshold:=" , 50, "OverwritePreviousDatasets:=", True])</pre>

SetCoreLoss

Turns on the core loss effect for AC Magnetic and Transient solutions. You must list all the objects that need to turn on the core loss effect. If an object is not in the list, then it is treated as core loss effect "off". This setting only takes effect if the object has the

corresponding core loss definition in its material library.

UI Access	Maxwell > Excitation > Set Core Loss		
Parameters	Name	Type	Description
	<objectName>	Array	Name or names of objects to apply core loss to
	<ApplyCoreloss>	Boolean	True to apply core loss effect on field; False means do not apply core loss effect. Default is False.
Return Value	None		

Python Syntax	SetCoreLoss (<objectName>, <ApplyCoreloss>)
Python Example	<pre>oModule = oDesign.GetModule("BoundarySetup") oModule.SetCoreLoss(["PhB0", "PhA1", "PhA0", "Stator"], False)</pre>

SetEddyEffect

Sets the eddy effect for an excitation.

UI Access	Maxwell > Excitations > Set Eddy Effects		
Parameters	Name	Type	Description
	<EddyEffectSettingArray>	Array	Array with <EddyEffectVectorArray> and <EddyEffectDataArray>
	<EddyEffectDataArray>		
	<ObjectName>	String	Name of object
	<Eddy Effect>	Boolean	true to compute the induced eddy currents on the selected object, false to leave the object out of the eddy current calculation.
	<Displacement Current>	Boolean	Available for 3D AC Magnetic only. Set to true to calculate the displacement current on the selected object; set to false to leave the object

	<table><tr><td></td><td></td><td>out the displacement current calculation. Note that for 3D Magnetic A-Phi, the displacement current is set for all objects, not individual objects, and the induced eddy current is automatically calculated for all objects.</td></tr></table>			out the displacement current calculation. Note that for 3D Magnetic A-Phi, the displacement current is set for all objects, not individual objects, and the induced eddy current is automatically calculated for all objects.
		out the displacement current calculation. Note that for 3D Magnetic A-Phi, the displacement current is set for all objects, not individual objects, and the induced eddy current is automatically calculated for all objects.		
Return Value	None			

Python Syntax	SetEddyEffect (<EddyEffectSettingArray>)
Python Example	Eddy effect for transient: <pre>oModule.SetEddyEffect(["NAME:Eddy Effect Setting", ["NAME:EddyEffectVector", ["NAME:Data", "Object Name:=", "Box1", "Eddy Effect:=", true,], ["NAME:Data", "Object Name:=", "Box1_1", "Eddy Effect:=", false],],]])</pre> Eddy effect for AC Magnetic T-Omega:


```
oModule.SetEddyEffect(
[
  "NAME:Eddy Effect Setting",
  [
    "NAME:EddyEffectVector",
    [
      "NAME:Data",
      "Object Name:=", "Box1", "Eddy Effect:=", true,
      "Displacement Current:=", true
    ],
  ],
  [
    "NAME:Data",
    "Object Name:=", "Box1_1", "Eddy Effect:=", true,
    "Displacement Current:=", false
  ]
])
```

Displacement Current for AC Magnetic A-Phi (Eddy Current is automatically calculated for all conductors):

```
oModule.SetEddyEffect(
[
  "NAME:Eddy Effect Setting",
  [
    "NAME:EddyEffectVector",
    "Displacement Current:=", True
  ]
])
```

SetMinimumTimeStep

Use: Sets the minimum time step for an external circuit excitation.

Command: Maxwell > Excitations > External Circuit > Set Minimum Time Step**Syntax:** SetMinimumTimeStep <value>**Return Value:** None**Parameters:** None**Example:** oModule.SetMinimumTimeStep "1ps"**SetupPWMExcitation**

Assigns a pulse-width modulation (PWM) voltage excitation on windings for 3D or 2D transient T-Omega solutions. It supports three-phase Space Vector Pulse Width Modulation (SVPWM). The generated waveform represents the Phase-Neutral voltage, and it is only applicable to Y(star)-connected and delta-connected windings.

For more details, see [Setting Up an SVPWM Excitation](#).

UI Access	Excitations > Setup PWM Voltage Excitations		
Parameters	Name	Type	Description
	<NAME>	string	Name of the SVPWM excitation
	<Windings>	list	List of windings to which the excitation is applied; at least three windings have to be specified.
	<DCBusVoltage>	string	The voltage available at the main DC bus as input to the inverter. This value can be entered directly or defined using a design variable.
	<ModulationIndex>	float	Specifies the ratio between the reference signal to the carrier signal; it determines the amplitude of the fundamental output voltage with respect to the available DC bus voltage. The range of modulation index is $0 < \text{ModulationIndex} \leq 1.1547$. This parameter cannot be defined with a variable.
	<OutputVoltageFrequency>	string	Fundamental frequency of the reference signal, and therefore the PWM output voltage. This value should be consistent with the speed of the motor. Specify value with a unit.

	<SwitchingFrequency>	string	Carrier signal frequency that is used to generate the PWM switching signal. A typical value for traction inverters is between 10 – 20 kHz. Specify value with a unit.
	<Phase>	string	The phase delay angle of the reference signal (and hence the output voltage). The phase delay depends on the operating point, and it can be provided from system-level simulation or by running a transient simulation with an ideal sinusoidal current and check the phase delay on the output of the phase voltage. Specify value with a unit.
	<ModulationIndexType>	integer	Values are 0 (activates the ReferencePhaseVoltage) and 1 (activates the ModulationIndex setting).
	<ReferencePhaseVoltage>	string	Defines the reference voltage signal. If this option is selected, the ModulationIndex setting is not accessible and will be automatically calculated according to $\text{ModulationIndex} = \text{ReferencePhaseVoltage} / (\text{DCBusVoltage}/2)$.
	<MinTimeStep>	string	Defines the rate at which the carrier signal and the reference signal are compared to generate the switching signals. This value is important to correctly capture the switching events. The recommended value is $1 / (\text{Fsw} * 100)$ or smaller.
	<MaxTimeStep>	string	Largest time step that can be used to generate the switching signal; by default, it is 10*Minimum Time Step.
	<WindingConnectionType>	string	Defines the winding connection type as "YConnection" or "DeltaConnection"
	<UseMaxTimeStep>	Boolean	If <code>True</code> , The MaxTimeStep parameter will be used to modify the time step data (see above). If <code>False</code> , no MaxTimeStep will be used.
Return Value	None		

Python Syntax	SetupPWMExcitation ([<NAME>, <Windings>, <DCBusVoltage>, <ModulationIndex>, <OutputVoltageFrequency>, <SwitchingFrequency>, <Phase>, <ModulationIndexType>, <ReferencePhaseVoltage>, <MinTimeStep>, <MaxTimeStep>, <WindingConnectionType>, <UseMaxTimeStep>])

Python Example	<pre>oModule.SetupPWMExcitation(["NAME:PWMExcitation", "Windings:=" , "Winding1,Winding2,Winding3", "DCBusVoltage:=" , "Vdc", "ModulationIndex:=" , 1, "OutputVoltageFrequency:=" , "100Hz", "SwitchingFrequency:=" , "10000Hz", "Phase:=" , "0deg", "ModulationIndexType:=" , 1, "ReferencePhaseVoltage:=" , "50V", "MinTimeStep:=" , "1us", "MaxTimeStep:=" , "100us", "WindingConnectionType:=" , "YConnection" "UseMaxTimeStep:=" , True])</pre>

SetupYConnection

Use: Sets up a Y connection among voltage windings. It will connect their negative voltage terminals to a common node of the circuit. Only one Y Connection is allowed per design.

Note:

- A mix of stranded winding and solid winding in one Y-connection group is not allowed.
- A mix of PWM voltage winding and (normal) voltage winding in one Y-connection group is not allowed.

Syntax: SetupYConnection <YConnectionArray>

Return Value: None

Parameters: <YConnectionArray>

Type: Array of strings

When the *YConnectionArray* is empty, it is used to delete the Y Connection.

Array(<*YConnection*>)

<*YConnection*>

Type: Array of strings

Array("Name:YConnection", "Windings:=", <*WindingNames*>)

<*WindingNames*>

Type: String

A list of winding names separated by commas

Python Syntax	SetupYConnection(<YConnectionArray>)
Python Example	Add Y Connection <pre>oModule.SetupYConnection([["NAME:YConnection", "Windings:=" , "Winding1,Winding2,Winding3,Winding4"]])</pre> Delete Y Connection <pre>oModule.SetupYConnection()</pre>

SetPermeabilityComputation

Use: Enables a nonlinear analysis that will calculate permeability data, which will then be available for 3D AC Magnetic T-Omega design simulation.

Command: In the Project Manager tree, right click **Excitations > Set Permeability Computation**

Syntax: SetPermeabilityComputation <ObjArray>

Return Value: None

Parameters: <ObjectArray>

List of objects with nonlinear BH curve material

Example:

```
oModule = oDesign.GetModule("BoundarySetup")
oModule.SetPermeabilityComputation(
[
    "NAME:MuLinkSrcObjs",
    "Objs:=" , ["Core_1st","Core_2nd","Magnet"]
])
```

14 - Mesh Operations Module Script Commands

Commands for mesh setup and operations should be executed by the "MeshSetup" module.

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

```
oModule.CommandName <args>
```

Conventions Used in this Chapter

<OpName>

Type: <string>

Name of a mesh operation.

<AssignmentObjects>

Type: Array of strings

An array of object names.

<AssignmentFaces>

Type: Array of integers.

An array of face IDs. The ID of a face can be determined through the user interface using the **3DModeler > Measure > Area** command. The face ID is given in the **Measure Information** dialog box.

The topics for this section include:

[General Commands Recognized by the Mesh Operations Module](#)

[Script Commands for Creating and Modifying Mesh Operations](#)

General Commands Recognized by the Mesh Operations Module

General commands recognized by the Mesh Operations Module:

[DeleteOp](#)

[GetMeshOpAssignment](#)

[GetOperationName](#)

[RenameOp](#)

DeleteOp

Deletes the specified mesh operations.

UI Access	Delete command in the List dialog box.		
Parameters	Name	Type	Description
	<NameArray>	Array	Array of mesh operation names.
Return Value	None.		

Python Syntax	DeleteOp(<NameArray>)
Python Example	<code>oModule.DeleteOp(["Length1", "SkinDepth1", "Length2"])</code>

GetOperationNames

Gets the names of mesh operations defined in a design.

UI Access	N/A		
Parameters	Name	Type	Description
	<OperationType>	String	Specified operation type.
Return Value	Array of strings containing mesh operation names.		

Python Syntax	GetOperationNames(<OperationType>)		
Python Example	oModule.GetOperationNames("Length Based")		

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	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	oModule.RenameOp("SkinDepth1", "NewName")		

Script Commands for Creating and Modifying Mesh Operations

Script commands for creating and modifying mesh operations are as follows:

[AssignCylindricalGapOp](#)

[AssignDensityControlOp](#)

[AssignLengthOp](#)

[AssignModelResolutionOp](#)

[AssignSkinDepthLayerSetting](#)

[AssignSkinDepthOp](#)

[AssignTrueSurfOp](#)

[EditCylindricalGapOp](#)

[EditDensityControlOp](#)

[EditEdgeCutLayerOp](#)

[EditLengthOp](#)

[EditModelResolutionOp](#)

[EditSkinDepthOp](#)

[EditTrueSurfOp](#)

[InitialMeshSettings](#)

AssignCylindricalGapOp

3D Designs

For 3D designs, assigns a cylindrical gap 3D mesh operation to enable use of Clone Mesh and associated Band Mapping Angle. For Maxwell 2D designs, [see below](#).

=====

Assigns a cylindrical gap 3D mesh operation to enable use of Clone Mesh and associated Band Mapping Angle.

UI Access	Maxwell3D > Mesh > Assign Mesh Operation > Cylindrical Gap Treatment		
Parameters	Name	Type	Description
	<CylindricalGapOpParams>	Array	Structured array. <pre>Array("NAME:<OpName>", "CloneMesh:=", <bool>, "BandMappingAngle:=", <value> <i>[use if CloneMesh is "true"]</i> "MovingSideLayers:=", <integer>, "StaticSideLayers:=", <integer>, "Objects:=", <AssignmentObjects>)</pre>
Return Value	None.		

Python Syntax	AssignCylindricalGapOp(<CylindricalGapOpParams>)
Python Example	<pre>oModule.AssignCylindricalGapOp (["NAME:CylindricalGap1", "CloneMesh:=", True,</pre>

	<pre> "BandMappingAngle:=", 3deg, "MovingSideLayers:=", 1, "StaticSideLayers:=", 3, "Objects:=", ["Box2"]] </pre>

Maxwell 2D Designs

For Maxwell 2D Designs, assigns a cylindrical gap 2D mesh operation.

UI Access	Maxwell 2D > Mesh > Assign Mesh Operation > Cylindrical Gap Treatment		
Parameters	Name	Type	Description
	<CylindricalGapOpParams>	Array	Structured array. <pre> Array("NAME:<OpName>", "UseBandMappingAngle:=" , <bool>, "BandMappingAngle:=", <value> [use if UseBandMappingAngle is "true". Range is 0.0005deg through 3deg inclusive] "Objects:=", <AssignmentObjects>) </pre>
Return Value	None.		

Python Syntax	AssignCylindricalGapOp(<CylindricalGapOpParams>)
Python Example	oModule.AssignCylindricalGapOp

	<pre>(["NAME:CylindricalGap1", "UseBandMappingAngle:=" , True, "BandMappingAngle:=" , "3deg", "Objects:=", ["Circle2"]])</pre>

AssignDensityControlOp

Assigns a density control mesh operation to specify refinement of clone mesh.

UI Access	Maxwell3D > Mesh > Assign Mesh Operation > Inside Selection > Clone Mesh Density		
Parameters	Name	Type	Description
	<DensityControlParams>	Array	Structured array. <pre>Array("NAME:<OpName>", "RefineInside:=", <bool>, "Objects:=", <AssignmentObjects>, "RestrictMaxElemLength:=", <bool>, "MaxElemLength:=", <value>, "RestrictLayersNum:=", <bool>, "LayersNum:=", <integer>)</pre> RefineInside If true, Objects should be specified. Implies

			apply restrictions to tetrahedra inside the object. If false, Faces and/or Objects can be specified. Implies apply restrictions to triangles on the surface of the face or object. RestrictMaxElemLength If true, MaxElemLength should be specified. RestrictLayersNum If true, LayersNum should be specified.
Return Value	None.		

Python Syntax	AssignDensityControlOp(<DensityControlParams>)
Python Example	<pre>oModule.AssignDensityControlOp(["NAME:Clone Mesh Density3", "RefineInside:=", True, "Objects:=", ["Inner_arm"], "RestrictMaxElemLength:=", True, "MaxElemLength:=", "0.015mm", "RestrictLayersNum:=", True,</pre>

	<pre>"LayersNum:=" , "15"])</pre>

AssignEdgeCutLayerOp

Assigns the edge cut mesh operation to specify refinement of layer mesh.

UI Access	Maxwell 3D > Mesh > Assign Mesh Operation > Inside Selection > Edge Cut Based		
Parameters	Name	Type	Description
	<EdgeCutLayerOpParams>	Array	Structured array. <pre>Array("NAME:<OpName>", "Objects:=", <AssignmentObjects>, "Layer Thickness:=", <value>)</pre>
Return Value	None.		

Python Syntax	AssignEdgeCutLayerOp(<EdgeCutLayerOpParams>)
Python Example	<pre>oModule.AssignEdgeCutLayerOp(["NAME:EdgeCut1", "Objects:=" , ["Rotor"], "Layer Thickness:=" , "0.55mm"])</pre>

AssignLengthOp

Assigns length-based operations to the selection.

UI Access	Maxwell 3D > Mesh > Assign Mesh Operation > On Selection Inside Selection > Length Based		
Parameters	Name	Type	Description
	<LengthOpParams>	Array	Structured array. <pre>Array("NAME:<OpName>", "RefineInside:=", <bool>, "Objects:=", <AssignmentObjects>, "Faces:=", <AssignmentFaces>, "Edges:=", <AssignmentEdges>, "RestrictElem:=", <bool> "NumMaxElem:=", <integer> "RestrictLength:=", <bool> "MaxLength:=", <value> "ApplyToInitialMesh:=", <bool>)</pre> RefineInside If true, Objects should be specified. Implies apply restrictions to tetrahedra inside the object. If false, Edges, Faces and/or Objects can be specified. Implies apply restrictions to triangles on the surface of the face or object.

	<table><tr><td></td><td></td><td> RestrictElem <i>If true</i>, NumMaxElem should be specified. RestrictLength <i>If true</i>, MaxLength should be specified. </td></tr></table>			RestrictElem <i>If true</i>, NumMaxElem should be specified. RestrictLength <i>If true</i>, MaxLength should be specified.
		RestrictElem <i>If true</i>, NumMaxElem should be specified. RestrictLength <i>If true</i>, MaxLength should be specified.		
Return Value	None.			

Python Syntax	AssignLengthOp(<LengthOpParams>)
Python Example	<pre>oModule.AssignLengthOp (["NAME:Length1", "RefineInside:=", True, "Objects:=", ["Box1"], "RestrictElem:=", True, "NumMaxElem:=", 1000, "RestrictLength:=", True, "MaxLength:=", "1mm" "ApplyToInitialMesh:=", True]) oModule.AssignLengthOp</pre>

	<pre>(["NAME:Length1", "RefineInside:=", False, "Edges:=", [58,73], "RestrictElem:=", True, "NumMaxElem:=", 1000, "RestrictLength:=", True, "MaxLength:=", "1mm" "ApplyToInitialMesh:=", True])</pre>

AssignModelResolutionOp

Assigns a model resolution name, value and unit for mesh operations, or specify to UseAutoFeaturelength. If UseAutoFeature length is true, the Defeature length is not used.

UI Access	Maxwell 3D > Mesh > Assign Mesh Operation > Model Resolution		
Parameters	Name	Type	Description
	<ModelResParams>	Array	Structured array. Array("NAME:<ModelResName>", "Objects:=", <array of object names>, "UseAutoLength:=", <boolean>,

			"DefeatureLength:=", "<value><units>")
Return Value	None.		

Python Syntax	AssignModelResolutionOp(<ModelResParams>)
Python Example	<pre>oModule.AssignModelResolutionOp (["NAME:ModelResolution1", "Objects:=", ["waveguide"], "UseAutoLength:=", True, "DefeatureLength:=", "71.5891053163818mil"])</pre>

AssignSkinDepthLayerSetting

Use: Assigns a skin-depth layer settings to the selected edges.

Command: Click **Mesh > Assign Mesh Operation > On Selection > Skin Depth Based**

Syntax: AssignSkinDepthLayerSetting Array("NAME:<LayerSettingName>", "Enabled:=", <SettingEnabled>, "Edges:=", Array(<EdgeIDArray>), "SkinDepth:=", "<SkinDepthValue>", "NumLayers:=", "<LayerNumber>")

Return Value: None

Parameters: <LayerSettingName>

The name of the skin depth layer setting.

```
<SettingEnabled>
  Boolean: "Enabled:=", true
<EdgeIDArray>
  Array of edge IDs. "Edges:=", Array(10, 12, 13)
<SkinDepthValue>
  Skin depth value with unit. "SkinDepth:=", "1mm"
<LayerNumber>
  Integer, the number of layers. "NumLayers:=", "2"
```

Example: oModule.AssignSkinDepthLayerSetting Array("NAME:SkinDepthLayer1",
"Enabled:=", true
"Edges:=", Array(10, 12, 13),
"SkinDepth:=", "1mm",
"NumLayers:=", 2)

AssignSkinDepthOp

Assigns a skin-depth based operations to the selection.

UI Access	Maxwell 3D > Mesh > Assign Mesh Operation > On Selection > Skin Depth Based		
Parameters	Name	Type	Description
	<SkinDepthOpParams>	Array	Structured array. Array ("NAME:<OpName>", "Faces:=", <AssignmentFaces>,

			<pre>"RestrictElem:=", <bool>, "NumMaxElem:=", <int>, "SkinDepth:=", <value>, "SurfTriMaxLength:=", <value>, "NumLayers:=", <int>)</pre> RestrictElem <i>If true, NumMaxElem should be specified.</i>
Return Value	None.		

Python Syntax	AssignSkinDepthOp(<SkinDepthOpParams>)
Python Example	<pre>oModule.AssignSkinDepthOp (["NAME:SkinDepth1", "Faces:=", [7], "RestrictElem:=", True, "NumMaxElem:=", 1000, "SkinDepth:=", "1mm", "SurfTriMaxLength:=", "1mm", "NumLayers:=", 2])</pre>

AssignTrueSurfOp

Assigns a true surface-based mesh operation on the selection.

UI Access	Maxwell 3D > Mesh > Assign Mesh Operation > Surface Approximation		
Parameters	Name	Type	Description
	<TrueSurfOpParams>	Array	Structured array. Array("NAME:<OpName>", "Faces:=", <AssignmentFaces>, "SurfDevChoice:=", <RadioOption>, "SurfDev:=", <value>, "NormalDevChoice:=", <RadioOption>, "NormalDev:=", <value>, "AspectRatioChoice:=", <RadioOption>, "AspectRatio:=", <double>) <RadioOption> Type: <int> 0: Ignore 1: Use defaults 2: Specify the value

Return Value	None.

Python Syntax	<code>AssignTrueSurfOp(<TrueSurfOpParams>)</code>
Python Example	<pre>oModule.AssignTrueSurfOp (["NAME:TrueSurf1", "Faces:=", [9], "SurfDevChoice:=", 2, "SurfDev:=", "0.04123105626mm", "NormalDevChoice:=", 2, "NormalDev:=", "15deg", "AspectRatioChoice:=", 1])</pre>

EditCylindricalGapOp

Modifies an existing cylindrical gap 3D mesh operation to enable/disable use of Clone Mesh and associated Band Mapping Angle.

Modifies an existing cylindrical gap 2D mesh operation to enable/disable use of Band Mapping Angle.

UI Access	Double-click the operation in the project tree to modify its settings.		
Parameters	Name	Type	Description
	<OpName>	String	Name of the operation to be edited.

	<i><CylindricalGapOpParams></i>	Array	Structured array for 3D mesh. <pre>Array("NAME:<OpName>", "CloneMesh:=", <bool>, "BandMappingAngle:=", <value> <i>[use if CloneMesh is "true"]</i> "MovingSideLayers:=", <integer>, "StaticSideLayers:=", <integer>)</pre> Structured array for 2D mesh. <pre>Array("NAME:<OpName>", "UseBandMappingAngle:=", <bool>, "BandMappingAngle:=", <real>) <i>[range is 0.0005deg through 3deg inclusive]</i></pre>
Return Value	None		

Python Syntax	<code>EditCylindricalGapOp(<OpName>, <CylindricalGapOpParams>)</code>
Python Example	<pre>oModule.EditCylindricalGapOp("CylindricalGap1", ["NAME:CylindricalGap1", "CloneMesh:=", True, "BandMappingAngle:=", 3deg, "MovingSideLayers:=", 1,</pre>

	<pre>"StaticSideLayers:=", 3]) Maxwell 2D Example oModule.EditCylindricalGapOp("CylindricalGap1", ["NAME:CylindricalGap1", "UseBandMappingAngle:=", True, "BandMappingAngle:=", 3deg,)</pre>

EditDensityControlOp

Edits an existing density control operation. This cannot be used to modify assignments. Instead, the mesh operation should be deleted and a new one created.

UI Access	Double-click the operation in the project tree to modify its settings.		
Parameters	Name	Type	Description
	<OpName>	String	Name of the operation to be edited.
	<DensityControlParams>	Array	Structured array.
			<pre>Array("NAME:<OpName>", "RefineInside:=", <bool>, "RestrictMaxElemLength:=", <bool>, "MaxElemLength:=", <value>, "RestrictLayersNum:=", <bool>, "LayersNum:=", <integer>)</pre>

			RefineInside If true, Objects should be specified. Implies apply restrictions to tetrahedra inside the object. If false, Faces and/or Objects can be specified. Implies apply restrictions to triangles on the surface of the face or object. RestrictMaxElemLength If true, MaxElemLength should be specified. RestrictLayersNum If true, LayersNum should be specified.
Return Value	None.		

Python Syntax	EditDensityControlOp(<OpName>, <DensityControlOpParams>)
Python Example	<pre>oModule.EditDensityControlOp("Clone Mesh Density1", ["NAME:Clone Mesh Density1", "RefineInside:=" , True, "RestrictMaxElemLength:=", True, "MaxElemLength:=" , "0.02mm", "RestrictLayersNum:=" , True,</pre>

	<pre>"LayersNum:=" , "15"])</pre>

EditEdgeCutLayerOp

Edits an existing edge cut operation. This cannot be used to modify assignments. Instead, the mesh operation should be deleted and a new one created.

UI Access	Double-click the operation in the project tree to modify its settings.		
Parameters	Name	Type	Description
	<OpName>	String	Name of the operation to be edited.
	<EdgeCutLayerOpParams>	Array	Structured array. <pre>Array("NAME:<OpName>", "Layer Thickness:=", <value>)</pre>
Return Value	None.		

Python Syntax	<pre>EditEdgeCutLayerOp(<OpName>, <EdgeCutLayerOpParams>)</pre>
Python Example	<pre>oModule.EditEdgeCutLayerOp("EdgeCut1", ["NAME:EdgeCut1", "Layer Thickness:=" , "0.66mm"])</pre>

EditLengthOp

Edits an existing length-based operation. This cannot be used to modify assignments. Instead, the mesh operation should be deleted and a new one created.

UI Access	Double-click the operation in the project tree to modify its settings.		
Parameters	Name	Type	Description
	<OpName>	String	Name of the operation to be edited.
	<LengthOpParams>	Array	Structured array. <pre>Array("NAME:<OpName>", "RefineInside:=", <bool>, "Objects:=", <AssignmentObjects>, "Faces:=", <AssignmentFaces>, "Edges:=", <AssignmentEdges>, "RestrictElem:=", <bool>, "NumMaxElem:=", <integer>, "RestrictLength:=", <bool>, "MaxLength:=", <value> "ApplyToInitialMesh:=", <bool>)</pre> RefineInside If true, Objects should be specified. Implies apply restrictions to tetrahedra inside the object. If false, Edges, Faces and/or Objects can be specified. Implies

			apply restrictions to triangles on the surface of the face or object. RestrictElem <i>If true</i>, NumMaxElem should be specified. RestrictLength <i>If true</i>, MaxLength should be specified.
Return Value	None.		

Python Syntax	EditLengthOp(<OpName>, <LengthOpParams>)
Python Example	<pre>oModule.EditLengthOpK("Length1", ["NAME:Length1", "RefineInside:=", True, "Objects:=", ["Circle2", "Rectangle1"], "RestrictElem:=", False, "NumMaxElem:=", 1000, "RestrictLength:=", True, "MaxLength:=", "2mm", "ApplyToInitialMesh:=", False])</pre>

EditModelResolutionOp

Assigns a model resolution name, value and unit for mesh operations. If UseAutoLength is true, the Defeature length is not used.

UI Access	Double-click the operation in the project tree to modify its settings.		
Parameters	Name	Type	Description
	<OpName>	String	Name of the operation to be edited.
	<ModelResParams>	Array	Structured array. Array ("NAME:<ModelResName>", "UseAutoLength:=", <boolean>, "DefeatureLength:=", "<value><units>")
Return Value	None.		

Python Syntax	EditModelResolutionOp(<OpName>, <ModelResParams>
Python Example	<pre>oModule.EditModelResolutionOp("ModelResolution1", ["NAME:ModelResolution1", "UseAutoLength:=", False, "DefeatureLength:=", "71.5891053163818mil"])</pre>

EditSkinDepthOp

Modifies an existing skin-depth based mesh operation. Assignments cannot be changed using this command. To change the assignment, you must delete operation and create it using a new assignment.

UI Access	Double-click the operation in the project tree to modify its settings.		
Parameters	Name	Type	Description
	<OpName>	String	Name of the operation to be edited.
	<SkinDepthOpParams>	Array	Structured array. <pre>Array("NAME:<OpName>", "RestrictElem:=", <bool>, "NumMaxElem:=", <int>, "SkinDepth:=", <value>, "SurfTriMaxLength:=", <value>, "NumLayers:=", <int>)</pre> RestrictElem <i>If true, NumMaxElem should be specified.</i>
Return Value	None.		

Python Syntax	EditSkinDepthOp(<OpName>, <SkinDepthOpParams>)
Python Example	<pre>oModule.EditSkinDepthOp("SkinDepth1", ["NAME:SkinD",</pre>

	<pre>"RestrictElem:=", False, "SkinDepth:=", "2mm", "SurfTriMaxLength:=", "1mm", "NumLayers:=", 2])</pre>

EditTrueSurfOp

Modifies an existing true surface approximation-based mesh operation. Assignments cannot be changed using this command. To change the assignment, delete this operation and create it using a new assignment.

UI Access	Double-click the operation in the project tree to modify its settings.		
Parameters	Name	Type	Description
	<OpName>	String	Name of the operation to be edited.
	<TrueSurfOpParams>	Array	Structured array.
			<pre>Array ("NAME:<OpName>", "SurfDevChoice:=", <RadioOption>, "SurfDev:=", <value>, "NormalDevChoice:=", <RadioOption>, "NormalDev:=", <value>, "AspectRatioChoice:=", <RadioOption>, "AspectRatio:=", <double>)</pre>

	<table><tr><td></td><td></td><td> <RadioOption> Type: <int> 0: Ignore 1: Use defaults 2: Specify the value </td></tr></table>			<RadioOption> Type: <int> 0: Ignore 1: Use defaults 2: Specify the value
		<RadioOption> Type: <int> 0: Ignore 1: Use defaults 2: Specify the value		
Return Value	None.			

Python Syntax	EditTrueSurfOp(<OpName>, <TrueSurfOpParams>)
Python Example	<pre>oModule.EditTrueSurfOp("TrueSurf2", ["NAME:trusurf", "SurfDevChoice:=", 2, "SurfDev:=", "0.03mm", "NormalDevChoice:=", 1, "AspectRatioChoice:=", 2, "AspectRatio:=", 10])</pre>

InitialMeshSettings

Assigns a true surface-based mesh operation to the selection.

UI Access	Mesh > Initial Mesh Settings		
Parameters	Name	Type	Description
	<InitialMeshSettingsParams>	Array	Structured array. Array ("NAME:<MeshSettings>", "CurvedSurfaceApproxChoice:=", <"UseSlider" or "Manual Settings"> "SliderMeshSettings:=", <integer 1 through 9>, "UseLegacyFaceterForTauVolumeMesh:=", <boolean>, "DynamicSurfaceResolution:=", <boolean>, "UseFlexMeshingForTAUvolumeMesh:=", <boolean>, "UseAlternativeMeshMethodsAsFallBack:=", <boolean>, "AllowPhiForLayeredGeometry:=", <boolean>)
Return Value	None.		

Python Syntax	InitialMeshSettings(<InitialMeshSettingsParams>)
Python Example	oModule.InitialMeshSettings

```
([
  "NAME:MeshSettings",
  [
    "NAME:GlobalSurfApproximation",
    "CurvedSurfaceApproxChoice:=", "UseSlider",
    "SliderMeshSettings:=" , 5
  ],
  [
    "NAME:GlobalCurvilinear",
    "Apply:=", False
  ],
  [
    "NAME:GlobalModelRes",
    "UseAutoLength:=", True
  ],
  [
    "NAME:GlobalLengthMeshSetup",
    "RefineInside:=", True,
    "ID:=", -1,
    "Type:=", "LengthBased",
    "RestrictElem:=", False,
```

	<pre>"NumMaxElem:=", "1000", "RestrictLength:=", True, "MaxLength:=", "45mm", "ApplyToInitialMesh:=", True, "IsGlobal:=", True], "MeshMethod:=", "Auto", "UseLegacyFaceterForTAUVolumeMesh:=", False, "DynamicSurfaceResolution:=", False, "UseFlexMeshingForTAUvolumeMesh:=", False, "UseAlternativeMeshMethodsAsFallBack:=", False, "AllowPhiForLayeredGeometry:=", True])</pre>

15 - Analysis Setup Module Script Commands

Maxwell analysis setup commands should be executed by the Analysis module, referred to in Maxwell scripts as the "AnalysisSetup" module.

Set oModule = oDesign.GetModule("AnalysisSetup")

[ClearLinkedData](#)

[CopySetup](#)

[DeleteSetups](#)

[EditSetup](#)

[ExportCircuit](#)

[ExportIcepak](#)

[ExportSolnData](#)

[GetSetupCount](#)

[GetSetups](#)

[GetSweepCount](#)

[GetSweeps](#)

[InsertSetup \(Maxwell\)](#)

[PasteSetup](#)

[PasteSweep](#)

[RenameSetup](#)

[ResetAllToTimeZero](#)

[ResetSetupToTimeZero](#)

[RevertAllToInitial](#)

[RevertAllToInitialTemperature](#)

[RevertSetupToInitial](#)

[RevertSetupToInitialTemperature](#)

ClearLinkedData (Module)

Clear the linked data of the specified solution setups. (This command is similar to the ClearLinkedData command for the *design* level, which clears the linked data for all solution setups in the design.)

UI Access	Project Manager > <i>{Design name}</i> > Analysis > right-click <i>{Setup name}</i> > Clear Linked Data or, with a solution setup selected in the Project Manager: Maxwell > Analysis > Clear Linked Data		
Parameters	Name	Type	Description
	<SetupNameArray>	Array	Specify the name of the setups whose linked data are to be cleaned
Return Value	None		

Python Syntax	ClearLinkedData(<SetupNameArray>)
Python Example	<code>oModule.ClearLinkedData(["setup1"])</code>

CopySetup

Copy the specified Optimetrics setup.

UI Access	NA		
Parameters	Name	Type	Description
	<SetupName>	String	Name of the setup.
Return Value	None.		

Python Syntax	CopySetup (<SetupName>)		
Python Example	oModule.CopySetup ("OptimizationSetup1")		

DeleteSetups

Deletes one or more solution setups, which are specified by an array of solution setup names.

UI Access	Right-click a solution setup in the project tree and then click Delete on the shortcut menu, or delete selected solution setups in the List dialog box.		
Parameters	Name	Type	Description
	<SetupArray>	Array	Array of solution setup names.
Return Value	None.		

Python Syntax	DeleteSetups (<SetupArray>)		
Python Example	oModule.DeleteSetups (["Setup1", "Setup2"])		

EditSetup [Maxwell]

Modifies an existing solution setup.

UI Access	Double-click a solution setup in the Project Manager tree to modify its settings.		
Parameters	Name	Type	Description
	<SetupName>	String	Name of the solve setup being edited.
	<Attributes>	Array	Structured array. Array("NAME:<NewSetupName>", <NamedParameters>) See the InsertSetup [Maxwell] command for details and examples.
Return Value	None		

Note: For AC Magnetic designs, `Frequency` also supports use of variables and expressions.

Python Syntax	EditSetup (<SetupName>, <Attributes>)
Python Example	<pre>3D AC Magnetic with Include DC Fields: oModule.EditSetup("AC Magnetic with DC", ["NAME:EddyCurrentDCFields", "Enabled:=" , True, ["NAME:MeshLink", "ImportMesh:=" , False], "MaximumPasses:=" , 10, "MinimumPasses:=" , 3,</pre>


```

"MinimumConvergedPasses:=", 1,
"PercentRefinement:=", 30,
"SolveFieldOnly:=", False,
"PercentError:=", 1,
"SolveMatrixAtLast:=", True,
"UseNonLinearIterNum:=", False,
"UseCacheFor:=", ["Pass"],
"UseIterativeSolver:=", False,
"RelativeResidual:=", 1e-05,
"NonLinearResidual:=", 0.0001,
"RelaxationFactor:=", 1,
"SmoothBHCurve:=", False,
"Frequency:=", "60Hz",
"HasSweepSetup:=", False,
"UseHighOrderShapeFunc:=", False,
"ImportMeshForMuLink:=", False,
"LossAdaptiveCtrl:=", "0.5",
"UseMuLink:=", False,
"DCPercentRefinement:=", 25,
"DCMinimumPasses:=", 4,
"DCMinConvergedPasses:=", 2,
"DCNonLinearResidual:=", 0.002,
"DCSmoothBHCurve:=", True,
"DCMaxumPasses:=", 15,
"DCPercentError:=", 2
])

```

Python Example: Transient EditSetup Objectlist and Facelist to limit field output to certain entities

```

oModule = oDesign.GetModule("AnalysisSetup")
oModule.EditSetup("Setup1",
[
    "NAME:Setup1",
    "Enabled:=", True,
    [
        "NAME:MeshLink",
        "ImportMesh:=", False
    ]
]
)

```

```
],
"NonlinearSolverResidual:=", "0.005",
"ScalarPotential:=", "Second Order",
"SmoothBHCurve:=", False,
"StopTime:=", "10ms",
"TimeStep:=", "1.66ms",
"OutputError:=", False,
"OutputPerObjectCoreLoss:=", False,
"OutputPerObjectSolidLoss:=", False,
"UseControlProgram:=", False,
"ControlProgramName:=", "",
"ControlProgramArg:=", "",
"CallCtrlProgAfterLastStep:=", False,
"FastReachSteadyState:=", False,
"AutoDetectSteadyState:=", False,
"IsGeneralTransient:=", True,
"IsHalfPeriodicTransient:=", False,
"ListsForFields:=", ["Objectlist1", "Facelist1"],
"SaveFieldsType:=", "None",
"UseNonLinearIterNum:=", False,
"CacheSaveKind:=", "Count",
"NumberSolveSteps:=", 1,
"RangeStart:=", "0s",
"RangeEnd:=", "0.1s",
"ContinuumAir:=", False
])
```

3D Electric Transient EditSetup:

```
oModule.EditSetup ("Setup1",
[
  "NAME:Setup1",
  "Tolerance:=", 0.005,
  [
```

```

    "NAME:Data",
    "SaveField:=", true,
    "Stop:=", "100s",
    "InitialStep:=", "0.01s",
    "MaxStep:=", "0.5s"
  ],
  "Initial Voltage:=", "0V",
  "NumberOfOutputVars:=", 0
])

```

3D Transient EditSetup:

```

oModule.EditSetup("Setup1",
[
  "NAME:Setup1", "Enabled:=", true,
  [
    "NAME:MeshLink", "ImportMesh:=", false
  ],
  "NonlinearSolverResidual:=", "0.005",
  "ScalarPotential:=", "Second Order",
  "SmoothBHCurve:=", false,
  "StopTime:=", "10000000ns",
  "TimeStep:=", "2000000ns",
  "OutputError:=", false,
  "OutputPerObjectCoreLoss:=", true,
  "OutputPerObjectSolidLoss:=", false,
  "UseControlProgram:=", false,
  "ControlProgramName:=", " ",
  "ControlProgramArg:=", " ",
  "CallCtrlProgAfterLastStep:=", false,
  "FastReachSteadyState:=", false,
  "AutoDetectSteadyState:=", false,
  "IsGeneralTransient:=", true,
  "IsHalfPeriodicTransient:=", false,
  "SaveFieldsType:=", "None",
  "UseNonLinearIterNum:=", false,

```

```
"CacheSaveKind:=", "Count",  
"NumberSolveSteps:=", 1,  
"RangeStart:=", "0s",  
"RangeEnd:=", "0.1s"  
])
```

3D AC Conduction EditSetup:

```
oModule.EditSetup("Setup1",  
[  
  "NAME:Setup1", "Enabled:=", true,  
  [  
    "NAME:MeshLink", "ImportMesh:=", false  
  ],  
  "MaximumPasses:=", 10,  
  "MinimumPasses:=", 2,  
  "MinimumConvergedPasses:=", 1,  
  "PercentRefinement:=", 30,  
  "SolveFieldOnly:=", false,  
  "PercentError:=", 1,  
  "SolveMatrixAtLast:=", true,  
  "UseNonLinearIterNum:=", true,  
  "MinIterNum:=", 1,  
  "MaxIterNum:=", 100,  
  "CacheSaveKind:=", "Delta",  
  "ConstantDelta:=", "0s",  
  "UseIterativeSolver:=", false,  
  "RelativeResidual:=", 1E-06,  
  "NonLinearResidual:=", 0.001,  
  "Frequency:=", "60Hz",  
  "HasSweepSetup:=", false  
])
```

2D DC Conduction EditSetup:

```
oModule.EditSetup("Setup1",
[
    "NAME:Setup1",
    "MaximumPasses:=", 20,
    "MinimumPasses:=", 2,
    "MinimumConvergedPasses:=", 1,
    "PercentRefinement:=", 30,
    "SolveFieldOnly:=", false,
    "PercentError:=", 1,
    "SolveMatrixAtLast:=", true,
    "UseOutputVariable:=", false,
    "PreAdaptMesh:=", false
])
```

2D Transient EditSetup:

```
Set oModule = oDesign.GetModule("AnalysisSetup")
oModule.EditSetup("Setup1",
[
    "NAME:Setup1",
    "Enabled:=", true,
    "NonlinearSolverResidual:=", "0.0001",
    "TimeIntegrationMethod:=", "BackwardEuler",
    "SmoothBHCurve:=", true,
    "StopTime:=", "0.2s",
    "TimeStep:=", "0.0002s",
    "OutputError:=", false,
    "OutputPerObjectCoreLoss:=", true,
    "OutputPerObjectSolidLoss:=", false,
    "UseControlProgram:=", false,
    "ControlProgramName:=", " ",
    "ControlProgramArg:=", " ",
    "CallCtrlProgAfterLastStep:=", false,
    "FastReachSteadyState:=", true,
    "AutoDetectSteadyState:=", true,
    "FrequencyOfAddedVoltageSource:=", "50Hz",
```

```
"StopCriterion:=", 0.005,  
"IsGeneralTransient:=", false,  
"IsHalfPeriodicTransient:=", true,  
"HasSweepSetup:=", true,  
"SaveFieldsType:=", "Custom",  
"Sweep Ranges:="   
  [  
    "SweepSetupType:=", "LinearStep",  
    "StartValue:=", "0",  
    "StopValue:=", "200ms",  
    "StepSize:=", "25ms"  
  ]  
1)
```

ExportCircuit

Use: Export equivalent circuit data.

Command: Right-click a setup in the project tree or the **Analysis** folder and choose **Export Circuit**.

Syntax: ExportCircuit <Solution>, <Variation>, <FileName>, <ExportSettings>, <ModelName>, <Freq>

Return Value: none

Parameters: <Solution>

<SetupName>:<SolutionName>

<SetupName>

Type: <string>

Name of the setup where circuit is being exported

<SolutionName>

Type: <String>

Name of the solution.

<Variation>

Type: <string>

The variation where circuit is being exported

<FileName>

Type: <string>

The name of the file where circuit is being exported

<ModelName>

Type: <String>

Model name or name of the sub circuit (Optional). If not specified then <FileName> is considered as model name.

<Freq>

Type: <double>

Sweep frequency in hz.

<ExportSettings>

Array("NAME:CircuitData", "MatrixName:=", _

<ReduceMatrix>, "NumberOfCells:=", <NumCell>, "UserHasChangedSettings:=", <User-ChangedSettings>, "IncludeCap:=", <IncludeCap>, "IncludeCond:=", <IncludeCond>, Array("NAME:CouplingLimits", <CouplingLimitsArray>, "IncludeDCR:=", <IncludeDCR>, "IncludeDCL:=", <IncludeDCL>, "IncludeACR:=", <IncludeACR>, "IncludeACL:=", <IncludeACL>, "ADDResistance:=", <AddResistance>)

Parameters:

<ReduceMatrix>

Type: <string>

One of the reduced matrix setup or "Original"

<NumCell>

Type: <string>

Number of cells in export. Can be a variable.

<UserChangedSettings>

Type: <bool>

Whether user changed settings or use default settings.wirebond

<IncludeCap>

Type: <bool>

Flag indicates whether to export Capacitance matrix.

<IncludeCond>

Type: <bool>

Flag indicates whether to export Conductance matrix.

<IncludeDCR>

Type: <bool>

Flag indicates whether to export DC resistance matrix.

<IncludeDCL>

Type: <bool>

Flag indicates whether to export DC Inductance matrix.

<IncludeACR>

Type: <bool>

Flag indicates whether to export AC resistance matrix.

<IncludeACL>

Type: <bool>

Flag indicates whether to export AC inductance matrix.

<AddResistance>

Type: <bool>

Adds the DC and AC resistance.

Note:

You cannot export both AC and DC matrices unless **AddResistance** is selected.

<CouplingLimitsArray>

```
Array("NAME:CouplingLimits", "CouplingLimitType=", <CouplingLimitType>, _  
<CouplingLimitsParameters>, 0.01, "CondFraction=", 0.01)
```

<CouplingLimitType> = "None"

Argument not needed

<CouplingLimitType> = "ByFraction"

<CouplingLimitsParameters>

"CapFraction=", <Fraction>, "IndFraction=", <Fraction>,

"ResFraction=", <Fraction>, "CondFraction=", <Fraction>,

Parameters:

<Fraction>

Type: <double>

Fraction of the self term

```
<CouplingLimitType> = "ByValue"
```

```
<CouplingLimitsParameters>
```

```
"CapLimit:=", <Limit>, "IndLimit:=", <Limit>, "ResLimit:=", <Limit>,
```

```
"CondLimit:=", <Limit>,
```

Parameters:

<Limit>

Type: <string>

Value of the limit.

ExportIcepak

Use: Exports Icepak data to the specified output directory.

Command: Right-click a setup under **Analysis** in the project tree, and select **Create Icepak Design**. From the dialog box, select an appropriate solution setup and variation, enter the path to the output directory, and click the **OK** button.

Syntax: ExportIcepak <SetupName>, <Variation>, <Output Directory>

Return Value: None

Parameters: <SetupName>

Type: string

Setup name.

<Variation>

Type: string

Variation values.

<Output Directory>

Type: string

Path to the output directory in which the Icepak design files will be saved.

Example: oModule.ExportIcepak "Setup1", "air_gap='1mm' diameter='120mm", "D:\Work\Icepak\"

ExportSolnData

Use: Exports solution data to a file.

Command: Right-click a parameter setup under **Parameters** in the project tree, and select **ViewSolution**. From the dialog box, select an appropriate solution type, and click the **Export Solution** button.

Syntax: ExportSolnData <SetupName>, <SolutionName>, <IsPostProcessed>, <Variation>, <ExportFileName>,

Return Value: None

Parameters: <SetupName>

Type: string

Setup name.

<SolutionName>

Type: string

Solution setup name.

<IsPostProcessed>

Type: bool

Is post processed (true/false).

<Variation>

Type: string

Variation values.

<ExportFileName>

Type: string

Export file name.

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 ()

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>

GetSweepCount

Gets the number of sweeps in a specified analysis setup.

UI Access	N/A		
Parameters	Name	Type	Description
	<SetupName>	String	Name of specified setup.
Return Value	Integer containing number of sweeps for the named setup.		

Python Syntax	GetSweepCount (<SetupName>)
Python Example	<code>oModule.GetSweepCount ("Setup1")</code>

GetSweeps

Gets the names of all sweeps in a given analysis setup.

UI Access	N/A		
Parameters	Name	Type	Description
	<SetupName>	String	Name of specified setup.
Return Value	Array of sweep names.		

Python Syntax	GetSweeps (<SetupName>)
Python Example	<code>oModule.GetSweeps ("Setup1")</code>

InsertSetup [Maxwell]

This command adds a new solution setup. The information on this page is specific to the implementation of the InsertSetup command for Maxwell 2D and 3D designs.

The parameters depend on the following model characteristics:

- whether the design is 2D or 3D
- which type of solution you are setting up (*SetupType* = "[Magnetostatic](#)", "[AC Magnetic](#)", "[AC Magnetic APhi](#)", "[AC Magnetic with DC](#)", "[Transient](#)", "[Transient APhi](#)", "[Electrostatic](#)", "[DC Conduction](#)", "[Electric DC Conduction](#)", "[Electric Transient](#)", and "[AC Conduction](#)")

- whether you are importing the mesh from a source design
- whether there are any variables mapped between the source and target designs (when importing a mesh)

Certain options, when selected, activate additional parameters. For example, Solver limiting parameters (**Minimum** and **Maximum**) are only applicable if the **Number of Nonlinear Iterations** is checked.

UI Access	Maxwell 3D or Maxwell 2D > Analysis Setup > Add Solution Setup		
Parameters	Name	Type	Description
	<SetupName>	String	Type of the solve setup being added. Possible values for 3D designs are "Magnetostatic", "AC Magnetic", "AC Magnetic APhi", "AC Magnetic with DC", "Transient", "Transient APhi", "Electrostatic", "DC Conduction", "Electric DC Conduction", "Electric Transient", "AC Conduction". Possible values for 2D designs are "Magnetostatic", "AC Magnetic", "Transient", "Electrostatic", "DC Conduction", "AC Conduction".
	<Attributes>	Array	Structured array. <code>Array("NAME:<NewSetupName>", <NamedParameters>)</code> The named parameters vary according to the setup type and settings selected for a setup. See the typical examples below for each solution type. The parameters correspond to fields in the Setup window.
Return Value	None		

Python Syntax	InsertSetup (<SetupName>, <Attributes>)
Python Example	3D AC Magnetic with Permeability Computation <code>oModule.InsertSetup("AC Magnetic",</code>

```
[
  "NAME:ACMagnetic",
  "Enabled:="                , True,
  [
    "NAME:MeshLink",
    "ImportMesh:="           , False
  ],
  "MaximumPasses:="          , 4,
  "MinimumPasses:="          , 2,
  "MinimumConvergedPasses:=" , 1,
  "PercentRefinement:="      , 30,
  "SolveFieldOnly:="         , False,
  "PercentError:="           , 0.1,
  "SolveMatrixAtLast:="      , True,
  "UseNonLinearIterNum:="     , False,
  "UseCacheFor:="            , ["Pass"],
  "UseIterativeSolver:="     , False,
  "RelativeResidual:="       , 1E-05,
  "NonLinearResidual:="      , 0.0001,
  "RelaxationFactor:="       , 1,
  "SmoothBHCurve:="         , False,
  "Frequency:="              , "150kHz",
  "HasSweepSetup:="          , False,
  "UseHighOrderShapeFunc:="  , False,
  "UseMuLink:="              , True,
  "ImportMeshForMuLink:="    , True,
  "LossAdaptiveCtrl:="       , "0.5",
  "Project:="                , "This Project*",
  "Product:="                , "Maxwell",
  "Design:="                 , "Wireless_Charging_PM_Source",
  "Soln:="                   , "Setup1 : LastAdaptive",
  [
    "NAME:Params"
  ]
]
```



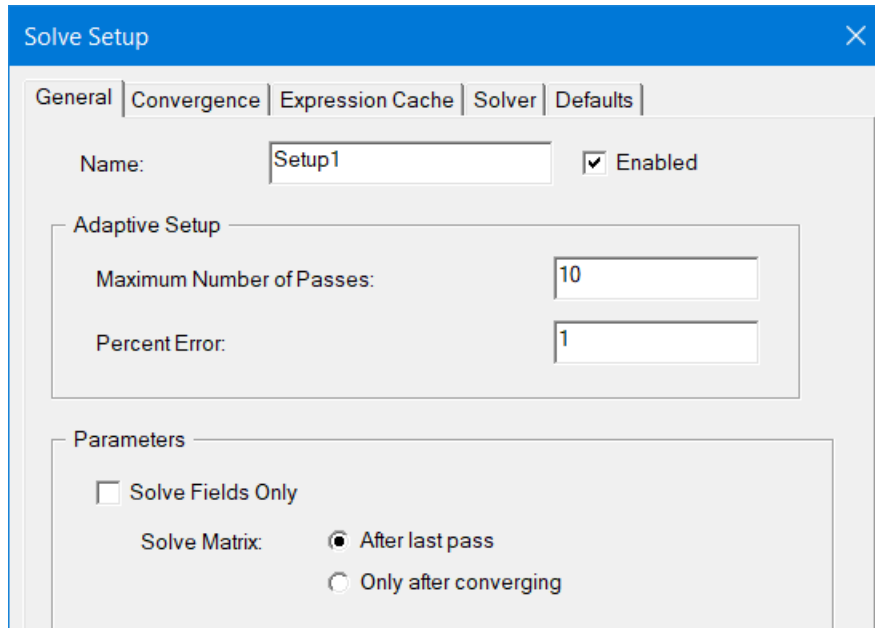
```
],
"ForceSourceToSolve:=" , True,
"PreservePartnerSoln:=" , True,
"PathRelativeTo:=" , "TargetProject"
])
```

Tip: For the Maxwell solver, there are numerous parameter commands for which the availability is determined by whether or not the design is 2D or 3D and by the solution type. Listing all of these commands in this topic would be unwieldy and difficult to navigate. The easiest way to identify all the parameters for a certain solver is to record a *.py* or *.vbs* script of the solver setup in the software, and then view the output, which will list all the parameter commands used that setup. The resulting script can then be easily edited and run. See [Recording a Script](#) for more details.

To understand how the parameter commands correspond to settings in the Solve Setup dialog box, let's look at a script for a Maxwell 3D [Magnetostatic](#) design:

```
oModule.InsertSetup("Magnetostatic",
["NAME:Setup1",
"MaximumPasses:=", 10,
"MinimumPasses:=", 2,
"MinimumConvergedPasses:=", 1,
"PercentRefinement:=", 30,
"SolveFieldOnly:=", false,
"PercentError:=", 1,
"SolveMatrixAtLast:=", true,
"UseOutputVariable:=", false,
"PreAdaptMesh:=", false,
"UseNonLinearIterNum:=" , True,
"MinIterNum:=" , 5,
"MaxIterNum:=" , 10,
"NonLinearResidual:=", 0.001,
"MuNonLinearBH:=", true,
"ComputeHc:=", false,
```

```
"HcNonLinearBH:=", true,  
"UserOverrideMu:=", true,  
"UserExePath:=", "E:\views\projects\ucp.exe",  
"UserExeArg:=", "-3D"])
```



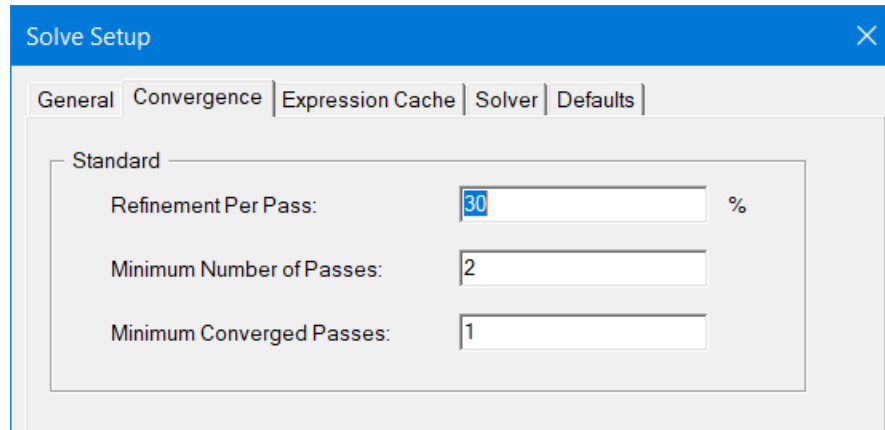
Some of the Array attributes correspond to these fields on the **General** tab of the Setup.

```
oModule.InsertSetup("Magnetostatic",  
["NAME: Setup1",  
"MaximumPasses:=", 10,
```

...

```
"SolveMatrixAtLast:=", true,
```

Other parameters correspond to fields in the **Convergence** tab of the Setup.



```
"MinimumPasses:=", 2,
"MinimumConvergedPasses:=", 1,
"PercentRefinement:=", 30,
```

Still other parameters correspond to fields in the **Solver** tab of the Setup.

```
"NonLinearResidual:=", 0.001,
UseNonLinearIterNum:=" , True,
"MinIterNum:=" , 5,
"MaxIterNum:=" , 10,
```

The examples that follow demonstrate the commands for different setups.

3D [AC Magnetic](#) solve setup:

```
oModule.InsertSetup("AC Magnetic",
[
  "NAME:ACMagnetic_03",
  "Enabled:=" , True,
  [
    "NAME:MeshLink",
    "ImportMesh:=" , False
  ],
  "MaximumPasses:=" , 4,
  "MinimumPasses:=" , 2,
  "MinimumConvergedPasses:=", 1,
  "PercentRefinement:=" , 30,
  "SolveFieldOnly:=" , False,
  "PercentError:=" , 0.1,
  "SolveMatrixAtLast:=" , True,
  "UseNonLinearIterNum:=" , False,
  [
    "NAME:ExpressionCache",
    [
      "NAME:CacheItem",
      "Title:=" , "eddy_loss1",
      "Expression:=" , "eddy_loss",
      "Intrinsics:=" , "Phase=\'0deg\'",
      "ReportType:=" , "Fields",
      [
        "NAME:ExpressionContext"
      ]
    ]
  ],
  "UseCacheFor:=" , ["Pass"],
  "UseIterativeSolver:=" , False,
  "RelativeResidual:=" , 0.0001,
  "NonLinearResidual:=" , 0.0001,
```

```

    "SmoothBHCurve:=" , False,
    "Frequency:=" , "200Hz",
    "HasSweepSetup:=" , False,
    "UseHighOrderShapeFunc:=" , False,
    "UseMuLink:=" , False,
    "LossAdaptiveCtrl:=" , "0.3"
  ])

```

2D [AC Magnetic](#) solve setup with two-way thermal coupling

```

oModule.InsertSetup("AC Magnetic",
[
  "NAME:Setup1",
  "Enabled:=" , True,
  [
    "NAME:MeshLink",
    "ImportMesh:=" , False
  ],
  "MaximumPasses:=" , 10,
  "MinimumPasses:=" , 2,
  "MinimumConvergedPasses:=" , 1,
  "PercentRefinement:=" , 30,
  "SolveFieldOnly:=" , False,
  "PercentError:=" , 1,
  "SolveMatrixAtLast:=" , True,
  "UseNonLinearIterNum:=" , False,
  "NonLinearResidual:=" , 0.0001,
  "RelaxationFactor:=" , 0.4, # only available for 2D designs with two-way thermal coupling
  "SmoothBHCurve:=" , False,
  "Frequency:=" , "60Hz",
  "UseLossConvergence:=" , False,
  "HasSweepSetup:=" , False
])

```

DC Biased AC Magnetic solve setup:

```

oModule.InsertSetup("AC Magnetic with DC",

```

```
[
  "NAME:Setup2",
  "Enabled:=" , True,
  [
    "NAME:MeshLink",
    "ImportMesh:=" , False
  ],
  "MaximumPasses:=" , 10,
  "MinimumPasses:=" , 2,
  "MinimumConvergedPasses:=" , 1,
  "PercentRefinement:=" , 30,
  "SolveFieldOnly:=" , False,
  "PercentError:=" , 1,
  "SolveMatrixAtLast:=" , True,
  "UseNonLinearIterNum:=" , False,
  "UseIterativeSolver:=" , False,
  "RelativeResidual:=" , 1E-05,
  "NonLinearResidual:=" , 0.0001,
])
```

Note: For AC Magnetic designs, `Frequency` also supports use of variables and expressions.

AC Magnetic with A-Phi solve setup:

```
oModule.InsertSetup("AC Magnetic APhi",
[
  "NAME:Setup2",
  "Enabled:=" , True,
  [
    "NAME:MeshLink",
    "ImportMesh:=" , False
  ],
],
```

```

"MaximumPasses:=" , 10,
"MinimumPasses:=" , 2,
"MinimumConvergedPasses:=" , 1,
"PercentRefinement:=" , 30,
"SolveFieldOnly:=" , False,
"PercentError:=" , 1,
"SolveMatrixAtLast:=" , True,
"UseNonLinearIterNum:=" , False,
"NonLinearResidual:=" , 0.0001,
"Frequency:=" , "60Hz",
"HasSweepSetup:=" , False
])

```

3D [Transient](#) solve setup with ObjectLists and FaceLists to save fields on specific entities:

```

oModule = oDesign.GetModule("AnalysisSetup")
oModule.InsertSetup("Transient",
[
    "NAME:Setup2",
    "Enabled:=" , True,
    [
        "NAME:MeshLink",
        "ImportMesh:=" , False
    ],
    "NonlinearSolverResidual:=" , "0.005",
    "ScalarPotential:=" , "Second Order",
    "SmoothBHCurve:=" , False,
    "StopTime:=" , "10000000ns",
    "TimeStep:=" , "2000000ns",
    "OutputError:=" , False,
    "OutputPerObjectCoreLoss:=" , False,
    "OutputPerObjectSolidLoss:=" , False,
    "UseControlProgram:=" , False,
    "ControlProgramName:=" , " ",
    "ControlProgramArg:=" , " ",
    "CallCtrlProgAfterLastStep:=" , False,

```

```
"FastReachSteadyState:=", False,  
"AutoDetectSteadyState:=", False,  
"IsGeneralTransient:=", True,  
"IsHalfPeriodicTransient:=", False,  
"ListsForFields:=", ["Objectlist1","Facelist1"],  
"SaveFieldsType:=", "None",  
"UseNonLinearIterNum:=", False,  
"CacheSaveKind:=", "Count",  
"NumberSolveSteps:=", 1,  
"RangeStart:=", "0s",  
"RangeEnd:=", "0.1s",  
"ContinuumAir:=", False
```

3D [Transient A-Phi](#) solve setup:

```
oModule.InsertSetup("Transient APhi",  
[  
  "NAME:Setup2",  
  "Enabled:=", True,  
  [  
    "NAME:MeshLink",  
    "ImportMesh:=", False  
  ],  
  "NonlinearSolverResidual:=", "0.005",  
  "ScalarPotential:=", "Second Order",  
  "SmoothBHCurve:=", False,  
  "StopTime:=", "0.07s",  
  "TimeStep:=", "0.0013s",  
  "OutputError:=", False,  
  "UseControlProgram:=", False,  
  "ControlProgramName:=", "",  
  "ControlProgramArg:=", "",  
  "CallCtrlProgAfterLastStep:=", False,  
  "FastReachSteadyState:=", False,
```



```

"AutoDetectSteadyState:=", False,
"IsGeneralTransient:=", True,
"IsHalfPeriodicTransient:=", False,
"SaveFieldsType:=", "None",
"UseNonLinearIterNum:=", False,
"CacheSaveKind:=", "Count",
"NumberSolveSteps:=", 1,
"RangeStart:=", "0s",
"RangeEnd:=", "0.1s"
])

```

3D [Electrostatic](#) solve setup

```

oModule.InsertSetup (Electrostatic",

["NAME:Setup1",
"MaximumPasses:=", 10,
"MinimumPasses:=", 2,
"MinimumConvergedPasses:=", 1,
"PercentRefinement:=", 30,
"SolveFieldOnly:=", false,
"PercentError:=", 1,
"SolveMatrixAtLast:=", true,
"UseOutputVariable:=", false,
"PreAdaptMesh:=", false)]

```

3D Electric [DC Conduction](#) solve setup

```

oModule.InsertSetup("Electric DC Conduction",

["NAME:Setup3",
"MaximumPasses:=", 10,
"MinimumPasses:=", 2,
"MinimumConvergedPasses:=", 1,
"PercentRefinement:=", 30,
"SolveFieldOnly:=", false,
"PercentError:=", 1,
"SolveMatrixAtLast:=", true,

```

```
"UseOutputVariable:=", false,  
"PreAdaptMesh:=", false])
```

3D [Electric Transient](#) solve setup

```
oModule.InsertSetup("Electric Transient",  
[  
  "NAME:Setup1",  
  "Tolerance:=", 0.005,  
  
  ["NAME:Data",  
   "SaveField:=", true,  
   "Stop:=", "100s",  
   "InitialStep:=", "0.01s",  
   "MaxStep:=", "5s"],  
  
  "Initial Voltage:=", "0V",  
  "NumberOfOutputVars:=", 0])
```

3D [AC Conduction](#) solve setup:

```
oModule.InsertSetup("AC Conduction",  
[  
  "NAME:Setup2",  
  "Enabled:=" , True,  
  [  
    "NAME:MeshLink",  
    "ImportMesh:=" , False  
  ],  
  "MaximumPasses:=" , 10,  
  "MinimumPasses:=" , 2,  
  "MinimumConvergedPasses:=", 1,  
  "PercentRefinement:=" , 30,  
  "SolveFieldOnly:=" , False,
```

```

"PercentError:=" , 1,
"SolveMatrixAtLast:=" , True,
"UseNonLinearIterNum:=" , True,
"MinIterNum:=" , 1,
"MaxIterNum:=" , 100,
"UseIterativeSolver:=" , False,
"RelativeResidual:=" , 1E-06,
"NonLinearResidual:=" , 0.001,
"Frequency:=" , "60Hz",
"HasSweepSetup:=" , False
])

```

PasteSetup

Use: Paste a solve setup.

Syntax: PasteSetup

Return Value: None

PasteSweep

Pastes a copied sweep.

UI Access	Right-click on a setup, select Paste		
Parameters	Name	Type	Description
	<SetupName>	String	Name of setup where the copied sweep is pasted to.
Return Value	None.		

Python Syntax	PasteSweep (<SetupName>)
Python Example	oModule.PasteSweep("Setup1")

RenameSetup

Renames an existing solution setup.

UI Access	Right-click a solution setup in the Project Manager and then click Rename on the shortcut menu.		
Parameters	Name	Type	Description
	<OldSetupName>	String	Name of the solution setup being renamed.
	<NewSetupName>	String	New name for the solution setup.
Return Value	None.		

Python Syntax	RenameSetup (<OldSetupName>, <NewSetupName>)
Python Example	<code>oModule.RenameSetup("Setup1", "MySetup")</code>

ResetAllToTimeZero

Forces the next solve to start from time 0 for all setups. Applies only to the Transient solution type.

Command: Right-click **Analysis** in the project tree, and select **Revert to Time Zero**.

Syntax:ResetAllToTimeZero

Return Value: None

Parameters: None

Example: `oModule.ResetAllToTimeZero`

ResetSetupToTimeZero

Use: Forces the next solve to start from time 0 for a given setup. Applies only to the Transient solution type.

Command: Right-click a setup in the project tree, and select **Revert to Time Zero**.

Syntax: ResetSetupToTimeZero <SetupName>

Return Value: None

Parameters: <SetupName>

Type: string

Setup name

Example: oModule.ResetSetupToTimeZero "Setup1"

RevertAllToInitial

Marks the current mesh for all solution setups as invalid. This will force the next simulation to begin with the initial mesh.

UI Access	Maxwell > Analysis Setup > Revert to Initial Mesh.
Parameters	None.
Return Value	None.

Python Syntax	RevertAllToInitial ()
Python Example	oModule.RevertAllToInitial ()

RevertAllToInitialTemperature

Reverts all setups to their initial temperature. Used when coupling with thermal designs.

UI Access	Maxwell > Set Object Temperature > Enable Feedback. Then in the Project Manager, right-click Analysis > Revert to Initial Temperature . Repeat for all setups.
Parameters	None.
Return Value	None.

Python Syntax	RevertAllToInitialTemperature()
Python Example	<code>oModule.RevertAllToInitialTemperature()</code>

RevertSetupToInitial

Marks the current mesh for a solution setup as invalid. This will force the next simulation to begin with the initial mesh.

UI Access	Right-click a solution setup in the Project Manager and then click Rename on the shortcut menu.		
Parameters	Name	Type	Description
	<SetupName>	String	Name of specified setup.
Return Value	None.		

Python Syntax	<code>RevertSetupToInitial(<SetupName>)</code>
Python Example	<code>oModule.RevertSetupToInitial("Setup1")</code>

RevertSetupToInitialTemperature

Reverts a specified setup to its initial temperature. Used when coupling with thermal designs.

UI Access	Maxwell > Set Object Temperature > Enable Feedback. Then in the Project Manager, right-click Analysis > Revert to Initial Temperature.		
Parameters	Name	Type	Description
	<code><setupName></code>	String	Name of solution setup.
Return Value	None		

Python Syntax	<code>RevertSetupToInitialTemperature(<setupName>)</code>
Python Example	<code>oModule.RevertSetupToInitialTemperature('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](#)

General Commands Recognized by the Optimetrics Module

Following are general script commands recognized by the **Optimetrics** module:

[CopySetup](#)

[DistributedAnalyzeSetup](#)

[EditSetup](#)

[ExportDXConfigFile](#)

[ExportOptimetricsProfile](#)

[ExportOptimetricsResult](#)

[ExportParametricResults](#)

[GetOptimetricResult](#)

[GetPropNames \[Optimetrics\]](#)

[GetPropValue \[Optimetrics\]](#)

[GetSetupNames \[Optimetrics\]](#)

[GetSetupNamesByType \[Optimetrics\]](#)[ImportSetup](#)[PasteSetup \[Optimetrics\]](#)[RenameSetup \[Optimetrics\]](#)[SetPropValue \[Optimetrics\]](#)[SolveSetup \[Optimetrics\]](#)[SolveAllSetup](#)

CopySetup

Copy the specified Optimetrics setup.

UI Access	NA		
Parameters	Name	Type	Description
	<SetupName>	String	Name of the setup.
Return Value	None.		

Python Syntax	CopySetup (<SetupName>)
Python Example	oModule.CopySetup ("OptimizationSetup1")

DeleteSetups [Optimetrics]

Deletes the specified Optimetrics setups.

UI Access	Right-click the setup in the project tree, and then click Delete on the shortcut menu

Parameters	Name	Type	Description
	<NameArray>	Array of Strings	An Array of Setup Names
Return Value	None		

Python Syntax	DeleteSetups (<NameArray>)
Python Example	<code>oModule.DeleteSetups (["OptimizationSetup1"])</code>

DistributedAnalyzeSetup

Distributes all variable value instances within a parametric sweep to different machines already specified from within the user interface

UI Access	Right-click the parametric setup name in the project tree and select Distribute Analysis.		
Parameters	Name	Type	Description
	<ParametricSetupName>	String	Name of the Setup
	isBlocking	Boolean	An optional arg that defaults to <code>true</code> . If it's <code>false</code> , the command immediately returns while the analysis or analyses run in the background. The status of the analyses can be checked with the <code>AreThereSimulationsRunning</code> command.
Return Value	None		

Python Syntax	DistributedAnalyzeSetup (<ParametricSetupName>)

Python Example	<code>oModule.DistributedAnalyzeSetup("ParametricSetup1")</code>

ExportDXConfigFile

Create an xml file with the setup information for Design Xplorer

UI Access	Right click on the Design Xplorer setup in the project tree and choose Export External Connector Addin Configuration...		
Parameters	Name	Type	Description
	<SetupName>	String	Must be one of existing DesignExplorer setup names
	<FileName>	String	Must be a valid file path and name
Return Value	None		

Python Syntax	<code>ExportDXConfigFile (<SetupName>, <FileName>)</code>
Python Example	<code>oModule.ExportDXConfigFile ("DesignXplorerSetup1", "c:/exportdir/DXSetup1.xml")</code>

ExportOptimetricsProfile

Export Optimetrics profile data

UI Access	Right click on the Optimetrics setup in the project tree and choose View Analysis Result... On the Post Analysis Display dialog box, click the Profile tab and click on the Export button.

Parameters	Name	Type	Description
	<SetupName>	String	Must be one of the existing Parametric, Optimization, Sensitivity, Statistical or DesignXplorer setup names
	<FileName>	String	Must be a valid file path and name with extension of csv, tab, dat, or txt
	[profileNum]	String	Must be a numeric string. Optional: defaulted to last profile number. It should be a zero indexed profile number.
Return Value	None		

Python Syntax	ExportOptimetricsProfile (<SetupName>, <FileName>, [profileNum])
Python Example	<pre>oModule.ExportOptimetricsProfile ("StatisticalSetup1", "c:/exportdir/test.csv")</pre>

ExportOptimetricsResult

Export an existing Optimization, Sensitivity, Statistical or DesignXplorer result. (Does not export Parametric results.)

UI Access	Right click on the desired Optimetrics setup in the project tree and choose View Analysis Result... On the Post Analysis Display dialog box, click the Result tab, then select Table view, and click on the Export button		
Parameters	Name	Type	Description
	<SetupName>	String	Must be one of the existing Optimization, Sensitivity, Statistical, or DesignXplorer setup names
	<FileName>	String	Must be a valid file path and name with extension of csv, tab, dat, or txt..
	[useFullOutputName]	Boolean	Optional: defaulted to false. If set to true values will be printed with units. This parameter is ignored for Optimization and Statistical results.

Return Value	None

Python Syntax	ExportOptimetricsResult (<SetupName>, <FileName>, [useFullOutputName])
Python Example	<pre>oModule.ExportOptimetricsResult ("StatisticalSetup1", "c:/exportdir/test.csv", false)</pre>

ExportParametricResults

Export existing Parametric results.

UI Access	Right click on the desired Parametric setup in the project tree and choose View Analysis Result... On the Post Analysis Display dialog box, click the Result tab, then select Table view, and click on the Export button.		
Parameters	Name	Type	Description
	<SetupName>	String	Must be one of the existing Parametric setup names
	<FileName>	String	Must be a valid file path and name with extension of csv, tab, dat or txt
	<bOutputUnits>	Boolean	If set to true, values will be printed with units
Return Value	None		

Python Syntax	ExportParametricResults (<SetupName>, <FileName>, <bOutputUnits>)
Python Example	<pre>oModule.ExportParametricResults ("ParametricSetup1", "c:/exportdir/test.csv", False)</pre>

GetSetupNames [Optimetrics]

Gets a list of Optimetrics setup names

UI Access	NA		
Parameters	Name	Type	Description
	None		
Return Value	IAnsoftCollectionObj – a collection of Optimetrics setup names		

Python Syntax	GetSetupNames()		
Python Example	<pre>oModule = oDesign.GetModule("Optimetrics") setupNames = oModule.GetSetupNames()</pre>		

GetSetupNamesByType [Optimetrics]

Gets a list of Optimetrics setup names by type.

UI Access	NA		
Parameters	Name	Type	Description
	<Optimetrics type>	String	Examples: parametric, optimization, statistical, sensitivity
Return Value	Array of Optimetrics setup names of the given type.		

Python Syntax	GetSetupNamesByType (<Optimetrics type>)
Python Example	<pre>for name in oModule.GetSetupNamesByType("optimization") AddInfoMessage(str(name))</pre>

ImportSetup

Import an Optimetric setup from a file.

UI Access	NA		
Parameters	Name	Type	Description
	<SetupTypeName>	String	Must be one of "OptiParametric" , "OptiOptimization", "OptiSensitivity", "OptiStatistical", or "OptiDesignExplorer".
	<SetupInfo>	Array	Array("NAME:<SetupName>", "FilePath") <SetupName> Type: <string> Name of the setup. <FilePath> Type : <string: file path> Must be a valid file path and name.
Return Value	None		

Python Syntax	ImportSetup (<SetupTypeName>, <SetupInfo>)

Python Example	<pre>oModule.ImportSetup ("OptiStatistical", ["NAME:StatisticalSetup1", "c:/importdir/mySetupInfoFile"])</pre>

PasteSetup [Optimetrics]

Pastes the specified Optimetrics setup.

UI Access	NA		
Parameters	Name	Type	Description
	<SetupName>	String	Name of the Setup
Return Value	None		

Python Syntax	PasteSetup (<SetupName>)
Python Example	<pre>oModule.PasteSetup ("OptimizationSetup1")</pre>

RenameSetup [Optimetrics]

Renames the specified Optimetrics setup.

UI Access	Right-click the setup in the project tree, and then click Rename on the shortcut menu.

Parameters	Name	Type	Description
	<OldName>	String	The name that needs to be replaced
	<NewName>	String	Replacement name
Return Value	None		

Python Syntax	RenameSetup (<OldName> <NewName>)
Python Example	<code>oModule.RenameSetup ("OptimizationSetup1" "MyOptimization")</code>

SolveSetup [Optimetrics]

Solves the specified Optimetrics setup.

UI Access	Right-click the setup in the project tree, and then click Analyze on the shortcut menu.		
Parameters	Name	Type	Description
	<SetupName>	String	Name of the setup to be solved
	isBlocking	Boolean	An optional arg that defaults to <code>true</code> . If it's <code>false</code> , the command immediately returns while the analysis or analyses run in the background. The status of the analyses can be checked with the <code>AreThereSimulationsRunning</code> command.
Return Value	None		

Python Syntax	SolveSetup (<SetupName>)
Python Example	<code>oModule.SolveSetup ("OptimizationSetup1")</code>

SolveAllSetup

Solves all Optimetrics setups

UI Access	Right-click on Optimetrics in Project Manager and select Analyze>All from context menu		
Parameters	Name	Type	Description
	isBlocking	Boolean	An optional arg that defaults to <code>true</code> . If it's <code>false</code> , the command immediately returns while the analysis or analyses run in the background. The status of the analyses can be checked with the <code>AreThereSimulationsRunning</code> command.
Return Value	None		

Python Syntax	<code>SolveAllSetup()</code>
Python Example	<code>oModule.SolveAllSetup()</code>

Parametric Script Commands

[EditSetup \[Parametric\]](#)

[ExportParametricSetupTable](#)

[GenerateVariationData \[Parametric\]](#)

[InsertSetup \[Parametric\]](#)**EditSetup [Parametric]**

Modifies an existing parametric setup

UI Access	Right-click the setup in the project tree, and then click Properties on the shortcut menu.		
Parameters	Name	Type	Description
	<SetupName>	String	Name of the Setup
	<ParametricParams>	List	List that defines the parameters of the parametric setup; examples are listed below.
Return Value	None		

Python Syntax	EditSetup (<SetupName>, <ParametricParams>)
Python Example	See EditSetup [Optimization]

ExportParametricSetupTable

Exports the parametric setup table as a CSV file.

UI Access	Double-click parametric setup. Select Table tab. Click Export .		
Parameters	Name	Type	Description
	<SetupName>	String	Name of the setup.
	<filePath>	String	Full path for file export.
Return Value	None		

Python Syntax	ExportParametricSetupTable (<SetupName>, <filePath>)
Python Example	<code>oModule.ExportParametricSetupTable('ParametricSetup1', 'E:/Files/ParametricSetup1_Table.csv')</code>

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.

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:=",

			<code><SimSetups></code> , <code><SweepDefs></code> , <code><SweepOps></code> , <code>Array("NAME:Goals", Array("NAME:Goal", <code><OptiGoalSpec></code>), ... Array("NAME:Goal", <code><OptiGoalSpec></code>))</code>
	<code><SetupName></code>	String	Name of the parametric setup.
	<code><SimSetups></code>	Array of Strings	An array of Twin Builder solution setup names.
	<code><SweepDefs></code>	Array	<code>Array("NAME:Sweeps",</code> <code>Array("NAME:SweepDefinition", "Variable:=",</code> <code><VarName></code> , <code>"Data:=", <SweepData></code> , <code>"Synchronize:=", <SyncNum>), ...</code> <code>Array("NAME:SweepDefinition", "Variable:=",</code> <code><VarName></code> , <code>"Data:=", <SweepData></code> , <code>"Synchronize:=", <SyncNum>))</code>
	<code><SweepData></code>	String	<code>"<SweepType></code> , <code><StartV></code> , <code><StopV></code> , <code><StepV>"</code>
	<code><SweepType></code>	String	The type of sweep data.
	<code><SyncNum></code>	Integer	<code>SweepData</code> s with the same value are synchronized.
	<code><SweepOps></code>		<code>Array("NAME:Sweep Operations", "<OpType>:=",</code> <code>Array(<VarValue></code> , <code>..., <VarValue>), ...</code> <code><OpType>:=", Array(<VarValue></code> , <code>..., <VarValue>))</code>
	<code><OpType></code>	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",</pre>

```
"Variable:=", "$length",  
"Data:=", "LIN 8mm 12mm 2mm",  
"OffsetF1:=", false,  
"Synchronize:=", 0  
]  
],  
[  
  "NAME:Sweep Operations"  
],  
[  
  "NAME:Goals",  
  [  
    "NAME:Goal",  
    "Solution:=", "Setup1 : LastAdaptive",  
    "Calculation:=", "returnloss",  
    "Context:=", ""  
    [  
      "NAME:Ranges",  
      "Range:=",  
      [  
        "Var:=", "Freq", "Type:=", "s"
```

```
        "Start:=", "8GHz",  
        "Stop:=", "8GHz"  
    ]  
],  
[  
    "NAME:Goal",  
    "Solution:=", "Setup1 : LastAdaptive"  
    "Calculation:=", "reflect",  
    "Context:=", "",  
    [  
        "NAME:Ranges",  
        "Range:=",  
        [  
            "Var:=", "Freq",  
            "Type:=", "s",  
            "Start:=", "8GHz",  
            "Stop:=", "8GHz"  
        ]  
    ]  
]
```

```

    ]
  ]
])

```

Example: Parametric Setup with Harmonic Distortions

```

oModule.InsertSetup("OptiParametric",
[
  "NAME:ParametricSetup1",
  "IsEnabled:=" , True,
  [
    "NAME:ProdOptiSetupDataV2",
    "SaveFields:=" , False,
    "CopyMesh:=" , False,
    "SolveWithCopiedMeshOnly:=", False
  ],
  "InterpolationPoints:=" , 0,
  "HarmonicDistortionNum:=", 10,
  [
    "NAME:StartingPoint"
  ],
  "Sim. Setups:=" , ["Setup1"],
  [

```

	<pre> "NAME:Sweeps"], ["NAME:Sweep Operations"], ["NAME:Goals"]]])</pre>

Optimization Script Commands

[EditSetup \[Optimization\]](#)

[InsertSetup \[Optimization\]](#)

EditSetup [Optimization]

Modifies an existing optimization setup.

UI Access	Right-click the setup in the Project Manager tree, and then click Properties from the shortcut menu		
Parameters	Name	Type	Description
	<SetupName>	String	Name of the Setup
	<OptimizationParams>	List	Parameters that define the setup; examples are listed below.

Return Value	None

Python Syntax	EditSetup (<SetupName>, <OptimizationParams>)
Python Example	<pre>oModule.EditSetup("OptimizationSetup1", ["NAME:OptimizationSetup1", "UseFastCalculationUpdateAlgo:=", False, "FastCalcOptCtrledByUser:=", False, "IsEnabled:=", True, "SaveSolutions:=", False, ["NAME:StartingPoint"], "Optimizer:=", "Quasi Newton", ["NAME:AnalysisStopOptions", "StopForNumIteration:=" , True, "StopForElapsTime:=", False, "StopForSlowImprovement:=", False, "StopForGrdTolerance:=", False,</pre>

```
"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  
])
```

Example with Harmonic Distortions

```
oModule.EditSetup("ParametricSetup1",  
[  
  "NAME:ParametricSetup1",  
  "IsEnabled:="          , True,  
  [  
    "NAME:ProdOptiSetupDataV2",  
    "SaveFields:="        , False,  
    "CopyMesh:="          , False,  
    "SolveWithCopiedMeshOnly:=", False  
  ],  
  "InterpolationPoints:=" , 0,  
  "HarmonicDistortionNum:=", 12,
```

	<pre>["NAME:StartingPoint"], "Sim. Setups:=" , ["Setup1"], ["NAME:Sweeps"], ["NAME:Sweep Operations"], ["NAME:Goals"]])</pre>

InsertSetup [Optimization]

Use: Inserts a new optimization setup.

UI Access	Right-click the Optimetrics folder in the project tree, and then click Add > Optimization on the shortcut menu.		
Parameters	Name	Type	Description
	<OptimizationParams>	Array	Array("NAME:<SetupName>", "SaveFields:=", <SaveField>, <StartingPoint>, "Optimizer:=",

		<pre> <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>
<OptimizationVars>	Array	<pre> 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>)) </pre>
<MinStepV>	VarValue	The minimum step of the variable.
<MaxStepV>	VarValue	The maximum step of the variable.
<AcceptableCost>	Double	The acceptable cost value for the optimizer to stop.
<Noise>	Double	The noise of the design.
<UpdateDesign>	Boolean	Specifies whether or not to apply the optimal variation to the design after the optimization is done.
<OptimizationGoalSpec>	Array	"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 expres- sion.
Return Value	None		

<MinStepV>

Type : <VarValue>

The minimum step of the variable.

<MaxStepV>

Type: <VarValue>

The maximum step of the variable.

<AcceptableCost>

Type: <double>

The acceptable cost value for the optimizer to stop.

<Noise>

Type: <double>

The noise of the design.

<UpdateDesign>

Type: <bool>

Specifies whether or not to apply the optimal variation to the design after the optimization is done.

<OptimizationGoalSpec>

"Condition:=", <OptimizationCond>,

Array("NAME:GoalValue", "GoalValueType:=",

<GoalValueType>,

"Format:=", <GoalValueFormat>, "bG:=",

Array("v:=", <GoalValue>)), "Weight:=", <Weight>)

<OptimizationCond>

Type: <string>

Either "<=", "==" or ">="

<GoalValueType>

Type: <string>

Either "Independent" or "Dependent"

<GoalValueFormat>

Type: <string>

Either "Real/Imag" or "Mag/Ang".

<GoalValue>

Type: <string>

Value in string. Value can be a real number, complex number, or expression.

Python Syntax	InsertSetup ("OptiOptimization", <OptimizationParams>)
Python Example	<pre>oModule.InsertSetup("OptiOptimization", ["NAME:OptimizationSetup1", "SaveFields:=", false, _ ["NAME:StartingPoint", "\$length:=", "8mm", "\$width:=", "14.5mm"], "Optimizer:=", "Quasi Newton", "MaxIterations:=", 100, "PriorPSetup:=", "ParametricSetup1", "PreSolvePSetup:=", true, ["NAME:Variables", "\$length:=", ["i:=", true, "Min:=", "6mm", "Max:=", "18mm", "MinStep:=", "0.001mm", "MaxStep:=", "1.2mm"], "\$width:=", ["i:=", true, "Min:=", "6.5mm", "Max:=", "19.5mm", "MinStep:=", "0.001mm", "MaxStep:=", "1.3mm"]],</pre>

```
[ "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
])
```

InsertSetup for System Coupling

Accesses the [System Coupling Setup](#) window for setting up exchange of data\cosimulation with other Ansys solvers.

UI Access	Right click on Optimetrics in the Project Manager tree and select Add > System Coupling Setup		
Parameters	Name	Type	Description
	"OptiSystemCoupling"	string	Designates setup as System Coupling
	<NAME>	string	Specifies the name of the system coupling setup
	<IsEnabled>	bool	Internal command required for system coupling
	<SolveSetupParameters>	Array	Specifies the parameters of the magnetic simulation that will be used in the system coupling; see the Python example that follows.
	<SystemCouplingQuantityData>	Array	Sets the quantities that will be exchanged with Maxwell during the system coupling; see the Python example that follows.
Return Value	None		

Note: Some of the commands listed in the examples that follow are generated when a script is recorded in the Maxwell software interface. These internal commands are required. ANSYS, Inc. recommends [recording a script](#) from the interface, and then editing that script to expand functionality.

Python Syntax	InsertSetup ("OptiSystemCoupling", <NAME>, <IsEnabled>, <SolveSetupParameters>, <SystemCouplingQuantityData>)

Python Example

```
oModule.InsertSetup("OptiSystemCoupling",
[
    "NAME:SystemCouplingSetup2",
    "IsEnabled:="                , True,
    [
        "NAME:ProdOptiSetupDataV2",
        "SaveFields:="          , True,
        "CopyMesh:="            , False,
        "SolveWithCopiedMeshOnly:=", True
    ],
    [
        "NAME:StartingPoint"
    ],
    "Sim. Setups:="              , ["Setup1"],

    [
        "NAME:Sweeps",
        [
            "NAME:SweepDefinition",
            "Variable:="          , "SimTime",
            "Data:="              , "5e-05s",
            "OffsetF1:="          , False,
            "Synchronize:="       , 0
        ]
    ],
    [
        "NAME:Sweep Operations"
    ],
    "CostFunctionName:="        , "Cost",
    "CostFuncNormType:="       , "L2",
```

```

[
  "NAME:CostFunctionGoals"
],
[
  "NAME:ProdSystemCouplingContextData",
  "SimSetup:=" , "Setup1"
],
[
  "NAME:ProdSystemCouplingQuantityData",
  "IsIncludeTimeChecked:=", True,
  "TimeVariable:=" , "SimTime",
  "ParameterVariablePairs:=", ["Current:=", "SimTime"],
  "IncludedParameters:=" , [
    "Current:=" , True,
    "Electrical Conductivity:=", False,
    "Lorentz Force:=" , False,
    "Surface Loss:=" , False,
    "System Coupling Time:=", False,
    "Temperature:=" , False,
    "Voltage:=" , True,
    "Volume Loss:=" , False]
],
[
  "NAME:Goals"
]
]
)

```

InitializeSystemCoupling

Used for the System Coupling workflow. The function in the interface is accessed by right clicking a SystemCouplingSetup*n* nested under Optimetrics in the Project Manager Tree, and selecting **Generate Configuration Files**.

When you select **Generate Configuration Files**, a System Coupling configuration (.sp) file and .a py file are generated. The .py file contains this line *oDesign.InitializeSystemCoupling("Optimetrics:SystemCouplingSetup*n*")* and must be preserved.

Python Syntax	InitializeSystemCoupling()
Python Example	<code>oDesign.InitializeSystemCoupling()</code>

Sensitivity Script Commands

[EditSetup \[Sensitivity\]](#)

[InsertSetup \[Sensitivity\]](#)

EditSetup [Sensitivity]

Modifies an existing sensitivity setup.

UI Access	Right-click the setup in the project tree, and then click Properties on the shortcut menu		
Parameters	Name	Type	Description
	<SetupName>	String	Name of the Setup
Return Value	None		

Python Syntax	EditSetup (<SetupName>, <SensitivityParams>)
Python Example	<pre> oModule.EditSetup("OptimizationSetup1", ["NAME:OptimizationSetup1", "UseFastCalculationUpdateAlgo:=", False, "FastCalcOptCtrledByUser:=", 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,
```

	<pre>"UpdateDesign:=", False, "UpdateIteration:=", 5, "KeepReportAxis:=", True, "UpdateDesignWhenDone:=", True])</pre>

InsertSetup [Sensitivity]

Inserts a new sensitivity setup.

UI Access	Right-click Optimetrics in the project tree, and then click Add>Sensitivity on the shortcut menu.		
Parameters	Name	Type	Description
	<SensitivityParams>	Array	Array("NAME:<SetupName>", "SaveFields:=", <SaveField>, <StartingPoint>, "MaxIterations:=", <MaxIter>, "PriorPSetup:=", <PriorSetup>, "PreSolvePSetup:=", <Preceed>, <SensitivityVars>, <Constraint>, Array("NAME:Goals", Array("NAME:Goal", <OptiGoalSpec>), ..., Array("NAME:Goal", <OptiGoalSpec>)), "Primary Goal:=". <PrimaryGoalID>, "PrimaryError:=", <PrimaryError>)
	<SensitivityVars>	Array	Array("NAME:Variables",

			"VarName:=", Array("i:=", <IncludeVar>, "Min:=", <MinV>, "Max:=", <MaxV>, "IDisp:=", <InitialDisp>),... "VarName:=", Array("i:=", <IncludeVar>, "Min:=", <MinV>, "Max:=", <MaxV>, "IDisp:=", <InitialDisp>))
	<InitialDisp>	VarValue	Index of the Primary goal. Index starts from zero.
	<PrimaryError>	Double	Error associated with the Primary goal.
Return Value	None		

Python Syntax	InsertSetup ("OptiSensitivity", <SensitivityParams>)
Python Example	<pre>oModule.InsertSetup("OptiSensitivity", _ ["NAME:SensitivitySetup1", _ "SaveFields:=", true, _ ["NAME:StartingPoint"], _ "MaxIterations:=", 20, _ "PriorPSetup:=", "", _ "PreSolvePSetup:=", true, _ ["NAME:Variables"], _ ["NAME:LCS"], _</pre>

```
"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])
```

Statistical Script Commands

[EditSetup \[Statistical\]](#)

[InsertSetup Statistical](#)

EditSetup [Statistical]

Modifies an existing statistical setup.

UI Access	Right-click the setup in the project tree, and click Properties on the shortcut menu.		
Parameters	Name	Type	Description
	<SetupName>	String	Name of the Setup
Return Value	None		

Python Syntax	EditSetup (<SetupName>, <StatisticalParams>)
Python Example	See EditSetup [Optimization]

InsertSetup [Statistical]

Inserts a new statistical setup.

UI Access	Right-click Optimetrics in the project tree, and then click Add>Statistical on the shortcut menu.		
Parameters	Name	Type	Description
	<StatisticalParams>	Array	Array("NAME:<SetupName>", "SaveFields:=", <SaveField>, <StartingPoint>, "MaxIterations:=",

			<code><MaxIter>, "PriorPSetup=", <PriorSetup>, "PreSolvePSetup=", <Preceed>, <StatisticalVars>, Array("NAME:Goals", Array("NAME:Goal", <OptiGoalSpec>), ..., Array("NAME:Goal", <OptiGoalSpec>)))</code>
	<code><StatisticalVars></code>	Array	<code>Array("NAME:Variables", "VarName=", Array("i=", <IncludeVar>, "Dist=", <DistType>, "Tol=", <Tolerance>, "StdD=", <StdD>, "Min=", <MinCutoff>, "Max=", <MaxCutoff>, ... "VarName=", Array("i=", <IncludeVar>, "Dist=", <DistType>, "Tol=", <Tolerance>, "StdD=", <StdD>, "Min=", <MinCutoff>, "Max=", <MaxCutoff>))</code>
	<code><DistType></code>	String	Distribution can be "Gaussian" or "Uniform".
	<code><Tolerance></code>	VarValue	The tolerance for the variable when distribution is Uniform
	<code><StdD></code>	VarValue	The standard deviation for the variable when distribution is Gaussian.
	<code><MinCutoff></code>	Double	The minimum cut-off for the variable when distribution is Gaussian.
	<code><MaxCutoff></code>	Double	The maximum cut-off for the variable when distribution is Gaussian.
Return Value	None		

Python Syntax	InsertSetup ("OptiStatistical", <StatisticalParams>)
Python Example	<pre>oModule.InsertSetup("OptiStatistical", _ ["NAME:StatisticalSetup1", _ "SaveFields:=", true, _ ["NAME:StartingPoint"], _ "MaxIterations:=", 50, _ "PriorPSetup:=", "", _ ["NAME:Variables"], _ ["NAME:Goals", _ ["NAME:Goal", _ "Solution:=", "Setup1 : LastAdaptive", _ "Calculation:=", "returnloss", _ "Context:=", "", _ ["NAME:Ranges", _ "Range:=", ["Var:=", "Freq", _ "Type:=", "s", _ "Start:=", "8GHz", "Stop:=", "8GHz"]], _ ["NAME:Goal", _ "Solution:=", "Setup1 : LastAdaptive", _</pre>

	<pre>"Calculation:=", "reflect",_ "Context:=", "", _ ["NAME:Ranges",_ "Range:=", ["Var:=", "Freq", "Type:=",_ "s", "Start:=", "8GHz", "Stop:=", "8GHz"]]]])</pre>

For Q3D Extractor and Circuit the command details are as follows:

Inserts a new statistical setup.

Command: Right-click **Optimetrics** in the project tree, and then click **Add>Statistical** on the shortcut menu.

Syntax: InsertSetup "OptiStatistical", <StatisticalParams>

Return Value: None

Parameters: <StatisticalParams>

```
Array("NAME:<SetupName>", "SaveFields:=",  
<SaveField>, <StartingPoint>, "MaxIterations:=",  
<MaxIter>, "PriorPSetup:=", <PriorSetup>,  
"PreSolvePSetup:=", <Preceed>, <StatisticalVars>,  
Array("NAME:Goals", Array("NAME:Goal",  
<OptiGoalSpec>), ..., Array("NAME:Goal",  
<OptiGoalSpec>))),  
<StatisticalVars>
```



```
Array("NAME:Variables",  
      "VarName:=", Array("i:=", <IncludeVar>, "Dist:=",  
                          <DistType>, "Tol:=", <Tolerance>,  
                          "StdD:=", <StdD>, "Min:=", <MinCutoff>, "Max:=",  
                          <MaxCutoff>, ...  
      "VarName:=", Array("i:=", <IncludeVar>, "Dist:=",  
                          <DistType>, "Tol:=", <Tolerance>, "StdD:=",  
                          <StdD>, "Min:=", <MinCutoff>, "Max:=",  
                          <MaxCutoff>))
```

Parameters:

<DistType>

Type : <string>

Distribution can be "Gaussian" or "Uniform".

<Tolerance>

Type: <VarValue>

The tolerance for the variable when distribution is Uniform.

<StdD>

Type: <VarValue>

The standard deviation for the variable when distribution is Gaussian.

<MinCutoff>

Type: <double>

The minimum cut-off for the variable when distribution is Gaussian.

<MaxCutoff>

Type: <double>

The maximum cut-off for the variable when distribution is Gaussian.

17 - Solutions Module Script Commands

Solutions commands should be executed by the "Solutions" module.

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

```
oModule.CommandName <args>
```

[DeleteSolutionVariation](#)

[ExportNetworkData](#)

[ExportTransientData](#)

[FFTONReport](#)

[GetAdaptiveSettings](#)

[GetAllSourceMagnitudes](#)

[GetAllSourceModes](#)

[GetAllSourcePhases](#)

[GetAllSources](#)

[GetAntennaParameters](#)

[GetFieldType](#)

[GetIncludePortPostProcessing](#)

[GetLayerAndNetOfMeshBody](#)

[GetMeshBodiesOnLayerAndNet](#)

[GetMultipactionBreakdown](#)

[GetNetworkDataSolution](#)

[GetNetworkDataSolutionDefinition](#)[GetNetworkPostprocSetup](#)[GetTerminalExcitationType](#)[GetTransientSolveTimes](#)[GetValidSolutionList](#)[TDROnReport](#)

ExportLayoutLayerGridField

Export the maximum field value on the top/bottom surface of a layer within a user-defined grid.

UI Access	Maxwell > Export Layout Field Data > Grid Max Field Data		
Parameters	Name	Type	Description
	fieldExpression	string	Field type for calculation, supports "Mag_J", "Mag_B", and "Mag_H"
	layoutComponent	string	Specifies the layout component to be used for the export
	layerName	string	Specifies which layer to perform the calculation. Use "All-Layers" to create an export field that includes all the layers.
	isTopFace	boolean	True for calculating on the top face of the layer. False for specifying the bottom face.
	gridSize	string	Specifies the length of each cell of the user-defined grid.
	solutionName		Maxwell design solution name, for example, "Setup1:Transient"
	context	Array	Structured array to specify Time and other geometry variables; for example, ["Time:=": "1ms"]
	exportFile	string	Specifies the path of export file. Supports macros like \$ProjectDir
	threshold	string	Specifies the minimum field value for the data to be exported
	fieldUnit	string	Specifies unit for exporting the field data
	lengthUnit	string	Specifies the unit for exporting the grid coordinates.

Return Value	None

Python Syntax	ExportLayoutLayerGridField (fieldExpression, layoutComponent, layerName, isTopFace, gridSize, solutionName, context, exportFile, threshold, fieldUnit, lengthUnit)
Python Example	<pre>oProject = oDesktop.SetActiveProject("SimpleGrid") oDesign = oProject.SetActiveDesign("Maxwell3DDesign1") oModule = oDesign.GetModule("Solutions") oModule.ExportLayoutLayerGridField("Mag_J", "LC1_1", "PWR", True, "5mm", "Setup1: Transient", ["Time:=" , "20.00000ms"], "\$ProjectDir/Mag_J.json", "0", "A_per_m2", "mm") oModule.ExportLayoutLayerGridField("Mag_B", "LC1_1", "PWR", True, "5mm", "Setup1: Transient", ["Time:=" , "20.00000ms"], "\$ProjectDir/Mag_B.json", "0", "mTesla", "mm") oModule.ExportLayoutLayerGridField("Mag_H", "LC1_1", "PWR", True, "5mm", "Setup1: Transient", [</pre>

	<pre>"Time:=" , "20.00000ms"], "\$ProjectDir/Mag_H.json", "0", "kA_per_m", "mm")</pre>

ExportLayoutViaCurrent

Export the current between signal layers for all the vias in a layout component.

UI Access	Maxwell > Export Layout Field Data > Export Via Current		
Parameters	Name	Type	Description
	layoutComponent	string	Name of layout component to be used for the export
	solutionName	string	Maxwell design solution name, e.g. Setup1:Transient
	context	Array	Structured array to specify Time and other geometry variables; for example, ["Time:=":"1ms"]
	exportFile	string	Specifies the path of export file. Supports macros like \$ProjectDir
	threshold	string	Specifies the minimum field value for the data to be exported
	fieldUnit	string	Specifies unit for exporting the field data
	lengthUnit	string	Specifies the unit for exporting the grid coordinates.
Return Value	None		

Python Syntax	ExportLayoutViaCurrent(layoutComponent, solutionName, context, exportFile, threshold, fieldUnit, lengthUnit)
Python Example	<pre>oProject = oDesktop.SetActiveProject("ViaTest") oDesign = oProject.SetActiveDesign("Maxwell3DDesign1") oModule = oDesign.GetModule("Solutions")</pre>

```
oModule.ExportLayoutViaCurrent("LC1_1", "Setup1:Transient", ["Time:=", "20ms"],
"$ProjectDir/current.json", "0mA", "mA", "mm")
```

ExportLayoutViaCurrentDensity

Export the average current density between signal layers for all the vias in a layout component.

UI Access	Maxwell > Export Layout Field Data > Export Via Average Current Density		
Parameters	Name	Type	Description
	layoutComponent	string	Name of layout component to be used for the export
	solutionName	string	Maxwell design solution name, e.g. Setup1:Transient
	context	Array	Structured array to specify Time and other geometry variables; for example, ["Time:=": "1ms"]
	exportFile	string	Specifies the path of export file. Supports macros like \$ProjectDir
	threshold	string	Specifies the minimum field value for the data to be exported
	fieldUnit	string	Specifies unit for exporting the field data
	lengthUnit	string	Specifies the unit for exporting the grid coordinates.
Return Value	None		

Python Syntax	ExportLayoutViaCurrentDensity(layoutComponent, solutionName, context, exportFile, threshold, fieldUnit, lengthUnit)
Python Example	<pre>oProject = oDesktop.SetActiveProject("SimpleVia") oDesign = oProject.SetActiveDesign("Maxwell3DDesign1") oModule = oDesign.GetModule("Solutions") oModule.ExportLayoutViaCurrentDensity("LC1_1", "Setup1:Transient", ["Time:-</pre>

	<pre>=", "20ms"], "\$ProjectDir/current_density.json", "0mA", "mA", "mm")</pre>

ExportLayoutLayerInducedCurrent

Export the induced eddy current on selected layers of a layout component. For more detailed information on this function, see [Export Layer Induced Eddy Current](#).

Important: The Export Layer-Induced Eddy Current function is only accessible if it has been activated from **Tools > Options > General Options > Desktop Configuration > Beta Options** window. Select the **Maxwell Export ECAD Induced Eddy Current** feature.

Important: This feature is new with Ansys Maxwell 25r1; you cannot export layer-induced eddy current from field data generated with an earlier version of Maxwell. To use this export function, rerun your analysis before exporting.

UI Access	Maxwell > Export Layout Field Data > Export Layer Induced Current		
Parameters	Name	Type	Description
	layoutComponent	string	Name of layout component to be used for the export
	layerName	string	Specifies which layer to perform the calculation. Use "All-Layers" for export field on all the layers
	maxInterval	string	Specifies the maximum length of sample cross sections on the same layout object; include unit
	solutionName	string	Maxwell design solution name, e.g. Setup1:Transient
	context	Array	Structured array to specify Time and other geometry variables; for example, ["Time:=": "1ms"]

	exportFile	string	Specifies the path of export file. Supports macros like \$ProjectDir
	fieldUnit	string	Specifies unit for exporting the field data
	lengthUnit	string	Specifies the unit for exporting the grid coordinates.
Return Value	None		

Python Syntax	ExportLayoutLayerInducedCurrent(layoutComponent, layerName, maxInterval, solutionName, [context Array], exportFile, fieldUnit, lengthUnit)
Python Example	<pre> oProject = oDesktop.SetActiveProject("SimpleGrid") oDesign = oProject.SetActiveDesign("Maxwell3DDesign1") oModule = oDesign.GetModule("Solutions") oModule.ExportLayoutLayerInducedCurrent("LC1_1", "PWR", "1mm", "Setup1 : Tran- sient", ["Time:=" , "20ms"], "\$ProjectDir/current.json", "mA", "mm") </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. See also DeleteVariation.

UI Access	Right-click on Results , select Browse Solutions... , click Delete button in the dialog.		
Parameters	Name	Type	Description
	<SoluParams>	Array	Structured array.

			Array(<DataSpecifierArray>, ...)
	<DataSpecifierArray>	Array	Structured array. Array(<DesignVariationKey>, <SetupName>, <SolutionName>)
Return Value	None.		

Python Syntax	DeleteSolutionVariation(<SoluParams>)
Python Example	<pre>oModule.DeleteSolutionVariation([["width='2in'", "Setup1", "Adaptive_1"], ["width='2in'", "Setup1", "Sweep1"]])</pre>

ExportNetworkData

Exports matrix solution data to a file.

UI Access	N/A		
Parameters	Name	Type	Description
	<DesignVariationKey>	String	Design variation key. Pass empty string for the current nominal variation.
	<SolnSelectionArray>	Array	Array of selected solutions.

		Array(<SolnSelector>, <SolnSelector>, ...) If more than one array entry, this indicates a combined Interpolating sweep.
<SolnSelector>	String	Solution setup name and solution name, separated by a colon.
<FileFormat>	Integer	File format value. 2 : Tab delimited spreadsheet format (.tab) 3 : Touchstone (.sNp) 4 : CitiFile (.cit) 7 : Matlab (.m) 8 : Terminal Z0 spreadsheet
<OutFile>	String	Full path to the file to write out.
<FreqsArray>	Array	The frequencies to export. The <FreqsArray> argument contains a vector (e.g. "1GHz", "2GHz", ...) to use, or "all". To export all frequencies, use Array("all"). If no frequencies are specified, all frequencies are used.
<DoRenorm>	Boolean	For Touchstone format only. Specifies whether to renormalize the data before export.
<RenormImped>	Double	For Touchstone format only. Real impedance value in ohms, for renormalization. Required in syntax, but ignored if DoRenorm is false.
<DataType>	Array	Optional. Type: "S", "Y", or "Z". The matrix to export.
<pass>	Integer	Optional. The pass to export. This is ignored if the sourceName is a frequency sweep. Leaving out this value or specifying -1 gets all passes.
<ComplexFormat>	Integer	Optional. Type: "0", "1", or "2" The format to use for the exported data. 0 = Magnitude/Phase. 1 = Real/Imaginary.

			2= db/Phase.
	<Precision>	Integer	Optional. Touchstone number of digits precision. Default if not specified is 15.
	<UseExportFreqs>	Boolean	Specifies whether to use export frequencies.
	<IncludeGammaComments>	Boolean	Touchstone only. Specifies whether to include Gamma and Impedance comments.
	<SupportNonStdExport>	Boolean	Specifies whether to support non-standard Touchstone extensions for mixed reference impedance.
Return Value	None.		

Python Syntax	ExportNetworkData(<DesignVariationKey>, <SolnSelectionArray>, <SolnSelector>, <FileFormat>, <OutFile>, <FreqsArray>, <DoRenorm>, <RenormImped>, [Optional <DataType>], [Optional <pass>], [Optional <ComplexFormat>], [Optional <Precision>], [Optional <UseExportFreqs>], [Optional <IncludeGammaComments>], [Optional <SupportNonStdExport>])		
Python Example			

ExportTransientData

Exports transient solution data to a file.

UI Access	HFSS > Results > Solution Data... , then click Export .		
Parameters	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_solu1.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	Name	Type	Description
	<ReportName>	String	Name of specified report.
	<WindowName>	String	Name of window to apply for FFT. Possible values are "Rectangular", "Tri", "Van Hann", "Hamming", "Blackman", "Lanczos", "Weber", "Welch".
	<Function>	String	Function to apply on transformation values. Possible values are "none", "ang_deg", "ang_rad", "arg", "cang_deg", "cang_rad", "dB", "dB1 normalize", "dB20normalize", "dBc", "im", "mag", "normalize", "re".
Return Value	None.		

Python Syntax	<code>FFTOnReport <ReportName>, <WindowName>, <Function></code>
Python Example	<code>oModule.FFTOnReport("XY Plot 1", "Rectangular", "dB")</code>

GetAdaptiveSettings

Obtains the adaptive frequency settings of a specified setup.

UI Access	N/A		
Parameters	Name	Type	Description
	<SetupName>	String	Name of specified setup.
Return Value	Array of strings represents adaptive frequency settings.		

Python Syntax	<code>GetAdaptiveSettings(<SetupName>)</code>
Python 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	<code>GetAllSourceMagnitudes()</code>
Python 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	<code>GetAllSourceModes()</code>
Python 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	<code>GetAllSourcePhases()</code>
Python 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	<code>GetAllSources()</code>
Python Example	<code>oModule.GetAllSources()</code>

GetAntennaParameters

Retrieves antenna parameters defined in a specified setup.

UI Access	N/A		
Parameters	Name	Type	Description
	<SetupName>	String	Name of specified setup.
Return Value	Array of strings containing antenna parameters.		

Python Syntax	GetAntennaParameters(<SetupName>)		
Python Example	<code>oModule.GetAntennaParameters("Setup1")</code>		

GetAvailableVariations

Returns the available variation for a solution.

UI Access	N/A		
Parameters	Name	Type	Description
	<SoluName>	String	Name of specified solution.
Return Value	Array of strings containing available variations.		

Python Syntax	GetAvailableVariations(<SoluName>)		
Python Example	<code>oModule.GetAvailableVariations("Setup1 : LastAdaptive")</code>		

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>

GetIncludePortPostProcessing

Determines whether the Include Port Post Processing Effects option is enabled.

UI Access	N/A
Parameters	None.
Return Value	Integer. • 1 - Include Port Post Processing Effects is enabled. • 0 - Include Port Post Processing Effects is disabled.

Python Syntax	<code>GetIncludePortPostProcessing()</code>
Python Example	<code>oModule.GetIncludePortPostProcessing()</code>

GetLayerAndNetOfMeshBody

For a given mesh body name, returns a pair of layer and net name to which this body belongs.

UI Access	N/A
Parameters	The input is a mesh body name.
Return Value	Returns a pair of layer and net name to which this body belongs.

Python Syntax	<code>GetLayerAndNetOfMeshBody()</code>
Python Example	<pre>LayerAndNetOfMeshBody("LC1_1:METAL-1") print layerNetInfo #prints ["METAL-1", "trace_P"] Example usage to plot on the objects on certain layers and nets using GetLayerAndNetOfMeshBody with GetMeshBodiesOnLayerAndNet. meshBodies = oModule.GetMeshBodiesOnLayerAndNet(["NAME:LayerNetInfo", "METAL-1:=" , ["trace_P", "ground"]]) plotGeomInfo = [1, "Volume", "ObjList",]</pre>

```
plotGeomInfo.append(len(meshBodies))

plotGeomInfo += meshBodies

oFieldsReporterModule = oDesign.GetModule("FieldsReporter")
oFieldsReporterModule.CreateFieldPlot(
    [
        "NAME:Mesh1",
        "SolutionName:=" , "Setup1 : LastAdaptive",
        "UserSpecifyName:=" , 0,
        "UserSpecifyFolder:=" , 0,
        "QuantityName:=" , "Mesh",
        "PlotFolder:=" , "MeshPlots",
        "FieldType:=" , "Fields",
        "StreamlinePlot:=" , False,
        "AdjacentSidePlot:=" , False,
        "FullModelPlot:=" , False,
        "IntrinsicVar:=" , "Freq='5GHz\' Phase='0deg\'",
        "PlotGeomInfo:=" , plotGeomInfo,
        "FilterBoxes:=" , [0],
        "Real time mode:=" , True,
    ]
)
```

```
"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:=" , [0,0,255],  
"Filled color:=" , [255,255,255]  
],  
"EnableGaussianSmoothing:=" , False,  
"SurfaceOnly:=" , False  
], "Field")
```

GetMeshBodiesOnLayerAndNet

Gets the field type of a driven modal design solution.

UI Access	N/A
Parameters	The inputs are a layer net block and a vector of component names (optional).
Return Value	The output is a vector of mesh body names that belong to the specified nets and layers and the component..

Python Syntax	GetMeshBodiesOnLayerAndNet()
Python Example	<pre>import ScriptEnv ScriptEnv.Initialize("Ansoft.ElectronicsDesktop") oDesktop.RestoreWindow() oProject = oDesktop.SetActiveProject("Tests") oDesign = oProject.SetActiveDesign("Reference_multiple") oModule = oDesign.GetModule("Solutions") meshBodies = oModule.GetMeshBodiesOnLayerAndNet(["NAME:LayerNetInfo", "METAL-1:=" , ["trace_P","ground"]]) print meshBodies #prints["LC1_1:METAL-1", "LC1_2:METAL-1"] meshBodiesForLC1_1 = oModule.GetMeshBodiesOnLayerAndNet(["NAME:LayerNetInfo", "METAL-1:=" , ["trace_P","ground"]], ["LC1_1"])</pre>

```

print meshBodiesForLC1_1    #prints ["LC1_1:METAL-1"]
moreMeshBodies = oModule.GetMeshBodiesOnLayerAndNet(
    [
        "NAME:LayerNetInfo",
        "METAL-1:="          , ["trace_P","Ground"],
        "METAL-2:="          , ["TRACE_p","GROUND"] #case insensitive
    ])
print moreMeshBodies    #prints ['LC1_1:METAL-1', 'LC1_2:METAL-1', 'LC1_1:METAL-2',
                              'LC1_2:METAL-2']
moreMeshBodiesForLC1_1 = oModule.GetMeshBodiesOnLayerAndNet(
    [
        "NAME:LayerNetInfo",
        "METAL-1:="          , ["trace_P","ground"],
        "METAL-2:="          , ["trace_P","ground"]
    ], ["LC1_1"])
print moreMeshBodiesForLC1_1    #prints['LC1_1:METAL-1', 'LC1_1:METAL-2']

```

GetMultipactionBreakdown

Gets multipaction breakdown from the specified setup.

UI Access	N/A		
Parameters	Name	Type	Description
	<SetupName>	String	Name of specified setup.
	<Variation>	String	Design variation. Pass empty string for the current nominal variation.
Return Value	Array of string containing the multipaction breakdown.		

Python Syntax	GetMultipactionBreakdown(<SetupName>, <Variation>)
Python Example	<code>oModule.GetMultipactionBreakdown("Setup1", "")</code>

GetNetworkDataSolution

Gets matrix data solution from a specified setup.

UI Access	N/A		
Parameters	Name	Type	Description
	<SoluName>	String	Name of solution.
	<VariationKey>	String	Design variation key. Pass empty string for the current nominal variation.
Return Value	String containing matrix data.		

Python Syntax	GetNetworkDataSolution(<SoluName>, <VariationKey>)
Python 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
	<SoluName>	String	Name of specified solution.
Return Value	String containing network data definition.		

Python Syntax	GetNetworkDataSolutionDefinition(<SoluName>)
Python Example	<code>oModule.GetNetworkDataSolutionDefinition("Setup1:Sweep1")</code>

GetNetworkPostprocSetup

Obtains network postprocessing setup.

UI Access	N/A		
Parameters	None.		
Return Value	String containing postprocessing setup.		

Python Syntax	GetNetworkPostprocSetup()

Python Example	<code>oModule.GetNetworkPostprocSetup()</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	<code>GetTerminalExcitationType()</code>
Python 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.
	<code><Variation></code>	String	Design variation. Pass empty string for the current nominal variation.
Return Value	Array of strings containing solve time.		

Python Syntax	<code>GetTransientSolveTimes(<SoluName>, <Variation>)</code>
Python Example	<code>oModule.GetTransientSolveTimes("Setup1:Transient", "")</code>

GetValidSolutionList

Gets all available solution names that exist in a design.

UI Access	N/A		
Parameters	Name	Type	Description
	<code><IncludeImportedSolutions></code>	Boolean	Optional. Specifies whether to include imported solutions. Default is false.
Return Value	Array of strings containing solution names.		

Python Syntax	<code>GetValidSolutionList([Optional <IncludeImportedSolutions>])</code>
Python Example	<code>oModule.GetValidSolutionList()</code>

TDROnReport

Performs a time-domain reflectometry (TDR) analysis on a specified report.

UI Access	Right-click on Results > Perform TDR On Report...

Parameters	Name	Type	Description
	<ReportName>	String	Name of specified report.
	<IsStep>	Boolean	Specifies the input signal as Step or Impulse.
	<RiseTime>	Double	Specifies rise time with unit in 'ps'.
	<WindowName>	String	Name of window to apply. Possible values are "Rectangular", "Bartlett", "Blackman", "Hamming", "Hanning", "Kaiser", "Welch".
	<WindowWidth>	Integer	Width of window in percentage.
	<KaiserParams>	Double	Optional. If Kaiser window is specified, set corresponding parameters.
Return Value	None.		

Python Syntax	TDROnReport(<ReportName>, <IsStep>, <RiseTime>, <WindowName>, <WindowWidth>, [Optional <KaiserParams>])
Python Example	<code>oModule.TDROnReport("Rept2DRectFreq", True, 1E-12, "Hanning", 100, 1)</code>

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>

AddAntennaOverlay

[AddMarkerToPlot](#)

[CreateFieldPlot](#)

[DeleteFieldPlot](#)

[EditSurfaceMeshSummaryData](#)

[ExportMarkerTable](#)

[ExportPlotImageWithViewToFile](#)

[ExportSurfaceMeshSummary](#)

[GetFieldPlotName](#)

HideAntennaParametersOverlay

[ModifyFieldPlot](#)

[ModifyInceptionParameters](#)

[RenameFieldPlot](#)

[RenamePlotFolder](#)

[SetFieldPlotSettings](#)

[SetPlotFolderSettings](#)

[SetPlotsViewSolutionContext](#)

[UpdateAllFieldsPlots](#)

[UpdateQuantityFieldsPlots](#)

AddMarkerToPlot

Adds a marker to a trace on a named field plot.

UI Access	N/A		
Parameters	Name	Type	Description
	<Location>	Array	Array of strings containing X.Y, and Z coordinates for the marker.
	<PlotName>	String	Name of the field plot.
Return Value	None.		

Python Syntax	AddMarkerToPlot(<Location>, <PlotName>)
Python Example	<pre>oModule.AddMarkerToPlot (["0.290455877780914in", "-0.616900205612183in", "1.77635683940025e-015in"], "Mag_H1") oModule.AddMarkerToPlot (["-0.317279517650604in", "1.22481322288513in", "0in"],</pre>

	"Mag_H1")

AddMarker[Fields Reporter]

Adds a marker to the current fields plot.

UI Access	Right-click Field Overlays > Plot Fields > Marker > Add Marker .		
Parameters	Name	Type	Description
	<Position>	Array	Array containing X, Y and Z coordinates.
Return Value	None.		

Python Syntax	AddMarker(<Position>)
Python Example	<pre>oModule.AddMarker(["-267.007756778494mil", "-640.898759461403mil", "9.09494701772928e-13mil"])</pre>

AddNamedExpression

Creates a named expression using the expression at the top of the stack.

UI Access	Click Add in the Fields Calculator window.

Parameters	Name	Type	Description
	<ExprName>	String	Name for the new named expression.
	<FieldType>	String	Type of field.
	<Expression>	Array	Defines an expression to be solved in the calculator. See Support Expression Input in Fields Calculator for more details.
Return Value	None		

Python Syntax	AddNamedExpression(<ExprName>, <FieldType>, < Expression>)
Python Example	<code>oModule.AddNamedExpression("Mag_JxE", "Fields")</code>

Maxwell users can create an expression that uses the [Algebraic option](#). In this case, no Stack commands are necessary; you can define the expression within `AddNamedExpression` command:

```
oModule.AddNamedExpression("new2", "Fields",  
[  
    "Algebraic:=" , "Mean( VolumeValue(Box1), Mag_B )"  
])
```

The same command in Stack form would look like this:

```
oModule.AddNamedExpression("new2", "Fields",  
[  
    "NameOfExpression:=" , ["Mag_B"],
```



```

"EnterVolume:=" , ["Box1"],
"Operation:=" , ["VolumeValue"],
"Operation:=" , ["Mean"]
])

```

For information, see [Syntax for Algebraic Option](#).

CalcOp

Performs a calculator operation.

UI Access	Operation commands like Mag , + , etc.		
Parameters	Name	Type	Description
	<OpString>	String	The text on the corresponding calculator button.
Return Value	None.		

Python Syntax	CalcOp(<OpString>)		
Python Example	oModule.CalcOp ("+")		

CalcStack

Performs an operation on the stack.

UI Access	Stack operation buttons such as Push and Pop .		
Parameters	Name	Type	Description

	<table><tr><td><OpString></td><td>String</td><td>The text on the corresponding calculator button.</td></tr></table>	<OpString>	String	The text on the corresponding calculator button.
<OpString>	String	The text on the corresponding calculator button.		
Return Value	None.			

Python Syntax	CalcStack(<OpString>)
Python Example	<code>oModule.CalcStack("push")</code>

CalculatorRead

Gets a register file and applies it to the calculator stack.

UI Access	Click Read... in the Fields Calculator dialog.		
Parameters	Name	Type	Description
	<FileName>	String	Path to and including name of input register file.
	<SoluName>	String	Specified solution to read in.
	<FieldType>	String	Type of specified field.
	<VarArray>	Array	Array of variable names, value pairs.
Return Value	None.		

Python Syntax	CalculatorRead(<FileName>, <SoluName>, <FieldType>, <VarArray>)
Python Example	<code>oModule.CalculatorRead("c:\example.reg",</code>

	<pre>"Setup1: LastAdaptive", "Fields", ["Freq:=", "10GHz", "Phase:=", "0deg"])</pre>

CalculatorWrite

Writes contents of top register to file.

UI Access	Click Write... in the Fields Calculator window.		
Parameters	Name	Type	Description
	<OutputFilePath>	String	Path to and including name of output register file.
	<SoluNameArray>	Array	Array of strings containing solution names.
	<Expression>	Array	Defines an expression to be solved in the calculator. It can be an algebraic or stacked(default) expression; the <code>Algebraic</code> option requires a beta feature; see Algebraic option . Note that for legacy scripts, this parameter is optional because the expression may already exist.
	<VarArray>	Array	Array of variable names, value pairs.
Return Value	None		

Python Syntax	CalculatorWrite(<OutputFilePath>, <SoluNameArray>, <Expression>, <VarArray>)
Python Example	Legacy script with no expression: <pre>oModule.CalculatorWrite("C:\test.reg", ["Solution:=", "Setup1 : LastAdaptive"], ["Freq:=", "1GHz", "Phase:=", "0deg"])</pre>

```
)
```

Example with algebraic expression:

```
oModule.CalculatorWrite("C:\test.reg",  
  [  
    "NAME:Write",  
    [  
      "NAME:Setup",  
      "Solution:=" , "setup1 : LastAdaptive"  
    ],  
    [  
      "NAME:Algebraic",  
      "Name:=" , [""],  
      "Algebraic:=" , "Mean( Volume(pri), Mag_H )"  
    ]  
  ],  
  [  
    "Freq:=" , "60Hz",  
    "Ihv:=" , "(833.3*1.414*232) A",  
    "Ilv:=" , "-Ihv/1404",  
    "Phase:=" , "0deg",
```

```
    "height:="          , "1mm"
```

```
  ])
```

Example with stack expression:

```
oModule.CalculatorWrite("C:\test.reg",
```

```
  [
```

```
    "NAME:Write",
```

```
    [
```

```
      "NAME:Setup",
```

```
      "Solution:="          , "setup1 : LastAdaptive"
```

```
    ],
```

```
    [
```

```
      "NAME:Expression",
```

```
      "Name:="              , [""],
```

```
      "Expression:="        , ["Mean(Volume(sec_1_1), Mag_H)"],
```

```
      "NameOfExpression:="  , ["Mag_H"],
```

```
      "EnterVolume:="       , ["sec_1_1"],
```

```
      "Operation:="         , ["VolumeValue"],
```

```
      "Operation:="         , ["Mean"]
```

```
    ],
```

```
    [
```

```
      "NAME:Context",
```

	<pre>"Freq:=" , "60Hz", "Ihv:=" , "(833.3*1.414*232) A", "Ilv:=" , "-Ihv/1404", "Phase:=" , "0deg", "height:=" , "1mm"]], "Fields")</pre>

ChangeProperty

Changes the properties of a field plot marker.

UI Access	Select a field plot marker in the marker table.		
Parameters	Name	Type	Description
	<propertyArgs>	Array	Structured array.
Return Value	None.		

Python Syntax	ChangeProperty(<propertyArgs>)

Python Example

```
oDesign.ChangeProperty(
    [
        "NAME:AllTabs",
        [
            "NAME:FieldPlotMarkerDataTab",
            [
                "NAME:PropServers",
                "FieldsReporter:FieldsPlotMarker:m1"
            ],
            [
                "NAME:ChangedProps",
                [
                    "NAME:Name",
                    "Value:=", "Temp"
                ]
            ]
        ]
    ])
oDesign.ChangeProperty(
    [
        "NAME:AllTabs",
        [
            "NAME:FieldPlotMarkerDataTab",
            [
                "NAME:PropServers",
                "FieldsReporter:FieldsPlotMarker:Temp"
            ],
            [
                "NAME:ChangedProps",
```

```
        [
            "NAME:Display Value",
            "Value:=",
            , True
        ]
    ]
]
])
oDesign.ChangeProperty(
[
    "NAME:AllTabs",
    [
        "NAME:FieldPlotMarkerDataTab",
        [
            "NAME:PropServers",
            "FieldsReporter:FieldsPlotMarker:Temp"
        ],
        [
            "NAME:ChangedProps",
            [
                "NAME:Display Name",
                "Value:=",
                , False
            ]
        ]
    ]
]
])
oDesign.ChangeProperty(
[
    "NAME:AllTabs",
    [
```



```

Value:="                , True
                        ]
                    ]
                ])
oDesign.ChangeProperty(
[
    "NAME:AllTabs",
    [
        "NAME:FieldPlotMarkerDataTab",
        [
            "NAME:PropServers",
            "FieldsReporter:FieldsPlotMarker:Temp"
        ],
        [
            "NAME:ChangedProps",
            [
                "NAME:Color",
                "R:="                , 255,
                "G:="                , 255,
                "B:="                , 255
            ]
        ]
    ]
])
oDesign.ChangeProperty(
[
    "NAME:AllTabs",
    [
        "NAME:FieldPlotMarkerDataTab",
        [
            "NAME:PropServers",
            "FieldsReporter:FieldsPlotMarker:Temp"
        ],
    ],

```


ChangeGeomSettings

Changes the line discretization setting.

UI Access	In the Fields Calculator dialog box, click on Geom Settings...		
Parameters	Name	Type	Description
	<LineDiscr>	Integer	Line discretization value.
Return Value	None.		

Python Syntax	ChangeGeomSettings(<LineDiscr>)
Python Example	oModule.ChangeGeomSettings(500)

ClcEval

Evaluates the expression at the top of the stack using the provided solution name and variable values.

UI Access	Click Eval in the Fields Calculator dialog.		
Parameters	Name	Type	Description
	<SoluName>	String	Name of specified solution.
	<VarArray>	Array	Array of variable name, value pairs.

	<Expression>	Array	Defines an expression to be solved in the calculator. It can be an algebraic or stacked (default) expression; the <code>Algebraic</code> option requires a beta feature; see Algebraic option . Note that for legacy scripts, this parameter is optional because the expression may have been created with another command.
	<FieldType>	String	Optional. Type of specified field.
Return Value	None		

Python Syntax	<code>ClcEval(<SoluName>, <Expression>, <VarArray>, [Optional <FieldType>])</code>
Python Example	Legacy script with no expression: <pre>oModule.ClcEval("Setup1: LastAdaptive", ["Freq:=", "10GHz", "Phase:=", "0deg"])</pre> Example with algebraic expression: <pre>oModule.ClcEval("setup1 : LastAdaptive", ["NAME:Evaluate", ["NAME:Algebraic", "Name:=" , [""], "Algebraic:=" , "Mean(Volume(sec_1_1), Mag_H)" ,],], []</pre>

```
"NAME:Context",  
  "Freq:="                , "60Hz",  
  "Ihv:="                  , "(833.3*1.414*232) A",  
  "Ilv:="                  , "-Ihv/1404",  
  "Phase:="                , "0deg",  
  "height:="               , "1mm"  
]  
], "Fields")
```

Example with stack expression:

```
oModule.ClcEval("setup1 : LastAdaptive",  
  [  
    "NAME:Evaluate",  
    [  
      "NAME:Expression",  
      "Name:="                , [""],  
      "Expression:="          , ["Mean(Volume(sec_1_1), Mag_H)"],  
      "NameOfExpression:="    , ["Mag_H"],  
      "EnterVolume:="         , ["sec_1_1"],  
      "Operation:="           , ["VolumeValue"],  
      "Operation:="           , ["Mean"]
```

	<pre>], ["NAME:Context", "Freq:=" , "60Hz", "Ihv:=" , "(833.3*1.414*232) A", "Ilv:=" , "-Ihv/1404", "Phase:=" , "0deg", "height:=" , "1mm"]], "Fields")</pre>

ClcMaterial

Performs a material operation on the top stack element.

UI Access	Click Matl... in the Fields Calculator dialog.		
Parameters	Name	Type	Description
	<MatString>	String	The material property to apply.
	<OpString>	String	Name of operation. Possible values are "mult", or "div".
Return Value	None.		

Python Syntax	ClcMaterial(<MatString>, <OpString>)

Python Example	<code>oModule.ClcMaterial("Permeability (mu)" "mult")</code>

ClcMaterialValue

Shows the value of the material property without performing any operation.

UI Access	Select None in the Material Operation dialog.		
Parameters	Name	Type	Description
	<MaterialName>	String	Name of specified material property.
Return Value	None.		

Python Syntax	<code>ClcMaterialValue(<MaterialName>)</code>
Python Example	<code>oModule.ClcMaterialValue("Mass Density")</code>

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	<code>ClearAllMarkers()</code>
Python 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	<code>ClearAllNamedExpr()</code>
Python Example	<code>oModule.ClearAllNamedExpr()</code>

CopyNamedExprToStack

Copies the named expression selected to the calculator stack.

UI Access	Select a named expression and then click Copy to stack .		
Parameters	Name	Type	Description
	<ExprName>	String	Name of the expression to be copied to the top of the stack.
Return Value	None.		

Python Syntax	CopyNamedExprToStack(<ExprName>)
Python Example	oModule.CopyNamedExprToStack("Mag_JxE")

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: **Maxwell3D** or **Maxwell2D** > **Fields** > **Fields** > <field_quantity>

Command: **Maxwell3D** > **Fields** > **Fields** > **Field Line Trace** (for 3D Electrostatic designs - see [Examples](#) below)

Syntax: CreateFieldPlot <PlotParameterArray> ["Field" | "VRT"]

Return Value: None

Parameters: <PlotParameterArray> "Field"

```
Array("NAME:<PlotName>",  
      "SolutionName:=", <string>,  
      "QuantityName:=", <string>,
```



```
"PlotFolder:=", <string>,
"UserSpecifyName:=", <int>,
"UserSpecifyFolder:=", <int>,
"IntrinsicVar:=", <string>,
"PlotGeomInfo:=", <PlotGeomArray>,
"FilterBoxes:=", <FilterBoxArray>,
<PlotOnPointsSettings>,
<PlotOnLineSettings>,
<PlotOnSurfaceSettings>,
<PlotOnVolumeSettings>)
```

SolutionName:

Name of the solution setup and solution formatted as

```
"<SolveSetupName>: <WhichSolution>",
```

where <WhichSolution> can be "Adaptive_<n>", "LastAdaptive", or "PortOnly".

For example: "Setup1 : Adaptive_2"

A space is required on either side of the colon (:) character. If omitted, the plot will not be created.

QuantityName:

Type of plot to create. Possible values vary according to the solver.

Mesh plots: "Mesh"

For Maxwell AC Magnetic with DC Fields, the "Mesh" quantity has a subparameter: "FieldType:=" , "DC Fields"

Maxwell Field plots include

```
"Mag_E", "Mag_H", "E_Vector", "H_Vector", "Mag_J", "J_Vector",
"ComplexMag_J", "Mag_Jc", "Jc_Vector", "ComplexMag_Jc",
"Mag_B_DC", "Vector_B_DC", "Mag_B_AC_DC", "Vector_B_AC_DC",
"Mag_B_AC_DC_Peak", "Vector_B_AC_DC_Peak", "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"
```

For AC Magnetic T-Omega solution with Include DC Fields:

```
"Mag_B_DC_Src", "Vector_B_DC_Src", "Mag_H_DC_Src", "Vector_H_DC_Src", "Mag_J_DC_Src",  
"Vector_J_DC_Src"
```

PlotFolder:

Name of the folder to which the plot should be added. (Values vary with the design type.) Some possible values include

```
"E Field", "H Field", "Jvol", "Jsurf", "SAR Field",  
"Jc", "Surface-Loss", "QSurf", "Temperature", "Energy",  
"Average-Surface-Loss-Density", "Dielectric_Loss",  
"MeshPlots", "Heat Flux", and "Displacement"
```

UserSpecifyName

0 if default name for plot is used, 1 otherwise.

Not needed. <PlotName> will be respected regardless of whether thisflag is set.

UserSpecifyFolder

0 if default folder for plot is used, 1 otherwise.

Not needed. The specified PlotFolder will be respected regardless of whether this flag is set.

IntrinsicVar (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 two layer-nets combinations. It starts with type "Volume", and subtype "LayerNets", followed by the number of layer-nets combinations which is 2 in this example.

The two layer-nets combinations are (1) "Top" layer, two nets, net names are "net1" and "net2", i.e., [Top, net1] pair, and [Top, net2] pair. (2) "Ground" layer and one net, net name is "GND", i.e., [Ground, GND] pair.

HFSS example with "Plot on surface" option checked:

```
"PlotGeomInfo:=", [1,"Surface","LayerNetsExtFace",2,  
"Top",2,"net1","net2","_Ground",1,"GND"]
```

The command creates field plot on two layer-nets combinations. It starts with type "Surface", and subtype "LayerNetsExtFace", followed by the number of layer-nets combinations which is 2 in this example.

The two layer-nets combinations are (1) "Top" layer, two nets, net names are "net1" and "net2", i.e., [Top, net1] pair, and [Top, net2] pair. (2) "Ground" layer and one net, net name is "GND", i.e., [Ground, GND] pair.

Limitations:

- No Layer/Nets name is supported for field plot command recording should be able to add it based on this change.
- No Layer/Nets name support for pure Layout design field plot command.

PlotGeomInfo (Maxwell)

For example: "PlotGeomInfo :=" , [1,"Volume","ObjList",2,
"LC1_1:Top","LC1_1:Top#L1"]

<PlotGeomArray>

```
Array(<NumGeomTypes>, <GeomTypeData>, <GeomTypeData>, ...)
```

For example: Array(4, "Volume", "ObjList", 1, "Box1",
"Surface", "FacesList", 1, "12", "Line", 1,
"Polyline1", "Point", 2, "Point1", "Point2")

<NumGeomTypes>

Type: <int>

Number of different geometry types (volume, surface, line, point) plotted on at the same time.

<GeomTypeData>

<GeomType>, <ListType>, <NumIDs>, <ID>, <ID>, ...)

<GeomType>

Type: <string>

Possible values are "Volume", "Surface", "Line", "Point".

<ListType>

Type: <string>

Possible values are "ObjList", or "FacesList".

These are used for the GeomType of "Line" or "Point".

<NumIDs>

Type: <int>

Number of IDs or object names that will follow.

<ID>

Type: <int> or <string>

ID of a face or name of an object, line, or point on which to plot.

<FilterBoxArray>

Array of names of objects to use to restrict the plot range.

Array(<NumFilters>, <ObjName>, <ObjName>, ...)

Example: Array(1, "Box1")

Example: Array(0) *no filtering*

<PlotOnPointSettings>

Array("NAME:PlotOnPointSettings",
"PlotMarker:=", <bool>,
"PlotArrow:=", <bool>)

<PlotOnLineSettings>

Array("NAME:PlotOnLineSettings",
Array("NAME:LineSettingsID",
"Width:=", <int>,
"Style:=", <string>),
"IsoValType:=", <string>,
"ArrowUniform:=", <bool>,
"NumofArrow:=", <int>)

Style

Possible values are: "Cylinder", "Solid", "Dashdash", "Dotdot", "Dotdash"

IsoValType

Possible values are "Tone", "Fringe", "Gourard"

<PlotOnSurfaceSettings>

Array("NAME:PlotOnSurfaceSettings",
"Filled:=", <bool>,
"IsoValType:=", <string>,

```
"SmoothShade:=", <bool>,
"AddGrid:=", <bool>,
"MapTransparency:=", <bool>,
"Transparency:=", <double>,
"ArrowUniform:=", <bool>
"ArrowSpacing:=", <double>
"GridColor:=", Array(<int>, <int>, <int>)
```

IsoValType

Possible values are "Tone", "Line", "Fringe", "Gourard"

GridColor

Array containing the R, G, B components of the color. Components should be in the range 0 to 255.

<PlotOnVolumeSettings>

```
Array("NAME:PlotOnVolumeSettings",
"PlotIsoSurface:=", <bool>,
"CloudDensity:=", <double>,
"PointSize:=", <int>,
"ArrowUniform:=", <bool>,
"ArrowSpacing:=", <double>)
```

Python Example – E Magnitude Field Plot:

```
oModule.CreateFieldPlot(
[
    "NAME:Mag_E1",
    "SolutionName:=",          , "Setup1 : LastAdaptive",
    "QuantityName:=",          , "Mag_E",
    "PlotFolder:=",            , "E Field1",
    "UserSpecifyName:=",       , 0,
```

```

"UserSpecifyFolder:=" , 0,
"IntrinsicVar:="      , "Freq='1GHz' Phase='0deg'",
"PlotGeomInfo:=" , [1 , "Surface", "FacesList", 1, "7"],
"FilterBoxes:=" , [0],
[
  "NAME:PlotOnSurfaceSettings",
  "Filled:="          , False,
  "IsoValType:="      , "Fringe",
  "SmoothShade:="     , True,
  "AddGrid:="         , False,
  "MapTransparency:=" , True,
  "Transparency:="    , 0,
  "ArrowUniform:="    , True,
  "ArrowSpacing:="    , 0.100000001490116,
  "GridColor:="       , [255, 255, 255]
]
], "Field")

```

Electron Density Field Plot:

```

oModule = oDesign.GetModule("FieldsReporter")
oModule.CreateFieldPlot(
[
  "NAME:Electron_Density3",
  "SolutionName:="      , "AIR : RFDischarge",
  "UserSpecifyName:="   , 0,
  "UserSpecifyFolder:=" , 0,
  "QuantityName:="      , "Electron_Density",
  "PlotFolder:="        , "RF Discharge Fields",
  "StreamlinePlot:="    , False,
  "AdjacentSidePlot:="  , False,
  "FullModelPlot:="     , False,

```

```
"IntrinsicVar:="          , "Freq=\'0.20000000000000001GHz\'
                           GasPressure=\'0.02kPascal\'",
"PlotGeomInfo:="         , [1,"Surface","FacesList",1,"48"],
"FilterBoxes:="           , [0],
[
  "NAME:PlotOnSurfaceSettings",
  "Filled:="              , False,
  "IsoValType:="          , "Tone",
  "AddGrid:="             , False,
  "MapTransparency:="     , True,
  "Refinement:="          , 0,
  "Transparency:="        , 0,
  "SmoothingLevel:="      , 0,
  "ShadingType:="         , 0,
  [
    "NAME:Arrow3DSpacingSettings",
    "ArrowUniform:="      , True,
    "ArrowSpacing:="      , 0,
    "MinArrowSpacing:="   , 0,
    "MaxArrowSpacing:="   , 0,
    "GridColor:="         , [255,255,255]
  ],
],
"EnableGaussianSmoothing:=" , False,
"SurfaceOnly:="            , False
], "Field")
```

HFSS Multipaction SEY Plot

```
# -----
```

```
# Script Recorded by Ansys Electronics Desktop Version 2025.2.0
```



```
# 15:36:54 Jan 15, 2025
# -----
import ScriptEnv
ScriptEnv.Initialize("Ansoft.ElectronicsDesktop")
oDesktop.RestoreWindow()
oProject = oDesktop.SetActiveProject("JPL_coax_r251")
oDesign = oProject.SetActiveDesign("Coaxial_7-8")
oModule = oDesign.GetModule("FieldsReporter")
oModule.CreateFieldPlot(
[
    "NAME:SEY_Plot1",
    "SolutionName:=" , "300MHz_low : Power",
    "UserSpecifyName:=" , 0,
    "UserSpecifyFolder:=" , 0,
    "QuantityName:=" , "Unknown",
    "PlotFolder:=" , "SEY_Plot",
    "StreamlinePlot:=" , False,
    "AdjacentSidePlot:=" , False,
    "FullModelPlot:=" , False,
    "IntrinsicVar:=" , "PowerMultiplier=\'250\' Time=\'0s\'",
    "PlotGeomInfo:=" , [1,"Surface","FacesList",4,"16","17","18","24"],
```

```
"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,
"QuantityName:=" , "QuantityName_SecondaryEmission",
"PlotFolder:=" , "SEY_Plot",
"IntrinsicVar:=" , "PowerMultiplier=\'250\' Time=\'0s\'"
], "SecondaryEmission")

```

Mesh Plot of Maxwell 3D AC Magnetic T-Omega Solution with Include DC Fields

```

oModule.CreateFieldPlot(
[
  "NAME:Mesh1",
  "SolutionName:=" , "Setup1 : LastAdaptive",
  "UserSpecifyName:=" , 0,
  "UserSpecifyFolder:=" , 0,
  "QuantityName:=" , "Mesh",
  "PlotFolder:=" , "MeshPlots",
  "FieldType:=" , "DC Fields",
  "StreamlinePlot:=" , False,
  "AdjacentSidePlot:=" , False,
  "FullModelPlot:=" , False,
  "IntrinsicVar:=" , "Freq=\'60Hz\' Phase=\'0deg\'",
  "PlotGeomInfo:=" , [1,"Volume","ObjList",1,"Cylinder1"],
  "FilterBoxes:=" , [0],
  "Real time mode:=" , True,
  [
    "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:=" , [0,0,255],
"Filled color:=" , [255,255,255]
],
"EnableGaussianSmoothing:=" , False,
"SurfaceOnly:=" , False
], "Field")
```

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=\ '0.000200000000000000001s\ '",
  "PlotGeomInfo:=" , [1,"Surface","FacesList",1,"2955"],
  "FilterBoxes:=" , [0],
  ...
])
```

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"],
  ...
])
```

Maxwell ACConduction Design Plot:

```
oModule.CreateFieldPlot(
[
  "NAME:Mag_Jc1",
  "SolutionName:="      , "Setup1 : LastAdaptive",
  "UserSpecifyName:="   , 0,
  "UserSpecifyFolder:=" , 0,
  "QuantityName:="      , "Mag_Jc",
  "PlotFolder:="        , "Jc",
  "StreamlinePlot:="    , False,
  "AdjacentSidePlot:="  , False,
  "FullModelPlot:="     , False,
  "IntrinsicVar:="      , "Freq=\'1000000Hz\' Phase=\'0deg\'",
  "PlotGeomInfo:="      , [1,"Volume","ObjList",1,"Box1"],
])
```

```
"FilterBoxes:="          , [1,""],
[
  "NAME:PlotOnVolumeSettings",
  "PlotIsoSurface:="      , True,
  "PointSize:="           , 1,
  "Refinement:="          , 0,
  "CloudSpacing:="        , 0.5,
  "CloudMinSpacing:="     , -1,
  "CloudMaxSpacing:="     , -1,
  "ShadingType:="         , 0,
  [
    "NAME:Arrow3DSpacingSettings",
    "ArrowUniform:="      , True,
    "ArrowSpacing:="      , 0,
    "MinArrowSpacing:="   , 0,
    "MaxArrowSpacing:="   , 0
  ]
],
"EnableGaussianSmoothing:=", False
], "Field")

oModule.CreateFieldPlot(
[
  "NAME:ComplexMag_Jc1",
  "SolutionName:="       , "Setup1 : LastAdaptive",
  "UserSpecifyName:="    , 0,
  "UserSpecifyFolder:="  , 0,
  "QuantityName:="       , "ComplexMag_Jc",
  "PlotFolder:="         , "Jc",
  ...
], "Field")
```

```
oModule.CreateFieldPlot(  
[  
  "NAME:Jc_Vector2",  
  "SolutionName:="      , "Setup1 : LastAdaptive",  
  "UserSpecifyName:="   , 0,  
  "UserSpecifyFolder:=" , 0,  
  "QuantityName:="      , "Jc_Vector",  
  "PlotFolder:="        , "Jc",  
  ...  
], "Field")  
  
oModule.CreateFieldPlot(  
[  
  "NAME:Dielectric_Loss1",  
  "SolutionName:="      , "Setup1 : LastAdaptive",  
  "UserSpecifyName:="   , 0,  
  "UserSpecifyFolder:=" , 0,  
  "QuantityName:="      , "Dielectric_Loss",  
  "PlotFolder:="        , "Dielectric-Loss",  
  ...  
], "Field")
```

Maxwell Field Line Trace Plot Examples

Maxwell Field Line Trace Plot:

```
oModule.CreateFieldPlot(  
[  
  "NAME:FieldLineTrace_Plot3",  
  "SolutionName:="      , "Setup1 : LastAdaptive",  
  "UserSpecifyName:="   , 0,  
  "UserSpecifyFolder:=" , 0,  
  ...  
], "Field")
```

```
"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")
```

DeleteFieldPlot

Deletes one or more field plots.

UI Access	Right-click on one filed plot, select Delete .		
Parameters	Name	Type	Description
	<NameArray>	Array	Array of strings containing the names of the plots to delete.
Return Value	None.		

Python Syntax	DeleteFieldPlot(<NameArray>)
Python Example	oModule.DeleteFieldPlot(["Mag_E1", "Vector_E1"])

DeleteMarker[Fields Reporter]

Deletes one or more marker in the current field plot.

UI Access	Right-click Field Overlays > Plot Fields > Marker > Delete Marker .		
Parameters	Name	Type	Description
	<MarkerNames>	Array	Array of strings containing marker names.
Return Value	None.		

Python Syntax	DeleteMarker(<MarkerNames>)
Python Example	<code>oModule.DeleteMarker(["m1", "m2"])</code>

DeleteNamedExpr

Deletes the selected named expression from the list.

UI Access	Select a named expression and then click Delete .		
Parameters	Name	Type	Description
	<ExprName>	String	Name of specified named expression.
Return Value	None.		

Python Syntax	DeleteNamedExpr(<ExprName>)
Python Example	<code>oModule.DeleteNamedExpr("Mag_JxE")</code>

DeleteUneditablePlot

Deletes one or more uneditable plots.

UI Access	N/A		
Parameters	Name	Type	Description
	<NameArray>	Array	Array of the plot names to delete.

Return Value	None.

Python Syntax	DeleteUneditablePlot(<NameArray>)
Python Example	oModule.DeleteUneditablePlot(["Mag_E1", "Vector_E1"])

DoesNamedExpressionExists

Determines whether specified named expression exists.

UI Access	N/A		
Parameters	Name	Type	Description
	<ExpName>	String	Name of the specified named expression.
Return Value	Boolean: • 1 - named expression exists. • 0 - named expression does not exist.		

Python Syntax	DoesNamedExpressionExists(<ExpName>)
Python Example	oModule.DoesNamedExpressionExists("Mag_JxE")

EditSurfaceMeshSummaryData

Use: Defines or Edits Surface Mesh Summary Data.

Command: Create Surface Mesh Summary, **Setup...**

Syntax: EditSurfaceMeshSummaryData Array(Array(<Parameters Array>))

Return Value: None

Parameters:

"NAME:SurfaceMeshSummary"

Type: <string>

"NAME:SurfaceMeshSummary"

"SolutionName:=" <string>

Type: <string>

Solution Name.

"Variation Name:=" <string>

Type: <string>

Variation name.

<RowItemsArray>

Array<MeshRowItems>

Row data.

<ItemPerRow>

Array<EntityPerRow>

Entity ID.

<MeshDataPer Item>

Array<Mesh Data Category and Stats>

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,</pre>

```
"Min Tri Area:="      , 3.4982570283924E-09,  
"Max Tri Area:="      , 3.73116346661249E-07,  
"Mean Tri Area:="     , 8.91526236529066E-08,  
"Std Devn Tri Area:=" , 6.89482807990471E-08,  
"DataState:="         , 2  
]  
],  
[  
  "NAME:ItemPerRow",  
    "EntityPerRow:="   , "gap",  
  [  
    "NAME:MeshDataPerItem",  
    "Min Edge Len:="   , 0.000105681286042456,  
    "Max Edge Len:="   , 0.000362084671538928,  
    "Mean Edge Len:="  , 0.000233090954707978,  
    "Std Devn Edge Len:=" , 7.55387369793154E-05,  
    "Min Tri Area:="   , 1.74640378345369E-09,  
    "Max Tri Area:="   , 3.78166789228631E-08,  
    "Mean Tri Area:="  , 1.56497981245286E-08,  
    "Std Devn Tri Area:=" , 1.04691637802914E-08,
```

```

        "DataState:="          , 2
    ]
]
]
]
] )

```

EnterComplex

Enters a complex number onto the stack.

UI Access	Click Number , and then click Scalar. Complex option is selected.		
Parameters	Name	Type	Description
	<ComplexNum>	String	String of complex value. Ex. "1 + 2j".
Return Value	None.		

Python Syntax	EnterComplex(<ComplexNum>)
Python Example	oModule.EnterComplex("1 + 2 j")

EnterComplexVector

Enters a complex vector onto the stack.

UI Access	Click Number , and then click Vector. Complex option is selected.

Parameters	Name	Type	Description
	<ComplexVector>	Array	Array of strings containing X, Y and Z complex values.
Return Value	None.		

Python Syntax	EnterComplexVector(<ComplexVector>)
Python Example	<pre>oModule.EnterComplexVector (["1 + 2 j", "1 + 2 j", "1 + 2 j"])</pre>

EnterCoord

Enters a coordinate system defined in the 3D Modeler editor.

UI Access	Click Geometry and then select Coord .		
Parameters	Name	Type	Description
	<CoordName>	String	Name of a coordinate system defined in the 3D Modeler editor.
Return Value	None.		

Python Syntax	EnterCoord(<CoordName>)
Python Example	oModule.EnterCoord("Global")

EnterEdge

Enters an edge defined in the 3D Modeler editor.

UI Access	N/A		
Parameters	Name	Type	Description
	<EdgeName>	String	Name of an edge defined in the 3D Modeler editor.
Return Value	None.		

Python Syntax	EnterEdge(<EdgeName>)
Python Example	oModule.EnterPoint("Edge_1")

EnterLine

Enters a line defined in the 3D Modeler editor.

UI Access	Click Geometry and then select Line		
Parameters	Name	Type	Description
	<LineName>	String	Name of a line defined in the 3D Modeler editor.
Return Value	None.		

Python Syntax	EnterLine(<LineName>)
Python Example	oModule.EnterLine("Line1")

EnterOutputVar

Enters Output Vars, only valid for Eigenmode problems.

UI Access	Click Geometry and then select Output Vars .		
Parameters	Name	Type	Description
	<VarName>	String	Name of the output vars. Only 'freq' is supported.
	<VarType>	String	Type of the output vars. 'Complex' type.
Return Value	None.		

Python Syntax	EnterOutputVar(<VarName>, <VarType>)
Python Example	oModule.EnterOutputVar("Freq", "Complex")

EntryPoint

Enters a point defined in the 3D Modeler editor.

UI Access	Click Geometry and then select Point .		
Parameters	Name	Type	Description
	<PointName>	String	Name of a point defined in the 3D Modeler editor.
Return Value	None.		

Python Syntax	EnterPoint(<PointName>)		
Python Example	oModule.EnterPoint("Point1")		

EnterQty

Enters a field quantity.

UI Access	Click Quantity , and then select from the list.		
Parameters	Name	Type	Description
	<FieldQty>	String	The field quantity to be entered onto the stack.
Return Value	None.		

Python Syntax	EnterQty(<FieldQty>)		
Python Example	oModule.EnterQty("E")		

EnterScalar

Enters a scalar onto the stack.

UI Access	Click Number and then click Scalar . Complex option not selected.		
Parameters	Name	Type	Description
	<Scalar>	Double	The real number to enter onto the stack.
Return Value	None.		

Python Syntax	EnterScalar(<Scalar>)
Python Example	oModule.EnterScalar(3.14159265358979)

EnterScalarFunc

Enters a scalar function.

UI Access	Click Function and then select Scalar .		
Parameters	Name	Type	Description
	<VarName>	String	Name of a variable to enter as a scalar function onto the stack.
Return Value	None.		

Python Syntax	EnterScalarFunc(<VarName>)
Python Example	<code>oModule.EnterScalarFunc("Phase")</code>

EnterSurf

Enters a surface defined in the 3D Modeler editor.

UI Access	Click Geometry and then select Surface .		
Parameters	Name	Type	Description
	<SurfName>	String	Name of a surface defined in the 3D Modeler editor.
Return Value	None.		

Python Syntax	EnterSurf(<SurfName>)
Python Example	<code>oModule.EnterSurf("Rectangle1")</code>

EnterVector

Enters a vector onto the stack.

UI Access	Click Number , and then click Vector . Complex option not selected.		
Parameters	Name	Type	Description
	<VectorArray>	Array	Array of strings containing X, Y and Z components of the vector.
Return Value	None.		

Python Syntax	EnterVector(<VectorArray>)
Python Example	oModule.EnterVector([1.0, 1.0, 1.0])

EnterVectorFunc

Enters a vector function.

UI Access	Click Function and then select Vector .		
Parameters	Name	Type	Description
	<VarNames>	Array	Array of strings containing names of a variable for the X, Y, and Z coordinates, respectively, to enter as a vector function on the stack.
Return Value	None.		

Python Syntax	EnterVectorFunc(<VarNames>)
Python Example	oModuleEnterVectorFunc(["X", "Y", "Z"])

EnterVol

Enters a volume defined in the 3D Modeler editor.

UI Access	Click Geometry and then select Volume .		
Parameters	Name	Type	Description
	<VolumeName>	String	Name of a volume defined in the 3D Modeler editor.
Return Value	None.		

Python Syntax	EnterVol(<VolumeName>)
Python Example	<code>oModule.EnterVol("Box1")</code>

ExportFieldPlot

Exports a field plot to a file.

UI Access	N/A		
Parameters	Name	Type	Description
	<PlotName>	String	Name of the field plot to export.
	<ShowHeader>	Boolean	True - export field plot header to a file. False - export field plot.
	<FileName>	String	Name of file to save as, including the file path.
Return Value	None.		

Python Syntax	ExportFieldPlot(<PlotName>, <ShowHeader>, <FileName>)
Python Example	<code>oModule.ExportFieldPlot("Mag_E1",</code>

	True, "C:/field_report.dsp")

ExportMarkerTable

Exports the marker table to a .csv or .tab file.

UI Access	[product] > Fields > Plot Fields > Marker > Export Marker Table .		
Parameters	Name	Type	Description
	<FileName>	String	Name of export file include path.
Return Value	None.		

Python Syntax	ExportMarkerTable(<FileName>)
Python Example	oModule.ExportMarkerTable("C:/work/FieldMarkerTable.csv")

ExportOnGrid [Field Overlays]

Evaluates the top stack element at a set of points specified by a grid and exports the data to a file.

UI Access	Click Export , and then click On Grid .		
Parameters	Name	Type	Description
	<OutputFile>	String	Name of the output file.
	<Min>	Array	Minimum values for the coordinate components of the grid system
	<Max>	Array	Maximum values for the coordinate components of the grid system

	<Spacing>	Array	Spacing values for the coordinate components of the grid system
	<SolnName>	String	Name of the simulation setup
	<VarVals>	Array	Array of strings containing setup definitions.
	<IncludePoints>	Boolean	Optional. Specifies whether include points in the output file.
	<CSType>	String	Optional. Type of coordinate system. "Cartesian" (default) "Cylindrical" "Spherical"
	<Offsets>	Array	Optional. Origin for the offset coordinate system. For Cartesian, x, y, z, for Cylindrical, R, Phi, Z, for Spherical, Rho, Theta, Phi.
	<ByCount>	Boolean	Optional.
Return Value	None.		

Note: Regarding the **ExportOnGrid** legacy script which only has “IncludePtInOutput” argument (the last Boolean one), AEDT can still read it and assign other new arguments as default values. Those default values are RefCSName = “global”, PtInSI = “True”, FieldInReCS = “False”

Python Syntax	ExportOnGrid(<OutputFile>, <Min>, <Max>, <Spacing>, <SolnName>, <SolnParameters>, [<ExportOption>, "IncludePointsInOutput:=", <boolean>], "RefCSName:=", <CSName>, "PtsInSI:=", <boolean>, "FieldRefCS:=", <boolean>], "<CSType>", [<Offsets>, <ByCount>])
Python Example	<pre>oModule.ExportOnGrid("C:\offset_grid_model_unit_ref.fld", ["-1mm", "16mm", "0mm"], ["1mm", "18mm", "1mm"], ["2mm", "2mm", "1mm"], "4500MHz : LastAdaptive", ["Freq:=", "4.5GHz", "Phase:=" , "0deg"], ["NAME:ExportOption", "IncludePtInOutput:=" , True,</pre>

	<pre>"RefCSName:=" , "offset", "PtInSI:=" , False, "FieldInRefCS:=" , True], "Cartesian", ["0mm", "0mm", "0mm"], False)</pre>

ExportPlotImageToFile

Deprecated. Use [ExportPlotImageWithViewToFile](#).

Creates field plot exports of existing field plots from a given view points, and with the model being auto-sized automatically for each view.

UI Access	N/A		
Parameters	Name	Type	Description
	<FileName>	String	Full path plus file name.
	<FolderName>	String	Plot folder name.
	<ItemName>	String	Name of fields to plot.
	<SetViewTopDownDirectionByRCS>	String	Optional. Name of relative coordinate system to use for the field plot.
Return Value	None.		

Python Syntax	<code>ExportPlotImageToFile(<FileName>, <FolderName>, <ItemName>, <SetViewTopDownDirectionByRCS>)</code>
Python Example	<pre>oModule.ExportPlotImageToFile ("C:\TestEPITF2.jpg", "", "Mag_E2", "RelativeCS1")</pre>

ExportPlotImageWithViewToFile [Reporter]

Exports a field plot image to a specified file.

Note:

This script replaces ExportPlotImageToFile, which is deprecated.

UI Access	N/A		
Parameters	Name	Type	Description
	<FileName>	String	Full path of file to export.
	<PlotQuantityName>	String	Name of quantity to plot.
	<PlotItemName>	String	Name of fields to plot.
	<PixelSizeX>	Integer	Desired image width, in pixels.
	<PixelSizeY>	Integer	Desired image height, in pixels.
	<ViewOrientation>	String	<i>Optional.</i> Name of orientation to use for plot.
Return Value	Image file is exported to the specified path.		

Python Syntax	<code>ExportPlotImageWithViewToFile(<FileName>, <PlotQuantityName>, <PlotItemName>, <PixelSizeX>, <PixelSizeY>, <ViewOrientation>)</code>

Python 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.

Python Syntax	ExportSurfaceMeshSummary (<SolutionName>, <DesignVariationKey>, <ExportFileName>)
Python Example	<pre>oModule.ExportSurfaceMeshSummary(["SolutionName:=" , "Setup1 : LastAdaptive",</pre>

	<pre>"DesignVariationKey:=" , "Nominal", "ExportFileName:=" , "D:\\Program Files\\Docu- ments\\surfaceMeshSummaryReport.csv"])</pre>

ExportToFile

Note:

The ExportToFile script command has replaced the script command [ExportReport](#). ExportReport remains in order to retain backward compatibility for existing scripts, but it is strongly recommended that you now use ExportToFile.

From a data table or plot, generates text format, comma delimited, tab delimited, or .dat type output files.

UI Access	Right-click on report name in the project tree and select Export Data .		
Parameters	Name	Type	Description
	<ReportName>	String	Name of report to be exported.
	<FileName>	String	Full path of the exported image file name; with extension of .txt - Post processor format file .csv - Comma-delimited data file .tab - Tab-separated file .dat - Ansys plot data file
	<UnitSpec>	String	For example, "kV, Mhz, yd"
	<UseTraceNumberFormat>	Boolean	"True", "False"

Return Value	None.

Python Syntax	<code>ExportToFile(<ReportName>, <FileName>)</code>
Python Example	<code>oModule.ExportToFile("rETotal", "C:/Users/Documents/rETotal.csv")</code> <code>oModule.ExportToFile("S Parameter Table 1", "D:/Users/Documents/cfft.csv", False, " kV, MHz, yd ", True)</code>

GetFieldFolderNames

Gets the folder names of the field plots.

UI Access	N/A
Parameters	None.
Return Value	Array of folder names.

Python Syntax	<code>GetFieldFolderNames()</code>
Python 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>

GetFieldPlotQuantityName

Gets the quantity name of a specified field plot.

UI Access	N/A		
Parameters	Name	Type	Description
	<PlotName>	String	Name of specified plot.
Return Value	String plot quantity name.		

Python Syntax	GetFieldPlotQuantityName(<PlotName>)
Python Example	<code>oModule.GetFieldPlotQuantityName("Mag_H1")</code>

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>

ModifyFieldPlot

Modifies a plot definition.

UI Access	[product] > Fields > Modify Plot		
Parameters	Name	Type	Description
	<OriginalName>	String	Name of the plot to be modified.
	<PlotParams>	Array	Array containing modified settings.
Return Value	None.		

Python Syntax	ModifyFieldPlot(<OriginalName>, <PlotParams>)
Python Example	<code>oModule.ModifyFieldPlot("Vector_E1" ,</code>


```
[ "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, _
    "ArrowUniform:=", True,
    "ArrowSpacing:=", 0.100000001490116,
    "GridColor:=", [255, 255, 255]]
  ])
```

GetTopEntryValue

Gets the value of the top entry of the calculator stack.

UI Access	N/A		
Parameters	Name	Type	Description
	<SolnName>	String	Name of specified solution.
	<VarVals>	Array	Array of variable name, value pairs.
Return Value	Array of strings containing top entry values.		

Python Syntax	GetTopEntryValue(<SolnName>, <VarVals>)
Python Example	<pre>oModule.GetTopEntryValue("Setup1:LastAdaptive", ["Freq:=", "1GHz", "Phase:=", "0deg", "x_size:=", "2mm"])</pre>

HidePolarPlot

Hides a currently visible polar plot.

UI Access	N/A

Parameters	Name	Type	Description
	<PlotName>	String	Name of the polar plot.
Return Value	None.		

Python Syntax	HidePolarPlot(<PlotName>)
Python Example	<code>oModule.HidePolarPlot("rE Plot 1")</code>

HideRadiatedPlotOverlay

Hides overlay radiation fields plot.

UI Access	N/A		
Parameters	Name	Type	Description
	<PlotName>	String	Name of specified plot.
Return Value	None.		

Python Syntax	HideRadiatedPlotOverlay(<PlotName>)
Python Example	<code>oModule.HideRadiatedPlotOverlay("Gain Plot 1")</code>

LoadNamedExpressions

Loads a named expression definition from a saved file.

UI Access	In the Fields Calculator, click Load From... in the Library area.		
Parameters	Name	Type	Description
	<FileName>	String	Filename and full path to the file to hold the named expression definition.
	<FieldType>	String	For products with just one filed type, it is set to "Fields".
	<NamedExpr>	Array	rray of strings containing the names of expression definitions to load from the file.
Return Value	None.		

Python Syntax	LoadNamedExpressions(<FileName>, <FieldType>, <NamedExpr>)		
Python Example	<pre>oModule.LoadNamedExpressions("C:\Ansoft\PersonalLib\smth.clc", "Fields", ["smoothedtemp"])</pre>		

ModifyFieldPlot

Modifies a plot definition.

UI Access	[product] > Fields > Modify Plot		
Parameters	Name	Type	Description
	<OriginalName>	String	Name of the plot to be modified.
	<PlotParams>	Array	Array containing modified settings.
Return Value	None.		

Python Syntax	ModifyFieldPlot(<OriginalName>, <PlotParams>)
Python Example	<pre>oModule.ModifyFieldPlot("Vector_E1" , ["NAME:Vector_E2", "SolutionName:=", "Setup1 : LastAdaptive", "QuantityName:=", "Vector_E", "PlotFolder:=", "E Field1", "UserSpecifyName:=", 0, "UserSpecifyFolder:=", 0, "IntrinsicVar:=", "Freq='1GHz' Phase='30deg'", "PlotGeomInfo:=", [1, "Surface", "FacesList", 1, "7"], "FilterBoxes:=", [0], ["NAME:PlotOnSurfaceSettings", "Filled:=", False, "IsoValType:=", "Fringe", "SmoothShade:=", True, "AddGrid:=", False, "MapTransparency:=", True, "Transparency:=", 0, _ "ArrowUniform:=", True,</pre>

	<pre>"ArrowSpacing:=", 0.100000001490116, "GridColor:=", [255, 255, 255]]])</pre>

ModifyInceptionParameters

Use: For field line trace plots, allows you to set inception voltage evaluation parameters.

UI Access	Right-click on a Field Line Trace Plot in the project tree and select Inception Voltage Evaluation		
Parameters	Name	Type	Description
	<NAME>	string	The name of the inception value settings
	<Gas Type>	int	0 for Dry Air, 1 for SF6, 2 for User Defined
	<Gas Pressure>	int	Gas pressure in Bar
	<Use Inception>	bool	True to use the inception parameters U0, K, A
	<Potential U0>	int	U0 parameter
	<Potential K>	int	Streamer constant (empirical value)
	<Potential A>	int	A parameter
Return Value	None		

Python Syntax	ModifyInceptionParameters (<NAME>, <Gas Type>, <Gas Pressure>, <Use Inception>, <Potential U0>, <Potential K>, <Potential A>)
Python Example	<pre>oModule.ModifyInceptionParameters("FieldLineTrace_Plot1", [</pre>

	<pre> "NAME:InceptionEvaluationSettings", "Gas Type:=" , 1, "Gas Pressure:=" , 1, "Use Inception:=" , True, "Potential U0:=" , 0, "Potential K:=" , 1, "Potential A:=" , 1]) </pre>

ReassignFieldPlot

Reassigns a field plot.

UI Access	Right-click on a field plot > Reassign .		
Parameters	Name	Type	Description
	<PlotName>	String	Name of specified plot to reassign.
	<PlotParameterArray>	Array	See CreateFieldPlot
Return Value	None.		

Python Syntax	ReassignFieldPlot(<PlotName>, <PlotParameterArray>)
Python Example	<pre> oModule.ReassignFieldPlot "Mag_H1", Array("NAME:Mag_H1", _ "SolutionName:=", "Setup1 : LastAdaptive", _ "UserSpecifyName:=", 0, _ "UserSpecifyFolder:=", 0, _ </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  
])
```

RenameFieldPlot

Renames a plot.

UI Access	Right-click the plot you want to rename in the project tree, and then click Rename on the shortcut menu.		
Parameters	Name	Type	Description
	<OldName>	String	Original name of the plot.
	<NewName>	String	New name of the plot.
Return Value	None.		

Python Syntax	RenameFieldPlot(<OldName>, <NewName>)
Python Example	<pre>oModule.RenameFieldPlot("Vector_E1", "Vector_E2")</pre>

RenamePlotFolder

Renames a plot folder.

UI Access	Right-click a plot folder in the project tree, and then click Rename on the shortcut menu.		
Parameters	Name	Type	Description
	<OldName>	String	Original name of the folder.
	<NewName>	String	New name of the folder.
Return Value	None.		

Python Syntax	RenamePlotFolder(<OldName>, <NewName>)		
Python Example	<pre>oModule.RenamePlotFolder("E Field", "Surface Plots")</pre>		

SaveFieldsPlots

Saves field plot(s) to a file.

UI Access	HFSS > Fields > Save As...		
Parameters	Name	Type	Description
	<PlotNames>	Array	Array of plot names.
	<FileName>	String	Name of the file to save as, including the file path.

Return Value	None.

Python Syntax	SaveFieldsPlots(<PlotNames>, <FileName>)
Python Example	<pre>oModule.SaveFieldsPlots(["Mag_E1", "Mag_E2"], "C:/field_report.dsp")</pre>

SetFieldPlotSettings

Sets plot attributes.

UI Access	[product] > Fields > Modify Plot Attributes		
Parameters	Name	Type	Description
	<PlotName>	String	Name of the plot to modify.
	<PlotItemAttributes>	Array	Structured array.
			<pre>Array("NAME:FieldsPlotItemSettings", <PlotOnPointsSettings>, <PlotOnLineSettings>, <PlotOnSurfaceSettings>, <PlotOnVolumeSettings>)</pre> <i>See description of CreateFieldPlot command for details.</i>
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), ["NAME:PlotOnSurfaceSettings", "Filled:=", False, "IsoValType:=", "Tone", "SmoothShade:=", True, "AddGrid:=", False, "MapTransparency:=", True, "Transparency:=", 0, "ArrowUniform:=", True, "ArrowSpacing:=", 0.100000001490116, "GridColor:=", [255, 255, 255]]]</pre>

	)

SaveNamedExpressions

Saves a named expression definition to a file.

UI Access	In the Fields Calculator, click Save To... in the Library area.		
Parameters	Name	Type	Description
	<FileName>	String	Filename and full path to the file to hold the named expression definition.
	<NamedExprs>	Array	Array of strings containing the names of expression definitions to load from the file.
	<Overwrite>	Boolean	Specifies whether to overwrite the file.
Return Value	None.		

Python Syntax	SaveNamedExpressions(<FileName>, <NamedExprs>, <Overwrite>)
Python Example	<pre>oModule.SaveNamedExpressions("C:\Ansoft\PersonalLib\smth.clc", ["smoothedtemp"], True)</pre>

SetFieldPlotSettings

Sets plot attributes.

UI Access	[product] > Fields > Modify Plot Attributes

Parameters	Name	Type	Description
	<PlotName>	String	Name of the plot to modify.
	<PlotItemAttributes>	Array	Structured array.
			<pre>Array("NAME:FieldsPlotItemSettings", <PlotOnPointsSettings>, <PlotOnLineSettings>, <PlotOnSurfaceSettings>, <PlotOnVolumeSettings>)</pre> <i>See description of CreateFieldPlot command for details.</i>
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,</pre>

	<pre>"NumofArrow:=", 100), ["NAME:PlotOnSurfaceSettings", "Filled:=", False, "IsoValType:=", "Tone", "SmoothShade:=", True, "AddGrid:=", False, "MapTransparency:=", True, "Transparency:=", 0, "ArrowUniform:=", True, "ArrowSpacing:=", 0.100000001490116, "GridColor:=", [255, 255, 255]]])</pre>

SetPlotFolderSettings

Sets the attributes of all plots in the specified folder.

UI Access	[product] > Fields > Modify Plot Attributes		
Parameters	Name	Type	Description
	<PlotFolderName>	String	Name of the folder with the attributes to modify.
	<PlotFolderAttributes>	Array	Structured array. Array("NAME:FieldsPlotSettings", "Real time mode:=", <bool>,

			<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>

		"IntrinsicMin:=", <double>, "IntrinsicMax:=", <double>)
<Marker3DSettings>	Array	Structured array. Array("NAME:Marker3DSettings", "MarkerType:=", <integer 9: Sphere 10: Box 11: Tetrahedron 12: Octahedron default: Sphere>, "MarkerMapSize:=", <bool>, "MarkerMapColor:=", <bool>, "MarkerSize:=", <double>)
<Arrow3DSettings>	Array	Structured array. Array("NAME:Arrow3DSettings", "ArrowType:=", <integer 0: Line 1: Cylinder 2: Umbrella default: Line>, "ArrowMapSize:=", <bool>, "ArrowMapColor:=", <bool>,

			"ShowArrowTail:=", <bool>, "ArrowSize:=", <double>)
Return Value	None.		

Python Syntax	SetPlotFolderSettings(<PlotFolderName>, <PlotFolderAttributes>)
Python Example	<pre>oModule.SetPlotFolderSettings("E Field1", ["NAME:FieldsPlotSettings", "Real time mode:=", True, ["NAME:ColorMapSettings", "ColorMapType:=", "Spectrum", "SpectrumType:=", "Rainbow", "UniformColor:=", [127, 255, 255], "RampColor:=", [255, 127, 127]], ["NAME:Scale3DSettings", "m_nLevels:=", 27, "m_autoScale:=", True, "minvalue:=", 9.34379863739014, "maxvalue:=", 13683.755859375, "log:=", False,</pre>

	<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]])</pre>

SetPlotsViewSolutionContext

Sets the context of named field overlay plots to the specified time using the intrinsic Time value of the active window. Also updates the associated view solution context for each plot.

UI Access	[product] > Fields > Set Context to Active Window		
Parameters	Name	Type	Description
	<Plots>	Array	Names of the plots to modify.

	<SolName>	String	Name of a transient analysis solution.
	<SolTime>	String	Intrinsic variable name (must be "Time") and value.
Return Value	None.		

Python Syntax	SetPlotsViewSolutionContext(<Plots>, <SolName>, <SolTime>)		
Python Example	<pre>oModule.SetPlotsViewSolutionContext(["Flux_Lines1", "Mag_H1"], "Setup1 : Transient", "Time=-1")</pre>		

UpdateAllFieldsPlots

Updates the All Fields Plots.

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

Python Syntax	UpdateAllFieldsPlots()
Python Example	<pre>oModule.UpdateAllFieldsPlots()</pre>

UpdateQuantityFieldsPlots

Updates the Quantity Fields Plots.

UI Access	N/A		
Parameters	Name	Type	Description
	<FolderName>	String	Folder containing the plotted quantities.
Return Value	None.		

Python Syntax	UpdateQuantityFieldsPlots(<FolderName>)		
Python Example	<code>oModule.UpdateQuantityFieldsPlots("fldplt")</code>		

This page intentionally
left blank.

19 - Fields Calculator Script Commands

Fields Calculator commands should be executed by the Field Overlays module, which is called "FieldsReporter" in Maxwell scripts.

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

```
oModule.CommandName <args>
```

The command associated with each of the following scripting commands will be a button pressed in the Fields Calculator.

[AddNamedExpression](#)

[CalcOp](#)

[CalculatorRead](#)

[CalcStack](#)

[CalculatorWrite](#)

[ChangeGeomSettings](#)

[ClcEval](#)

[ClcMaterial](#)

[ClearAllNamedExpr](#)

[CopyNamedExprToStack](#)

[DeleteNamedExpr](#)

[EnterComplex](#)

[EnterComplexVector](#)

[EnterLine](#)

[EnterPoint](#)

[EnterQty](#)

[EnterScalar](#)

[EnterScalarFunc](#)

[EnterSurf](#)

[EnterVector](#)

[EnterVectorFunc](#)

[EnterVol](#)

[ExportOnGrid \[Fields Calculator\]](#)

[ExportToFile \[Fields Calculator\]](#)

[GetTopEntryValue](#)

[LoadNamedExpressions](#)

[SaveNamedExpressions](#)

AddNamedExpression

Creates a named expression using the expression at the top of the stack.

UI Access	Click Add in the Fields Calculator window.		
Parameters	Name	Type	Description
	<ExprName>	String	Name for the new named expression.
	<FieldType>	String	Type of field.
	<Expression>	Array	Defines an expression to be solved in the calculator. See Support Expression Input in Fields Calculator for more details.

Return Value	None

Python Syntax	AddNamedExpression(<ExprName>, <FieldType>, < Expression>)
Python Example	oModule.AddNamedExpression("Mag_JxE", "Fields")

Maxwell users can create an expression that uses the [Algebraic option](#). In this case, no Stack commands are necessary; you can define the expression within AddNamedExpression command:

```
oModule.AddNamedExpression("new2", "Fields",  
    [  
        "Algebraic:=" , "Mean( VolumeValue(Box1), Mag_B )"  
    ])
```

The same command in Stack form would look like this:

```
oModule.AddNamedExpression("new2", "Fields",  
    [  
        "NameOfExpression:=" , ["Mag_B"],  
        "EnterVolume:=" , ["Box1"],  
        "Operation:=" , ["VolumeValue"],  
        "Operation:=" , ["Mean"]  
    ])
```

For information, see [Syntax for Algebraic Option](#).

Syntax for Algebraic Option

When [adding a named expression](#), Maxwell users have the option of defining it using algebraic expressions.

An algebraic expression is either a `namedExpression`, a `constant`, a `name`, an `operator`, two expressions separated by a `binaryOperator`, or an expression enclosed in parentheses.

An `operator` is an `opName` followed by arguments.

An `argument` is either empty parentheses, an expression enclosed by parentheses, or two expressions separated by a comma and enclosed by parentheses.

A `binaryOperator` is one of the following literals: `'+'`, `'-'`, `'*'`, `'/'`, `'%'`, `'^'`

A `name` is either a project, design, or intrinsic variable (scalar or vector), or a geometry.

A `constant` can be one of the following:

- scalar literal; for example, "1.5e-10"
- complex number; for example, "3.5-1.0i" or "<(3.5, -1.0)>"
- 3D vector; for example, "<1.0, 2.0, 3.0>"
- complex 3D vector; for example, "<1.0-2.0i, 2.0-3.0i, 3.0-4.0i>" or "<(1.0, -2.0), (2.0, -3.0), (3.0, -4.0)>"

The syntax for an algebraic expression is summarized below.

```
expression : namedExpression
            | constant
            | name
            | operator
            | expression binaryOperator expression
            | '(' expression ')'
```

```
operator : opName arguments
binaryOperator : + | - | * | / | % | ^
arguments : <empty>
           | '(' expression ')'
           | '(' expression ',' expression ')'
```

Note the following:

- % is used for the vector dot product; for example, `Integrate(AllObjects, Real(E%H))`
- * supports vectors on both sides as the operand and the cross-product. For example, `Integrate(AllObjects, Mag(E*H))` can be used instead of `Integrate(AllObjects, Mag(Cross(E,H))`.

Note: For more information on the Algebraic option's syntax, see [Fields Calculator Algebraic Mode](#).

CalcOp

Performs a calculator operation.

UI Access	Operation commands like Mag , + , etc.		
Parameters	Name	Type	Description
	<OpString>	String	The text on the corresponding calculator button.
Return Value	None.		

Python Syntax	CalcOp(<OpString>)
Python Example	oModule.CalcOp ("+")

CalcRead(deprecated)

Reads a file that is written out by the CalcWrite command, and puts the result into a calculator numeric.

UI Access	Click Read... in the Fields Calculator dialog.		
Parameters	Name	Type	Description
	<FileName>	String	Name of the file to be read.
	<SoluName>	String	Specified solution to read in.
	<VarArray>	Array	Array of variables to read.
Return Value	None.		

Python Syntax	CalcRead(<FileName>, <SoluName>, <VarArray>)
Python Example	<pre>oModule.CalcRead("c:\example.reg", "Setup1: LastAdaptive", ["Freq:=", "10GHz", "Phase:=", "0deg"])</pre>

CalculatorRead

Gets a register file and applies it to the calculator stack.

UI Access	Click Read... in the Fields Calculator dialog.		
Parameters	Name	Type	Description
	<FileName>	String	Path to and including name of input register file.
	<SoluName>	String	Specified solution to read in.
	<FieldType>	String	Type of specified field.
	<VarArray>	Array	Array of variable names, value pairs.
Return Value	None.		

Python Syntax	CalculatorRead(<FileName>, <SoluName>, <FieldType>, <VarArray>)		
Python Example	<pre>oModule.CalculatorRead("c:\example.reg", "Setup1: LastAdaptive", "Fields", ["Freq:", "10GHz", "Phase:", "0deg"])</pre>		

CalcStack

Performs an operation on the stack.

UI Access	Stack operation buttons such as Push and Pop .		
Parameters	Name	Type	Description
	<OpString>	String	The text on the corresponding calculator button.
Return Value	None.		

Python Syntax	<code>CalcStack(<OpString>)</code>
Python Example	<code>oModule.CalcStack("push")</code>

CalculatorWrite

Writes contents of top register to file.

UI Access	Click Write... in the Fields Calculator window.		
Parameters	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><Expression></code>	Array	Defines an expression to be solved in the calculator. It can be an algebraic or stacked(default) expression; the <code>Algebraic</code> option requires a beta feature; see Algebraic option . Note that for legacy scripts, this parameter is optional because the expression may already exist.
	<code><VarArray></code>	Array	Array of variable names, value pairs.
Return Value	None		

Python Syntax	<code>CalculatorWrite(<OutputFilePath>, <SoluNameArray>, <Expression>, <VarArray>)</code>
Python Example	Legacy script with no expression: <code>oModule.CalculatorWrite("C:\test.reg",</code>

```
["Solution:=", "Setup1 : LastAdaptive"],
["Freq:=", "1GHz", "Phase:=", "0deg"]
)
```

Example with algebraic expression:

```
oModule.CalculatorWrite("C:\test.reg",
[
  "NAME:Write",
  [
    "NAME:Setup",
    "Solution:=" , "setup1 : LastAdaptive"
  ],
  [
    "NAME:Algebraic",
    "Name:=" , [""],
    "Algebraic:=" , "Mean( Volume(pri), Mag_H )"
  ]
],
[
  "Freq:=" , "60Hz",
  "Ihv:=" , "(833.3*1.414*232) A",
  "Ilv:=" , "-Ihv/1404",
```

```
"Phase:=" , "0deg",  
"height:=" , "1mm"  
])
```

Example with stack expression:

```
oModule.CalculatorWrite("C:\test.reg",  
[  
  "NAME:Write",  
  [  
    "NAME:Setup",  
    "Solution:=" , "setup1 : LastAdaptive"  
  ],  
  [  
    "NAME:Expression",  
    "Name:=" , [""],  
    "Expression:=" , ["Mean(Volume(sec_1_1), Mag_H) "],  
    "NameOfExpression:=" , ["Mag_H"],  
    "EnterVolume:=" , ["sec_1_1"],  
    "Operation:=" , ["VolumeValue"],  
    "Operation:=" , ["Mean"]  
  ],  
]
```


	<pre>["NAME:Context", "Freq:=" , "60Hz", "Ihv:=" , "(833.3*1.414*232) A", "Ilv:=" , "-Ihv/1404", "Phase:=" , "0deg", "height:=" , "1mm"]</pre> <pre>], "Fields")</pre>

ChangeGeomSettings

Changes the line discretization setting.

UI Access	In the Fields Calculator dialog box, click on Geom Settings...		
Parameters	Name	Type	Description
	<LineDiscr>	Integer	Line discretization value.
Return Value	None.		

Python Syntax	ChangeGeomSettings(<LineDiscr>)
Python Example	oModule.ChangeGeomSettings(500)

ClcEval

Evaluates the expression at the top of the stack using the provided solution name and variable values.

UI Access	Click Eval in the Fields Calculator dialog.		
Parameters	Name	Type	Description
	<SoluName>	String	Name of specified solution.
	<VarArray>	Array	Array of variable name, value pairs.
	<Expression>	Array	Defines an expression to be solved in the calculator. It can be an algebraic or stacked (default) expression; the <code>Algebraic</code> option requires a beta feature; see Algebraic option . Note that for legacy scripts, this parameter is optional because the expression may have been created with another command.
	<FieldType>	String	Optional. Type of specified field.
Return Value	None		

Python Syntax	ClcEval(<SoluName>, <Expression>, <VarArray>, [Optional <FieldType>])
Python Example	Legacy script with no expression: <pre>oModule.ClcEval("Setup1: LastAdaptive", ["Freq=", "10GHz", "Phase=", "0deg"])</pre> Example with algebraic expression: <pre>oModule.ClcEval("setup1 : LastAdaptive", [</pre>

```

"NAME:Evaluate",
[
  "NAME:Algebraic",
  "Name:="                      , [""],
  "Algebraic:="                  , "Mean( Volume(sec_1_1), Mag_H )"
],
[
  "NAME:Context",
  "Freq:="                      , "60Hz",
  "Ihv:="                      , "(833.3*1.414*232) A",
  "Ilv:="                      , "-Ihv/1404",
  "Phase:="                    , "0deg",
  "height:="                   , "1mm"
]
], "Fields")

```

Example with stack expression:

```

oModule.ClcEval("setup1 : LastAdaptive",
[
  "NAME:Evaluate",
  [
    "NAME:Expression",

```

```
"Name:="                , [""],  
"Expression:="          , ["Mean(Volume(sec_1_1), Mag_H)"],  
"NameOfExpression:="    , ["Mag_H"],  
"EnterVolume:="         , ["sec_1_1"],  
"Operation:="           , ["VolumeValue"],  
"Operation:="           , ["Mean"]  
],  
[  
  "NAME:Context",  
  "Freq:="              , "60Hz",  
  "Ihv:="                , "(833.3*1.414*232) A",  
  "Ilv:="                , "-Ihv/1404",  
  "Phase:="             , "0deg",  
  "height:="            , "1mm"  
]  
], "Fields")
```

ClcMaterial

Performs a material operation on the top stack element.

UI Access	Click Matl... in the Fields Calculator dialog.		
Parameters	Name	Type	Description
	<MatString>	String	The material property to apply.
	<OpString>	String	Name of operation. Possible values are "mult", or "div".
Return Value	None.		

Python Syntax	ClcMaterial(<MatString>, <OpString>)		
Python Example	oModule.ClcMaterial("Permeability (mu)" "mult")		

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	oModule.ClearAllNamedExpr()		

CopyNamedExprToStack

Copies the named expression selected to the calculator stack.

UI Access	Select a named expression and then click Copy to stack .		
Parameters	Name	Type	Description
	<ExprName>	String	Name of the expression to be copied to the top of the stack.
Return Value	None.		

Python Syntax	CopyNamedExprToStack(<ExprName>)
Python Example	<code>oModule.CopyNamedExprToStack("Mag_JxE")</code>

DeleteNamedExpr

Deletes the selected named expression from the list.

UI Access	Select a named expression and then click Delete .		
Parameters	Name	Type	Description
	<ExprName>	String	Name of specified named expression.
Return Value	None.		

Python Syntax	DeleteNamedExpr(<ExprName>)
Python Example	<code>oModule.DeleteNamedExpr("Mag_JxE")</code>

EnterComplex

Enters a complex number onto the stack.

UI Access	Click Number , and then click Scalar. Complex option is selected.		
Parameters	Name	Type	Description
	<ComplexNum>	String	String of complex value. Ex. "1 + 2j".
Return Value	None.		

Python Syntax	EnterComplex(<ComplexNum>)
Python Example	<code>oModule.EnterComplex("1 + 2 j")</code>

EnterComplexVector

Enters a complex vector onto the stack.

UI Access	Click Number , and then click Vector. Complex option is selected.		
Parameters	Name	Type	Description
	<ComplexVector>	Array	Array of strings containing X, Y and Z complex values.
Return Value	None.		

Python Syntax	EnterComplexVector(<ComplexVector>)
Python Example	<pre>oModule.EnterComplexVector (["1 + 2 j", "1 + 2 j", "1 + 2 j"])</pre>

EnterLine

Enters a line defined in the 3D Modeler editor.

UI Access	Click Geometry and then select Line		
Parameters	Name	Type	Description
	<LineName>	String	Name of a line defined in the 3D Modeler editor.
Return Value	None.		

Python Syntax	EnterLine(<LineName>)
Python Example	<pre>oModule.EnterLine("Line1")</pre>

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>

EnterQty

Enters a field quantity.

UI Access	Click Quantity , and then select from the list.		
Parameters	Name	Type	Description
	<FieldQty>	String	The field quantity to be entered onto the stack.
Return Value	None.		

Python Syntax	EnterQty(<FieldQty>)
Python Example	<code>oModule.EnterQty("E")</code>

EnterScalar

Enters a scalar onto the stack.

UI Access	Click Number and then click Scalar . Complex option not selected.		
Parameters	Name	Type	Description
	<Scalar>	Double	The real number to enter onto the stack.
Return Value	None.		

Python Syntax	EnterScalar(<Scalar>)
Python Example	<code>oModule.EnterScalar(3.14159265358979)</code>

EnterScalarFunc

Enters a scalar function.

UI Access	Click Function and then select Scalar .		
Parameters	Name	Type	Description
	<VarName>	String	Name of a variable to enter as a scalar function onto the stack.

Return Value	None.

Python Syntax	EnterScalarFunc(<VarName>)
Python Example	<code>oModule.EnterScalarFunc("Phase")</code>

EnterSurf

Enters a surface defined in the 3D Modeler editor.

UI Access	Click Geometry and then select Surface .		
Parameters	Name	Type	Description
	<SurfName>	String	Name of a surface defined in the 3D Modeler editor.
Return Value	None.		

Python Syntax	EnterSurf(<SurfName>)
Python Example	<code>oModule.EnterSurf("Rectangle1")</code>

EnterVector

Enters a vector onto the stack.

UI Access	Click Number , and then click Vector . Complex option not selected.

Parameters	Name	Type	Description
	<VectorArray>	Array	Array of strings containing X, Y and Z components of the vector.
Return Value	None.		

Python Syntax	EnterVector(<VectorArray>)
Python Example	oModule.EnterVector([1.0, 1.0, 1.0])

EnterVectorFunc

Enters a vector function.

UI Access	Click Function and then select Vector .		
Parameters	Name	Type	Description
	<VarNames>	Array	Array of strings containing names of a variable for the X, Y, and Z coordinates, respectively, to enter as a vector function on the stack.
Return Value	None.		

Python Syntax	EnterVectorFunc(<VarNames>)
Python Example	oModuleEnterVectorFunc(["X", "Y", "Z"])

EnterVol

Enters a volume defined in the 3D Modeler editor.

UI Access	Click Geometry and then select Volume .		
Parameters	Name	Type	Description
	<VolumeName>	String	Name of a volume defined in the 3D Modeler editor.
Return Value	None.		

Python Syntax	EnterVol(<VolumeName>)
Python Example	<code>oModule.EnterVol("Box1")</code>

ExportOnGrid [Fields Calculator]

Evaluates the top stack element at a set of points specified by a grid and exports the data to a file.

UI Access	Click Export , and then click On Grid .		
Parameters	Name	Type	Description
	<OutputFile>	String	Name of the output file.
	<Min>	Array	Minimum values for the coordinate components of the grid system
	<Max>	Array	Maximum values for the coordinate components of the grid system
	<Spacing>	Array	Spacing values for the coordinate components of the grid system
	<SolnName>	String	Name of the simulation setup
	<VarVals>	Array	Array of strings containing setup definitions.
	<IncludePoints>	Boolean	Optional. Specifies whether include points in the output file.

	<CSType>	String	Optional. Type of coordinate system. "Cartesian" (default) "Cylindrical" "Spherical"
	<Offsets>	Array	Optional. Origin for the offset coordinate system. For Cartesian, x, y, z, for Cylindrical, R, Phi, Z, for Spherical, Rho, Theta, Phi.
	<ByCount>	Boolean	Optional.
Return Value	None.		

Note: Regarding the **ExportOnGrid** legacy script which only has “IncludePtInOutput” argument (the last Boolean one), AEDT can still read it and assign other new arguments as default values. Those default values are RefCSName = “global”, PtInSI = “True”, FieldInRefCS = “False”

Python Syntax	ExportOnGrid(<OutputFile>, <Min>, <Max>, <Spacing>, <SolnName>, <SolnParameters>, [<ExportOption>, "IncludePointsInOutput:=", <boolean>], "RefCSName:=", <CSName>, "PtsInSI:=", <boolean>, "FieldRefCS:=", <boolean>], "<CSType>", [<Offsets>, <ByCount>])
Python Example	<pre>oModule.ExportOnGrid("C:\offset_grid_model_unit_ref.fld", ["-1mm", "16mm", "0mm"], ["1mm", "18mm", "1mm"], ["2mm", "2mm", "1mm"], "4500MHz : LastAdaptive", ["Freq:=", "4.5GHz", "Phase:=" , "0deg"], ["NAME:ExportOption", "IncludePtInOutput:=" , True, "RefCSName:=" , "offset",</pre>

	<pre>"PtInSI:=" , False, "FieldInRefCS:=" , True], "Cartesian", ["0mm", "0mm", "0mm"], False)</pre>

ExportToFile [Fields Calculator]

Evaluates the top stack element at a set of points specified in an external file and exports the data to a file.

Note: Regarding to legacy **ExportToFile** script which only has “IncludePtInOutput” argument (the last Boolean one), AEDT can still read it and assign other new arguments as default values. Those default values are RefCSName = “global”, PtInSI = “True”, FieldInRefCS = “False”

UI Access	Click Export , and then click To File .		
Parameters	Name	Type	Description
	<ReportName>	String	Name of report to be exported.
	<FileName>	String	Full path of the exported image file name; with extension of .txt - Post processor format file .csv - Comma-delimited data file .tab - Tab-separated file .dat - Ansys plot data file
	<solution>		Solution Name

	<table><tr><td><freq></td><td>frequency</td></tr><tr><td><phase></td><td>phase</td></tr><tr><td><Export Options></td><td>Whether to include points in output, RefCSName, Pts in SI units <boolean>, Field in RefCS, <boolean>.</td></tr></table>	<freq>	frequency	<phase>	phase	<Export Options>	Whether to include points in output, RefCSName, Pts in SI units <boolean>, Field in RefCS, <boolean>.
<freq>	frequency						
<phase>	phase						
<Export Options>	Whether to include points in output, RefCSName, Pts in SI units <boolean>, Field in RefCS, <boolean>.						
Return Value	None.						

Python Syntax	ExportToFile(<ReportName>, <FileName>, <solution>, ["Freq:=", <freq>, "Phase:=", "<phase>"], <ExportOptions>)
Python Example	<pre>oModule.ExportToFile("C:\\offset_file_model_unit_ref.fld", "C:\\offset_SI.pts", "4500MHz : LastAdaptive", ["Freq:=" , "4.5GHz", "Phase:=" , "0deg"], ["NAME:ExportOption", "IncludePtInOutput:=" , True, "RefCSName:=" , "offset", "PtInSI:=" , False, "FieldInRefCS:=" , True])</pre>

	])

GetTopEntryValue

Gets the value of the top entry of the calculator stack.

UI Access	N/A		
Parameters	Name	Type	Description
	<SolnName>	String	Name of specified solution.
	<VarVals>	Array	Array of variable name, value pairs.
Return Value	Array of strings containing top entry values.		

Python Syntax	GetTopEntryValue(<SolnName>, <VarVals>)
Python Example	<pre>oModule.GetTopEntryValue("Setup1:LastAdaptive", ["Freq:=", "1GHz", "Phase:=", "0deg", "x_size:=", "2mm"])</pre>

LoadNamedExpressions

Loads a named expression definition from a saved file.

UI Access	In the Fields Calculator, click Load From... in the Library area.		
Parameters	Name	Type	Description
	<FileName>	String	Filename and full path to the file to hold the named expression definition.
	<FieldType>	String	For products with just one filed type, it is set to "Fields".
	<NamedExpr>	Array	rray of strings containing the names of expression definitions to load from the file.
Return Value	None.		

Python Syntax	LoadNamedExpressions(<FileName>, <FieldType>, <NamedExpr>)		
Python Example	<pre>oModule.LoadNamedExpressions("C:\Ansoft\PersonalLib\smth.clc", "Fields", ["smoothedtemp"])</pre>		

SaveNamedExpressions

Saves a named expression definition to a file.

UI Access	In the Fields Calculator, click Save To... in the Library area.		
Parameters	Name	Type	Description
	<FileName>	String	Filename and full path to the file to hold the named expression definition.
	<NamedExprs>	Array	Array of strings containing the names of expression definitions to load from the file.
	<Overwrite>	Boolean	Specifies whether to overwrite the file.

Return Value	None.

Python Syntax	SaveNamedExpressions(<FileName>, <NamedExprs>, <Overwrite>)
Python Example	<pre>oModule.SaveNamedExpressions("C:\Ansoft\PersonalLib\smth.clc", ["smoothedtemp"], True)</pre>

This page intentionally
left blank.

20 - Motion Setup Script Commands

Motion setup commands should be executed by the **ModelSetup** module.

```
Set oModule = oDesign.GetModule("ModelSetup")
oModule.CommandName <args>
```

Conventions Used in the Motion Setup Chapter

<AssignmentObjects>

Type: Array of strings

An array of object names.

General Motion Setup Script Commands

Following are general script commands recognized by the **ModelSetup** module:

- [DeleteMotionSetup](#)
- [ReassignMoving](#)

DeleteMotionSetup

Use: Deletes the motion setup.

Command: Maxwell3D or Maxwell2D>Model>Motion Setup>Unassign Band

Syntax: DeleteMotionSetup

Return Value: None

Parameters: List of strings. A list of names of motion setups to delete.

Example: oModule.DeleteMotionSetup(["MotionSetup2"])

The [] can contain more than one name as it is a list.

ReassignMoving

Use: Specifies a new geometry assignment for moving objects.

Command: Maxwell3D or Maxwell2D>Model>Motion Setup>Add Selected Objects or
Maxwell3D or Maxwell2D>Model>Motion Setup>Remove Selected Objects

Syntax: ReassignMoving Array("Name:Moving", "Objects:=", <AssignmentObjects>)

Return Value: None

Example: oModule.ReassignMoving ["NAME:Moving",
"Objects:=", ["Box4", "Box1"]]

Commands to Create and Edit the Band

Following are script commands for creating and editing bands that are recognized by the **ModelSetup** module:

Note:

In the following commands, all named data can be included/excluded as desired and may appear in any order.

- [AssignBand](#)
- [EditMotionSetup](#)
- [GetMotionSetupNames](#)

AssignBand

Use: Assigns the selected object as a band.

Command: Maxwell3D or Maxwell2D>Model>Motion Setup>Assign Band

Syntax: AssignBand <BandDataArray>

Return Value: None*Parameters:*

```
<BandDataArray>

  Array("NAME:Band",
    "Move Type:=", <Translate/Rotate>,
    "Coordinate System:=", <CoordinateSystemName>,
    "Axis:=", <X/Y/Z>,
    "Is Positive:=", <bool>,
    "InitPos:=", <value>,
    "NegativePos:=", <value>,
    "PositivePos:=", <value>,
    "TranslatePeriodic:=", <bool>,
    "Consider Mechanical Transient:=", <bool>,
    "velocity:=", <value>,
    "objects:=", <AssignmentObjects>)
```

Example: Assign band as translate, do not consider mechanical transient.

```
oModule.AssignBand _
Array("NAME:Band", "Move Type:=", "Translate",
"Coordinate System:=", "Global", "Axis:=", "Z",
"Is Positive:=", true, "InitPos:=", "0mm", _
"NegativePos:=", "0mm", "PositivePos:=", "1mm",
"TranslatePeriodic:=", true,
```

```
"Consider Mechanical Transient:=", false,  
"Velocity:=", "0m_per_sec",  
"Objects:=", Array("band"))
```

Example: Assign band as translate, consider mechanical transient.

```
oModule.AssignBand _  
Array("NAME:Band", _  
"Move Type:=", "Translate",  
"Coordinate System:=", "Global", "Axis:=", "Z",  
"Is Positive:=", true, "InitPos:=", "0mm",  
"NegativePos:=", "0mm", "PositivePos:=", "1mm",  
"TranslatePeriodic:=", true,  
"Consider Mechanical Transient:=", true,  
"Velocity:=", "0m_per_sec", "Mass:=", "1kg",  
"Damping:=", "1", "Load Force:=", "1nNewton"  
"Objects:=", Array("band"))
```

Example: Assign band as rotate, do not consider mechanical transient.

```
oModule.AssignBand _  
Array("NAME:Band", "Move Type:=", "Rotate",  
"Coordinate System:=", "Global", "Axis:=", "Z",  
"Is Positive:=", true, "InitPos:=", "0deg",
```



```
"HasRotateLimit:=", false, "NonCylindrical:=",  
false, "Consider Mechanical Transient:=", false,  
"Angular Velocity:=", "0deg_per_sec", "Objects:=",  
Array("band"))
```

Example: Assign band as rotate, consider mechanical transient.

```
oModule.AssignBand _  
Array("NAME:Band", "Move Type:=", "Rotate",  
"Coordinate System:=", "Global", "Axis:=", "Z",  
"Is Positive:=", true, "InitPos:=", "0deg",  
"HasRotateLimit:=", false, "NonCylindrical:=",  
false, "Consider Mechanical Transient:=", true,  
"Angular Velocity:=", "0deg_per_sec",  
"Moment of Inertia:=", "1", "Damping:=", "0",  
"Load Torque:=", "0NewtonMeter", "Objects:=",  
Array("band"))
```

EditMotionSetup

Use: Edits the motion setup.

Command: Double-click the moving item in the project tree to edit it.

Syntax: EditMotionSetup <SetupName>, <Array>

Return Value: None

Parameters: <SetupName>

The name of the motion setup to be edited.

```
<BanddataArray>  
  Array("NAME:Data",  
    "Move Type:=", <Translate/Rotate>,  
    "Coordinate System:=", <CoordinateSystemName>,  
    "PostProcessing Coordinate System:=", <CoordinateSystemName>  
    "Axis:=", <X/Y/Z>,  
    "Is Positive:=", <bool>,  
    "InitPos:=", <value>,  
    "NegativePos:=", <value>,  
    "PositivePos:=", <value>,  
    "TranslatePeriodic:=", <bool>,  
    "Consider Mechanical Transient:=", <bool>,  
    "velocity:=", <value>,  
    "Mass:=", <value>,  
    "Damping:=", <value>,  
    "Load Force:=", <value>)
```

GetMotionSetupNames

Use: Gets the motion setup parameters being used for the object. Applicable to 2D and 3D Transient solution types.

Command: None

Syntax: GetMotionSetupNames ()

Return Value: List of motion setup names

Parameters: None

Example: oModule.GetMotionSetupNames ()

Other Commands Recognized By the ModelSetup Module

The following command is also recognized by the **ModelSetup** module:

- [GetSymmetryMultiplier](#)
- [SetSymmetryMultiplier](#)

GetSymmetryMultiplier

Use: Returns the symmetry multiplier of the model. Applicable to 2D and 3D Transient solution types.

Command: None

Syntax: oModule.GetSymmetryMultiplier ()

Return Value: The symmetry multiplier integer value.

Parameters: None

Example: oModule.GetSymmetryMultiplier ()

SetSymmetryMultiplier

Use: Sets the symmetry multiplier. This symmetry multiplier will be automatically applied to all input quantities including: input voltage, inductance, resistance, load torque, mass, damping, external circuit; and all output quantities including: induced voltages, flux linkages in every winding, stranded loss, solid loss, core loss, torque and force.

Command: Maxwell3D or Maxwell2D>Model>Set Symmetry Multiplier

Syntax: SetSymmetryMultiplier <int>

Return Value: None

Parameters: <int>

Example: oModule.SetSymmetryMultiplier 2

21 - Parameter Setup Script Commands

Parameter setup commands should be executed by the **MaxwellParameterSetup** module.

```
oModule = oDesign.GetModule("MaxwellParameterSetup")
```

```
oModule.CommandName <args>
```

Conventions Used in the Parameter Chapter

<ParameterName>

Type: <string>

Name of a parameter.

<ParameterNameArray>

Type: Array of strings

An array of the names in a group of parameters.

<AssignmentObjects>

Type: Array of strings

An array of object names.

General Parameter Setup Script Commands

Following are general script commands recognized by the **MaxwellParameterSetup** module:

- [DeleteParameters](#)
- [DeleteAllParameters](#)

- [RenameParameter](#)
- [ReassignParameter](#)

Commands to Create and Edit Parameters

Following are script commands for creating and editing parameters that are recognized by the **MaxwellParameterSetup** module:

- [AssignForce](#)
- [EditForce](#)
- [AssignTorque](#)
- [EditTorque](#)
- [AssignMatrix](#)
- [EditMatrix](#)
- [DeleteReduceMatrix](#)
- [RenameReduceMatrix](#)
- [AddReduceOp](#)
- [DeleteReduceOp](#)
- [EditReduceOp](#)
- [RenameReduceOp](#)

AddReduceOp

Adds a reduce matrix group.

UI Access	Right-click on the matrix and select either Join in Series... or Join in Parallel...		
Parameters	Name	Type	Description
	<MatrixName>	String	Name of parent matrix
	<ReduceMatrixName>	String	Name of reduce matrix

	<ReduceOpArray>	Array	Array("NAME:<GroupName>", "Type:=", one of "Join in Series" or "Join in Parallel", "Sources:=", "<Source1>, <Source2, ...")
Return Value	None		

Python Syntax	AddReduceOp (<MatrixName>, <ReduceMatrixName>, <ReduceOpArray>)
Python Example	<pre>oModule = oDesign.GetModule("MaxwellParameterSetup") oModule.AddReduceOp ("Matrix1", "ReduceMatrix1", ["NAME:JoinSeries1", "Type:=", "Join in Series", "Sources:=", "Current1,Current3"])</pre>

AssignForce

Creates a force.

UI Access	Maxwell3D or Maxwell2D > Parameters > Assign > Force		
Parameters	Name	Type	Description
	<ForceArray>	Array	Array containing force data: Array("NAME:<ForceName>", "IsVirtual:=", <boolean>,

	<table><tr><td></td><td>"ReferenceCS:=", <string>, "Objects:=", <array of objects>)</td></tr></table>		"ReferenceCS:=", <string>, "Objects:=", <array of objects>)
	"ReferenceCS:=", <string>, "Objects:=", <array of objects>)		
Return Value	None		

Note: For 3D AC Magnetic A-Phi simulations, only the Lorentz force type is available, so there is no type argument for force.

Python Syntax	AssignForce (<ForceArray>)
Python Example	<pre>oModule = oDesign.GetModule("MaxwellParameterSetup") oModule.AssignForce(["NAME:Force1", "Reference CS:=" , "Global", "Is Virtual:=" , True, "Objects:=" , ["Core"]])</pre>

[Beta] AssignLayoutForce

Creates a layout force.

UI Access	Maxwell3D > Parameters > Assign > Force from Layout		
Parameters	Name	Type	Description
	<ForceArray>	Array	Array containing force data: Array("NAME:<ForceName>", "ReferenceCS:=", <string>, <array of nets and layers choices>)
Return Value	None		

Python Syntax	AssignLayoutForce (<ForceArray>)
Python Example	<pre>oModule = oDesign.GetModule("MaxwellParameterSetup") oModule.AssignLayoutForce(["NAME:LayoutForce1", "Reference CS:=" , "Global", ["NAME:NetsAndLayersChoices", ["NAME:LC1_1", ["NAME:NetLayerSetMap", ["NAME:net1", "LayerSet:=" , ["Ground","Top"]</pre>

	<pre>], ["NAME:net2", "LayerSet:=" , ["Top"]]]]]])</pre>

AssignMatrix

Creates a matrix.

UI Access	Maxwell 3D or Maxwell 2D > Parameters > Assign > Matrix		
Parameters	Name	Type	Description
	<MatrixName>	String	Specifies the matrix name; for example, RLMatrix, GCMatrix
	<MatrixEntry>	Array	Specifies the sources and number of turns for the sources in the matrix.
	<Source>	String	Specifies the sources to be included in the matrix calculation; this is equivalent to checking the Signal box for the object in the Ansys Electronic Desktop interface.
	<NumberofTurns>	Integer	Specifies the number of turns for the sources.
	<MatrixGroup>	Array	If more than one source is selected, you can combine the sources into one series winding.

	<GroupName>	String	Specifies the name of the winding
	<NumberOfBranches>	Integer	Specifies the number of parallel branches the winding should be divided into
	<Sources>	Array	Specifies the sources to include in the winding.
	<GroundSources>	String	Available for 2D Electrostatic, 2D and 3D DC Conduction, 2D and 3D AC Conduction, and 3D AC Magnetic A-Phi solvers. Excitations designated as ground will be kept at 0 V and will not be part of the matrix.
Return Value	None		

Python Syntax	AssignMatrix (<MatrixArray>)
Python Example	Example: <pre>oModule.AssignMatrix(["NAME:Matrix1", ["NAME:MatrixEntry", ["NAME:MatrixEntry", "Source:=", "Current1", "NumberOfTurns:=", "1"], ["NAME:MatrixEntry", "Source:=", "Current3", "NumberOfTurns:=", "1"]], ["NAME:MatrixGroup", ["NAME:MatrixGroup", "GroupName:=", "Group1", "NumberOfBranches:=", "1", "Sources:=", "Current1,Current3"]]]) </pre> Example: GC and RL matrices specified for AC Magnetic A-Phi simulation

```
oModule.AssignMatrix(  
  [  
    "NAME:Matrix1",  
    [  
      "NAME:RLMatrix",  
      [  
        "NAME:MatrixEntry",  
        [  
          "NAME:MatrixEntry",  
          "Source:=" , "Current1",  
          "NumberOfTurns:=" , "1"  
        ],  
        [  
          "NAME:MatrixEntry",  
          "Source:=" , "Current2",  
          "NumberOfTurns:=" , "1"  
        ]  
      ],  
      [  
        "NAME:MatrixGroup"  
      ],  
      "GroundSources:=" , "Voltage1,Voltage2"  
    ],  
    [  
      "NAME:GCMatrix",  
      [  
        "NAME:MatrixEntry",  
        [  
          "NAME:MatrixEntry",  
          "Source:=" , "Voltage2",  
          "NumberOfTurns:=" , "1"  
        ]  
      ]  
    ]  
  ]  
)
```

	<pre>], ["NAME:MatrixEntry", "Source:=" , "Current1", "NumberOfTurns:=" , "1"], ["NAME:MatrixEntry", "Source:=" , "Current2", "NumberOfTurns:=" , "1"]], ["NAME:MatrixGroup"], "GroundSources:=" , "Voltage1"]])</pre>

AssignTorque

Creates a torque.

UI Access	Maxwell3D or Maxwell2D > Parameters > Assign > Torque		
Parameters	Name	Type	Description
	<TorqueArray>	Array	Array containing torque data: Array("NAME:<TorqueName>", "Is Virtual:=", <bool>, "Coordinate System:=", <string>

			<code>"Axis:=", <string>, "Is Positive:=", <bool> "Objects:=", <AssignmentObjects>)</code>
Return Value	None		

Note: For 3D AC Magnetic A-Phi simulations, only the Lorentz torque type is available, so there is no type argument for torque.

Python Syntax	AssignTorque (<TorqueArray>)
Python Example	<pre>oModule = oDesign.GetModule("MaxwellParameterSetup") oModule.AssignTorque(["NAME:Torque1", "Coordinate System:=" , "Global", "Axis:=" , "Z", "Is Positive:=" , True, "Objects:=" , ["Cylinder1"]]) </pre>

DeleteAllParameters

Use: Deletes all parameters.

Command: Maxwell3D or Maxwell2D>Parameters>Delete All

Syntax: DeleteAllParameters

Return Value: None

Example: `oModule.DeleteAllParameters`

DeleteParameters

Deletes one or more specified parameters.

UI Access	Delete button in Maxwell List dialog box (Maxwell3D or Maxwell2D>List)		
Parameters	Name	Type	Description
	<NameArray>	string	An array of parameter names
Return Value	None		

Python Syntax	DeleteParameters ([<NameArray>])
Python Example	<code>oModule.DeleteParameters(["Force1", "Torque1"])</code>

DeleteReduceMatrix

Deletes a specified Reduce Matrix in the project.

UI Access	Right-click on the desired reduce matrix item in Program Manager, then select Delete		
Parameters	Name	Type	Description
	<MatrixName>	String	The name of the parent matrix
	<ReduceMatrixName>	String	The name of the reduce matrix
Return Value	None		

Python Syntax	DeleteReduceMatrix (<MatrixName>, <ReduceMatrixName>)
Python Example	<code>oModule.DeleteReduceMatrix ("Matrix1", "ReduceMatrix1")</code>

DeleteReduceOp

Deletes a specified Reduce Matrix group in the project.

UI Access	Right-click on the desired reduce matrix group item in Program Manager, then select Delete		
Parameters	Name	Type	Description
	<MatrixName>	String	The name of the parent matrix
	<ReduceMatrixName>	String	The name of the reduce matrix
	<GroupName>	String	The name of the reduce matrix group
Return Value	None		

Python Syntax	DeleteReduceOp (<MatrixName>, <ReduceMatrixName>, <GroupName>)
Python Example	oModule.DeleteReduceOp ("Matrix1", "ReduceMatrix1", "JoinSeries1")

EditForce

Use: Edits a force parameter.

Command: Double-click the parameter in the project tree to edit it.

Syntax: EditForce <ParameterName>, <ForceArray>

Return Value: None

[Beta] EditLayoutForce

Creates a layout force.

UI Access	Right-click the desired Layout Force Parameter in the project manager and select Properties... to open the Layout Force Setup dialog for editing.		
Parameters	Name	Type	Description
	<ForceArray>	Array	Array containing force data: Array("NAME:<ForceName>", "ReferenceCS:=", <string>, <array of nets and layers choices>)
Return Value	None		

Python Syntax	EditLayoutForce (<ForceArray>)
Python Example	<pre>oModule = oDesign.GetModule("MaxwellParameterSetup") oModule.EditLayoutForce(["NAME:LayoutForce4", "Reference CS:=" , "Global", ["NAME:NetsAndLayersChoices", ["NAME:LC1_1", ["NAME:NetLayerSetMap", ["NAME:net1", "LayerSet:=" , ["Ground"]], ["NAME:net2", "LayerSet:=" , ["Top"]]]]]])</pre>

	]
	])

EditMatrix

Use: Edits a matrix parameter.

Command: Double-click the parameter in the project tree to edit it.

Syntax:EditMatrix <ParameterName>, <MatrixArray>

Return Value: None

EditReduceOp

Edits a reduce matrix group.

UI Access	Right-click on the reduce matrix group and select Properties....		
Parameters	Name	Type	Description
	<MatrixName>	String	Name of parent matrix
	<ReduceMatrixName>	String	Name of reduce matrix
	<GroupName>	String	Current name of the reduce matrix group
	<ReduceOpArray>	Array	Array("NAME:<NewGroupName>", "Type:=", one of "Join in Series" or "Join in Parallel", "Sources:=", "<Source1>, <Source2, ...")
Return Value	None		

Python Syntax	EditReduceOp (<MatrixName>, <ReduceMatrixName>, <GroupName>, <ReduceOpArray>)
Python Example	<pre>oModule = oDesign.GetModule("MaxwellParameterSetup") oModule.EditReduceOp ("Matrix1", "ReduceMatrix1", "Group1" ["NAME:NewSeriesGroup", "Type:=", "Join in Series", "Sources:=", "Current1,Current3"])</pre>

EditTorque

Use: Edits a torque parameter.

Command: Double-click the parameter in the project tree to edit it.

Syntax: EditTorque <ParameterName>, <TorqueArray>

Return Value: None

ReassignParameter

Use: Specifies a new geometry assignment for a parameter.

Command: Maxwell3D or Maxwell2D>Parameters>Reassign

Syntax: ReassignParameter Array("Name:<ParameterName>","Objects:=", <AssignmentObjects>)

Return Value: None

Example: oModule.ReassignParameter ["NAME:Forcel","Objects:=", ["Box2"]]

RenameParameter

Use: Renames a parameter.

Command: Right-click the parameter item in the project tree, and click **Rename**.

Syntax: RenameParameter <OldName>, <NewName>

Return Value: None

Parameters:

<OldName>

Type: <string>

<NewName>

Type: <string>

Example: oModule.RenameParameter "Forcel", "test"

RenameReduceMatrix

Renames a specified Reduce Matrix in the project.

UI Access	Right-click on the desired reduce matrix item in Program Manager, then select Rename		
Parameters	Name	Type	Description
	<MatrixName>	String	The name of the parent matrix
	<OldReduceMatrixName>	String	The old name of the reduce matrix
	<NewReduceMatrixName>	String	The new name of the reduce matrix
Return Value	None		

Python Syntax	<code>RenameReduceMatriz (<MatrixName>, <OldReduceMatrixName>, <NewReduceMatrixName>)</code>
Python Example	<code>oModule.RenameReduceMatrix ("Matrix1", "ReduceMatrix1", "ReduceMatrix2")</code>

RenameReduceOp

Renames a specified Reduce Matrix group in the project.

UI Access	Right-click on the desired reduce matrix group item in Program Manager, then select Rename		
Parameters	Name	Type	Description
	<MatrixName>	String	The name of the parent matrix
	<ReduceMatrixName>	String	The name of the reduce matrix
	<OldGroupName>	String	The old name of the reduce matrix group
	<NewGroupName>	String	The new name of the reduce matrix group
Return Value	None		

Python Syntax	<code>RenameReduceOp (<MatrixName>, <ReduceMatrixName>, <OldGroupName>, <NewGroupName>)</code>
Python Example	<code>oModule.RenameReduceOp ("Matrix1", "ReduceMatrix1", "Series1", "Series2")</code>

22 - User Defined Document Script Commands

The product has to implement the [GetModule](#) call to create the UserDefinedDocument scripting object (e.g., Check AltraSimDesign.cpp (function GetMgrIDispatch())). To access the UserDefinedDocuments scripting object, use:

```
oModule = oDesign.GetModule("UserDefinedDocuments")
```

Once you have the scripting object, you can use the following methods:

- [AddDocument](#)
- [EditDocument](#)
- [RenameDocument](#)
- [DeleteDocument](#)
- [UpdateDocument](#)
- [ViewHtmlDocument](#)
- [ViewPdfDocument](#)
- [SaveHtmlDocumentAs](#)
- [SavePdfDocumentAs](#)
- [GetDocumentDefinitionNames](#)
- [DeleteAllDocuments](#)
- [UpdateAllDocuments](#)

You can find examples of how to use these methods on each of the methods' pages. A complete [example of a Python script](#) is also available, as is an example with a line by line explanation.

AddDocument

Creates a new document based on provided data and traces. The document names come from UserDefinedDocument folder under syslib, userlib, and personallib. This creates a document and places it in the Project Manager under **Results > Documents**.

UI Access	Right click on Results > Create Document . Choose a document name.		
Parameters	Name	Type	Description
	<data>	Array	Data the defines the document.
	<traces>	Array	trace data for the inputs in the document.
Return Value	None		

Python Syntax	AddDocument (<data>, <traces>)
Python Example	<pre>oModule.AddDocument (["NAME:Design Summary", "", "SysLib", "DesignSummary", ["NAME:Inputs"]], ["NAME:DocTraces"])</pre>

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	<code>oModule.DeleteAllDocuments()</code>

DeleteDocument

Deletes a specified document.

UI Access	Right click on the created document in the Project Manager under Results > Documents and click Delete .		
Parameters	Name	Type	Description
	<name>	String	Name of the document to be deleted.
Return Value	None.		

Python Syntax	DeleteDocument(<names>)
Python 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"]],</pre>

	<pre>["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
- personallb/UserDefinedDocuments

UI Access	NA		
Parameters	Name	Type	Description
	<separator>	String	Separator used to convey the directory "level"
Return Value	None.		

Python Syntax	GetDocumentDefinitionNames(<separator>)
Python Example	oModule.GetDocumentDefinitionNames ("/")

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>

RenameDocument

Changes the name of a document.

UI Access	Right click on the created document in the Project Manager under Results> Documents and click Rename .		
Parameters	Name	Type	Description
	<oldName>	String	Current name of the document
	<newName>	String	New name of the document
Return Value	None.		

Python Syntax	RenameDocument(<oldName>, <newName>)
Python Example	<code>oModule.RenameDocument("Design Summary", "Project Design Summary")</code>

SaveHtmlDocumentAs

Saves a pre-existing HTML file to a different name and/or location.

UI Access	Right click on the created document in the Project Manager under Results > Documents and click Save As > HTML .		
Parameters	Name	Type	Description
	<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>

SavePdfDocumentAs

Saves a pre-existing PDF file to a different name and/or location.

UI Access	Right click on the created document in the Project Manager under Results > Documents and click Save As > PDF .		
Parameters	Name	Type	Description
	<name>	String	Name of the document to be saved.
	<saveTo>	String	File path to save the document at.
Return Value	None.		

Python Syntax	SavePdfDocumentAs(<name>, <saveTo>)
Python Example	<code>oModule.SavePdfDocumentAs("Design Summary 1", "DS1.pdf")</code>

UpdateAllDocuments

Refreshes the contents of all created documents. This action is made on the folder rather than the individual document.

UI Access	Right click on Results > Documents in the Project Manager and click Update All Documents .
Parameters	None.
Return Value	None.

Python Syntax	UpdateAllDocuments()
Python Example	<code>oModule.UpdateAllDocuments()</code>

UpdateDocument

Refreshes the contents of the selected document.

UI Access	Right click on the created document in the Project Manager under Results > Documents and click Update Document .

Parameters	Name	Type	Description
	<name>	String	Name of the document to be updated
Return Value	None.		

Python Syntax	UpdateDocument(<name>)
Python Example	oModule.UpdateDocument("Test UDD Report")

ViewHtmlDocument

Displays a pre-existing document as HTML.

UI Access	Right click on the created document in the Project Manager under Results > Documents and click View Xml Document .		
Parameters	Name	Type	Description
	<name>	String	Name of the document to be viewed as a HTML
Return Value	None		

Python Syntax	ViewHtmlDocument(<name>)
Python Example	oModule.ViewHtmlDocument("Design Summary 1")

ViewPdfDocument

Displays a pre-existing document as a PDF file.

UI Access	Right click on the created document in the Project Manager under Results > Documents and click View PDF Document .		
Parameters	Name	Type	Description
	<name>	String	Name of the document to be viewed as a PDF
Return Value	None.		

Python Syntax	ViewPdfDocument(<name>)
Python Example	<code>oModule.ViewPdfDocument("Design Summary 1")</code>

Example Python Script: Defining a Document

This script creates a user-defined solution and a document.

```
import ScriptEnv
ScriptEnv.Initialize("Ansoft.ElectronicsDesktop")
oDesktop.RestoreWindow()
oProject = oDesktop.SetActiveProject("BJT_Inverter")
oDesign = oProject.SetActiveDesign("Nexxim1")
oModule = oDesign.GetModule("UserDefinedSolutionModule")

oModule.CreateUserDefinedSolution("UDS Distance Trace Arithmetic Result1", "SysLib",
```



```

"TraceArithmetic/Distance Sweep Trace Arithmetic",
[
    "Offset 1:=", "0",
    "Scale 1:=", "1",
    "Offset 2:=", "0",
    "Scale 2:=", "1",
    "Operation:=", "Add"
],
[
    [
        "Standard",
        "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:=", [""]  
],  
[]  
]  
])
```

23 - User Defined Solutions Commands

User Defined Solution commands should be executed by the "UserDefinedSolutionModule" module.

```
oDesign = oProject.SetActiveDesign("TestDesign1")
```

```
oModule = oDesign.GetModule("UserDefinedSolutionModule")
```

The list of commands is as follows:

[CreateUserDefinedSolution](#)

[DeleteUserDefinedSolutions](#)

[EditUserDefinedSolution](#)

CreateUserDefinedSolution

Creates a new user defined solution.

UI Access	Right-click on Results > Create User Defined Solution .		
Parameters	Name	Type	Description
	<SoluName>	String	Name of user defined solution.
	<LibType>	String	Indicates the library where the UDS plugin file is located. This parameter must be one of the following values: "SysLib", "UserLib", "Personallib".
	<RelativePath>	String	The path of the UDS plugin file relative to the "UserDefinedOutputs" sub-directory of the library specified by <LibType>.
	<PropList>	Array	Strings specify name-value pairs corresponding to the UDS properties specified in the plugin file. For example: Array("multiply_factor:=", "2.0", "component_name:=", "resistor1")
	<ProbeSelections>	Array	Name of the probe being specified. Note: this must match a probe name spe-

			cified in the UDS plugin file.
	<DynamicProbes>	Array	Array of <ProbeSelection>'s, representing the probes that are used by dynamic-probes.
Return Value	String name of created user defined solution.		

Python Syntax	CreateUserDefinedSolution(<SoluName>, <LibType>, <RelativePath>, <PropList>, <ProbeSelections>, <DynamicProbes>)		
Python Example	<pre>oModule.CreateUserDefinedSolution("ConstantTimestep1", "SysLib", "ConstantTimestep", [], [], [])</pre>		

DeleteUserDefinedSolutions

Deletes one or more user defined solutions.

UI Access	Delete button from the User Defined Solutions dialog.		
Parameters	Name	Type	Description
	<SoluNames>	Array	Array of strings containing names of User Defined Solutions to be deleted.
Return Value	None.		

Python Syntax	DeleteUserDefinedSolutions(<SoluNames>)
Python Example	<code>oModule.DeleteUserDefinedSolutions(["Solution1", "Solution2"])</code>

EditUserDefinedSolution

Modifies an existing user defined solution.

UI Access	Edit button from the User Defined Solutions dialog box.		
Parameters	Name	Type	Description
	<SoluName>	String	Name of user defined solution to be edited.
	<NewSoluName>	String	New name for the specified user defined solution.
	<LibType>	String	Indicates the library where the UDS plugin file is located. This parameter must be one of the following values: "SysLib", "UserLib", "PersonalLib".
	<RelativePath>	String	The path of the UDS plugin file relative to the "UserDefinedOutputs" sub-directory of the library specified by <LibType>.
	<PropList>	Array	Strings specify name-value pairs corresponding to the UDS properties specified in the plugin file. For example: <code>Array("multiply_factor:=", "2.0", "component_name:=", "resistor1")</code>
	<ProbeSelections>	Array	Name of the probe being specified. Note: this must match a probe name specified in the UDS plugin file.
	<DynamicProbes>	Array	Array of <ProbeSelection>'s, representing the probes that are used by dynamic-probes.
Return Value	String name of update user defined solution.		

Python Syntax	<code>EditUserDefinedSolution(<SoluName>, <NewSoluName>, <LibType>, <RelativePath>, <PropList>, <ProbeSelections>, <DynamicProbes>)</code>
Python Example	<pre>oModule.EditUserDefinedSolution("ConstantTimestep1" "ConstantTimestep1After", "SysLib", "ConstantTimestep", [], [], [])</pre>

24 - Network Data Explorer Script Commands

Network Data Explorer (NDE) scripting uses three objects, each with unique script commands:

- **Network Data Explorer** – top-level object obtained by calling `oNDE=oDesktop.GetTool("ndExplorer")`. Network Data Explorer commands are called using `oNDE`.
- **Network Data** – single set of S-parameters, corresponding to a single entry in the UI tree. Network Data commands are called using `oData`.
- **Post Process Settings** – settings that can be applied and removed from network data without making permanent changes to the underlying data. Post Process Settings commands are called using `oPostProc`.

Examples using the above objects:

```
oNDE = oDesktop.GetTool("ndExplorer")  
  
oData = oNDE.Open("D:\folder\test.s2p")  
  
success = oPostProc.AddDiffPair(2, 1, "Diff1", "Comm1", 100, 25)
```

The following commands are described in this section:

[AddDiffPair](#)

[Cascade \(SPISim\)](#)

[ClearDiffPairs](#)

[Clone](#)

[Close](#)

[Combine \(SPISim\)](#)

[Deembed \(SPISim\)](#)

[DeembedBack \(SPISim\)](#)

[DeembedFront \(SPISim\)](#)

[DisableDiffPairs](#)

[EnableDiffPairs](#)

[ExportCitiFile](#)

[ExportFullWaveSpice](#)

[ExportMatlab](#)

[ExportNMFData](#)

[ExportNetworkData](#)

[ExportSpreadsheet](#)

[ExportTouchstone](#)

[ExportTouchstone2](#)

[Extract \(SPISim\)](#)

[GetFrequencies](#)

[GetFrequencyCount](#)

[GetName](#)

[GetPortCount](#)

[GetPortNumber](#)

[GetPostProcSettings](#)

[GetSolutionVariation](#)

[GetVariation](#)

[HasSameData](#)

- [LoadSolution](#)
- [Open](#)
- [Rename \(SPISim\)](#)
- [Renormalize \(SPISim\)](#)
- [Reorder](#)
- [Reorder \(SPISim\)](#)
- [Reset](#)
- [SetAllPortImpedances](#)
- [SetAllPortImpedances](#)
- [SetPortDeembedDistance](#)
- [SetPortImpedance](#)
- [SetPostProcSettings](#)
- [Smooth](#)
- [Stretch \(SPISim\)](#)
- [Terminate](#)

Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.

AddDiffPair

Specifies a differential pair from two terminal ports.

UI Access	From the Network Data Explorer tab, select Differential Pairs in the NDE ribbon to open the Differential

	Pairs window. Then select differential pairs from the Pairs group box and click Add Pairs >> .		
Parameters	Name	Type	Description
	<positiveTerminal>	Integer	The portNumber of the positive terminal.
	<negativeTerminal>	Integer	The portNumber of the negative terminal.
	<diffName>	String	The new name of the differential pin (e.g., Diff1).
	<commName>	String	The new name of the common pin (e.g., Comm1).
	<diffImpedance>	Double	The differential pin characteristic impedance.
	<commImpedance>	Double	The common pin characteristic impedance.
	<matched>	Boolean	Specify whether to use matched pair.
			Note: The default value is False .
Return Value	Boolean.		

Python Syntax	AddDiffPair()
Python Example	<pre>success = oPostProc.AddDiffPair(2, 1, "Diff1", "Comm1", 100, 25)</pre>

Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.

Cascade (SPISim)

Creates new network data from a group of network data objects cascaded together. The network data must have an even number of terminal ports and must have the same number of ports.

Note: Cascade opens and interacts with **SPISim** to complete.

UI Access	From the Network Data Explorer tab, select Transforms in the NDE ribbon. Then select Cascade from the drop-down menu.		
Parameters	Name	Type	Description
	<dataArray>	Array	These network data objects are cascaded together to create the new network data.
Return Value	Network IDispatch. Note: Cascade returns None if the specified network data does not have compatible terminal ports defined.		

Python Syntax	Cascade()
Python Example	<pre>oNDE = oDesktop.GetTool("ndExplorer") oData2 = oNDE.Cascade([oData1, oData2, oData3])</pre>

Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.

ClearDiffPairs

Clears all differential pair definitions. All other post-processing settings remain the same.

UI Access	From the Network Data Explorer tab, select Differential Pairs in the NDE ribbon to open the Differential Pairs window. Then select differential pairs from the rightmost box and click << Remove Pairs .
Parameters	None.
Return Value	None.

Python Syntax	<code>ClearDiffPairs()</code>
Python Example	<pre>oNDE = oDesktop.GetTool("ndExplorer") oPostProc.ClearDiffPairs()</pre>

Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.

Clone

Creates a copy of the network data object.

UI Access	From the Network Data Explorer tab, select the network data object to clone. Then select Clone from the NDE ribbon.
Parameters	None.
Return Value	Network IDispatch.

Python Syntax	Clone()
Python Example	<pre>oNDE = oDesktop.GetTool("ndExplorer") oData1 = oData.Clone()</pre>

Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.

Close

Closes the network data object. The object will no longer be accessible.

UI Access	From the Network Data Explorer tab, click Close in the NDE ribbon, or select File > Close .
Parameters	None.
Return Value	None.

Python Syntax	Close()
Python Example	<pre>oNDE = oDesktop.GetTool("ndExplorer") oData.Close()</pre>

Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.

Combine (SPISim)

Combines frequencies from multiple network data objects to create a new network data with all of the frequencies.

Note: Combine opens and interacts with **SPISim** to complete.

UI Access	From the Network Data Explorer tab, select Transforms in the NDE ribbon. Then select Combine from the drop-down menu.		
Parameters	Name <dataArray>	Type Array	Description These network data objects are combined to create the new network data.
	<freqTol>	Double	If $\text{abs}(f1 - f2) < \text{freqTol}$, $f1$ and $f2$ are considered the same frequency. Note: The default value is 100 .
Return Value	Network IDispatch. Note: Combine returns None if the network data does not have the same number of ports.		

Python Syntax	Combine()
Python Example	<pre>oNDE = oDesktop.GetTool("ndExplorer") oData4 = oNDE.Combine([oData1, oData2, oData3], 1000)</pre>

Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.

Deembed (SPISim)

Creates a new network data, dembedding the frontData and the backData from the front and back of the originalData.

Note: Deembed opens and interacts with **SPISim** to complete.

UI Access	From the Network Data Explorer tab, select Transforms in the NDE ribbon. Then select Deembed from the drop-down menu to open the Deembed... window, and choose Given 3 S-params in order of: Total, Front, Back from the Settings group box.		
Parameters	Name	Type	Description
	<originalData>	Object	Original network data that is deembeded.
	<frontData>	Object	Network data that is deembeded from the front of the original.
	<backData>	Object	Network data that is deembeded from the back of the original.
Return Value	Network IDispatch. Note: Deembed returns None if the specified network data does not have compatible terminal ports defined.		

Python Syntax	Deembed()
Python Example	oNDE = oDesktop.GetTool("ndExplorer")

	<code>oData2 = oNDE.Deembed(oData1, oDataFront, oDataBack)</code>

Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.

DeembedBack (SPISim)

Creates a new network data, dembedding the backData from the back of the originalData.

Note: DeembedBack opens and interacts with **SPISim** to complete.

UI Access	From the Network Data Explorer tab, select Transforms in the NDE ribbon. Then select Deembed from the drop-down menu to open the Deembed... window, and choose Given 3 S-params in order of: Total, Back from the Settings group box.		
Parameters	Name	Type	Description
	<originalData>	Object	Original network data that is deembedded.
	<backData>	Object	Network data that is deembedded from the back of the original.
Return Value	Network IDispatch. Note: DeembedBack returns None if the specified network data does not have compatible terminal ports defined.		

Python Syntax	DeembedBack()
Python Example	<pre>oNDE = oDesktop.GetTool("ndExplorer") oData2 = oNDE.DeembedBack(oData1, oDataBack)</pre>

Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.

DeembedFront (SPISim)

Creates a new network data, dembedding the frontData from the front of the originalData.

Note: DeembedFront opens and interacts with **SPISim** to complete.

UI Access	From the Network Data Explorer tab, select Transforms in the NDE ribbon. Then select Deembed from the drop-down menu to open the Deembed... window, and choose Given 3 S-params in order of: Total, Front from the Settings group box.		
Parameters	Name	Type	Description
	<originalData>	Object	Original network data that is deembedded.
	<frontData>	Object	Network Data that is deembedded from the front of the original.
Return Value	Network IDispatch. Note: DeembedFront returns None if the specified network data does not have compatible terminal ports defined.		

Python Syntax	DeembedFront()
Python Example	<pre>oNDE = oDesktop.GetTool("ndExplorer") oData2 = oNDE.DeembedFront(oData1, oDataFront)</pre>

Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.

DisableDiffPairs

Deactivates all differential pair definitions. This script does not remove them, so they [can be reactivated](#).

UI Access	From the Network Data Explorer tab, select Differential Pairs in the NDE ribbon to open the Differential Pairs window. Once differential pairs have been added to the rightmost box (i.e., select differential pairs from the Pairs group box and click Add Pairs >>), remove check marks from the Enabled column to deactivate the corresponding differential pairs.
Parameters	None.
Return Value	None.

Python Syntax	DisableDiffPairs()
Python Example	<pre>oNDE = oDesktop.GetTool("ndExplorer") oPostProc.DisableDiffPairs()</pre>

Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.

EnableDiffPairs

Enables all differential pair definitions.

UI Access	From the Network Data Explorer tab, select Differential Pairs in the NDE ribbon to open the Differential Pairs window. Once differential pairs have been added to the rightmost box (i.e., select differential pairs from the Pairs group box and click Add Pairs >>), add check marks to the Enabled column to activate the corresponding differential pairs.
Parameters	None.
Return Value	None.

Python Syntax	EnableDiffPairs()
Python Example	<pre>oNDE = oDesktop.GetTool("ndExplorer") success = oPostProc.EnableDiffPairs()</pre>

Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.

ExportCitiFile

Writes the S-parameters to disk in Citifile format (*.cit text file).

UI Access	From the Network Data Explorer tab, select File > Save As to open a Save As explorer window. Then select Citifile (*.cit) from the Save as type: drop-down menu.		
Parameters	Name	Type	Description
	<filePath>	String	The full path to the file.
	<matrixType>	Integer	One of the following: • 0 – S-Parameters • 1 – Y-Parameters • 2 – Z-Parameters • 3 – Gamma • 4 – Impedance Note: The default value is 0.
	<formalType>	Integer	One of the following: • 0 – Magnitude/Angle (degrees) • 1 – Real/Imaginary • 2 – dB/Angle (degrees) Note: The default value is 0.
	<precision>	Integer	The number of significant figures. Note: The default value is 10.
	<frequencies>	Array of Doubles	A subset of the calculated frequencies. Any array values that don't have calculated data will be ignored. An empty array will cause all frequencies to be written.

			Note: The default value is an empty array.
Return Value	Boolean True if file was successfully written; else False.		

Python Syntax	ExportCitiFile()
Python Example	<pre>oNDE = oDesktop.GetTool("ndExplorer") success = oData.ExportCitiFile("D:\folder\ne133.cit", 0, 1, 12, [])</pre>

Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.

ExportMatlab

Writes the S-parameters to disk in Matlab format (*.m text file).

UI Access	From the Network Data Explorer tab, select File > Save As to open a Save As explorer window. Then select MATLAB (*.m) from the Save as type: drop-down menu.		
Parameters	Name	Type	Description
	<filePath>	String	The full path to the file.
	<matrixType>	Integer	One of the following: • 0 – S-Parameters • 1 – Y-Parameters • 2 – Z-Parameters

			• 3 – Gamma • 4 – Impedance Note: The default value is 0.
	<i><formalType></i>	Integer	One of the following: • 0 – Magnitude/Angle (degrees) • 1 – Real/Imaginary • 2 – dB/Angle (degrees) Note: The default value is 0.
	<i><precision></i>	Integer	The number of significant figures. Note: The default value is 10.
	<i><frequencies></i>	Array of Doubles	A subset of the calculated frequencies. Any array values that don't have calculated data will be ignored. An empty array will cause all frequencies to be written. Note: The default value is an empty array.
Return Value	Boolean True if file was successfully written; else False.		

Python Syntax	ExportMatlab()
Python	<code>oNDE = oDesktop.GetTool("ndExplorer")</code>

Example	<code>success = oData.ExportMatlab("D:\folder\ne133.m", 0, 1, 12, [])</code>

Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.

ExportSpreadsheet

Writes the S-parameters to disk in spreadsheet format (*.tab text file).

UI Access	From the Network Data Explorer tab, select File > Save As to open a Save As explorer window. Then select Data Table (spreadsheet) (*.tab) from the Save as type: drop-down menu.		
Parameters	Name	Type	Description
	<filePath>	String	The full path to the file.
	<matrixType>	Integer	One of the following: • 0 – S-Parameters • 1 – Y-Parameters • 2 – Z-Parameters • 3 – Gamma • 4 – Impedance Note: The default value is 0.
	<formalType>	Integer	One of the following: • 0 – Magnitude/Angle (degrees) • 1 – Real/Imaginary • 2 – dB/Angle (degrees)

			Note: The default value is 0 .
	<precision>	Integer	The number of significant figures. Note: The default value is 10 .
	<frequencies>	Array of Doubles	A subset of the calculated frequencies. Any array values that don't have calculated data will be ignored. An empty array will cause all frequencies to be written. Note: The default value is an empty array.
	<renormalize>	Boolean	If True , will renormalize the data to the value of renormImpedance before writing. Note: The default value is False .
	<renormImpedance>	Double	Data will be renormalized to this impedance before writing only if renormalize is True . Note: The default value is 50 ohms .
Return Value	Boolean True if file was successfully written; else False.		

Python Syntax	ExportSpreadsheet()
Python	oNDE = oDesktop.GetTool ("ndExplorer")

Example	<pre>oData = oNDE.Open(<model path>) success = oData.ExportSpreadsheet("D:\folder\nel33.tab", 0, 1, 12, [], True, 23)</pre>

Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.

ExportTouchstone

Writes the S-parameters to disk in Touchstone 1.0 format (*.snp text file).

UI Access	From the Network Data Explorer tab, select File > Save As to open a Save As explorer window. Then select Touchstone Format 1.0 (*.snp) from the Save as type: drop-down menu.		
Parameters	Name	Type	Description
	<filePath>	String	The full path to the file.
	<matrixType>	Integer	One of the following: • 0 – S-Parameters • 1 – Y-Parameters • 2 – Z-Parameters • 3 – Gamma • 4 – Impedance Note: The default value is 0.
	<formalType>	Integer	One of the following: • 0 – Magnitude/Angle (degrees) • 1 – Real/Imaginary

			• 2 – dB/Angle (degrees) Note: The default value is 0.
	<precision>	Integer	The number of significant figures. Note: The default value is 10.
	<frequencies>	Array of Doubles	A subset of the calculated frequencies. Any array values that don't have calculated data will be ignored. An empty array will cause all frequencies to be written. Note: The default value is an empty array.
	<renormalize>	Boolean	If True, will renormalize the data to the value of renormImpedance before writing. Note: The default value is False.
	<renormImpedance>	Double	Data will be renormalized to this impedance before writing only if renormalize is True. Note: The default value is 50 ohms.
	<freqUnits>	String	One of the following: "Hz", "KHz", "MHz", "GHz". Note: The default value is "Hz".
Return Value	Boolean True if file was successfully written; else False.		

Python Syntax	ExportTouchstone()
Python Example	<pre>oNDE = oDesktop.GetTool("ndExplorer") success = oData.ExportTouchstone("D:\folder\nel133.s2p", 0, 1, 12, [], True, 23, "GHz")</pre>

Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.

ExportTouchstone2

Writes the S-parameters to disk in Touchstone 2 format (*.ts text file).

UI Access	From the Network Data Explorer tab, select File > Save As to open a Save As explorer window. Then select Touchstone 2 Format (*.ts) from the Save as type: drop-down menu.		
Parameters	Name	Type	Description
	<filePath>	String	The full path to the file.
	<matrixType>	Integer	One of the following: • 0 – S-Parameters • 1 – Y-Parameters • 2 – Z-Parameters • 3 – Gamma • 4 – Impedance

		Note: The default value is 0 .
<code><formalType></code>	Integer	One of the following: • 0 – Magnitude/Angle (degrees) • 1 – Real/Imaginary • 2 – dB/Angle (degrees) Note: The default value is 0.
<code><precision></code>	Integer	The number of significant figures. Note: The default value is 10.
<code><frequencies></code>	Array of Doubles	A subset of the calculated frequencies. Any array values that don't have calculated data will be ignored. An empty array will cause all frequencies to be written. Note: The default value is an empty array.
<code><renormalize></code>	Boolean	If True, will renormalize the data to the value of renormImpedance before writing. Note: The default value is False.
<code><renormImpedance></code>	Double	Data will be renormalized to this impedance before writing only if renormalize is True. Note: The default value is 50 ohms.

	<table><tr><td><code><freqUnits></code></td><td>String</td><td>One of the following: "Hz", "KHz", "MHz", "GHz".</td></tr><tr><td colspan="3"> Note: The default value is "Hz". </td></tr></table>	<code><freqUnits></code>	String	One of the following: "Hz", "KHz", "MHz", "GHz".	Note: The default value is "Hz".		
<code><freqUnits></code>	String	One of the following: "Hz", "KHz", "MHz", "GHz".					
Note: The default value is "Hz".							
Return Value	Boolean True if file was successfully written; else False.						

Python Syntax	ExportTouchstone2()		
Python Example	<pre>oNDE = oDesktop.GetTool("ndExplorer") success = oData.ExportTouchstone2("D:\folder\ne133.ts", 0, 1, 12, [], True, 23, "GHz")</pre>		

Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.

Extract (SPISim)

Creates a new smaller network data after removing data for the specified terminal port numbers.

Note: Extract opens and interacts with **SPISim** to complete.

UI Access	From the Network Data Explorer tab, select Transforms in the NDE ribbon. Then select Extract from the drop-down menu.		
Parameters	Name	Type	Description
	<i><oData></i>	Object	Data from this object is copied to a new network data and the copy

			is transformed.
	<portNumbers>	Array of Integers	The port numbers that will be retained (integers between 1 and n).
Return Value	Network IDispatch.		

Python Syntax	Extract()
Python Example	<pre>oNDE = oDesktop.GetTool("ndExplorer") oData2 = oNDE.Extract(oData1, [3, 2])</pre>

Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.

GetFrequencies

Returns the frequency values with calculated data in the network data.

UI Access	None.
Parameters	None.
Return Value	Array of doubles.

Python Syntax	GetFrequencies()
Python Example	<pre>oNDE = oDesktop.GetTool("ndExplorer") Freqs = oData.GetFrequencies()</pre>

Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.

GetFrequencyCount

Returns the number of frequencies with calculated data in the network data.

UI Access	None.
Parameters	None.
Return Value	Integer number of frequencies.

Python Syntax	GetFrequencyCount()
Python Example	<pre>oData = oDesktop.GetTool("ndExplorer") numFreqs = oData.GetFrequencyCount()</pre>

GetName

Returns the design name of the active design, in that order separated by a semicolon.

UI Access	N/A
Parameters	None.
Return Value	String indicating the name of the active design.

Python Syntax	GetName()
Python Example	design_name = oDesign.GetName()

Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.

GetPortCount

Returns the total number of ports.

UI Access	From the Network Data Explorer tab, select Edit Ports in the NDE ribbon to open the Port properties window.
Parameters	None.
Return Value	Integer number of ports.

Python Syntax	GetPortCount()

Python Example	<pre>oNDE = oDesktop.GetTool("ndExplorer") num = oData.GetPortCount()</pre>

Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.

GetPortNumber

Returns the terminal port number for a given terminal port name.

Note: GetPortNumber can be used in other commands that require a port number, if a port name is preferable.

The PortNumber is the index of the name port in the original data and is not changed by any other settings applied to the oPostProc, including the Reorder settings. It is permanently linked to the port name and can be used interchangeably with the port name in many of the PostProcSettings script functions.

UI Access	From the Network Data Explorer tab, ensure the Full Port Names check box is activated. Port numbers will be represented in the object portNames.		
Parameters	Name	Type	Description
	<portName>	String	The designation of the port number.
Return Value	Integer.		

Python Syntax	GetPortNumber()
Python Example	<pre>oNDE = oDesktop.GetTool("ndExplorer")</pre>

	<code>index = oPostProc.GetPortNumber("Input35")</code>

Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.

GetPostProcSettings

Returns a copy of the IDispatch to the postprocessing settings for the network data (oPostProc).

Note: Making changes to the PostProcSettings will not change the network data. To make changes, call oData.SetPostProcSettings to change the network data.

UI Access	None.
Parameters	None.
Return Value	PostProcSettings IDispatch

Python Syntax	<code>GetPostProcSettings()</code>
Python Example	<code>oNDE = oDesktop.GetTool("ndExplorer") oPostProc = oData.GetPostProcSettings()</code>

Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.

GetVariation

Returns the variation key for the network data.

UI Access	None.
Parameters	None.
Return Value	String variation key.

Python Syntax	GetVariation()
Python Example	<pre>oNDE = oDesktop.GetTool("ndExplorer") variation = oData.GetVariation()</pre>

Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.

HasSameData

Compares one network data object with another network data. Actual calculated values will be compared and no interpolation will be done.

UI Access	None.

Parameters	Name	Type	Description
	<NetworkData>	Object	The second network data to compare against.
	<matrixType>	Integer	One of the following: • 0 – S-Parameters • 1 – Y-Parameters • 2 – Z-Parameters • 3 – Gamma • 4 – Impedance Note: The default value is 0.
	<comparePortNames>	Boolean	If True , port names must be identical. Note: The default value is True.
	<compareNoise>	Boolean	If True , and both network data have noise data, the comparison will fail unless the noise values also match. Note: The default value is 10.
	<relativeTolerance>	Double	If the absoluteTolerance test fails, then $\text{abs}(\text{value1} - \text{value2}) / \max(\text{abs}(\text{value1}), \text{abs}(\text{value2}))$ must be less than this to be considered equal. Note: The default value is 1e-14.
	<absoluteTolerance>	Double	If True , then $\text{abs}(\text{value1} - \text{value2}) < \text{absoluteTolerance}$.

			Note: The default value is 0 .
Return Value	Boolean True if the compared network data have the same frequency and matrix values; otherwise, False.		

Python Syntax	HasSameData()
Python Example	<pre>oNDE = oDesktop.GetTool("ndExplorer") success = oData.HasSameData(oData2, 0, True, False, 1e-10, 0)</pre>

Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.

LoadSolution

Loads the specified design (all variations) and returns an integer ID.

UI Access	After analyzing a design, from the Project Manager window, expand the Project Tree and Analysis folders. Then right-click on the completed analysis icon and select Network Data Explorer to open the solution in the Network Data Explorer tab.		
Parameters	Name	Type	Description
	<projectName>	String	Full path to the Electronics Desktop file.
	<designName>	String	Name of the design.
	<solutionName>	String	Name of the solution.
Return Value	Integer ID (identifying the solution); < 0 if the solution is not loaded.		

Python Syntax	LoadSolution()
Python Example	<pre>oNDE = oDesktop.GetTool("ndExplorer") id = oNDE.LoadSolution("D:\folder\Tee.aedt", "HFSS_Test", "Setup1:Sweep1")</pre>

Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.

Open

Reads contents of file and returns the IDispatch for the new network data.

UI Access	From the Network Data Explorer tab, select Open in the NDE ribbon, or select File > Open to open an explorer window. Then navigate to the required S-parameter file.		
Parameters	Name	Type	Description
	<fileName>	String	Full path to the file containing S-parameters.
Return Value	Network IDispatch.		

Python Syntax	Open()
Python Example	<pre>oNDE = oDesktop.GetTool("ndExplorer")</pre>

	<code>oData = oNDE.Open("D:\folder\test.s2p")</code>

Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.

Rename (SPISim)

Creates a new network data of the same size with the terminal ports renamed.

Note: Rename opens and interacts with **SPISim** to complete.

UI Access	From the Network Data Explorer tab, select Transforms in the NDE ribbon. Then select Rename from the drop-down menu.		
Parameters	Name	Type	Description
	<code><oData></code>	Object	Data from this object is copied to a new network data and the copy is transformed.
	<code><portNames></code>	Array of Strings	The new names (one for each port).
Return Value	Network IDispatch.		

Python Syntax	<code>Rename()</code>
Python Example	<pre>oNDE = oDesktop.GetTool("ndExplorer") oData2 = oNDE.Rename(oData1, ["input", "control", "output"])</pre>

Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.

Renormalize (SPISim)

Creates a new network data of the same size after renormalizing the terminal ports using the specified impedance values.

Note: Rename opens and interacts with **SPISim** to complete.

UI Access	From the Network Data Explorer tab, select Transforms in the NDE ribbon. Then select Renormalize from the drop-down menu.		
Parameters	Name	Type	Description
	<oData>	Object	Data from this object is copied to a new network data and the copy is transformed.
	<portImpedances>	Array of Doubles	The new impedance values to be applied to each port.
Return Value	Network IDispatch. Note: Renormalize returns None if there is not an impedance value for each port.		

Python Syntax	Renormalize()
Python Example	<pre>oNDE = oDesktop.GetTool("ndExplorer") oData2 = oNDE.Renormalize(oData1, [3, 2])</pre>

Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.

Reorder

Rearranges the terminal port entries but does not change the port names (i.e., switching first and third terminal port positions will give switched values for S(1,1) and S(3,3) but S(Port1,Port1) and S(Port3,Port3) do not change).

UI Access	From the Network Data Explorer tab, select Edit Ports in the NDE ribbon to open the Port properties window. Click+drag the terminal port rows to reorder them. Make changes, as necessary, then click OK .		
Parameters	Name	Type	Description
	<newOrder>	Array of Integers	Must have one entry for each port; all numbers 1 to <i>n</i> must be present.
Return Value	None.		

Python Syntax	Reorder()
Python Example	<pre>oNDE = oDesktop.GetTool("ndExplorer") oPostProc.Reorder([3, 2, 1])</pre>

Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.

Reorder (SPISim)

Creates a new network data with the same number of terminal ports and with the port names in the same order. The data is also reordered so it corresponds to the new order (e.g., the data for S(Port1, Port1) may correspond to S(Port3, Port3).

Note: Reorder opens and interacts with **SPISim** to complete.

UI Access	From the Network Data Explorer tab, select Transforms in the NDE ribbon. Then select Reorder from the drop-down menu.		
Parameters	Name	Type	Description
	<oData>	Object	Data from this object is copied to a new network data and the copy is transformed.
	<portNumbers>	Array of Integers	The port numbers in new order (integers between 1 and <i>n</i>).
Return Value	Network IDispatch. Note: Reorder returns None if the array argument does not contain all the terminal port numbers, in any order.		

Python Syntax	Reorder()
Python Example	<pre>oNDE = oDesktop.GetTool("ndExplorer") oData2 = oNDE.Reorder([3, 2, 1])</pre>

Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.

Reset

Resets the post-processing to what it was when the network data was created.

Note: This is not equivalent to setting post-processing to an empty state. The network data may have been read from a file or created from solution data that already had post-processing settings.

Resetting the oPostProcSetting object does not change any data; it only changes the settings object. If you want to change a data object to apply the changed settings, you must follow the Reset function call with oData.SetPostProcSettings (oPostProc)

UI Access	From the Network Data Explorer tab, select Reset postprocessing in the NDE ribbon.
Parameters	None.
Return Value	None.

Python Syntax	Reset()
Python Example	<pre>oNDE = oDesktop.GetTool("ndExplorer") oPostProc.Reset()</pre>

Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.

SetAllPortImpedances

Sets all terminal port impedances in a single call. The impedances are in an array of real or complex values.

UI Access	Through the UI, the terminal ports can only be edited individually. From the Network Data Explorer tab, select Edit Ports in the NDE ribbon to open the Port properties window. Make changes, then click OK .		
Parameters	Name	Type	Description
	<impedances>	Array of Doubles or String	Must have one entry for each port or a single complex (or real) value which will be applied to all ports.
Return Value	None.		

Python Syntax	SetAllPortImpedances()
Python Example	<pre>oNDE = oDesktop.GetTool("ndExplorer") oPostProc.SetAllPortImpedances([23, "2+3i", 50]) -or- oData.SetAllPortImpedances(50)</pre>

Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.

SetPortDeembedDistance

Sets terminal port impedance for a single port.

UI Access	From the Network Data Explorer tab, select Edit Ports in the NDE ribbon to open the Port properties window. Make changes, then click OK .		
Parameters	Name	Type	Description
	<portNumber>	Integer	The port number of the changed terminal port (integers between 1 and n).
	<distance>	String	Impedance with units (e.g., "20mm"); if no units, meters are assumed
Return Value	None.		

Python Syntax	SetPortDeembedDistance()
Python Example	<pre>oNDE = oDesktop.GetTool("ndExplorer") oPostProc.SetPortDeembedDistance(2, "20mm")</pre>

Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.

SetPortImpedance

Sets port impedance for a single port.

UI Access	From the Network Data Explorer tab, select Edit Ports in the NDE ribbon to open the Port properties win-

	dow. Make changes, then click OK .		
Parameters	Name	Type	Description
	<portNumber>	Integer	The port number of the changed terminal port (integers between 1 and n).
	<impedance>	Double or String	Double if impedance value is real; string format (e.g., 24+10i) for complex impedance.
Return Value	None.		

Python Syntax	SetPortImpedance()
Python Example	<pre>oNDE = oDesktop.GetTool("ndExplorer") oPostProc.SetPortImpedance(2, "25+3i")</pre>

Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.

SetPostProcSettings

Applies postprocessing settings (oPostProc) to the specified network data.

Note: Making changes to the PostProcSettings will not change the network data. To make changes, call oData.SetPostProcSettings to change the network data.

UI Access	None.		
Parameters	Name	Type	Description
	<PostProcSettings>	Object	The settings that will be applied to the data.
Return Value	None.		

Python Syntax	SetPostProcSettings()		
Python Example	<pre>oNDE = oDesktop.GetTool("ndExplorer") oData.SetPostProcSettings(oPostProc)</pre>		

Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.

Smooth

Creates a new network data with values smoothed. The number of adjacent points smoothed is user-specified.

UI Access	From the Network Data Explorer tab, select Transforms in the NDE ribbon. Then select Smooth from the drop-down menu.		
Parameters	Name	Type	Description
	<oData>	Object	Data from this object is copied to a new network data and the copy is transformed.
	<order>	Integer	The number of adjacent points that are smoothed.

			Note: The default value is 10 .
	<code><enforceCausality></code>	Boolean	Casual data is calculated before smoothing.
			Note: The default value is False .
Return Value	Network IDispatch Note: Network IDispatch is returned only if the Smooth function was successfully able to perform the smoothing operation on the supplied data and meet the maximum 0.025 error tolerance. Otherwise it will return a < null object >. The user may want to try changing other input parameters, for example, <code>enforceCausality</code> .		

Python Syntax	Smooth()
Python Example	<pre>oNDE = oDesktop.GetTool("ndExplorer") oData2 = oNDE.Smooth(oData1, 3, True)</pre>

Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.

Stretch (SPISim)

Creates a new network data representing a matrix of transmission lines with the length of those transmission lines multiplied by a factor of n .

Note: Stretch opens and interacts with **SPISim** to complete.

UI Access	From the Network Data Explorer tab, select Transforms in the NDE ribbon. Then select Stretch from the drop-down menu.		
Parameters	Name	Type	Description
	<oData>	Object	Data from this object is copied to a new network data and the copy is transformed.
	<factor>	Double	The transmission line lengths are multiplied by this number.
Return Value	Network IDispatch. Note: Stretch returns None if the network data argument does not represent a matrix of transmission lines.		

Python Syntax	Stretch()
Python Example	<pre>oNDE = oDesktop.GetTool("ndExplorer") oData2 = oNDE.Stretch(oData1, 3.1)</pre>

Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.

Terminate

Creates a new network data with specified ports terminated.

UI Access	From the Network Data Explorer tab, select Transforms in the NDE ribbon. Then select Terminate from the drop-down menu.		
Parameters	Name	Type	Description
	<oData>	Object	Data from this object is copied to a new network data and the copy is transformed.
	<portNumbers>	Array of Integers	The port numbers that are terminated (integers between 1 and <i>n</i>).
	<termImpedances>	Array of Complex Numbers	The impedance values used to terminate the specified ports.
Return Value	Network IDispatch. Note: Terminate returns None if the port numbers are not valid or there are no impedance value for the port numbers.		

Python Syntax	Terminate()
Python Example	<code>oNDE = oDesktop.GetTool("ndExplorer")</code>

	<code>oData2 = oNDE.Terminate(oData1, [2, 3], [23, "10+2i"])</code>

This page intentionally
left blank.

25 - CompInstance Script Commands

CompInstance commands should be executed by the **oDesign2** or **oPropHost2** object.

```
Set oDesign2 = CompInstance.GetParentDesign
```

```
Set oPropHost2 = CompInstance.GetPropHost
```

Callback Scripting Using CompInstance Object

Callback scripts are scripts that can be set in the Property Dialog for individual properties by clicking the button in the Callback column and choosing a script that is saved with the project. Callback scripts can contain any legal script commands including general Ansys script function calls (e.g., `GetApplicationName()`). In addition, Callback scripts can also call functions on a special object named `CompInstance`.

You can obtain an interface to a `CompInstance` in a schematic or layout by calling `oEditor.GetCompInstanceFromRefDes(refDes)`. For more information see Layout Scripting and [Schematic Scripting](#). This interface is also available as a `CompInstance` object in `CompInstance` event callbacks, such as placing a component in a layout or schematic.

Definitions

<propName> = text string

<value> = double

<valueText> = text string

<fileName> = full path file name

<choices> = string containing menu choices separated by commas

<initialChoice> = string containing initial choice for menu; must be one of the **<choices>**

<scriptName> = string containing name of script stored in project

<bool> is 1 for true or 0 for false

<editorName> is either "Layout" or "SchematicEditor"

The topics for this section include:

[ComplInstance Functions](#)

ComplInstance Functions

Following are commands that can be used to manipulate properties from a ComplInstance script.

The topics for this section include:

[GetComponentName](#)

GetEditor

[GetInstanceID](#)

[GetInstanceName](#)

[GetParentDesign](#)

[GetPropHost](#)

[GetPropServerName](#)

GetComponentName

Returns the name of the component corresponding to this ComplInstance.

UI Access	NA		
Parameters	Name	Type	Description

	None		
Return Value	String name of component (e.g. MS_TRL) and stores it in "name"		

Python Syntax	GetComponentName()
Python Example	<code>name = CompInstance.GetComponentName()</code>

GetInstanceID [Component Instance]

Use: Returns the instanceID of the CompInstance.

Command: None

Syntax: GetInstanceID()

Return Value: String

Python Syntax	GetInstanceID()
Python Example	<code>id = CompInstance.GetInstanceID()</code>

GetInstanceName [Component Instance]

Use: Returns the instance name of the component corresponding to this CompInstance.

Command: None

Syntax: GetInstanceName()

Return Value: String

Returns instanceName (e.g. A7) of compInstance and stores it in "name".

Note that the Instance Name is not the same as the RefDes.

GetParentDesign

Use: Returns an interface to the compInstance's parent design. This interface can be used to call Design functions. See: [Design Object Script Commands](#).

Command: None

Syntax: GetParentDesign()

Return Value: Returns interface to design.

Python Syntax	GetParentDesign()
Python Example	<code>oDesign2 = CompInstance.GetParentDesign()</code>

GetPropHost

Use: Returns an interface to the PropHost of the CompInstance, which gives access to its properties. This interface can be used to call PropHost functions; for more information see Callback Scripting Using PropHost Object.

Command: None

Syntax: GetPropHost()

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

GetPropServerName

Use: Returns the PropServerName of the Component corresponding to this CompInstance.

Command: None

Syntax: GetPropServerName()

Return Value: String

Python Syntax	GetPropServerName()
Python Example	<code>name = CompInstance.GetPropServerName ()</code>

This page intentionally
left blank.

26 - Schematic Scripting

The Schematic scripting interface is a set of commands that match the data changing methods available in the UI of the Schematic, plus some selection and query methods. Examples of the commands available through the scripting interface are adding items, removing items, and modifying items on the Schematic. Identifying objects on the Schematic is done by a unique ID that is generated and returned by all methods that add objects, and by the FindElements and GetSelections methods. This ID can then be passed into commands to modify or remove Schematic objects.

- Since most Schematic objects can have sub-components (such as property displays or segments on a wire, etc), IDs can also be in the format of "TopLevelID:SubComponentIndex".
- The SubComponentIndex is a one-based index into the sub components of the item thus making the second segment of component CAP1 identified by "CAP1:2".
- A SubComponentIndex of zero will refer to the TopLevel item only.
- When an ID is mentioned later in this document, it will refer to either a simple string ID representing a top-level Schematic item or a "TopLevelID:SubComponentIndex" pair unless otherwise stated.
- If a specified ID or SubComponentIndex does not exist, then the function will ignore that entry and proceed with other specified IDs. Even if no valid IDs are specified, the functions will still return success.

The topics for this section include:

- [Method Format](#)
- [Editor Scripting IDs](#)
- [Create Method List](#)
- [General Method List](#)
- [Property Method List](#)
- [Information Method List](#)

Method Format

In the following formats, [Opt=x] appearing after a parameter indicates that the parameter is optional and the default value is x.

When calling object-creation methods as functions, parenthesis are required in order to retrieve the name of the created object.

All methods to create Schematic objects have the following form:

```
Create[type] (VARIANT parameters,  
// Array of type specific parameters  
VARIANT attributes,  
// attributes is in the format of:  
Array("NAME:Attributes", _  
"Page:=", int, _ // [Opt=1] Page number (one-based)  
// Note: Page 1 always exists  
"X:=", double, _ // X position of the object  
"Y:=", double, _ // Y position of the object  
"Angle:=", double, _ // Rotation angle (radians)  
"Flip:=", bool) // True if mirrored  
[out,retval] string id)
```

The algorithm used in the create methods is:

- 1) Create SchAdd[*type*]Command object with parameters passed in
- 2) Execute the command

All methods to modify Schematic objects have the following form:

```
[modification] (Array("NAME:Selections",
```

```
"Page:=", Page number [optional: Default=1]
"Selections:=", IDs to modify see format below),
VARIANT params
// Array of modification specific parameters)
```

The algorithms in these types of methods will be:

- 1) Go through ids and get the SelectableObjs they correspond to
- 2) Select the SelectableObjs found in (1)
- 3) Select any sub components specified
- 4) Create the Sch[*modification*]Command
- 5) Execute the command

ID Format

When IDs are passed to a function it can be in one of the following formats:

- Array of integers of top-level items only
- String of IDs separated by spaces or commas
- Array of strings with each string element being an ID

Point Format

When a method takes an array of points, each element in the array is a string in the format of: "x y", "x,y", or "(x,y)".

Editor Scripting IDs

Objects in the schematic are identified by text IDs. These IDs are used in scripts to perform actions on objects. These IDs are returned by all methods that add objects, and by the FindElements and GetSelections methods. The IDs are then passed into commands to modify or remove Schematic objects.

Format of IDs for different schematic objects:

Components: CompInst@<CompName>;<compInstID>;<optional schematicID>

Geometric primitive: SchObj@<schematicID>

Global Port/ground :: GPort@<portName>;<schematicID>

Interface Port :: IPort@<portName>;<schematicID>

Page Port: PagePort@<portName>;<schematicID>

Wire : Wire@<netName>;<schematicID>;<segment index list>

Find this information in the Properties Window for the selected object as follows:

<CompName> General tab/CompName

<compInstID> General tab/ID

<schematicID> Symbol tab/SchematicID

<portName> Param Values tab/PortName

<netName> General tab/NetName

<segment index list> numbers separated by commas; Symbol tab/Segmentn - e.g. Segment0 refers to index "0"

In the Layout, the ID for a selected object is always the Name or Net property in the Footprint tab of the Properties Window

Format for Components

```
[ "CompInst@CAP_ ; 2 ; 4" ]
```


The format for Components is `CompInst@<CompName>;<compInstID>;<optional schematicID>`. In the documented example, `<CompName>` stands for the component name `CAP_`, `<compInstID>` stands for its ID 2, and `<optional schematicID>` is the schematic ID 4. If you select the component, the **Properties** window gets updated and displays its details. For example, the selected object's component name and ID appear in the **General** tab of the **Properties** window as shown below.

Properties	
Name	Value
ID	2
CompName	CAP_
Description	Capacitor
Manufacturer	
Datasource	Ansoft built-in component
Date	15:30:47 11/14/2005
PinCount	2
Refdes	C1
Symbol	nexx_cap
Footprint	__symbolFootprint_nexx_cap__positive__negative_0
Param Values General Symbol	

The SchematicID is shown in the **Symbol** tab.

Create Method List

This section lists the scripts that are available in the Schematic scripting interface to create and change data.

Script Position Parameters

In Create Method scripts, X and Y position parameters are expressed in meters.

- If your layout grid units are metric, you may enter the X and Y positions directly into a script as parameters.
- If your layout grid is expressed in mils, you must first convert the component's positional coordinates to meters.

To convert an X or Y position to meters based on the minor grid size:

$$\text{X or Y position in meters} = \text{desired\_position} \times (0.00254 / \text{minor\_grid\_size})$$

where desired_position and minor_grid_size have the same units. For example, to position an object at 500mm with a minor grid size of 20mm:

$$\text{X or Y position in meters} = 500 \times (0.00254 / 20)$$

This section lists the following commands:

[CreateArc \(Schematic Editor\)](#)

[CreateCircle \(Schematic Editor\)](#)

[CreateComponent \(Schematic Editor\)](#)

[CreateCurve \(Schematic Editor\)](#)

[CreateGlobalPort \(Schematic Editor\)](#)

[CreateGround \(Schematic Editor\)](#)

[CreateLine \(Schematic Editor\)](#)

[CreatePagePort \(Schematic Editor\)](#)

[CreateIPort \(Schematic Editor\)](#)

[CreatePolygon \(Schematic Editor\)](#)

[CreateRectangle \(Schematic Editor\)](#)

[CreateText \(Schematic Editor\)](#)

[CreateWire \(Schematic Editor\)](#)

CreateArc (Schematic Editor)

Create an arc

UI Access	Draw>Primitive>Arc		
Parameters	Name	Type	Description
	<ArcData>	Array	<pre>Array("NAME:ArcData" _ "x:=", double, _ // X position of the object "y:=", double, _ // Y position of the object "Radius:=", double, _ // Radius of the circle "StartAng:=", double, _ // Start angle of the arc (radians) "EndAng:=", double, _ // End angle of the arc (radians) "LineWidth:=", double, // the width of the line, in meters "Color:=", int, // the RGB value of the arc color "Id:=", int), // [Opt=New id] Id for this item</pre>
	<Attributes>	Array	<pre>Array("NAME:Attributes", _ "Page:=", int) _ // [Opt=1] Page number (one-based) [out,retval] string id)</pre>
Return Value	String Unique id		

Arc is created with the format SchObj@<schematicID>

The Schematic ID can be found on the **Symbols** tab of the **Properties** window when you select the arc.

Properties		
Name	Value	Unit
SchematicID	48	
Color		
Linewidth	0	mil
Fill Style	Hollow	
Center	4100 , 3300	mil
Radius	360.5551275464	mil
StartAngle	123.69006752598	deg
EndAngle	315	deg
◀		
Symbol		

Python Syntax	CreateArc()
Python Example	<pre>oEditor.CreateArc (["NAME:ArcData", _ "X:=", -0.004318, "Y:=", -0.00127, _ "Radius:=", 0.00297299377732279, _</pre>

```
"StartAng:=", 1.9195673303788, _
"EndAng:=", 3.32144615338227, _
"Id:=", 10], _
["NAME:Attributes", "Page:=", 1])
```

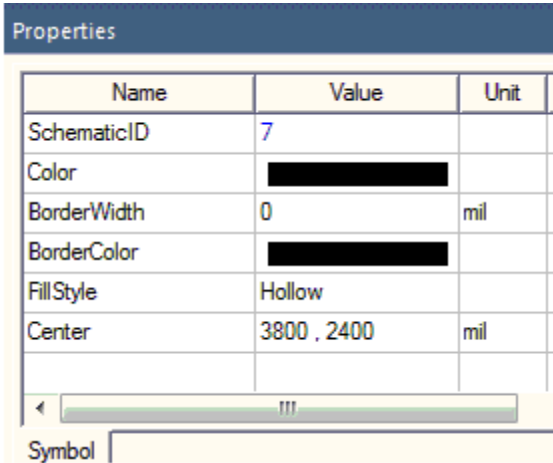
CreateCircle (Schematic Editor)

Create a circle

UI Access	Draw>Primitive>Circle		
Parameters	Name	Type	Description
	<CircleData>	Array	Array("NAME:CircleData" _ "x:=", double, _ // X position of the object "y:=", double, _ // Y position of the object "Radius:=", double, _ // Radius of the circle "LineWidth:=", double, // the width of the circle border, in meters "Bordercolor:=", int, // the RGB value of the border color "Fill:=", int, // the fill pattern id, 0 = hollow, 1 = solid, 2 = NEDiagonal, 3 = OrthoCross, 4 = DiagCross, 5 = NWDiagonal, 6 = Horizontal, 7 = Vertical "Color:=", int, // the RGB value of the circle fill color "Id:=", int), // [Opt=New id] Id for this item
	<Attributes>	Array	Array("NAME:Attributes", _ "Page:=", int) _ // [Opt=1] Page number (one-based)

	[out,retval] string id)
Return Value	String Unique id

Circle is created and the returned value has the format SchObj@<schematicID>
The schematic ID of the circle can be located in the **Properties** window on the **Symbols** tab.



Python Syntax	CreateCircle()
Python Example	oEditor.CreateCircle(["NAME:CircleData", _

	<pre>"X:=", -0.004572, "Y:=", -0.000508, _ "Radius:=", 0.001778, "Id:=", 12], _ ["NAME:Attributes", "Page:=", 1])</pre>

CreateComponent (Schematic Editor)

Creates a component of a given type.

UI Access	N/A.		
Parameters	Name	Type	Description
	<params>	Array	Structured array. Array("NAME:ComponentProps", "Name:=", <String path to component in library>, "Id:=", <String Component ID>)
	<attributes>	Array	Structured array. Array("NAME:Attributes", "Page:=", <Integer: page number>, Note: For the page number, page 1 always exists. "X:=", <Double: X position of object>, "Y:=", <Double: Y position of the object>,

			<pre>"Angle:=", <Double: rotation angle in degrees>, "Flip:=", <boolean: True if mirrored; else False>, "PinNumber:=", <Integer: index of pin in symbol 0 to n-1></pre> Note: PinNumber is optional and is set to -1 by default. PinNumber affects where the symbol is placed relative to the X and Y parameters. If PinNumber is set to -1, the symbol is placed with its center at (X,Y). If it is a pin index, the symbol is placed so that the pin is located at (X,Y). The PinNumber is not recorded.
Return Value	String containing unique ID in the format: <pre>CompInst@<CompName>;<compInstID>;<optional schematicID></pre> For example, the return value <code>CompInst@CAP;4;35</code> indicates that the component name is <code>CAP</code>, its ID is <code>4</code>, and its Schematic ID is <code>35</code>. You can also see these details in the Properties window when you select the created component.		

Python Syntax	<code>CreateComponent(<params>, <attributes>)</code>
Python Example	<pre>oEditor.CreateComponent(["NAME:ComponentProps", "Name:=", "Maxwell Circuit Elements\Passive Elements:Res",</pre>

	<pre>"Id:=", "1"], ["NAME:Attributes", "Page:=", 1, "X:=", -0.01524, "Y:=", 0.00508, "Angle:=", 0, "Flip:=", false])</pre>

CreateGlobalPort (Schematic Editor)

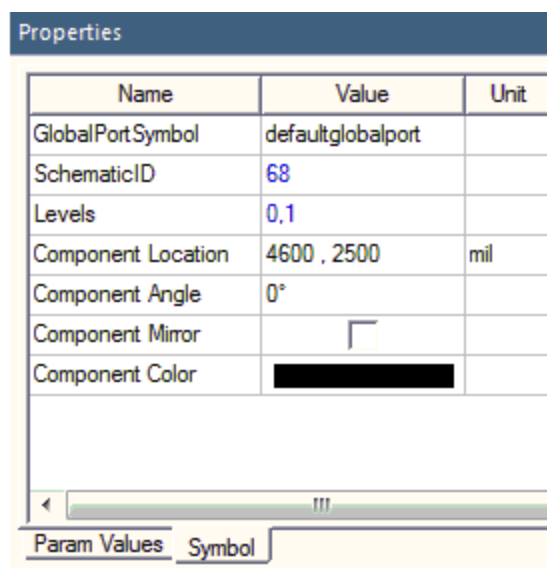
Create a global port and enable placement on the schematic

UI Access	Draw>Global Port		
Parameters	Name	Type	Description
	<portname>	String	Name for this port - leave empty to use default unique name
	<id>	Integer	ID number for the port - leave empty to use default (recommended)
	<pagenumber>	Integer	The (1-based) page number of the schematic page this port should be added to - leave empty to use default (1)
	<xval>	Double	X position of the object
	<yval>	Double	Y position of the object
	<angle>	Double	Rotation of the object, in radians. Rotations of multiples of 90 degrees are valid. Other values will be adjusted to the nearest valid angle.

	<degrees>	Double	Rotation of the object, in degrees. Rotations of multiples of 90 degrees are valid. Other values will be adjusted to the nearest valid angle
	<flip>	Boolean	True if mirrored
Return Value	String PortID, the identifier that can be used for this port in script operations involving this schematic		

Global port is created and the return value has the format GPort@<portName>;<schematicID>

The Schematic ID can be found in the **Properties** window on the **Symbols** tab when you select the global port.



Python Syntax	<code>CreateGlobalPort(["NAME:GlobalPortProps", "Name:=", <portname>, "Id:=", <id>], ["NAME:Attributes", "Page:=", <pagenumber>, "X:=", <xval>, "Y:=", <yval>, "Angle:=", <angle>, [or "Degrees:=", <degrees> ,] "Flip:=", <flip>])</code>
Python Example	<pre>oEditor.CreateGlobalPort (["NAME:GlobalPortProps", "Name:=", "G_0", "Id:=", 1], _ ["NAME:Attributes", "Page:=", 1, "X:=", -0.02286, "Y:=", 0.00254, "Angle:=", 0, "Flip:=", false])</pre>

CreateGround (Schematic Editor)

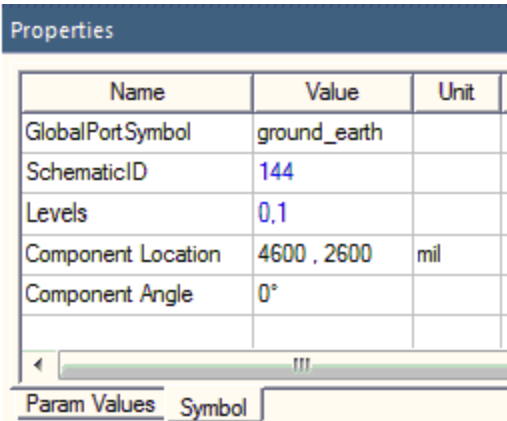
Create a ground

UI Access	Draw>Ground		
Parameters	Name	Type	Description
	<GroundProps>	Array	Array("NAME:GroundProps", _ "Id:=", int), // [Opt=New id] Id for this item
	<Attributes>	Array	Array("NAME:Attributes", _ "Page:=", int, _ // [Opt=1] Page number (one-based) Note: Page 1 always exists "X:=", double, _ // X position of the object "Y:=", double, _ // Y position of the object "Angle:=", double, _ // Rotation angle (radians) "Flip:=", bool), // True if mirrored [out,retval] string id)

Return Value	String Unique id

Ground is created and the returned value has the format GPort@<portname>;<schematicID>

The Schematic ID for ground can be found on the **Symbols** tab of the **Properties** window when you select the object.



Python Syntax	CreateGround()
Python Example	<pre>oEditor.CreateGround (["NAME:GroundProps", "Id:=", 8), ["NAME:Attributes", "Page:=", 1, "X:=",</pre>

```
-0.04064, "Y:=", -0.00254, "Angle:=",  
0, "Flip:=", false])
```

CreateLine (Schematic Editor)

Creates a line.

UI Access	Draw>Primitive>Line		
Parameters	Name	Type	Description
	<LineData>	Array	Array("NAME:LineData" _ "Points:=", Array of points, _ "LineWidth:=", double, // the width of the line, in meters "Color:=", int, // the RGB value of the line color "Id:=", int), // [Opt=New id] Id for this item
	<Attributes>	Array	Array("NAME:Attributes", _ "Page:=", int) _ // [Opt=1] Page number (one-based) [out,retval] string id)
Return Value	String Unique id		

Line is created and the returned value has the format SchObj@<schematicID>

The Schematic ID can be found on the **Symbol** tab of the **Properties** window when you select the line.

Name	Value	Unit
SchematicID	169	
Color		
Linewidth	0	mil
Line Style	Solid	
Begin Object	None	

Python Syntax	CreateLine()
Python Example	<pre>oEditor.CreateLine (["NAME:LineData",_ "Points:=", ["(-0.055652, 0.020669)", _ "(-0.036923, 0.011257)", "(-0.023144, 0.018049)"],_ "LineWidth:=", 0, "Color:=", 0, "Id:=", 22)], _ ["NAME:Attributes", "Page:=", 1])</pre>

CreatePagePort (Schematic Editor)

Create a PagePort

UI Access	Draw>Page Connector		
Parameters	Name	Type	Description
	<PagePortProps>	Array	Array ("NAME:PagePortProps") , Array ("NAME:Attributes", _ "Page:=", int, _ // [Opt=1] Page number (one-based) Note: Page 1 always exists "X:=", double, _ // X position of the object "Y:=", double, _ // Y position of the object "Angle:=", double, _ // Rotation angle (radians) "Flip:=", bool), // True if mirrored [out,retval] string id)
Return Value	String Unique id		

The pageport is created and it has the following format:

```
PagePort@<portName>;<schematicID>
```

The schematic ID can be found in the **Properties** window on the **Symbols** tab when you select the pageport.

Properties		
Name	Value	Unit
SchematicID	177	
Levels	0.1	
Component Location	3500 , 6500	mil
Component Angle	0°	
Component Mirror		
Param Values Symbol		

Python Syntax	CreatePagePort()
Python Example	<pre>oEditor.CreatePagePort (["NAME:PagePortProps", "Name:=", "pageport_0", "Id:=", _ 12], ["NAME:Attributes", "Page:=", _ 1, "X:=", -0.0381, "Y:=", -0.0127, "Angle:=", _ 0, "Flip:=", false])</pre>

CreteIPort (Schematic Editor)

Create an interface port

UI Access	Draw>Interface Port		
Parameters	Name	Type	Description
	<IPortProps>	Array	<pre>Array ("NAME:IPortProps" , "Name:=", string), // [Opt=default name] Name for this port "Id:=", int), // Port id number</pre>
	<Attributes>	Array	<pre>Array ("NAME:Attributes", _ "Page:=", int, _ // [Opt=1] Page number (one-based)</pre> Note: Page 1 always exists "X:=", double, _ // X position of the object "Y:=", double, _ // Y position of the object "Angle:=", double, _ // Rotation angle (radians) "Flip:=", bool), // True if mirrored [out,retval] string id)
Return Value	String Unique id		

Interface port is created and the returned value has the following format:

IPort@<portName>;<schematicID>

The interface port schematic ID can be found in the **Properties** window on the **Symbols** tab.

Properties		
Name	Value	Unit
SchematicID	189	
Levels	0.1	
Component Location	3500 , 6500	mil
Component Angle	0°	
Component Mirror		
Param Values Symbol		

Python Syntax	CreateIPort()
Python Example	<pre>oEditor.CreateIPort (["NAME:IPortProps", "Name:=", _ "Port1", "Id:=", 4], ["NAME:Attributes","Page:=", _ 1, "X:=", -0.0381, "Y:=", 0.0127, "Angle:=", _ 0, "Flip:=", false])</pre>

CreatePolygon (Schematic Editor)

Creates a polygon.

UI Access	Draw > Primitive > Polygon.

Parameters	Name	Type	Description
	<PolygonData>	Array	<pre>Array("NAME:PolygonData" _ "Points:=", Array of points, _ "LineWidth:=", double, // the width of the Polygon border, in meters "Bordercolor:=", int, // the RBG value of the border color "Fill:=", int, // the fill pattern id, 0 = hollow, 1 = solid, 2 = NEDiagonal, 3 = OrthoCross, 4 = DiagCross, 5 = NWDiagonal, 6 = Horizontal, 7 = Vertical "Color:=", int, // the RBG value of the Polygon fill color "Id:=", int), // [Opt=New id] Id for this item</pre>
	<Attributes>	Array	<pre>Array("NAME:Attributes", _ "Page:=", int) _ // [Opt=1] Page number (one-based) [out,retval] string id)</pre>
Return Value	String Unique id		

Polygon is created and the return value has the format

SchObj@<schematicID>

The Schematic ID can be found in the **Properties** window on the **Symbol** tab when you select the polygon.

Properties		
Name	Value	Unit
SchematicID	218	
Color		
BorderWidth	0	mil
BorderColor		
FillStyle	Hollow	
Symbol		

Python Syntax	CreatePolygon()
Python Example	<pre>oEditor.CreatePolygon(["NAME:PolygonData", "Points:=", [_ "(-0.058951, 0.006308)", "(-0.046142, 0.010480)",_ "(-0.046239, 0.001067)"), "LineWidth:=", _ 0, "BorderColor:=", 0, "Fill:=", 0, "Color:=",_ 0, "Id:=", 27], ["NAME:Attributes", "Page:=", 1])</pre>

CreateRectangle (Schematic Editor)

Create a rectangle

UI Access	Draw>Primitive>Rectangle		
Parameters	Name	Type	Description
	<RectData>	Array	<pre>Array("NAME:RectData" _ "x1:=", double, _ // X position of the upper left "y1:=", double, _ // Y position of the upper left "x2:=", double, _ // X position of the lower right "y2:=", double, _ // Y position of the lower right "LineWidth:=", double, // the width of the Rectangle border, in meters "Bordercolor:=", int, // the RGB value of the border color "Fill:=", int, // the fill pattern id, 0 = hollow, 1 = solid, 2 = NEDiagonal, 3 = OrthoCross, 4 = DiagCross, 5 = NWDiagonal, 6 = Horizontal, 7 = Vertical "Color:=", int, // the RGB value of the Rectangle fill color "Id:=", int), // [Opt=New id] Id for this item</pre>
	<Attributes>	Array	<pre>Array("NAME:Attributes", _ "Page:=", int) _ // [Opt=1] Page number (one-based) [out,retval] string id)</pre>
Return Value	String Unique id		

Rectangle is created and the return value has the format

SchObj@<schematicID>

The Schematic ID can be found in the **Properties** window on the **Symbol** tab when you select the rectangle.

UI Access	Draw>Primitive>Text		
Parameters	Name	Type	Description
	<TextData>	Array	Array ("NAME:TextData" _ "x:=", double, _ // X position of the object "y:=", double, _ // Y position of the object "Text:=", string, _ // Text to display "Color:=", int, // the RBG value of the text color "Id:=", int), // Id for this item "ShowRect:=", bool, // true or false, whether to show the highlight rectangle for the text "X1:=", double, // the text rectangle left X value, in meters "Y1:=", double, // the text rectangle upper Y value, in meters "X2:=", double, // the text rectangle right X value, in meters "Y2:=", double, // the text rectangle lower Y value, in meters "RectLineWidth:=", double, // the width of the rectangle border, in meters "RectBordercolor:=", int, // the RBG value of the rectangle border color "RectFill:=", int, // the rectangle fill pattern id, 0 = hollow, 1 = solid, 2 = NEDiagonal, 3 = OrthoCross, 4 = DiagCross, 5 = NWDiagonal, 6 = Horizontal, 7 = Vertical "RectColor:=", int, // the RBG value of the rectangle fill color
	<Attributes>	Array	Array ("NAME:Attributes", _ "Page:=", int) _ // [Opt=1] Page number (one-based) [out,retval] string id)

Return Value	String Unique id

Text is created and the return value has the format: SchObj@<SchematicID>

The Schematic ID can be found in the **Properties** window on the **Symbols** tab when you select the text.

Properties		
Name	Value	Unit
SchematicID	286	
Color		
Location	4100 , 2600	mil
Angle	0	deg
TextSize	12	
Symbol		

Python Syntax	CreateText()
Python Example	<pre>oEditor.CreateText(["NAME:TextData", "X:=", -0.0764183381088826, "Y:=", _ 0.040174212034384, "Size:=", 12, "Angle:=", _</pre>

	<pre>0, "Text:=", "Control Circuit", "Color:=", _ 0, "Id:=", 34, "ShowRect:=", false, "X1:=", _ -0.0793287547755609, "Y1:=", _ 0.0407033787010528, "X2:=", -0.0502245881087778, _ "Y2:=", 0.035411712034365, "RectLineWidth:=", _ 0, "RectBorderColor:=", 0, "RectFill:=", 0, "RectColor:=", 0], ["NAME:Attributes", "Page:=", 1])</pre>

CreateWire (Schematic Editor)

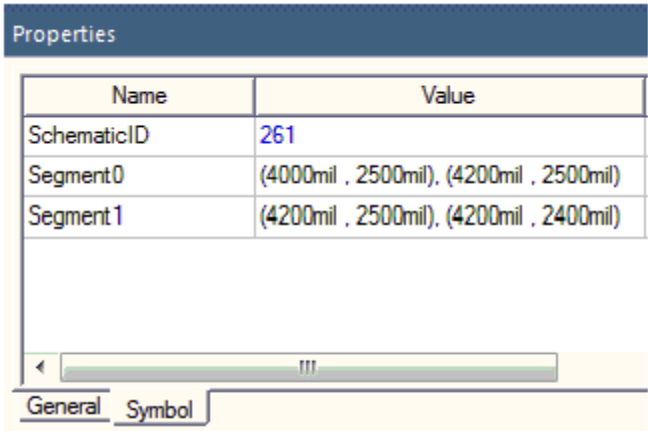
Create a wire

UI Access	Draw>Wire		
Parameters	Name	Type	Description
	<WireData>	Array	Array("NAME:WireData" _ "Name:=", string), // [Opt=default name] Name for this wire "Id:=", int), // Wire idnum "Points:=", Points on wire)
	<Attributes>	Array	Array("NAME:Attributes", _ "Page:=", int, _ // [Opt=1] Page number (one-based) Note: Page 1 always exists [out,retval] string id)
Return Value	String		

	Unique id

Wire is created and its return value has the format Wire@<netName>;<schematicID>;<segment index list>

The Schematic ID can be found in the **Symbol** tab of the **Properties** window when you select the wire. The wire net name appears in the **General** tab. The parameter segment index list indicates the number of segments i.e. the segment count.



Python Syntax	CreateWire()
Python Example	<pre>oEditor.CreateWire(["NAME:WireData", "Name:=", "", "Id:=", 41, "Points:=", [_ "(-0.033020, 0.015240)", "(-0.017780, 0.015240)",</pre>

	<code>"(-0.017780, 0.012700)"]], [{"NAME:Attributes", "Page:=", 1}]</code>

General Method List

This section presents the general script methods that are available.

The general method script commands are listed here.

[Activate \(Schematic Editor\)](#)

[AddPinGrounds \(Schematic Editor\)](#)

[AddPinPorts \(Schematic Editor\)](#)

[AddPinPageConnectors \(Schematic Editor\)](#)

[AddPinCustomTerminations \(Schematic Editor\)](#)

[AlignHorizontal \(Schematic Editor\)](#)

[AlignVertical \(Schematic Editor\)](#)

[BringToFront \(Schematic Editor\)](#)

[Close Editor \(Schematic Editor\)](#)

[Copy \(Schematic Editor\)](#)

[CopyData \[Schematic Editor\]](#)

[CopySubdesign \[Schematic Editor\]](#)

[CreatePage \(Schematic Editor\)](#)

[Cut \(Schematic Editor\)](#)

[DeactivateOpen \(Schematic Editor\)](#)

[DeactivateShort \(Schematic Editor\)](#)

[Delete \(Schematic Editor\)](#)

[DeletePage \(Schematic Editor\)](#)

[ElectricRuleCheck \(Schematic Editor\)](#)

[ExportImage \(Schematic Editor\)](#)

[FindElements \(Schematic Editor\)](#)

[FitToBorder \[Schematic Editor\]](#)

[GridSetup \(Schematic Editor\)](#)

[FlipHorizontal \(Schematic Editor\)](#)

[FlipVertical \(Schematic Editor\)](#)

[Move \(Schematic Editor\)](#)

[NameNets \(Schematic Editor\)](#)

[PageBorders \(Schematic Editor\)](#)

[Pan \(Schematic Editor\)](#)

[Paste \(Schematic Editor\)](#)

[PasteData \[Schematic Editor\]](#)

[PasteDesign \(Schematic Editor\)](#)

[PushExcitations \(Schematic Editor\)](#)

[Rotate \(Schematic Editor\)](#)

[ShowVariableBlock \(Schematic Editor\)](#)

[SelectAll \(Schematic Editor\)](#)

[SelectPage \(Schematic Editor\)](#)

[SendToBack\(Schematic Editor\)](#)

[SetPageData \[Schematic\]](#)

[SortComponents \(Schematic Editor\)](#)

[ZoomArea \(Schematic Editor\)](#)

[ZoomIn \(Schematic Editor\)](#)

[ZoomOut \(Schematic Editor\)](#)

[ZoomPrevious \(Schematic Editor\)](#)

[ZoomToFit \(Schematic Editor\)](#)

[Wire \(Schematic Editor\)](#)

Activate (Schematic Editor)

Enables items previously disabled.

UI Access	Edit > Activate.		
Parameters	Name	Type	Description
	<Selections>	Array	Description
Return Value	None		

Note:

For more information about the component name, schematic IDs, and formats, see the section [Editor Scripting IDs](#).

Python Syntax	Activate(["NAME:Selections","Selections:=", [<components>]])
Python Example	oEditor.Activate(["NAME:Selections","Selections:=", ["CompInst@STEP;1;1"]])

AddPinGrounds (Schematic Editor)

Adds grounds at all unconnected pins

```
AddPinGrounds (  
Array("NAME:Selections", _ // PagePort Name  
"Selections:=", _ // Net Name  
Array("CompInst@CAP_;2;4"))
```

Note:

The format for Components is CompInst@<CompName>;<compInstID>;<optional schematicID>. In the documented example, <CompName> stands for the component name CAP_, <compInstID> stands for its ID 2, and <optional schematicID> is the schematic ID 4. If you select the component, the **Properties** window gets updated and displays its details. For example, the selected object's component name and ID appear in the **General** tab of the **Properties** window as shown below.

Properties	
Name	Value
ID	2
CompName	CAP_
Description	Capacitor
Manufacturer	
Datasource	Ansoft built-in component
Date	15:30:47 11/14/2005
PinCount	2
Refdes	C1
Symbol	nexx_cap
Footprint	__symbolFootprint_nexx_cap__positive__negative_0
Param Values General Symbol	

To view the SchematicID click the **Symbol** tab.

Name	Value	Unit
SchematicID	4	
Levels	0.1	
Component Location	3900 , 2300	mil
Component Angle	0°	
Component Mirror	☐	
Use Symbol Color	☒	

Python Syntax	AddPinPageConnectors ()
Python Example	oEditor.AddPinPageconnectors["NAME:Selections", "Selections:=", [_ "CompInst@C;1;1"]]

AddPinIPorts (Schematic Editor)

Adds Interface Ports at unconnected pins of all selected components. Each port has a unique name.

UI Access	Draw > Add at unconnected pins > Interface Ports		
Parameters	Name	Type	Description
	<IntPortName,NetName>	Array	Array("NAME:Selections", _ // Interface Port Name "Selections:=", _// Net Name
Return Value	None		

Note:

For more information about the component name, schematic IDs, and formats, see the section [Editor Scripting IDs.](#)

Python Syntax	AddPinIPorts ()
Python Example	<pre>oEditor.AddPinIPorts ["NAME:Selections", "Selections:=", ["CompInst@R;2;37"]]</pre>

AddPinPageConnectors (Schematic Editor)

Adds Page Ports at unconnected pins of all selected components

UI Access	Draw > Add at unconnected pins > Grounds.

Parameters	Name	Type	Description
	Name	Type	Description
Return Value	None		

Note:

For more information about the component name, schematic IDs, and formats, see the section [Editor Scripting IDs](#).

Python Syntax	AddPinPageConnectors ()
Python Example	<pre>oEditor.AddPinPageconnectors ["NAME:Selections", "Selections:=", [_ "CompInst@C;1;1"]]</pre>

AlignHorizontal (Schematic Editor)

Align items horizontally

UI Access	Draw>Align Horizontal		
Parameters	Name	Type	Description
	<Selections>	Array	Array ("NAME:Selections", _ "Page:=", page number, _ // [Opt=1] Page number "Selections:=", IDs to modify)
	<AlignParameters>	Array	Array ("NAME:AlignParameters", _

			"Disconnect:=", bool _ // [Opt=0] Should wires disconnect "Rubberband:=", bool) _ // [Opt=1] Should wires staircase Note: Alignment occurs relative to the first item in ids
Return Value	Value		

Note:

For more information about the component name, schematic IDs, and formats, see the section [Editor Scripting IDs.](#)

Python Syntax	AlignHorizontal()
Python Example	<pre>oEditor.AlignHorizontal (["NAME:Selections", "Selections:=", [_ "CompInst@R;2;4", "CompInst@C;1;1"]], ["NAME:AlignParameters", "Disconnect:=", _ false, "Rubberband:=", false])</pre>

AlignVertical (Schematic Editor)

Align items vertically

UI Access	Draw>Align Vertical		
Parameters	Name	Type	Description
	<Selections>	Array	Array("NAME:Selections", _ "Page:=", page number, _ // [Opt=1] Page number "Selections:=", IDs to modify)

	<Align Parameters> Array Array("NAME:AlignParameters", _ "Disconnect:=", bool _ // [Opt=0] Should wires disconnect "Rubberband:=", bool) _ // [Opt=1] Should wires staircase Note: Alignment occurs relative to the first item in ids
Return Value	None

Note:

For more information about the component name, schematic IDs, and formats, see the section [Editor Scripting IDs](#).

Python Syntax	AlignVertical()
Python Example	<pre>oEditor.AlignVertical (["NAME:Selections", "Selections:=", ["CompInst@R;2;4", _ "CompInst@C;1;1"]], ["NAME:AlignParameters", "Disconnect:=", false, "Rubberband:=", _ false])</pre>

BringToFront (Schematic Editor)

Bring the selected object to the front of the view

UI Access	Draw>Bring to Front		
Parameters	Name	Type	Description
	<Selections>	Array	["NAME: Selections", "Selections:=", [<Selected Components>]] <Selected Components> CompInst@DefName; ID, schematic ID , CompInst@DefName; ID, schematic ID CompInst@R;1,2
Return Value	None		

Note:

For more information about the component name, schematic IDs, and formats, see the section [Editor Scripting IDs](#).

Python Syntax	BringToFront()
Python Example	<pre>oEditor.BringToFront (["NAME:Selections", "Selections:=", ["SchObj@1"]])</pre>

CloseEditor (Schematic Editor)

Close an Editor

UI Access	NA		
Parameters	Name	Type	Description

	None		
Return Value	None		

Note:

For more information about the component name, schematic IDs, and formats, see the section [Editor Scripting IDs](#).

Python Syntax	CloseEditor()
Python Example	oDefinitionEditor.CloseEditor()

Copy (Schematic Editor)

```
GeneralCopy
// Copy items for pasting
Copy(
Array("NAME:Selections", _
"Page:=", page number, _ // [Opt=1] Page number
"Selections:=", IDs to modify))
```

Note:

For more information about the component name, schematic IDs, and formats, see the section [Editor Scripting IDs](#).

Python Syntax	Copy()
Python Example	<pre>oEditor.Copy (["NAME:Selections", "Selections:=", ["CompInst@R;2;37", _ "CompInst@C;1;1"]])</pre>

CopyData [Schematic Editor]

Use: Copy graphic object or component properties for pasting.

Command: Edit> Copy Data (or Right-click on graphic object or component and select **Copy Data**)

Syntax: CopyData Array("NAME:Selections", "Selections:=", Array(<selections>))

Return Value: None.

Parameters: <selections>

Type: Comma-separated list of strings

Identifiers for each selected item to be copied, in the form CompInst@<id>, where id is the compname;ID;schematicID for each element.

CopySubdesign [Schematic Editor]

Copy components that represent subdesigns, as well as their represented designs, for pasting. The designs, when pasted, will be independent, with unique names based on the original design.

UI Access	(Right-click on hierarchical component) Copy as New Design		
Parameters	Name	Type	Description
	<selections>	Comma-separated list of strings	Identifiers for each selected item to be copied, in the form CompInst@<id>, where id is the compname;ID;schematicID for each element.
Return Value	None		

Python Syntax	CopySubdesign (["NAME:Selections", "Selections:=", [<selections>]])		
Python Example	<pre>oEditor.CopySubdesign (["NAME:Selections", "Selections:=", ["CompInst@Simplorer2;23;17"]])</pre>		

CreatePage (Schematic Editor)

Create a page at the end of the existing pages

UI Access	Schematic>New Page		
Parameters	Name	Type	Description
	<PageName>	String	Name of the page that has been created
	<PageNum>	Integer	Page Number
Return Value	None		

Note:

For more information about the component name, schematic IDs, and formats, see the section [Editor Scripting IDs](#).

Python Syntax	CreatePage(string name, // Page name [out,retval] int num) // Page number
Python Example	oEditor.CreatePage ("TestPage1")

Cut (Schematic Editor)

Cut Page Selection

UI Access	Edit>Cut		
Parameters	Name	Type	Description
	<Selections>	Array	Array("NAME:Selections", _ "Page:=", page number, _ // [Opt=1] Page number "Selections:=", IDs to modify))
Return Value	None		

Note:

For more information about the component name, schematic IDs, and formats, see the section [Editor Scripting IDs](#).

Python Syntax	Cut()

Python Example	<code>oEditor.Cut (["NAME:Selections", "Selections:=", ["CompInst@R;10;93"]])</code>

DeactivateOpen (Schematic Editor)

Disable items and replace with an open circuit

UI Access	Edit>Deactivate (Open)		
Parameters	Name	Type	Description
	<Selections>	Array	Array("NAME:Selections", _ "Page:=", page number, _// [Opt=1] Page number "Selections:=", IDs to modify))
Return Value	None		

Note:

For more information about the component name, schematic IDs, and formats, see the section [Editor Scripting IDs](#).

Python Syntax	DeactivateOpen()
Python Example	<code>oEditor.DeactivateOpen (["NAME:Selections", "Selections:=", ["CompInst@R;2;37"]])</code>

DeactivateShort (Schematic Editor)

Disable items and replace with a closed circuit

UI Access	Edit>Deactivate (Short)		
Parameters	Name	Type	Description
	<Selections>	Array	Array("NAME:Selections", _ "Page:=", page number, _ // [Opt=1] Page number "Selections:=", IDs to modify))
Return Value	None		

Note:

For more information about the component name, schematic IDs, and formats, see the section [Editor Scripting IDs](#).

Python Syntax	DeactivateShort ()
Python Example	oEditor.DeactivateShort(["NAME:Selections", "Selections:=", [_ "CompInst@R;2;37"]])

Delete (Schematic Editor)

Delete items

UI Access	Edit>Delete

Parameters	Name	Type	Description
	<Selections>	Array	Array("NAME:Selections", _ "Page:=", page number, _ // [Opt=1] Page number "Selections:=", IDs to modify))
Return Value	None		

Note:

For more information about the component name, schematic IDs, and formats, see the section [Editor Scripting IDs](#).

Python Syntax	Delete()
Python Example	<pre>oEditor.Delete(["NAME:Selections", "Selections:=", ["CompInst@R;10;93"]])</pre>

DeletePage (Schematic Editor)

To reduce the number of pages in the schematic

UI Access	Schematic>Remove Page		
Parameters	Name	Type	Description
	None		
Return Value	None		

Note:

For more information about the component name, schematic IDs, and formats, see the section [Editor Scripting IDs](#).

Python Syntax	DeletePage(<pagenum>, <pagenum>*)
Python Example	num = oEditor.DeletePage([1, 4])

ElectricRuleCheck (Schematic Editor)

Check Electric Rule

UI Access	Schematic>Electric Rule Check		
Parameters	Name	Type	Description
	<ElectricRules>	Array	Array(string, ...), bool Array Array of strings containing names of electric rules to perform. Valid names are: "PinRule": Checks for components with unconnected pins "OutputPinRule": Checks for nets with more than one output pin bool True if this should check all subcircuits False if this should just check this circuit

Return Value	None

Note:

For more information about the component name, schematic IDs, and formats, see the section [Editor Scripting IDs](#).

Python Syntax	ElectricRuleCheck()
Python Example	<pre>oEditor.ElectricRuleCheck(["PinRule", "OutputPinRule", "OverlapCompRule"], True)</pre>

ExportImage (Schematic Editor)

To export a picture for a specified page of the current design to a file. The image size can also be specified. The filename extension determines the type of image exported.

UI Access	None.		
Parameters	Name	Type	Description
	<filename>	String	The name of the file, with format-specific extension. Extensions supported are: bmp, gif, jpg, jpeg, png, tif, tiff.
	<pagenum>	Integer	The page number of the current schematic. Page numbers start at 1. If the number is out of range, the current schematic page will be printed.
	<dx>	Integer	The width of the image. If dx is less than 160, 160 will be used for the width.
	<dy>	Integer	The height of the image. If dy is less than 160, 160 will be used for the height.

Return Value	None

Note:

For more information about the component name, schematic IDs, and formats, see the section [Editor Scripting IDs](#).

Python Syntax	ExportImage (<filename>, <pagenum>, <dx>, <dy>)
Python Example	<code>oEditor.ExportImage ("c:\mysch.png", 1, 800, 400)</code>

ExportNetlist

Exports a netlist solution.

UI Access	None		
Parameters	Name	Type	Description
	<solution>	String	The Circuit solution in the design. For Maxwell Circuit, the solution must be an empty string, "".
	<fileName>	String	File and path to the netlist file. For Maxwell Circuit, the file name must be a SPH file.
Return Value	None.		

Python Syntax	ExportNetlist(<setupName>,<fileName>)
Python Example	<code>oDesign.ExportNetlist("", "SolutionData.sph")</code>

FindElements (Schematic Editor)

Select elements based on properties

UI Access	Edit>Find Elements		
Parameters	Name	Type	Description
	<propname>	String	The name of the property, or "" to search through all properties
	<propvalue>	String	The value of the property, or a substring
	<criterion>	Integer	0 = Contains 1 = Does not contain 2 = Exact match
	<types>	Integer	2 - components only 4 - graphic objects only 8 - wires only 16 - ports only Add the numbers to look through more types - 30 is a common number, searching through everything
	<match>	Boolean	True = match all of the properties specified False = match any of the properties specified
	<case>	Boolean	True = case-sensitive False = not case-sensitive
	<sub>	Boolean	True = search subcircuits as well as current schematic False = only search current schematic

	<table><tr><td><only></td><td>Boolean</td><td>True = search within current selections False = search schematic(s)</td></tr></table>			<only>	Boolean	True = search within current selections False = search schematic(s)
<only>	Boolean	True = search within current selections False = search schematic(s)				
Return Value	Array of strings, the names of the elements selected					

Note:

For more information about the component name, schematic IDs, and formats, see the section [Editor Scripting IDs](#).

Python Syntax	FindElements(["NAME:SearchProps", "Prop:=", [<propname>, <propvalue>, <criterion>], [<propname>, <propvalue>, <criterion>]]*, ["NAME:Parameters", "Filter:=", <types>, "MatchAll:=", <match>, "MatchCase:=", <case>, "SearchSubCkt:=", <sub>, "SearchSelectionOnly:=", <only>])
Python Example	<pre>oEditor.FindElement(["NAME:SearchProps", "Prop:=", ["CompName", "", 0]], ["NAME:Parameters", "Filter:=" , 30, "MatchAll:=" , False, "MatchCase:=" , False, "SearchSubCkt:=" , True,</pre>

	<pre>"SearchSelectionOnly:=" , False])</pre>

FitToBorder [Schematic Editor]

Use: To reset the schematic view to just include the page border, if any.

UI Access	View>Fit Border		
Parameters	Name	Type	Description
	None		
Return Value	None		

Python Syntax	FitToBorder()
Python Example	<pre>oEditor.FitToBorder()</pre>

GridSetup (Schematic Editor)

Changes the settings on the schematic grid

UI Access	Schematic>Grid Setup

	Name	Type	Description
	<Options>	Array	<pre>Array("NAME:Options", _ "MajorGrid:=", string, _ // Major grid size with units "Divisions:=", int, _ // Number of minor grid divisions "MajorColor:=", int, _ // RGB Color of major grid lines "MinorColor:=", int, _ // RGB Color of minor grid lines "ShowGrid:=", bool, _ // Should the grid be shown? "SnapToGrid:=", bool, _ // Should objects snap to grid? "BackgroundColor:=", int, _ // RGB Color of the background "SaveAsDefault:=", bool))</pre>
Parameters			
Return Value	None		

Note:

For more information about the component name, schematic IDs, and formats, see the section [Editor Scripting IDs](#).

Python Syntax	GridSetup()
Python Example	<pre>oEditor.GridSetup(["NAME:Options", _ "MajorGrid:=", "10mm", _ "Divisions:=", 10, _</pre>

	<pre>"MajorColor:=", 0x00ff0000, _ # Red "MinorColor:=", 0x0000ff00, _ # Green "ShowGrid:=", true, _ "SnapToGrid:=", true, _ "BackgroundColor:=", 0x00ffffff, _ # White "SaveAsDefault:=", false])</pre>

FlipHorizontal (Schematic Editor)

Flip items about the horizontal (x) axis

UI Access	NA		
Parameters	Name	Type	Description
	<Selections>	Array	Array("NAME:Selections", _ "Page:=", page number, _ // [Opt=1] Page number "Selections:=", IDs to modify))
	<FlipParameters>	Array	Array("NAME: FlipParameters", _ "Disconnect:=", bool _ // [Opt=0] Should wires disconnect "Rubberband:=", bool) _ // [Opt=1] Should wires staircase
Return Value	None		

Note:

For more information about the component name, schematic IDs, and formats, see the section [Editor Scripting IDs](#).

Python Syntax	FlipHorizontal(
Python Example	<pre>oEditor.FlipHorizontal(["NAME:Selections", "Selections:=", [_ "CompInst@R;2;4", "CompInst@C;1;1"]], ["NAME:FlipParameters", "Disconnect:=", true, "Rubberband:=", false])</pre>

FlipVertical (Schematic Editor)

Flip items about the vertical (y) axis

UI Access	NA		
Parameters	Name	Type	Description
	<Selections>	Array	<pre>["NAME:Selections", "Page:=", page number, _ // [Opt=1] Page number "Selections:=", IDs to modify]</pre>
	<FlipParameters>	Array	<pre>["NAME:FlipParameters", "Disconnect:=", bool // [Opt=0] Should wires disconnect</pre>

			"Rubberband:=", bool] // [Opt=1] Should wires staircase
Return Value	None		

Note:
For more information about the component name, schematic IDs, and formats, see the section [Editor Scripting IDs](#).

Python Syntax	FlipVertical(<Selections>, <FlipParameters>)
Python Example	<pre>oEditor.FlipVertical (["NAME:Selections", "Selections:=", ["CompInst@R;2;4", _ "CompInst@C;1;1"]], ["NAME:FlipParameters", "Disconnect:=", true, "Rubberband:=", false])</pre>

Move (Schematic Editor)

Move items

UI Access	NA		
Parameters	Name	Type	Description
	<Selections>	Array	Array("NAME:Selections", _

			"Page:=", page number, _ // [Opt=1] Page number "Selections:=", IDs to modify))
	<MoveParameters>	Array	Array("NAME:MoveParameters", _ "xdelta:=", double _ // X distance to move "ydelta:=", double _ // Y distance to move "Disconnect:=", bool _ // [Opt=0] Should wires disconnect "Rubberband:=", bool) _ // [Opt=1] Should wires staircase
Return Value	None		

Note:

For more information about the component name, schematic IDs, and formats, see the section [Editor Scripting IDs](#).

Python Syntax	Move()
Python Example	oEditor.Move (["NAME:elements", "circle_0", "rect_2"], [0.0165613577023499, -0.001])

NameNets (Schematic Editor)

Change names of nets on the schematic

UI Access	Schematic>Auto-Name Wires

Parameters	Name	Type	Description
	None		
Return Value	None		

Note:

For more information about the component name, schematic IDs, and formats, see the section [Editor Scripting IDs](#).

Python Syntax	<pre>NameNets(bool display, // Show Net names or not ["NAME:NetNames",_ "string:=", _ #Old name of the net (e.g. net_1) string, _ #New name of the net (e.g. my_net) ...]) # Repeat for additional nets to change</pre>
Python Example	<pre>oEditor.NameNets (True, ["NAME:NetNames", "Wire@net_2;41:1:=", "A[0]"])</pre>

PageBorders (Schematic Editor)

Note:

This command has been replaced with the [SetPageData](#) command. Legacy scripts using this command will play back for the first page of a schematic only.

Sets up visible page borders on the schematic

```
PageBorders (
Array("NAME:Options", _
"PageSize:=", _
Array("x:=", string, _ // Width of page border with units
"y:=", string), _ // Height of page border with units
"PageMargins:=", _
Array("x:=", string, _ // Margin Width with units
"y:=", string), _ // Margin Height with units
"ZonesHoriz:=", int, _ // Number of Horizontal zones
"ZonesVert:=", int)) // Number of Vertical zones
```

Note:

For more information about the component name, schematic IDs, and formats, see the section [Editor Scripting IDs](#).

Pan (Schematic Editor)

Pan Display

UI Access	View>Pan		
Parameters	Name	Type	Description
	None		
Return Value	None		

Note:

For more information about the component name, schematic IDs, and formats, see the section [Editor Scripting IDs](#).

Python Syntax	<pre>Pan([double, _ // Move the visible area by X meters double]) // Move the visible area by Y meters</pre>
Python Example	<pre>oDefinitionEditor.Pan ([-0.00922653869663931, 0.00686839904222147])</pre>

Paste (Schematic Editor)

Paste copied items

UI Access	Edit>Paste		
Parameters	Name	Type	Description
	None		

Return Value	None

Note:

For more information about the component name, schematic IDs, and formats, see the section [Editor Scripting IDs](#).

Python Syntax	Paste(VARIANT attrs)# specified attributes
Python Example	<pre>oEditor.Paste (["NAME:Attributes", "Page:=", 1, "X:=", -0.059131200000001, "Y:=", _ 0.030784799999999, "Angle:=", 0, "Flip:=",false])</pre>

PasteData [Schematic Editor]

Use: Paste graphic object or component properties to another object or component.

Command: Edit> Paste Data (or Right-click on graphic object or component and select **Paste Data**)

Syntax: PasteData Array("NAME:Selections", "Selections:=", Array(<selections>), "SelectList:=", Array(<select_prop_display>, <select_parameters>), "Exclude:=", Array(<ExcludeList>))

Return Value: None.

Parameters: Selections

Type: Comma separated component instance list where user wants to paste data

Identifiers for each selected item to be copied, in the form Complnst@<id>, where id is the compname;ID;schematicID for each element.

SelectList

Type: Comma separated Boolean list where:

<select_prop_display> true to paste property displays

<select_parameters> true to paste parameters

<ExcludeList>

Type: Comma separated list of property names to exclude from being pasted.

PasteDesign (Schematic Editor)

Paste a design that has already been copied to the clipboard into schematic as a subdesign.

Syntax:

```
PasteDesign(  
    pasteOption , // see below for values  
    Array("NAME:Attributes", _  
        "Page:=", int, _ // [Opt=1] Page number (one-based)  
        // Note: Page 1 always exists  
        "X:=", double, _ // X position of the object  
        "Y:=", double, _ // Y position of the object  
        "Angle:=", double, _ // Rotation angle (radians)  
        "Flip:=", bool) // True if mirrored  
    )
```

Return Value: Component is created and the returned value has the format CompInst@<CompName>;<compInstID>;<optional schematicID>

For instance a return value `CompInst@CAP_;4;35` indicates that the component name is `CAP_`, its ID is 5, and its Schematic ID is 35. These details can be found in the Properties window when you select the created component. The component name and ID can be found in the Gen-eral tab and the Schematic ID can be found in the Symbols tab.

Parameters: `pasteOption` should be one of the following:

0 to link to the existing design that was copied

1 to create a new copy of the design that was copied but keep the original layers of the design being copied

2 to create a new copy of the design and merge the layers of the design being copied into the design receiving the copy

PushExcitations

Allows access to computed excitations for transient and linear frequency solutions. The script command can be accessed from three locations.

UI Access	- Layout Editor - Schematic Editor - Select a Nexxim solution in a 3D Layout design, right click, and choose Push Excitations.		
Parameters	Name	Type	Description
	<Reference Designation>	String	"<refdes>" This argument is empty when excitations are pushed from a Nexxim solution.
	<PushExcitations Parameters>	Array	This parameter changes depending on whether the excitations comes from a transient or linear frequency solution. The keyword " <code>transient:="</code> indicates to AEDT which solution generated the excitations. If " <code>transient:="</code> is present, " <code>CalcThevenin =</code> " and its value are ignored. For a linear frequency solution, use this array:

		<pre>Array("NAME:options", "CalcThevenin =" , <true false>, "Sol:=", "<solution name>")</pre> If CalcThevenin is true, Thevenin's equivalent is calculated. Parameters for the linear frequency solution do not include a Freq argument, so all frequencies from the solution are used. For a transient solution, use: <pre>Array("NAME:options", "transient:=", Array("start:=", <start time>, "stop:=", <stop time>, "maxHarmonics:=", <max harmonics>, "winType:=", <window>, ["widthPct:=", <width percentage>], ["kaiser:=", <Kaiser value>], ["correctCoherentGain:=", true]), "Sol:=", "<solution name>")</pre> winType can have the following values: • Rectangular • Bartlett • Blackman

			• Hamming • Hanning • Kaiser • Welch • Weber • Lanczos
Return Value	None		

Note:

For more information about the component name, schematic IDs, and formats, see the section [Editor Scripting IDs](#).

Rotate (Schematic Editor)

Rotate items

UI Access	Draw>Rotate		
Parameters	Name	Type	Description
	<Selections>	Array	<pre>Array("NAME:Selections", _ "Page:=", page number, _ // [Opt=1] Page number "Selections:=", IDs to modify))</pre>
	<RotateParameters>	Array	<pre>Array("NAME:RotateParameters", _ "Disconnect:=", bool _ // [Opt=0] Should wires disconnect "Rubberband:=", bool _ // [Opt=1] Should wires staircase "Angle:=", double) // [Opt=/2] Angle to rotate</pre>

			Note: Rotation occurs around center of ids
Return Value	None		

Note:

When you record a script, the schematic editor's Rotate command shows the rotation angle for some older scripts in radians, while newer scripts show rotation angle in degrees.

```
oEditor.Rotate Array("NAME:Selections", "Selections:=", _  
Array("Complnst@RES_1;1:1")), Array("NAME:RotateParameters", _  
"Angle:=", 1.5707963267949, "Disconnect:=", false, "Rubberband:=", false)
```

If "Degrees=xxx" is not shown, the rotation must be scaled.

Note:

For more information about the component name, schematic IDs, and formats, see the section [Editor Scripting IDs](#).

Python Syntax	Rotate()

Python Example	<pre>oEditor.Rotate (["NAME:Selections", "Selections:=", ["CompInst@R;2;4", "CompInst@C;1;1"]], ["NAME:RotateParameters", "Degrees:=", 90, "Disconnect:=", _ false, "Rubberband:=", false])</pre>

SelectAll (Schematic Editor)

Select all elements on the given page.

UI Access	Edit>Select All		
Parameters	Name	Type	Description
	None		
Return Value	None		

Note:

For more information about the component name, schematic IDs, and formats, see the section [Editor Scripting IDs](#).

Python Syntax	SelectAll(int pageNum) # Page number The first page number is 1. If an invalid page number is passed in (i.e. 0, -1, etc), SelectAll will use the active page on the currently active view. If there is no active view, the first page, page 1, is used.
Python Example	<pre>oEditor.SelectAll(1)</pre>

SelectPage (Schematic Editor)

Select a page in the UI

UI Access	NA		
Parameters	Name	Type	Description
	None		
Return Value	None		

Note:

For more information about the component name, schematic IDs, and formats, see the section [Editor Scripting IDs](#).

Python Syntax	SelectPage(int page) # Page number
Python Example	<code>oEditor.SelectPage(1)</code>

SetPageData [Schematic Editor]

Note:

This command replaces the [PageBorders](#) command.

To add or remove page border, title block and page property data from the specified page. Title blocks will not be shown if the page border type is "None". Any number of title blocks may be specified, as well as any number of page properties. Some of the page properties are special and their values will be updated automatically as appropriate, such as Project, ProjectPath, Design, Number and Date.

UI Access	Schematic>Page Borders and Title Blocks...		
Parameters	Name	Type	Description
	<bordertype>	String	One of: "None", "Outline", "ANSI", "ISO", "DIN"
	<sizeX>	Double (with units)	Size of X axis
	<sizeY>	Double (with units)	Size of Y axis
	<marginX>	Double (with units)	Margin of X axis
	<marginY>	Double (with units)	Margin of Y axis
	<zonesH>	Integer	
	<zonesV>	Integer	
	<symbolName>	String	The name of the title block symbol.
	<libName>	String	The name of the library (with path from library directory root) that contains the title block symbol
	<Hsnap>	String	One of: "Left", "Right", "Center", "None"
	<Vsnap>	String	One of: "Top", "Bottom", "Center", "None"
	<rotation>	String	One of "0", "90", "270"
	<propName>	String	The name of a page property followed by "[:="
	<propVal>	String	The value of a page property
Return Value	None		

Python Syntax	<pre>SetPageData (<pagenum>, ["NAME:Border", "BorderType:=", <bordertype>, "PageSize:=", ["x:=", <sizeX>, "y:=", <sizeY>], "PageMargins:=", ["x:=", <marginX>, "y:=", <marginY>], "ZonesHoriz:=", <zonesH>, "ZonesVert:=", <zonesV>], ["NAME:TitleBlocks"[, "TitleBlock:=", [<symbolName>, <libName>, <Hsnap>, <Vsnap>, <rotation>, -1]]*], ["NAME:PageProps"[, <propName>, <propVal>]*])</pre>
Python Example	<pre>oEditor.SetPageData (1, ["NAME:Border", "BorderType:=", "ANSI", "PageSize:=", ["x:=", "11in", "y:=", "8.5in"], "PageMargins:=", ["x:=", "0.38in", "y:=", "0.25in"], "ZonesHoriz:=", 2, "ZonesVert:=", 2], ["NAME:TitleBlocks", "TitleBlock:=", ["TitleBlk1", "TitleBlocks", "Right", "Bottom", "0", -1]], ["NAME:PageProps", "Title:=", "Design Title", "Author:=", "I. M. User", "Number:=", "1", "Project:=", "Project8", "ProjectPath:=", "e:/projects/", "Design:=", "TwinBuilder1", "Date:=", "9:53:11 PM Jan 27, 2016"])</pre>

SendToBack

Send the selected object to the back of the view

UI Access	Draw>Send To Back		
Parameters	Name	Type	Description
	<Object>	String	Object to send to the back.
Return Value	None		

Note:

For more information about the component name, schematic IDs, and formats, see the section [Editor Scripting IDs](#)

Python Syntax	SendToBack(["NAME:Selections", "Selections:=", [<Object>, <Object>, ...]])
Python Example	<pre>oDefinitionEditor.SendToBack(["NAME:Selections", "Selections:=", ["SchObj@10"]])</pre>

ShowVariableBlock (Schematic Editor)

Use: Control display of Variable Text Block in Schematic Editor.

Command: Schematic Ribbon > Settings > Design Variables

Syntax: ShowVariableBlock("True") // Display Text Block

 ShowVariableBlock("False") // Hide Text Block

Return Value: None.

```
Parameters: True // Display Text Block
            False // Hide Text Block
```

Python Syntax	ShowVariableBlock ()
Python Example	<pre>oEditor.ShowVariableBlock [("True")]</pre>

SortComponents (Schematic Editor)

Sorts all components, or a block of components, using the specified method

UI Access	Schematic>Sort Components		
Parameters	Name	Type	Description
	<Type>	Array	<pre>Array(type, _ // string specifying sort type method)) // string specifying sort method type // string argument specifying sort type: "All Components" or "Blocks" // An empty string corresponds to "All Components" method // string argument specifying sort method: "By Name", "Left to Right", or "Signal Flow"</pre>
Return Value	None		

Note:

For more information about the component name, schematic IDs, and formats, see the section [Editor Scripting IDs](#).

Python Syntax	SortComponents ()
Python Example	<code>oEditor.SortComponents ()</code>

Wire (Schematic Editor)

Wire specified pairs of pins in two selected component instances.

UI Access	Select two component instances in the Schematic Editor, then right click on one and choose Wire .		
Parameters	Name	Type	Description
	<code><firstComponent></code>	String	Name of the first component to be wired
	<code><firstListOfPins></code>	String	List of pins separated by spaces
	<code><secondComponent></code>	String	Name of second component the first component will be wired to
	<code><secondListOfPins></code>	String	List of pins on the second component, separated by spaces
Return Value	None		

Python Syntax	<code>Wire(<firstComponent>,<firstListOfPins>,<secondComponent>,<secondListOfPins >)</code>
Python Example	<code>oEditor.Wire("CompInst@etch_sline_s20_z70_1;10;76", "A1 A2", "CompInst@CX0805MRNPO9BB220;1;1", "B1 B2")</code>

ZoomArea (Schematic Editor)

Zoom Area

UI Access	View>Zoom Area		
Parameters	Name	Type	Description
	<Size>	Array	Array(double, _ //Base X value of the area to zoom into double), _ // Base Y value of the area to zoom into Array(double, _ // X size of the area to zoom double)) // Y Size of the area to zoom NOTE: All values in meters
Return Value	None		

Note:

For more information about the component name, schematic IDs, and formats, see the section [Editor Scripting IDs](#).

Python Syntax	ZoomArea()
Python Example	<pre>oEditor.ZoomArea ([0.00307569800375911, 0.021101611633739], [-0.050433530091516, -0.0220260723256825])</pre>

ZoomIn (Schematic Editor)

Zoom into the schematic

UI Access	View>Zoom In		
Parameters	Name	Type	Description
	None		
Return Value	None		

Note:

For more information about the component name, schematic IDs, and formats, see the section [Editor Scripting IDs](#).

Python Syntax	ZoomIn()
Python Example	<code>oEditor.ZoomIn()</code>

ZoomOut (Schematic Editor)

Zoom out from the schematic

UI Access	View>Zoom Out		
Parameters	Name	Type	Description
	None		
Return Value	None		

Note:

For more information about the component name, schematic IDs, and formats, see the section [Editor Scripting IDs](#).

Python Syntax	ZoomOut()
Python Example	<code>oEditor.ZoomOut()</code>

ZoomPrevious (Schematic Editor)

Restore the zoom to the previous settings

UI Access	NA		
Parameters	Name	Type	Description
	None		
Return Value	None		

Note:

For more information about the component name, schematic IDs, and formats, see the section [Editor Scripting IDs](#).

Python Syntax	ZoomPrevious()

Python Example	<code>oEditor.ZoomPrevious()</code>

ZoomToFit (Schematic Editor)

Set the Zoom to Fit

UI Access	View>Fit Drawing		
Parameters	Name	Type	Description
	None		
Return Value	None		

Note:

For more information about the component name, schematic IDs, and formats, see the section [Editor Scripting IDs](#).

Python Syntax	<code>ZoomToFit()</code>
Python Example	<code>oEditor.ZoomToFit()</code>

Property Method List

This section presents the property script methods that are available.

[ChangeProperty \(Schematic Editor\)](#)

[GetEvaluatedPropertyValue \(Schematic Editor\)](#)

[GetProperties \(Schematic Editor\)](#)

[GetPropertyAttribute \[Schematic Editor\]](#)

[GetPropertyValue \(Schematic Editor\)](#)

[SetPropertyValue \(Schematic Editor\)](#)

ChangeProperty (Schematic Editor)

Change a property

UI Access	NA		
Parameters	Name	Type	Description
	Name	Type	Description
Return Value	Value		

Python Syntax	ChangeProperty()
Python Example	<pre>oEditor.ChangeProperty(["NAME:AllTabs", ["NAME:ComponentTab", ["NAME:PropServers", _ "Wire@net_1;14:1"], ["NAME:ChangedProps", ["NAME:NetName", "ExtraText:=", _ "net_10", "Name:=", "net_10", "SplitWires:=", false]]])</pre>

GetEvaluatedPropertyValue (Schematic Editor)

Get value.

Command: None.

Syntax: GetEvaluatedPropertyValue<tabDescription, componentID, propName>

Return Value: Evaluated value of variable property in double format.

Parameters: tabDescription = name of property tab where property is found

componentID = id of component instance where property is found,

in the format: "CompInst@<name of component type>;<id of compInstance>"

propName = name of the variable property

Notes:

1. This function is only available with the schematic editor.
2. Calling this function on non-numeric properties (e.g. Text properties) returns 0.

GetProperties (Schematic Editor)

To get the names of properties for a specified tab and element.

UI Access	NA		
Parameters	Name	Type	Description
	<tab>	String	One of six choices, "PassedParameterTab", "ComponentTab", "Component", "BaseElementTab", "Quantities", "Signals"
	<objectId>	String	Instance ID of object with the property
Return Value	Array of strings, the property names		

Python Syntax	GetProperties(<tab>, <objectId>)

Python Example	<pre>comparray = oEditor.GetAllComponents() for i in comparray: instval = oEditor.GetPropertyValue ("PassedParameterTab",i,"InstanceName") proparray = oEditor.GetProperties ("Quantities",i) for j in proparray: direction = oEditor.GetPropertyAttribute ("Quantities",i, j, "Direction") AddInfoMessage(str(j) + " = (direction) " + str(direction))</pre>

GetPropertyAttribute [Schematic Editor]

To find the value of an attribute of a specified property.

UI Access	NA		
Parameters	Name	Type	Description
	<tab>	String	One of six choices, "PassedParameterTab", "ComponentTab", "Component", "BaseElementTab", "Quantities", "Signals"
	<objectId>	String	Instance ID of object with the property

	<propName>	String	The name of the property
	<attribute>	String	Currently, only the "Direction" attribute is supported. Return values will be one of "In", "Out" or "InOut"
Return Value	A string, the attribute's value.		

Python Syntax	GetPropertyAttribute(<tab>, <objectId>, <propName>, <attribute>)
Python Example	<pre> comparray = oEditor.GetAllComponents() for i in comparray: instval = oEditor.GetPropertyValue ("PassedParameterTab",i,"InstanceName") proparray = oEditor.GetProperties ("Quantities",i) for j in proparray: direction = oEditor.GetPropertyAttribute ("Quantities",i, j, "Direction") AddInfoMessage(str(j)+ " = (direction) " + str(direction)) </pre>

GetPropertyValue (Schematic Editor)

Get a property

UI Access	NA		
Parameters	Name	Type	Description

	<tab>	String	Tab with the property
	<item>	String	Name of the object
	<propname>	String	Name of the property
	<value>	String	Value of the property
Return Value	None		

Python Syntax	GetPropertyValue(<tab>,<item>,<propname>,<value>)
Python Example	None

SetPropertyValue (Schematic Editor)

Set a property.

UI Access	NA		
Parameters	Name	Type	Description
	<tab>	String	Tab with the property
	<item>	String	Name of the objects
	<propname>	String	Name of the property
	<value>	String	The new value of the property
Return Value	None		

Python Syntax	SetPropertyValue(<params>)

Python Example	None

Information Method List

This section presents the information methods that are available.

[GetCompInstanceFromRefDes \(Schematic Editor\)](#)

[GetComponentInfo \(Schematic Editor\)](#)

[GetComponentPins \(Schematic Editor\)](#)

[GetComponentPinInfo \(Schematic Editor\)](#)

[GetComponentPinLocation \(Schematic Editor\)](#)

[GetEditorName \(Schematic Editor\)](#)

[GetNetConnections \(Schematic Editor\)](#)

[GetNumPages \(Schematic Editor\)](#)

[GetPortInfo \(Schematic Editor\)](#)

[GetSelections \(Schematic Editor\)](#)

[GetSignals \(Schematic Editor\)](#)

GetAllPorts

Get all ports in a design.

UI Access	NA		
Parameters	Name	Type	Description
	None		

Return Value	Ports in the design

Python Syntax	<code>GetAllPorts()</code>
Python Example	<code>oEditor.GetAllPorts()</code>

GetCompInstanceFromRefDes (Schematic Editor)

Informational

UI Access	NA		
Parameters	Name	Type	Description
	Component	String	Name of the Component
Return Value	Returns IDispatch for CompInstance		

Python Syntax	<code>GetCompInstanceFromRefDes(string)</code> # string is refDes for the component
Python Example	<code>GetCompInstanceFromRefDes ("R1 ")</code>

GetComponentInfo (Schematic Editor)

Informational

UI Access	NA		
Parameters	Name	Type	Description
	<compID>	String	The Schematic component's ID, obtained from GetSelections or through FindElements
Return Value	Array of Strings, as follows: "Page= <i>number</i> " "ComponentName= <i>string</i> " "GateCount= <i>number</i> " "LocationX= <i>number</i> " "LocationY= <i>number</i> " "BBoxLLx= <i>number</i> " "BBoxLLy= <i>number</i> " "BBoxURx= <i>number</i> " "BBoxURy= <i>number</i> " "Angle= <i>number</i> " (in degrees) "Flip=true or false"		

Python Syntax	GetComponentInfo(<compID>)
Python Example	For i in selectionArray:

```
compArray = oEditor.GetComponentInfo(i)

For Each k in compArray:
    AddInfoMessage(str(k) +
        "Got with GetComponentInfo")
```

GetComponentPins (Schematic Editor)

Informational

UI Access	NA		
Parameters	Name	Type	Description
	<compID>	String	The Schematic component's ID, obtained from GetSelections or through FindElements.
Return Value	Array of strings, which are the names of all the component's pins		

Python Syntax	GetComponentPins(<compID>)
Python Example	<pre>selectionArray = oEditor.GetSelections() for i in selectionArray pinArray = oEditor.GetComponentPins(i) for j in pinArray</pre>

	AddInfoMessage("Pin: " + str(j))

GetComponentPinInfo (Schematic Editor)

Informational - gets details about a pin on a schematic component.

UI Access	NA		
Parameters	Name	Type	Description
	<compID>	String	The Schematic component's ID
	<pinName>	String	The pin name
Return Value	String values, as follows: "X=number" "Y=number" "Angle=number" (in degrees) "Flip=true or false" "WireId=netID or empty string" (netID is the identifier of the schematic net, and can be used in other informational methods)		

Python Syntax	GetComponentPinInfo(<compID>, <pinName>)
Python Example	<pre>info = oEditor.GetComponentPinInfo ("1", "N1")</pre>

GetComponentPinLocation [Schematic Editor]

Informational - gets the X or Y location of a pin on a schematic component. Invoke once with TRUE for "XorY" to get the X value of the pin's location in the schematic, and once with FALSE to get the Y value.

UI Access	NA		
Parameters	Name	Type	Description
	<compID>	String	The Schematic component's ID
	<pinName>	String	The pin name
	<XorY>	Boolean	TRUE gets the X value; FALSE gets the Y value.
Return Value	String values, as follows: "X= <i>number</i> " "Y= <i>number</i> "		

Python Syntax	GetComponentPinLocation <compID>, <pinname>, <XorY>
Python Example	<pre>varx = oEditor.GetComponentPinLocation (compid, "N1", TRUE)</pre>

GetEditorName (Schematic Editor)

Get the name of the schematic editor.

UI Access	N/A
Parameters	None
Return Value	String containing the name of the schematic editor. "SchematicEditor"

Python Syntax	GetEditorName()
Python Example	<code>oEditor.GetEditorName()</code>

GetNetConnections (Schematic Editor)

Use: Informational.

Command: None.

Syntax: GetNetConnections(<netName>)

Return Value: Array of strings containing the connections for the net identified by the

netName argument. Strings in the array are in one of four formats:

"IPort portID x y pinName"

"GPort portID x y pinName"

"PagePort portID x y pinName"

"Component componentName compID x y pinName"

where *IPort*, *GPort*, *PagePort*, and *Component* are key words

that indicate what the rest of the string specifies:

IPort indicates that *portID* specifies the name of the interface port,

GPort indicates that *portID* specifies the name of the global port,

PagePort indicates that *portID* specifies the name of the page port,

Component indicates that *compID* specifies the component instance identifier,

x and *y* are the connection point,

pinName is the name of the connected pin,

componentName is the name of the component.

Parameters: <netName>

Type: String

The name of the net. Specifying "0" as the netName will return all objects connected to ground including all ground ports.

GetNumPages [Schematic Editor]

To determine how many pages there are in the schematic.

UI Access	NA		
Parameters	Name	Type	Description
	None		
Return Value	Integer, the number of pages in the schematic.		

Python Syntax	GetNumPages()
Python Example	<pre>num = oEditor. GetNumPages()</pre>

GetPortInfo (Schematic Editor)

Informational – get details about a port on a schematic component.

UI Access	NA		
Parameters	Name	Type	Description
	<portID>	String	The Schematic port's ID, obtained from GetSelections or through FindElements.
Return Value	Array of strings, as follows: "Name=string" // Name of the port. "X=val" // Connection point X location. "Y=val" // Connection point Y location. "Angle=val" // Component angle. "Flip=true or false" // Whether the component is flipped or not. "WireId=string" // The identifier of the schematic net, can be used in other informational methods.		

Python Syntax	GetPortInfo(<portID>)
Python Example	<pre>selectionArray = oEditor.GetSelections() for i in selectionArray:</pre>

	<pre>info = oEditor.GetPortInfo(i) for j in info: AddInfoMessage(str("GetPortInfo: " + j))</pre>

GetSelections (Schematic 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>

GetSignals (Schematic Editor)

Informational

UI Access	NA		
Parameters	Name	Type	Description

	<table><tr><td><wirename></td><td>String</td><td>The wire's name, as seen in the UI, obtained in script through parsing return data from <code>GetWireInfo</code>.</td></tr></table>	<wirename>	String	The wire's name, as seen in the UI, obtained in script through parsing return data from <code>GetWireInfo</code> .
<wirename>	String	The wire's name, as seen in the UI, obtained in script through parsing return data from <code>GetWireInfo</code> .		
Return Value	Array of strings which contains the signal names in the <i>wirename</i> argument. The <i>wirename</i> "Net1" would return an array with a single string "Net1". The <i>wirename</i> "WW[0-3],A,B" would return an array with six strings "WW[0]", "WW[1]", "WW[2]", "WW[3]", "A", :B".			

Python Syntax	GetSignals(<wirename>)
Python Example	<pre>selectionArray = oEditor.GetSelections() for i in selectionArray: netArray3 = oEditor.GetWireInfo(i) for n in netArray3: wname = n[0:9] if (wname == "WireName:"): nn = len(n) wn = n[nn-10:] AddInfoMessage("Wirename, parsed from ID :" + str(wn)) sigs = oEditor.GetSignals(wn) for ll in sigs: AddInfoMessage("Element in array</pre>

	<code>from GetSignals : " + str(ll))</code>

GetWireConnections (Schematic Editor)

Informational.

UI Access	NA		
Parameters	Name	Type	Description
	None		
Return Value	Array of strings containing the connections for the wire identified by the wireID argument. The strings in the array are in one of four formats: "InterfacePort=portname portID x,y" "GlobalPort=portname portID x,y" "PagePort=portname portID x,y" "ComponentPin=compld pinname x,y" where x,y is the connection point.		

Python Syntax	<code>GetWireConnections(<wireID>)</code>
Python Example	<pre>selectionArray = oEditor.GetSelections() for i in selectionArray: AddInfoMessage("Selected element" + i)</pre>

	<pre>netArray = oEditor.GetWireConnections(i) for m in netArray: AddInfoMessage("GetWireConnections" + str(m))</pre>

GetWireInfo (Schematic Editor)

Informational.

UI Access	NA		
Parameters	Name	Type	Description
	<wire id>	String	The Schematic wire’s ID, obtained from <code>GetSelections</code> or through <code>FindElements</code> .

Return Value	Array of strings containing the wire info in the wireID argument. The strings in the array are: "Page=number" "WireName=string" "SegmentCount=number" "IPortCount=number" "GPortCount=number" "PagePortCount=number" "PinCount=number" "BBoxLLx=val" "BBoxLLy=val" "BBoxURx=val" "BBoxURy=val"

Python Syntax	<code>GetWireInfo(<wireID>)</code>
Python Example	<pre>selectionArray = oEditor.GetSelections() for i in selectionArray: AddInfoMessage("Selected element" + str(i)) netArray2 = oEditor.GetWireInfo(i)</pre>

```
for n in netArray2:
    AddInfoMessage("GetWireInfo" + str(n))
```

GetWireSegments (Schematic Editor)

Informational.

UI Access	NA		
Parameters	Name	Type	Description
	<wireID>	String	The Schematic wire's ID, obtained from <code>GetSelections</code> or through <code>FindElements</code>
Return Value	Array of strings containing the wire segments in the wireID argument. The strings in the array are: "Segment=val,val val,val number" the "val" quantities are segment endpoints x1,y1 x2,y2 and the "number" is a schematic segment ID		

Python Syntax	GetWireSegments (<wireID>)
Python Example	<pre>selectionArray = oEditor.GetSelections() for i in selectionArray: AddInfoMessage("Selected element" + str(i)) netArray2 = oEditor.GetWireSegments(i) for n in netArray2:</pre>

	<code>AddInfoMessage("GetWireSegments" + str(n))</code>

27 - Definition Editor Script Commands

The Definition Editor controls the use of materials and scripts in a project. Symbol editor script commands and footprint editor script commands are accessed using the Definition Editor.

```
Set oDefinitionEditor = oProject.SetActiveDefinitionEditor("SymbolEditor", "MySymbol")
```

```
Set oDefinitionEditor = oProject.SetActiveDefinitionEditor("FootprintEditor", "MyFootprint")
```

The topics for this section include:

Also refer to the following subtopics:

[Symbol Editor Scripts](#)

Symbol Editor Scripts

Symbol editor script commands are accessed with a definition editor.

```
oDefinitionEditor = oProject.SetActiveDefinitionEditor("SymbolEditor", "MySymbol")
```

The symbol editor script commands are listed below.

[AddLevel \(Symbol Editor\)](#)

[AlignHorizontal \(Symbol Editor\)](#)

[AlignVertical \(Symbol Editor\)](#)

"BringToFront (Symbol Editor)" on page 27-5

[ChangeProperty \(Symbol Editor\)](#)

[CloseEditor \(Symbol Editor\)](#)

[CreateArc \(Symbol Editor\)](#)

[CreateCircle \(Symbol Editor\)](#)

[CreateImage \(Symbol Editor\)](#)

[CreateLine \(Symbol Editor\)](#)

[CreatePin \(Symbol Editor\)](#)

[CreatePolygon \(Symbol Editor\)](#)

[CreateRectangle \(Symbol Editor\)](#)

[CreateText \(Symbol Editor\)](#)

[Cut \(Symbol Editor\)](#)

[Delete \(Symbol Editor\)](#)

[DisplayPinNames \(Symbol Editor\)](#)

[ExportFile \(Symbol Editor\)](#)

[FlipHorizontal \(Symbol Editor\)](#)

[FlipVertical \(Symbol Editor\)](#)

[GetProperties \(Symbol Editor\)](#)

[GetPropertyValue \(Symbol Editor\)](#)

[GetVisibleLevels \(Symbol Editor\)](#)

[GridSetup \(Symbol Editor\)](#)

[Move \(Symbol Editor\)](#)

[Pan \(Symbol Editor\)](#)

[Paste \(Symbol Editor\)](#)

[Redo \(Symbol Editor\)](#)

[RemoveLevels \(Symbol Editor\)](#)

[RemovePort \(Symbol Editor\)](#)

[Rotate \(Symbol Editor\)](#)

[Save \(Symbol Editor\)](#)

[SelectAll \(Symbol Editor\)](#)

[SendToBack \(Symbol Editor\)](#)

[SetActiveLevel \(Symbol Editor\)](#)

[SetInitialLevels \(Symbol Editor\)](#)

[SetPropertyValue \(Symbol Editor\)](#)

[SetVisibleLevels \(Symbol Editor\)](#)

[ToggleLevel \(Symbol Editor\)](#)

[Undo \(Symbol Editor\)](#)

[ZoomArea \(Symbol Editor\)](#)

[ZoomIn \(Symbol Editor\)](#)

[ZoomOut \(Symbol Editor\)](#)

[ZoomPrevious \(Symbol Editor\)](#)

[ZoomToFit \(Symbol Editor\)](#)

AddLevel (Symbol Editor)

To add a graphics level to the symbol.

UI Access	Add Level in the View > Levels dialog box		
Parameters	Name	Type	Description
	<level>	Integer	The new level, an integer greater than 1.
Return Value	Integer, the previous active level. The current active level is the newly added level.		

AlignHorizontal (Symbol Editor)

Align items horizontally

Syntax: AlignHorizontal(

```
Array("NAME:Selections", _  
"Page:=", page number, _ // [Opt=1] Page number  
"Selections:=", IDs to modify)),  
Array("NAME:AlignParameters", _  
"Disconnect:=", bool _ // [Opt=0] Should wires disconnect  
"Rubberband:=", bool) _ // [Opt=1] Should wires staircase  
// Note: Alignment occurs relative to the first item in ids
```

Python Syntax	AlignHorizontal(["NAME:Selections", "Selections:=", [<selections>]], 0)
Python Example	<pre>oSymbolEditor = oProject.SetActiveDefinitionEditor("SymbolEditor", _ "Simplorer Elements\basic Elements\Circuit\PassiveElements:R") oSymbolEditor.AlignHorizontal (["NAME:Selections",_</pre>

	<code>"Selections:=", ["SchObj@11"]], 0)</code>

AlignVertical (Symbol Editor)

Align items vertically

Syntax: AlignVertical(

```
Array("NAME:Selections", _
"Page:=", page number, _ // [Opt=1] Page number
"Selections:=", IDs to modify)),
Array("NAME:AlignParameters", _
"Disconnect:=", bool _ // [Opt=0] Should wires disconnect
"Rubberband:=", bool) _ // [Opt=1] Should wires staircase
// Note: Alignment occurs relative
to the first item in ids
```

Python Syntax	<code>AlignVertical(["NAME:Selections", "Selections:=", [<selections>]], 0)</code>
Python Example	<pre>oSymbolEditor = oProject.SetActiveDefinitionEditor("SymbolEditor", _ "Simplorer Elements\basic Elements\Circuit\PassiveElements:R") oSymbolEditor.AlignVertical(["NAME:Selections", "Selections:=", ["SchObj@11"]], 0)</pre>

BringToFront (Symbol Editor)

Changes the drawing for the symbol so that the specified objects are drawn on top of other overlapping objects.

*Command:***Draw > Bring To Front**

Syntax: BringToFront Array("NAME:Selections", "Selections:=", Array (<Object>, <Object>, ...))

Return Value: None.

Parameters: <string><Object>

<string> // object to bring to the front

ChangeProperty (Symbol Editor)

Use: Changes to properties are scripted using the **ChangeProperty** command. This command can be executed by the **oEditor** to change editor properties, by the **oDesign** to change design level properties, and by the **oProject** to change project level properties. The command can be used to create, edit, and/or remove properties. In Circuit, only Variable and Separator properties can be deleted.

Command: None

Syntax: ChangeProperty Array("Name:AllTabs", <PropTabArray>, <PropTabArray>, ...)

Return Value: None

Parameters: <PropTabArray>

Array("Name:<PropTab>",
<PropServersArray>,
<NewPropArray>,
<ChangedPropsArray>,
<DeletedPropsArray>)

<PropServersArray>

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

<PropServer>, ...)

<NewPropArray>

Array("Name:NewProp", <PropDataArray>,

<PropDataArray>,...)

<ChangedPropsArray>

Array("Name:ChangedProps",<PropDataArray>,

<PropDataArray>, ...)

<DeletedPropsArray>

Array("Name:DeletedProps", <PropName>,

<PropName>, ...)

OR (for PropDisplay deletions only)

Array("Name:DeletedProps",<PropDataArray>,

<PropDataArray>, ...)

<PropDataArray>

Array("NAME:<PropName>",

"PropType:=", <PropType>,

```
"NewName:=", <string>,  
"Description:=", <string>,  
"Callback:=", <string>,  
"NewRowPosition:=", <int>,  
"ReadOnly:=", <bool>,  
"Hidden:=", <bool>,  
<PropTypeSpecificArgs>)OR (for PropDisplays only)  
Array("Name:<PropName>",<PropDisplayData>)
```

<PropDisplayData>

for adding, changing, deleting PropDisplays

<PropDisplayAttributes>

for changing PropDisplays only

<PropDisplayNewAttributes>

<PropDisplayAttributes>

Layer & Location only used for PropDisplays in layout

For adding PropDisplays, this will add a single PropDisplay with attributes as shown; if an attribute is missing, a default value will be assigned. Adding PropDisplay to schematic with attributes that are identical to one already existing there will fail without an error message.

For deleting PropDisplays, these attributes are used to identify an existing PropDisplay to delete. If there doesn't exist a PropDisplay that matches the given attributes, then nothing will be deleted. If multiple PropDisplays match the given attributes, then all of them will be deleted. If an attribute is missing, then all PropDisplays match that missing attribute. For example, if Layer is missing, then PropDisplays on all layers that match the remaining given attributes will be deleted.

For changing PropDisplays, these attributes are used to identify an existing PropDisplay to change. If no PropDisplay matching the attributes is found, no changes will be made. If multiple PropDisplays match the attributes, all of them will be changed. If an attribute is missing, it matches all PropDisplays. For example, to change the format of PropDisplays that are on the bottom, but have any layer, style or format to show the name only, this command should have Location set to "Bottom" and all other attributes omitted.

"Format:=", <PropDisplayType> ,

"Location:=", <PropDisplayLocation> ,

"Layer:=", <string> ,

"Style:=", <string>

<PropDisplayNewAttributes>

NewLayer & NewLocation only used for PropDisplays in layout

For changing PropDisplays, these attributes are used to identify which attributes to change and what the new value is. If the attribute should not be changed, the corresponding entry should be omitted.

"NewName:=", <string> ,

"NewFormat:=", <PropDisplayType> ,

"NewLocation:=", <PropDisplayLocation> ,

"NewLayer:=", <string>,

"NewStyle:=", <string>

<PropDisplayType>

Type: string

Identifies the format of PropDisplay.

"Name"

"Value"

"NameAndValue"

"EvaluatedValue"

"NameAndEvaluatedValue"

<PropDisplayLocation>

Type: string

Identifies where PropDisplay is located with respect to object

"Left"

"Top"

"Right"

"Bottom"

"Custom"

<PropType>

Type: string

Identifies the type of property when a new property is added. In Circuit, only separator properties and variable properties can be added.

"SeparatorProp"

"VariableProp"

"TextProp"

"NumberProp"

"ValueProp"

"CheckboxProp"

"MenuProp"

"PointProp"

"VPointProp"

"ButtonProp"

NewName

Specify the new name of a property if the property's name is being edited. In Circuit, the name can only be changed for separators and variables.

Description

Specify a description of the property. In Circuit, the description can only be changed for separators and variables.

Callback

Specify the name of the script callback to be run when the property value is changed.

NewRowPosition

Used to reorder rows in the **Property** dialog box. In Circuit, this only applies to the **Project>Project Variables** panel and the **Designer>DesignProperties** panel. Specify the new zero-based row index of the variable or separator.

ReadOnly

Used to mark a property as "read only" so it can not be modified. In Circuit, this flag can only be set for variables and separators.

Hidden

Used to hide a property so it can not be viewed outside of the **Property** dialog box. In Circuit, this flag can only be set for variables and separators.

<PropTypeSpecificArgs>

SeparatorProp: no arguments

TextProp: "Value:=", <string>

NumberProp: "Value:=", <double>

ValueProp: "Value:=", <value>

CheckboxProp: "Value:=", <bool>

MenuProp: "Value:=", <string>

PointProp "X:=", <double>, "Y:=", <double>

VPointProp: "X:=", <value>, "Y:=", <value>

Material Button: "Material:=", <string>

Color Button: "R:=", <int>, "G:=", <int>, "B:=", <int>

Transparency Button: "Value:=", <double>

<PropTypeSpecificArgs> for MenuProps

Syntax for NewProps array: "AllChoices:=",

<"choice1,choice2,..."> or <Array("choice1" "choice2", ...)>,

"Value:=", <string>

Syntax for ChangedProps array: "Value:=", <string>

<PropTypeSpecificArgs> for VariableProps

Syntax:

"Value:=", <value>, <OptimizationFlagsArray>,

<TuningFlagsArray>, <SensitivityFlagsArray>,

<StatisticsFlagsArray>

Parameters:

<OptimizationFlagsArray>

Array("NAME:Optimization",

"Included:=", <bool>,

"Min:=", <value>,

"Max:=", <value>)

<TuningFlagsArray>

Array("NAME:Tuning",

"Included:=", <bool>,

"Step:=", <value>,

"Min:=", <value>,

"Max:=", <value>)

<SensitivityFlagsArray>

Array("NAME:Sensitivity",

"Included:=", <bool>,

"Min:=", <value>,

"Max:=", <value>,

"IDisp:=", <value>)

<StatisticsFlagsArray>

```
Array("NAME:Statistical",  
      "Included:=", <bool>,  
      "Dist:=", <Distribution>,  
      "StdD:=", <value>,  
      "Min:=", <value>,  
      "Max:=", <value>,  
      "Tol:=", <string>)
```

<Distribution>

Type: string

Value should be **"Gaussian"** or **"Uniform"**

StdD

Standard deviation.

Min

Low cut-off for the distribution.

Max

High cut-off for the distribution.

Tol

Tolerance for uniform distributions. Format is "<int>%".

Example: "20%".

CloseEditor (Symbol Editor)

Closes the specified editor.

Command: None

Syntax: CloseEditor (VARIANT ptDelta)

Return Value: None

Python Syntax	CloseEditor()
Python Example	oDefinitionEditor.CloseEditor()

Copy (Symbol Editor)

Copy items for pasting.

*Command:***Edit > Copy**

Syntax: Copy Array("NAME:Selections", "Selections:=", Array(<selections>))

Return Value: None.

Parameters: <selections>

Type: Comma-separated list of strings

Identifiers for each selected item to be copied, in the form SchObj@<id>, where id is the value of the SchematicID property for each element.

CreateArc (Symbol Editor)

Draws an arc in the symbol editor

Command: None

Syntax: CreateArc

Return Value: None

Python Syntax	CreateArc()
Python Example	<pre>oDefinitionEditor.CreateArc (["NAME:ArcData", _ "X:=", -0.004318, "Y:=", -0.00127, _ "Radius:=", 0.00297299377732279, _ "StartAng:=", 1.9195673303788, _ "EndAng:=", 3.32144615338227, _ "Id:=", 10], _ ["NAME:Attributes", "Page:=", 1])</pre>

CreateCircle (Symbol Editor)

Draws a circle in the symbol editor

Command: None

Syntax: CreateCircle

Return Value: None

Python Syntax	CreateCircle()
Python Example	<pre>oDefinitionEditor.CreateCircle (["NAME:CircleData", _ "X:=", -0.004572, "Y:=", -0.000508, _ "Radius:=", 0.001778, "Id:=", 12], _ ["NAME:Attributes", "Page:=", 1])</pre>

CreateImage (Symbol Editor)

Create picture from a BMP or GIF file.

Command:Draw>Image

Syntax: CreateImage(

Return Value: None.

Parameters: Array("NAME:ImageData",
 "X1:=", double, // the image rectangle left X value, in meters
 "Y1:=", double, // the image rectangle upper Y value, in meters
 "X2:=", double, // the image rectangle right X value, in meters
 "Y2:=", double, // the image rectangle lower Y value, in meters
 "File:=", string, // the name of the BMP or GIF file

"DisplayBorder:=", bool, // true or false, whether to display the image border

"LineWidth:=", double, // the width of the image border, in meters

"Color:=", int, // the RGB value of the image border color

"Id:=", int), // Id for this item

Array("NAME:Attributes",

"Page:=", int) // [Opt=1] The page number (one-based)

CreateLine (Symbol Editor)

Draws a line in the symbol editor

Command: None

Syntax: CreateLine

Python Syntax	CreateLine()
Python Example	<pre>oDefinitionEditor.CreateLine (["NAME:LineData", _Return Value: None "Points:=", ["(0.001778, -0.001270)",_ "(0.004572, 0.002794)", "(0.003048, 0.003556)"],_ "Id:=", 14], ["NAME:Attributes", "Page:=",1])</pre>

CreatePin (Symbol Editor)

Draws a pin in the symbol editor

Command: None

Syntax: CreatePin

Return Value: None

Python Syntax	CreatePin()
Python Example	<pre>oDefinitionEditor.CreatePin (["NAME:PinData", _ "Name:=", "newpin3"], ["NAME:PinParams",_ "X:=", -0.00762, "Y:=", 0.00254, "Angle:=", _ 0, "Flip:=", false])</pre>

CreatePolygon (Symbol Editor)

Draws a polygon in the symbol editor

Command: None

Syntax: CreatePolygon

Return Value: None

Python Syntax	CreatePolygon()
Python Example	<pre>oDefinitionEditor.CreatePolygon (["NAME:PolygonData", _ "X1:=", -0.001524, "Y1:=", 0.000508, _ "X2:=", 0.002286, "Y2:=", -0.002032, "Id:=", 18], _ ["NAME:Attributes", "Page:=", 1])</pre>

CreateRectangle (Symbol Editor)

Draws a rectangle in the symbol editor

Command: None

Syntax: CreateRectangle

Return Value: None

Python Syntax	CreateRectangle()
Python Example	<pre>oDefinitionEditor.CreateRectangle (["NAME:RectData", _ "X1:=", -0.001524, "Y1:=", 0.000508, _ "X2:=", 0.002286, "Y2:=", -0.002032, "Id:=", 18], _ ["NAME:Attributes", "Page:=", 1])</pre>

CreateText (Symbol Editor)

Draws text in the symbol editor

Command: None

Syntax: CreateText

Return Value: None

Python Syntax	CreateText()
Python Example	<pre>oDefinitionEditor.CreateText(["NAME:TextData", "X:=", 0.001524, "Y:=",</pre>

	<pre>0.00127, "Text:=", "My text", "Id:=", 20], ["NAME:Attributes", "Page:=", 1])</pre>

Cut (Symbol Editor)

Cut page selection.

Command: None

Syntax: Cut(PageNumber, Selections)

Return Value: None

Python Syntax	Cut(["NAME:Selections", "Selections:=", [<selections>]])
Python Example	<pre>oSymbolEditor = oProject.SetActiveDefinitionEditor("SymbolEditor",_ "Simplorer Elements\basic Elements\Circuit\PassiveElements:R") oSymbolEditor.Cut (["NAME:Selections", "Selections:=", ["SchObj@11", "SchObj@3"]])</pre>

Delete (Symbol Editor)

Delete items - remove from the symbol.

Command: **Edit>Delete**

Syntax: Delete Array("NAME:Selections", "Selections:=", Array(<selections>))

Return Value: None.

Parameters: <selections>

Type: Comma-separated list of strings

Identifiers for each selected item to be deleted, in the form SchObj@<id>, where id is the value of the SchematicID property for each element.

DisplayPinNames (Symbol Editor)

To turn on or off the display of pin names.

*Command:*None.

Syntax: DisplayPinNames <onoff>

Return Value: None.

Parameters: <onoff>

Type: Boolean

TRUE or FALSE to turn on or off the display of pin names.

Example: Dim oSymbolEditor

```
Set oSymbolEditor = oProject.SetActiveDefinitionEditor("SymbolEditor", "Simplorer Elements\basic  
Elements\Circuit\Passive Elements:R")
```

```
oSymbolEditor.DisplayPinNames TRUE
```

ExportFile (Symbol Editor)

Export file of symbol.

*Command:***Symbol>Export File**

Syntax: ExportFile <type>, <filename>

Return Value: None.

Parameters: <type>

Type: String

Type of file to export - only "EMF" is supported currently.

<filename>

Type: String

Path, name and extension for the file to be saved.

FlipHorizontal (Symbol Editor)

To flip the specified objects about the horizontal (x) axis.

Command:Draw>Flip Horizontal

Syntax: FlipHorizontal Array("NAME:Selections", "Selections:=", Array(<selections>)), 0

Return Value: None.

Parameters: <selections>

Type: Comma-separated list of strings

Identifiers for each selected item to be flipped, in the form SchObj@<id>, where id is the value of the SchematicID property for each element.

FlipVertical (Symbol Editor)

To flip the specified objects about the horizontal (x) axis.

Command:Draw>Flip Vertical

Syntax: FlipVertical Array("NAME:Selections", "Selections:=", Array(<selections>)), 0

Return Value: None.

Parameters: <selections>

Type: Comma-separated list of strings

Identifiers for each selected item to be flipped, in the form SchObj@<id>, where id is the value of the SchematicID property for each element.

GetProperties (Symbol Editor)

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

Command: None

Syntax: GetProperties(<PropTab>, <PropServer>)

Return Value: Variant array of strings – the names of the properties belonging to the prop server.

GetPropertyValue (Symbol Editor)

Gets the value of a single property. This can be executed by the **oProject**, **oDesign**, or **oEditor** objects.

Command: None

Syntax: GetPropertyValue(<PropTab>, <PropServer>, <PropName>)

Return Value: String representing the property value.

GetVisibleLevels (Symbol Editor)

To determine the levels in the symbol.

*Command:*None.

Syntax: GetVisibleLevels

Return Value: An array of integers, which are the levels defined for the particular symbol.

Parameters: None.

GridSetup (Symbol Editor)

Changes the settings on the symbol grid.

Command:Symbol>Grid Setup

Syntax: GridSetup(

Return Value: None.

Parameters: Array("NAME:Options", _
 "MajorGrid:=", string, _ // Major grid size with units
 "Divisions:=", int, _ // Number of minor grid divisions
 "MajorColor:=", int, _ // RGB Color of major grid lines
 "MinorColor:=", int, _ // RGB Color of minor grid lines
 "ShowGrid:=", bool, _ // Should the grid be shown?
 "SnapToGrid:=", bool, _ // Should objects snap to grid?
 "BackgroundColor:=", int, _ // RGB Color of the background
 "SaveAsDefault:=", bool)) // Should these settings be the default for new schematics

Example: oEditor.GridSetup Array(
 "NAME:Options",
 "MajorGrid:=", "10mm",
 "Divisions:=", 10,
 "MajorColor:=", 0x00ff0000, // Red
 "MinorColor:=", 0x0000ff00, // Green

```
"ShowGrid:=", true,  
"SnapToGrid:=", true,  
"BackgroundColor:=", 0x00ffffff, // White  
"SaveAsDefault:=", false)
```

Move (Symbol Editor)

To move the specified objects in X and Y.

*Command:*None.

Syntax: Move Array("NAME:Selections", "Selections:=", Array(<selections>)), Array("NAME:FlipParameters", "xdelta:=", <xval>, "ydelta:=", <yval>)

Return Value: None.

Parameters: <selections>

Type: Comma-separated list of strings

Identifiers for each selected item to be moved, in the form SchObj@<id>, where id is the value of the SchematicID property for each element.

<xval>

Type: Real number

The X delta to move.

<yval>

Type: Real number

The Y delta to move.

Pan (Symbol Editor)

Pans the display.

UI Access	View > Pan.		
Parameters	Name	Type	Description
	<argSelections>	Array of doubles	First double pans along x-axis; second double pans along y-axis.
Return Value	None.		

Python Syntax	Pan (<argSelections>)
Python Example	<pre>oDefinitionEditor.Pan([-0.00922653869663931, 0.00686839904222147])</pre>

Paste (Symbol Editor)

To paste copied objects to a specified X and Y location.

Command: Edit>Paste

Syntax: Paste Array("NAME:Attributes", "X:=", <xval>, "Y:=", <yval>)

Return Value: None.

Parameters: <xval>

Type: Real number

The X placement value.

<yval>

Type: Real number

The Y placement value.

Redo (Symbol Editor)

Redo the last operation

Command: Click **Edit>Redo**

Syntax: Redo

Return Value: None

Parameters: None

RemoveLevels (Symbol Editor)

Removes one or more graphics levels from a symbol.

UI Access	Tools > Edit Libraries > Symbols. From Symbol editor, click View > Levels > Remove Level.		
Parameters	Name	Type	Description
	</levels>	Array of integers	The levels to be deleted.
Return Value	None.		

Python Syntax	RemoveLevels (</levels>)
Python Example	<code>oSymbolEditor.RemoveLevels ([3,4])</code>

RemovePort (Symbol Editor)

Selected pins are changed to vias. Selected edge terminal ports are removed.

Syntax: RemovePort <NAME:elements", <object_name> ... // objects to be removed

Return Value: None.

Parameters: <object_name>

Type: <String>

Example: oEditor.RemovePort ["NAME:elements", "via_195", "Port1"]

Rotate (Symbol Editor)

To rotate the specified objects about their common center.

Command:Draw>Rotate

Syntax: Rotate Array("NAME:Selections", "Selections:=", Array(<selections>)), Array("NAME:FlipParameters", <angle>)

Return Value: None.

Parameters: <selections>

Type: Comma-separated list of strings

Identifiers for each selected item to be rotated, in the form SchObj@<id>, where id is the value of the SchematicID property for each element.

<angle>

Type: A comma-separated string and real number

Either "Angle:=", <rad>, where <rad> is the angle in radians, or "Degrees:=", <deg>, where <deg> is the angle in degrees.

Save (Symbol Editor)

Saves any changes in the current symbol to the project.

Command:Symbol>Update Project

Syntax: Save

Return Value: None.

Parameters: None.

Example:

```
oSymbolEditor = oProject.SetActiveDefinitionEditor("SymbolEditor", "Simplorer Elements\basic Elements\Circuit\Passive Elements:R")
```

SelectAll (Symbol Editor)

Select all elements in the symbol editor.

Command: None.

Syntax: SelectAll()

Parameters: None

Example: removedDefs = oDefinitionEditor.SelectAll()

Python Syntax	SelectAll()
Python Example	removedDefs = oSymbolManager.SelectAll()

SendToBack (Symbol Editor)

Changes the drawing for the symbol so that the specified objects are drawn behind other overlapping objects.

Command: Draw > Send To Back

Syntax: SendToBack Array("NAME:Selections", "Selections:=", Array (<Object>, <Object>, ...))

Return Value: None

Parameters: <Object>

<string> // object to send to the back

Example: oDefinitionEditor.SendToBack ["NAME:Selections", "Selections:=", ["SchObj@10"]]

Python Syntax	SendToBack(["NAME:Selections", "Selections:=", <Object>, <Object>, ...])
Python Example	oDefinitionEditor.SendToBack(["NAME:Selections", "Selections:=", ["SchObj@10"]])

SetActiveLevel (Symbol Editor)

To set the active graphics level in the symbol editor.

Command: "Current" radio button for a level in the **View>Levels** dialog.

Syntax: SetActiveLevel <level>

Return Value: None.

Parameters: <level>

Type: Integer

The new active level.

Example:

```
oSymbolEditor = oProject.SetActiveDefinitionEditor("SymbolEditor", _  
    "Simplorer Elements\basic Elements\Circuit\Passive Elements:R")  
oldLevel = oSymbolEditor.SetActiveLevel (4)
```


SetInitialLevels (Symbol Editor)

Sets one or more graphics levels as the default visible levels for a symbol. The levels submitted do not need to contain 1 or 0, which are always visible.

UI Access	Tools > Edit Libraries > Symbols. From Symbol editor, click View > Levels . Select the Default check box for desired levels.		
Parameters	Name	Type	Description
	<levels>	Array of integers	The levels to be selected.
Return Value	None.		

Python Syntax	SetInitialLevels (<levels>)
Python Example	<code>oSymbolEditor.SetInitialLevels([3,4])</code>

SetPropertyValue (Symbol Editor)

Set a property.

Command: None.

Syntax: SetPropertyValue(tab, item, propname, value)

Parameters: None

Example:

```
removedDefs = oDefinitionEditor.SetPropertyValue(string tab, // Tab with the property
string item, // Name of the object
string propname, // Name of the property
```

string value) // The new value of the prop

SetVisibleLevels (Symbol Editor)

Makes one or more graphics levels visible in the symbol editor.

UI Access	Tools > Edit Libraries > Symbols. From Symbol editor, click View > Levels. Select the Visible check box for the desired levels.		
Parameters	Name	Type	Description
	</levels>	Array of integers	The levels to be made visible.
Return Value	None.		

Python Syntax	SetVisibleLevels (</levels>)
Python Example	oSymbolEditor.SetVisibleLevels ([3,4])

ToggleLevel (Symbol Editor)

To toggle display of the graphics level in the symbol editor.

Command:"Visible" checkbox for a level in the View>Levels dialog.

Syntax: ToggleLevel <level>

Return Value: None.

Parameters: <level>

Type: Integer

The level to turn on or off.

Example:

```
oSymbolEditor = oProject.SetActiveDefinitionEditor("SymbolEditor",
    "Simplorer Elements\basic Elements\Circuit\Passive Elements:R")
oSymbolEditor.ToggleLevel (2)
```

Undo (Symbol Editor)

Undo the last operation

Command: Edit>Undo

Syntax: Undo

Return Value: None

Parameters: None

Example: oDesign.Undo

Python Syntax	Undo()
Python Example	<pre>oSymbolEditor = oProject.SetActiveDefinitionEditor("SymbolEditor",_ "Simplorer Elements\basic Elements\Circuit\PassiveElements:R") oSymbolEditor.Undo()</pre>

ZoomArea (Symbol Editor)

To change the current graphics view area.

Command:View>Zoom Area

Syntax: ZoomArea Array(<xpt>, <ypt>), Array(<xdelta>, <ydelta>)

Return Value: None.

Parameters: <xpt>

Type: Real number

The X value of the zoom area box, in meters.

<ypt>

Type: Real number

The X value of the zoom area box, in meters.

<xdelta>

Type: Real number

The width of the zoom area box, in meters.

<ydelta>

Type: Real number

The height of the zoom area box, in meters.

Example:

```
oSymbolEditor = oProject.SetActiveDefinitionEditor("SymbolEditor",  
    "Simplorer Elements\basic Elements\Circuit\Passive Elements:R")  
oSymbolEditor.ZoomArea [0, 0], [0.004, 0.004]
```

ZoomIn (Symbol Editor)

To zoom in a standard amount to the current graphics view area.

Command:View>Zoom In

Syntax: ZoomIn

Return Value: None.

Parameters: None.

Example:

```
oSymbolEditor = oProject.SetActiveDefinitionEditor("SymbolEditor",  
    "Simplorer Elements\basic Elements\Circuit\Passive Elements:R")  
oSymbolEditor.ZoomIn
```

ZoomOut (Symbol Editor)

To zoom out a standard amount to the current graphics view area.

Command:View>Zoom Out

Syntax: ZoomOut

Return Value: None.

Parameters: None.

Example:

```
oSymbolEditor = oProject.SetActiveDefinitionEditor("SymbolEditor",  
    "Simplorer Elements\basic Elements\Circuit\Passive Elements:R")  
oSymbolEditor.ZoomOut
```

ZoomPrevious (Symbol Editor)

To zoom the current graphics view area to the previous extents.

Command:View>Zoom Previous

Syntax: ZoomPrevious

Return Value: None.

Parameters: None.

Example:

```
oSymbolEditor = oProject.SetActiveDefinitionEditor("SymbolEditor",  
    "Simplorer Elements\basic Elements\Circuit\Passive Elements:R")  
oSymbolEditor.ZoomPrevious
```

ZoomToFit (Symbol Editor)

Set the current symbol zoom to fit the contents of the currently visible page

Command: None

Syntax: ZoomToFit()

Return Value: None

Python Syntax	ZoomToFit()
Python Example	<pre>oSymbolEditor = oProject.SetActiveDefinitionEditor("SymbolEditor",_ "Simplorer Elements\basic Elements\Circuit\PassiveElements:R")</pre>

	<code>oSymbolEditor.ZoomToFit()</code>

This page intentionally
left blank.

28 - Core Global Script Context Commands

To run these commands:

```
import CoreGlobalScriptContextFunctions  
  
CoreGlobalScriptContextFunctions.[CommandName]
```

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

- [AddErrorMessage](#)
- [AddFatalMessage](#)
- [AddInfoMessage](#)
- [AddWarningMessage](#)
- [LogDebug](#)
- [LogError](#)

AddErrorMessage

Adds an error message to the **Message Manager** window. AddErrorMessage is a function of CoreGlobalScriptContextFunctions.

UI Access	N/A		
Parameters	Name	Type	Description
	<message>	String	Error message.
Return Value	None.		

Python Syntax	AddErrorMessage(<message>)
Python Example	<pre>import CoreGlobalScriptContextFunctions</pre>

	<code>CoreGlobalScriptContextFunctions.AddErrorMessage('My error message.')</code>

AddFatalMessage

Adds a fatal error message to the **Message Manager** window. AddFatalMessage is a function of CoreGlobalScriptContextFunctions.

UI Access	N/A		
Parameters	Name	Type	Description
	<message>	String	Error message.
Return Value	None.		

Python Syntax	AddFatalMessage(<message>)		
Python Example	<pre>import CoreGlobalScriptContextFunctions CoreGlobalScriptContextFunctions.AddFatalMessage('My fatal error message.')</pre>		

AddInfoMessage

Adds an informational message to the **Message Manager** window. AddInfoMessage is a function of CoreGlobalScriptContextFunctions.

UI Access	N/A		
Parameters	Name	Type	Description
	<message>	String	Informational message.
Return Value	None.		

Python Syntax	AddInfoMessage(<message>)
Python Example	<pre>import CoreGlobalScriptContextFunctions CoreGlobalScriptContextFunctions.AddInfoMessage('My info.')</pre>

AddWarningMessage

Adds a warning message to the **Message Manager** window. AddWarningMessage is a function of CoreGlobalScriptContextFunctions.

UI Access	N/A		
Parameters	Name	Type	Description
	<message>	String	Warning message.
Return Value	None.		

Python Syntax	AddWarningMessge(<message>)
Python Example	<pre>import CoreGlobalScriptContextFunctions CoreGlobalScriptContextFunctions.AddWarningMessage('My warning.')</pre>

LogDebug

Adds a debug line to the log specified at **Tools > Debug Logging**. LogDebug is a function of CoreGlobalScriptContextFunctions.

UI Access	N/A		
Parameters	Name	Type	Description
	<message>	String	Debug message.
Return Value	None.		

Python Syntax	LogDebug(<message>)		
Python Example	<pre>import CoreGlobalScriptContextFunctions CoreGlobalScriptContextFunctions.LogDebug('My debug message.')</pre>		

LogError

Adds an error line to the log specified at **Tools > Debug Logging**. LogError is a function of CoreGlobalScriptContextFunctions.

UI Access	N/A		
Parameters	Name	Type	Description
	<error>	String	Error to log.
Return Value	None.		

Python Syntax	LogError(<error>)		
Python Example	<pre>import CoreGlobalScriptContextFunctions</pre>		

	<code>CoreGlobalScriptContextFunctions.LogError('My error.')</code>

This page intentionally
left blank.

29 - Example Scripts

This section contains [IronPython example scripts](#).

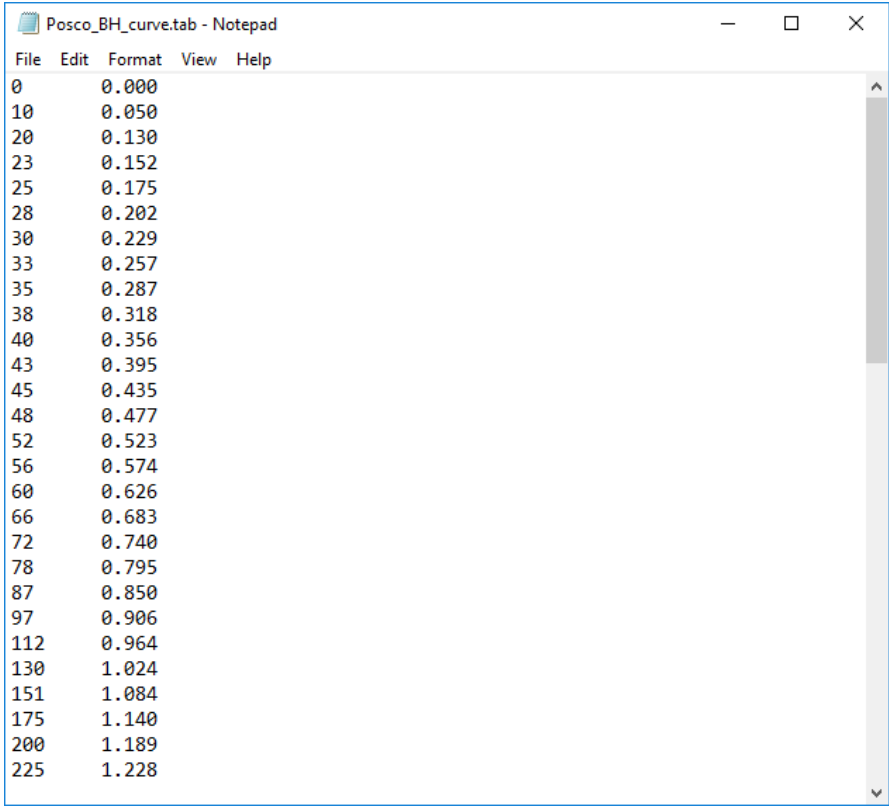
IronPython Example Scripts

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):



File	Edit	Format	View	Help
0	0.000			
10	0.050			
20	0.130			
23	0.152			
25	0.175			
28	0.202			
30	0.229			
33	0.257			
35	0.287			
38	0.318			
40	0.356			
43	0.395			
45	0.435			
48	0.477			
52	0.523			
56	0.574			
60	0.626			
66	0.683			
72	0.740			
78	0.795			
87	0.850			
97	0.906			
112	0.964			
130	1.024			
151	1.084			
175	1.140			
200	1.189			
225	1.228			

To recreate this tab file, paste [the text below the script](#) into a text editor and save as Posco_BH_curve.tab.

The script itself includes comment lines, which are preceded by # and offer explanations for the subsequent line(s).

Script Contents

```
# specify path to the file with BH coordinates
path_to_file = r"D:\Posco_BH_curve.tab"

# specify name of the material
material_name = "Posco 35PN250"

oProject = oDesktop.GetActiveProject()
oDefinitionManager = oProject.GetDefinitionManager()

""" create list with B and H points to pass to AddMaterial command
bh_coordinates list is a three dimensional array: 1D: name, 2D:
Coordinate list, 3D: BH point"""

bh_coordinates = ["NAME:BHCoordinates", ["NAME:DimUnits", "", ""]]

with open(path_to_file) as input_file:
    for line in input_file:
        h, b = line.split()

        bh_coordinates.append(["NAME:Coordinate", [
            "NAME:CoordPoint",
            float(h),
            float(b)
        ]
        ])

# create a new magnetic material with BH curve
oDefinitionManager.AddMaterial(
[
    "NAME:" + material_name,
    "CoordinateSystemType:=", "Cartesian",
    "BulkOrSurfaceType:=" , 1,
    [
        "NAME:PhysicsTypes",
        "set:=" , ["Electromagnetic"]
    ]
]
```



```

],
[
    "NAME:permeability",
    "property_type:=" , "nonlinear",
    "BTypeForSingleCurve:=" , "normal",
    "HUnit:=" , "A_per_meter", # unit can be specified as variable
    "BUnit:=" , "tesla", # unit can be specified as variable
    "IsTemperatureDependent:=", False,
    bh_coordinates,
    [
        "NAME:Temperatures"
    ]
],
"conductivity:=" , "1818181.82",
[
    "NAME:magnetic_coercivity",
    "property_type:=" , "VectorProperty",
    "Magnitude:=" , "0A_per_meter",
    "DirComp1:=" , "1",
    "DirComp2:=" , "0",
    "DirComp3:=" , "0"
]
])

```

Posco_BH_Curve.tab Contents

0	0.000
10	0.050
20	0.130
23	0.152
25	0.175
28	0.202

30	0.229
33	0.257
35	0.287
38	0.318
40	0.356
43	0.395
45	0.435
48	0.477
52	0.523
56	0.574
60	0.626
66	0.683
72	0.740
78	0.795
87	0.850
97	0.906
112	0.964
130	1.024
151	1.084
175	1.140
200	1.189
225	1.228
252	1.258
282	1.283
317	1.304
357	1.324
402	1.343
450	1.360
500	1.375
550	1.387

602	1.398
657	1.407
717	1.417
784	1.426
867	1.437
974	1.448
1117	1.461
1299	1.478
1515	1.495
1752	1.513
2000	1.529
2253	1.543
2521	1.557
2820	1.570
3167	1.584
3570	1.600
4021	1.617
4503	1.634
5000	1.652
5503	1.669
6021	1.685
6570	1.701
7167	1.717
7839	1.733
8667	1.749
9745	1.769
11167	1.793
12992	1.820
15146	1.851
17518	1.882

20000	1.909
22552	1.931
25417	1.948
28906	1.961
33333	1.971
38932	1.981

Equation Based Curve Python Script

This sample Python script creates an equation based curve that produces a helix.

```
from math import pi, sin, cos

oProject = oDesktop.GetActiveProject()
oDesign = oProject.GetActiveDesign()
oEditor = oDesign.SetActiveEditor("3D Modeler")

Start_t = 0
End_t = pi*2

Npoint = 128
Nsection = Npoint-1

d_t = (End_t-Start_t)/Nsection

for n in range(1,Nsection):

    P1 = Start_t+d_t*(n-1)
    P2 = P1+d_t
    X_t1 = cos(P1*6)
    Y_t1 = sin(P1*6)
    Z_t1 = P1
    X_t2 = cos(P2*6)
```

```
Y_t2 = sin(P2*6)
```

```
Z_t2 = P2
```

```
oEditor.CreatePolyline(  
  [  
    "NAME:PolylineParameters",  
    "IsPolylineCovered:=" , True,  
    "IsPolylineClosed:=" , False,  
    [  
      "NAME:PolylinePoints",  
      [  
        "NAME:PLPoint",  
        "X:=" , '1mm*' + str(X_t1),  
        "Y:=" , '1mm*' + str(Y_t1),  
        "Z:=" , '1mm*' + str(Z_t1)  
      ],  
      [  
        "NAME:PLPoint",  
        "X:=" , '1mm*' + str(X_t2),  
        "Y:=" , '1mm*' + str(Y_t2),  
        "Z:=" , '1mm*' + str(Z_t2)  
      ]  
    ],  
    [  
      "NAME:PolylineSegments",  
      [  
        "NAME:PLSegment",  
        "SegmentType:=" , "Line",  
        "StartIndex:=" , 0,
```

```
"NoOfPoints:=" , 2
]
],
[
  "NAME:PolylineXSection",
  "XSectionType:=" , "None",
  "XSectionOrient:=" , "Auto",
  "XSectionWidth:=" , "0mm",
  "XSectionTopWidth:=" , "0mm",
  "XSectionHeight:=" , "0mm",
  "XSectionNumSegments:=" , "0",
  "XSectionBendType:=" , "Corner"
]
],
[
  "NAME:Attributes",
  "Name:=" , "Polyline"+str(n),
  "Flags:=" , "",
  "Color:=" , "(132 132 193)",
  "Transparency:=" , 0,
  "PartCoordinateSystem:=" , "Global",
  "UDMId:=" , "",
  "MaterialValue:=" , "\"vacuum\"",
  "SurfaceMaterialValue:=" , "\"\"",
  "SolveInside:=" , True,
  "IsMaterialEditable:=" , True,
  "UseMaterialAppearance:=" , False,
  "IsLightweight:=" , False
])
])
```

Index

#

- 3D Modeler editor commands 10-1
 - AssignMaterial 10-125
 - Chamfer 10-126
 - ChangeProperty 6-7, 10-224, 18-10
 - Connect 10-128
 - Copy 10-108
 - CoverLines 10-129
 - CoverSurfaces 10-129
 - Create3DComponent 10-6
 - CreateBondwire 10-11
 - CreateBox 10-14
 - CreateCircle 10-16
 - CreateCone 10-18
 - CreateCutplane 10-20
 - CreateCylinder 10-22
 - CreateEllipse 10-24
 - CreateEntityList 10-130, 10-137
 - CreateEquationCurve 10-26
 - CreateEquationSurface 10-29
 - CreateFaceCS 10-131, 10-138
 - CreateGroup 10-135
 - CreateHelix 10-31
 - CreateObjectFromdges 10-144
 - CreateObjectFromFaces 10-145
 - CreatePoint 10-33
 - CreatePolyline 10-41
 - CreateRectangle 10-45
 - CreateRegion 10-47
 - CreateRegularPolygon 10-52
 - CreateRegularPolyhedron 10-50
 - CreateRelativeCS 10-146
 - CreateSphere 10-55
 - CreateSpiral 10-57
 - CreateTorus 10-58
 - CreateUserDefinedModel 10-60
 - CreateUserDefinedPart 10-34, 10-73
 - Delete 10-231
 - DeleteEmptyGroups 10-148
 - DeleteLastOperation 10-149
 - DeletePolylinePoint 10-108
 - DetachFaces 10-149
 - DuplicateAlongLine 10-109
 - DuplicateAroundAxi 10-111
 - DuplicateMirror 10-113
 - Edit3dComponent 10-81
 - EditEntityList 10-151, 10-156
 - EditFaceCS 10-152

EditObjectCS 10-157	GetProperties 6-33, 8-23
EditPolyline 10-84	GetPropertyValue 6-34, 8-24, 12-56
EditRelativeCS 10-163	GetPropEvaluatedValue 6-30
Export 10-165	GetPropSIValue 6-31
ExportModelImageToFile 10-166	GetUser Position 10-246
Fillet 10-170	GetUserDefinePartVersion 10-90
FlattenGroup 10-171	GetVertexIDsFromEdge 10-247
GenerateHistory 10-172	GetVertexIDsFromFace 10-247
Get3DComponentParameters 10-88	GetVertexIDsFromObject 10-248
GetActiveCoordinateSystem 10- 176	GetVertexPosition 10-248
GetAct- iveCoordin- ateSystemTransform 10-177	HealObject 10-173
GetBodyNamesByPosition 10- 232	Import 10-178
GetCoordinateSystems 10-177	ImportDXF 10-180
GetEdgeByPosition 10-233	ImportGDSII 10-184
GetEdgeIDsfromFace 10-234	Insert3DComponent 10-91
GetEdgeIDsfromObject 10-235	InsertNativeComponent 10-93
GetEntityListIDByName 10-235, 10-236	Intersect 10-187
GetFaceArea 10-236	Mirror 10-115
GetFaceByPosition 10-237	Move 10-117
GetFaceCenter 10-237	MoveCSToEnd 10-188
GetFaceIDs 10-239	MoveEntityToGroup 10-188
GetGeometryModelerMode 10- 239	MoveFaces 10-189
GetModelBoundingBox 10-240	OffsetFaces 10-118
GetObjectBoundingBox 10-240	PageSetup 10-249
	ProjectSheet 10-191
	RenamePart 10-250
	Rotate 10-120
	Scale 10-121

- Section 10-193
- SeparateBody 10-194
- SetModelUnits 10-195
- SetWCS 10-196
- Split 10-197
- Subtract 10-199
- SweepAlongPath 10-98
- SweepAlongVector 10-100
- SweepAroundAxis 10-101
- SweepFacesAlongNormal 10-103, 10-103, 10-200, 10-200
- Sweep-FacesAlongNormalWithAttributes 10-104
- ThickenSheet 10-202
- UncoverFaces 10-203
- Ungroup 10-205
- Unite 10-204
- UpdateComponentDefinition 10-106
- WrapSheet 10-206
- 3D Modeler editor object commands
 - GetNumObjects 10-245
 - GetObjectIDByName 10-241
 - GetObjectName 10-241
 - GetObjectNameByFaceID 10-242
- A**
 - Add 7-35, 7-92, 7-119, 7-141, 7-174
 - AddAddSolverOnDemandModel 7-54
 - AddAssignmentToBoundary 13-3
 - AddCartesianLimitLine 12-4
 - AddCartesianXMarker 12-5
 - AddCartesianYMarker 12-6
 - AddDataset 6-4
 - AddDefinitionFromBlock 10-209
 - AddDefinitionFromLibFile 10-214
 - AddDeltaMarker 12-7
 - AddDiffPair 24-3
 - AddLevel 27-3
 - AddMarker 12-8
 - AddMarker[Fields Reporter] 18-3
 - AddMarkerToPlot 18-2
 - AddMaterial 7-2, 7-14
 - AddMessage 4-5
 - AddNote 12-8
 - AddNPortData 7-50
 - AddPinGrounds 26-34
 - AddPinIPorts 26-37
 - AddPinPageConnectors 26-37
 - AddReduceOp 21-2
 - AddTerminalsToWinding 13-47
 - AddTraceCharacteristics 12-10
 - AddTraces 12-11
 - AlignHorizontal 26-38, 27-4
 - AlignVertical 26-39, 27-5
 - Analysis module commands
 - Analyze 9-4

CopySetup 15-2, 16-7	ApplyMeshOps 9-4
DeleteSetups 15-3	AssignBand B
EditSetup 15-4	AssignCharge 13-47
ExportIcepak 15-14	AssignCoilGroup 13-48
ExportSolnData 15-15	AssignCoilTerminal 13-49
GetSetupCount 15-16	AssignCoilTerminalGroup 13-49
GetSetups 15-17	AssignCurrent 13-50
GetSweepCount 15-17	AssignCurrentDensity 13-52
GetSweeps 15-18	AssignCurrentDensityGroup 13-53
InsertSetup [Maxwell] 15-18	AssignCurrentDensityTerminal 13-53
PasteSetup 15-31	AssignCurrentDensityTerminalGroup 13-53, 13-53
PasteSweep 15-31	AssignCurrentGroup 13-53
RenameSetup 15-32	AssignCylindricalGapOp 14-4
ResetAllToTimeZero 15-32	AssignCylindricalHField 13-17
ResetSetupToTimeZero 15-33	AssignFloating 13-54
ResetToTimeZero 9-35	AssignFluxTangential 13-21
RevertAllToInitial 15-33	AssignForce 21-3
RevertAllToInitialTemperature 15-34	AssignImpedance 13-21
RevertSetupToInitial 15-34	AssignInsulating 13-25
RevertSetupToInitialTemperature 15-35	AssignLayoutForce 21-4
Analysis Setup Module Script Commands 15-1	AssignLengthOp 14-10
Analyze	AssignMaterial 10-125
Analysis module command 9-4	AssignModelResolutionOp 14-12
AnalyzeAll 9-5	AssignRadiation 13-25
Ansoft Application Object commands 3-1	AssignResistiveSheet 13-26
GetAppDesktop 3-2	AssignSink 13-54
	AssignSkinDepthLayerSetting 14-13
	AssignSkinDepthOp 14-14

AssignSymmetry 13-28
AssignTangentialHField 13-28
AssignThinLayer 13-29
AssignTouching 13-32
AssignTrueSurfOp 14-16
AssignVectorPotential 13-33
AssignVoltageAPhi 13-56, 13-74
AssignVoltageDrop 13-57
AssignVoltageDropGroup 13-58
AssignVoltageGroup 13-58
AssignVolumeChargeDensity 13-58
AssignWindingGroup 13-59

B

Boundary/Excitation module commands

AddAssignmentToBoundary 13-3
AddTerminalsToWinding 13-47
AssignCharge 13-47
AssignCoilGroup 13-48
AssignCoilTerminal 13-49
AssignCoilTerminalGroup 13-49
AssignCurrentDensity 13-52
AssignCurrentDensityGroup 13-53
AssignCurrentDensityTerminal 13-53
AssignCurrentDensityTerminalGroup 13-53, 13-53

AssignCurrentGroup 13-53
AssignCylindricalHField 13-17
AssignFloating 13-54
AssignImpedance 13-21
AssignInsulating 13-25
AssignRadiation 13-25
AssignSink 13-54
AssignSymmetry 13-28
AssignTangentialHField 13-28
AssignVoltageDrop 13-57
AssignVoltageDropGroup 13-58
AssignVoltageGroup 13-58
AssignVolumeChargeDensity 13-58
AssignWindingGroup 13-59
DeleteAllBoundaries 13-4
DeleteAllExcitations 13-4
DeleteBoundaries 13-5
EditCharge 13-60
EditCoilTerminal 13-61
EditCurrentDensity 13-62
EditCurrentDensityTerminal 13-63
EditCylindricalHField 13-34
EditDependent 13-35
EditFloating 13-64
EditImpedance 13-36
EditIndependent 13-36
EditInsulating 13-37
EditRadiation 13-38

EditSink 13-64	SetMinimumTimeStep 13-74
EditSymmetry 13-40	SetPermeabilityLink 13-78
EditTangentialHField 13-41	boundary/excitation script commands 13-2
EditVoltage 13-64	BreakUDMConnection 10-223
EditVoltageDrop 13-67	BringToFront 7-127, 27-5
EditVolumeChargeDensity 13-67	C
EditWindingGroup 13-67	CalcOp 18-5, 19-5
EditZeroTangentialHField 13-45	CalcRead (deprecated) 19-6
GetBoundaries 13-6	CalcStack 18-5, 19-7
GetBoundariesOfType 13-7	CalculatorRead 18-6, 19-6
GetBoundaryAssignment 13-5	CalculatorWrite 18-7, 19-8
GetCoreLossEffect 13-7	Cascade 24-4
GetCoreLossEffectOnField 13-8	Chamfer 10-126
GetDisplacementCurrent 13-8	ChangeGeomSettings 18-14, 19-11
GetEddyEffect 13-8	ChangeProperty 6-7, 10-224, 12-13, 18-10, 26-80, 27-6
GetExcitations 13-9	ClcEval 18-14, 19-12
GetExcitationsOfType 13-9	ClcMaterial 18-17, 19-14
GetNumBoundaries 13-10	ClcMaterialValue 18-18
GetNumBoundariesOfType 13-11	ClearAllMarkers 12-21
GetNumExcitations 13-11	ClearAllMarkers[Fields Reporter] 18-18
GetNumExcitationsOfType 13-12	ClearAllNamedExpr 18-19, 19-15
ReassignBoundaries 13-3, 13-12	ClearAllTraceCharacteristics 12-22
ReassignBoundary 13-12	ClearDiffPairs 24-5
RemoveAssignmentFromBoundary 13-13	ClearMessages 2-34, 4-6, 6-14
RenameBoundary 13-14	ClearSolutionCache 7-55
ReprioritizeBoundaries 13-15	Clone 24-6
	CloneMaterial 7-12

- Close 2-35, 6-15, 24-7
- CloseAllWindows 4-10
- CloseEditor 26-41, 27-16
- CloseProject 4-10
- CloseProjectNoForce 4-11
- Combine 24-8
- CompInstance Functions 25-2
- CompInstance Script
Commands 25-1
- Component Manager Script Com-
mands 7-34
- ComputeCoreLossCoefficients 7-
13
- Con-
fig-
ureFlu-
entConductivityCoupling 9-6
- Connect 10-128
- ConstructVariationString 9-6
- conventions
scripting help 1-1
- ConvertToDynamic 7-100
- ConvertToParametric 7-101
- Copy 10-108, 26-42, 27-16
- CopyData 26-43
- CopyDesign 2-35, 6-15
- CopyNamedExprToStack 18-19,
19-16
- CopyReportDefinition 12-24
- CopyReportsData 12-23
- Copyright and Trademark Inform-
ation 2
- CopySetup
Analysis module command 15-2,
16-7
Optimetrics module command 15-
2, 16-7
- CopySubdesign 26-43
- CopyTraceDefinitions 12-24
- CopyTracesData 12-23
- Core Global Script Context Com-
mands 28-1
- Count 4-12
- CoverLines 10-129
- CoverSurfaces 10-129
- CPython 1-44
- Create3DComponent 10-6
- Create3DDesign 9-8
- CreateArc 27-17
- CreateBondwire 10-11
- CreateBox 10-14
- CreateCircle 10-16, 27-17
- CreateCone 10-18
- CreateCutplane 10-20
- CreateCylinder 10-22
- CreateEllipse 10-24
- CreateEntityList 10-130, 10-137
- CreateEquationCurve 10-26
- CreateEquationSurface 10-29
- CreateFaceCS 10-131, 10-138
- CreateFieldPlot 18-20
- CreateGroup 10-135

CreateHelix 10-31
 CreateImage 27-18
 CreateLine 27-19
 CreateObjectFromEdges 10-144
 CreateObjectFromFaces 10-145
 CreateOutputVariable 11-1
 CreatePage 26-44
 CreatePin 27-19
 CreatePoint 10-33
 CreatePolygon 27-20
 CreatePolyline 10-41
 CreateRectangle 10-45, 27-21
 CreateRegion 10-47
 CreateRegularPolygon 10-52
 CreateRegularPolyhedron 10-50
 CreateRelativeCS 10-146
 CreateReport 12-26
 CreateReportFromFile 12-25
 CreateReportFromTemplate 12-32
 CreateReportofAllQuantities 12-33
 CreateSphere 10-55
 CreateSpiral 10-57
 CreateText 27-21
 CreateTorus 10-58
 CreateUserDefinedModel 10-60
 CreateUserDefinedPart 10-34, 10-73
 CreateUserDefinedSolution 23-1
 Cut 27-22

CutDesign 2-36, 6-16

D

dataset commands

AddDataset 6-4
 DeleteDataset 6-17
 EditDataset 6-18
 ExportDataset 6-20
 ImportDataset 6-37

DeactivateOpen 26-46

DeactivateShort 26-47

Deembed 24-9

DeembedBack 24-10

DeembedFront 24-11

Definition Editor Script Commands 27-1

Definition Manager commands

AddDefinitionFromBlock 10-209
 AddDefinitionFromLibFile 10-214
 GetProjectMaterialNames 7-30

Definition Manager Script Commands 7-1

Delete 10-231, 26-47, 27-22

DeleteAllBoundaries 13-4

DeleteAllExcitations 13-4

DeleteAllParameters 21-11

DeleteAllReports 12-34

DeleteBoundaries 13-5

DeleteDataset 6-17

DeleteDesign 2-37, 6-17

DeleteEmptyGroups 10-148

- DeleteFieldPlot 18-37
- DeleteFieldVariation 9-9
- DeleteFullVariation 9-10
- DeleteLastOperation 10-149
- DeleteLinkedDataVariation 9-10
- DeleteMarker 12-34
- DeleteMarker[Fields Reporter] 18-37
- DeleteMotionSetup A
- DeleteNamedExpr 18-38, 19-16
- DeleteOp 14-2
- DeleteOutputVariable 11-2
- DeleteParameters 21-11
- DeletePolylinePoint 10-108
- DeleteProject 4-12
- DeleteReduceMatrix 21-11
- DeleteReduceOp 21-12
- DeleteReport 12-35
- DeleteSetups 15-3
 - Analysis module command 15-3
 - Optimetrics module command 16-7
- DeleteSolutionVariation 17-7
- Deletetraces 12-35
- DeleteUneditablePlot 18-38
- DeleteUserDefinedSolutions 23-2
- Design object commands 12-46, 12-47, 12-48, 12-49, 12-49, 12-60
 - AddCartesianLimitLine 12-4
 - AddCartesianXMarker 12-5
 - AddCartesianYMarker 12-6
 - AddDeltaMarker 12-7
 - AddMarker 12-8
 - AddMarkerToPlot 18-2
 - AddNote 12-8
 - AddTraceCharacteristics 12-10
 - AddTraces 12-11
 - AnalyzeAll 9-5
 - ApplyMeshOps 9-4
 - CalculatorRead 18-6, 19-6
 - ClearAllMarkers 12-21
 - ClearAllTraceCharacteristics 12-22
 - CloseAllWindows 4-10
 - CopyReportDefinition 12-24
 - CreateOutputVariable 11-1
 - CreateReport 12-26
 - CreateReportFromTemplate 12-32
 - DeleteAllReports 12-34
 - DeleteOutputVariable 11-2
 - DeleteReport 12-35
 - DeleteTraces 12-35
 - DoesOutputVariableExist 11-3
 - EditOutputVariable 11-4
 - ExportConvergence 9-20
 - ExportElementBasedHarmonicForce 9-20
 - ExportMeshStats 9-22

ExportPlotImageToFile 12-41, 18-54	GetVariableValue 6-36, 8-26
ExportPlotImageWithViewToFile 18-55	GetVariationVariableValue 9-32
ExportProfile 9-23	GetVersion 4-31
ExportReport 12-42	ImportIntoReport 12-64
ExportToFile 12-44	Is2D 9-32
Fast Transformation 9-45, 9-45	Is3D 9-33
GetDisplayType 12-54	PasteReports 12-68
GetLibraryDirectory 4-22	Redo 9-33
GetMatchedObjectName 10- 244	RenameDesignInstance 9-34
GetModelUnits 10-244	RenameReport 12-70
GetModule 9-28	RenameTraces 12-70
GetName 9-29, 24-25	RunToolkit 6-46, 9-56
GetObjectsByMaterial 10-242	SavePlotSettingsAsDefault 12-72
GetObjectsInGroup 10-243	SetActiveEditor 9-35
GetOutputVariableValue 11-5	SetConductivityThreshold 9-36
GetProjectDirectory 4-26	SetLibraryDirectory 4-49
GetProperties 6-33, 8-23	SetProjectDirectoryVBCommand> 4-49
GetPropertyValue 6-34, 8-24, 12-56	SetPropertyValue 6-53, 8-26
GetPropEvaluatedValue 6-30	SetTempDirectory 4-51
GetPropSIValue 6-31	SetupYConnection 13-76
GetRegistryInt 4-56	SetVariableValue 6-54, 8-28
GetRegistryString 4-57	Show Layout 9-52, 9-53
GetSelections 10-245	ShowWindow 10-197
GetSolutionType 9-31	SimulateLink 9-54
GetTempDirectory 4-29	Solve 9-55, 9-56
GetVariables 6-37, 8-25	UpdateAllReports 12-72
	ValidateDesign 9-57
	Design Object Script Commands 9-1

Desktop Commands For Registry
Values 4-54

Desktop object commands

AddMessage 4-5

ClearMessages 4-6

CloseProject 4-10

CloseProjectNoForce 4-11

Count 4-12

DeleteProject 4-12

EnableAutosave 4-13

ExportOptionsFiles 4-13

GetActiveProject 4-14

GetAutosaveEnabled 4-15

GetBuildTimeDateString 4-15

GetChildNames 6-22

GetChildTypes 6-24

GetCustomMenuSet 4-16

GetDesigns 4-18, 6-26

GetDesktopConfiguration 4-19

GetDistributedAnalysisMachines
4-20

GetDis-
trib-
utedAna-
lysisMachinesForDesignType
4-20

GetName 4-24

GetProjectList 4-27

GetProjects 4-23, 4-26

GetPropNames 6-31

GetSchematicEnvironment 4-27

LaunchJobMonitor 4-31

NewProject 4-32

OpenMultipleProjects 4-33

OpenProject 4-34

PauseRecording 4-35

PauseScript 4-36

Print 4-36

QuitApplication 4-37

RefreshJobMonitor 4-37

ResetLogging 4-38

RestoreWindow 4-40

ResumeRecording 4-40

RunProgram 4-41

RunScript 4-42

SelectScheduler 4-45

SetActiveProject 4-46

SetActiveProjectByPath 4-46

SetCustomMenuSet 4-47

SetDesktopConfiguration 4-48

SetSchematicEnvironment 4-50

Sleep 4-52

SubmitJob 4-53

TileWindows 4-54

Desktop Object Script Commands 4-1

Desktop Scripting Conventions 1-56

DetachFaces 10-149

DisableDiffPairs 24-12

DisplayPinNames 27-23

DistributedAnalyzeSetup, Opti-
metrics module command 16-
8

DoesNamedExpressionExists 18-
39

DoesOutputVariableExist 11-3

Draw Menu commands 10-4

DuplicateAlongLine 10-109

DuplicateAroundAxis 10-111

DuplicateMirror 10-113

E

Edit 7-55, 7-101, 7-127, 7-156, 7-
179

Edit Menu Commands 10-107

Edit3DComponent 10-80, 10-81

EditCharge 13-60

EditCoilTerminal 13-61

EditCurrent 13-61

EditCurrentDensity 13-62

EditCurrentDensityTerminal 13-63

EditCylindricalGapOp 14-17

EditCylindricalHField 13-34

EditDataset 6-18

EditDependent 13-35

EditEntityList 10-151, 10-156

EditFaceCS 10-152

EditFloating 13-64

EditFluxTangential_ 13-35

EditForce 21-13

EditImpedance 13-36

EditIndependent 13-36

EditInsulating 13-37

EditLayoutForce 21-13

EditLengthOp 14-22

EditMatrix 21-15

EditModelResolutionOp 14-24

EditMotionSetup E

EditNotes 9-11

EditObjectCS 10-157

Editor object commands

SetPropertyValue 6-53, 8-26

Editor Scripting IDs 26-4

EditOutputVariable 11-4

EditPolyline 10-84

EditRadiation 13-38

EditReduceOp 21-15

EditRelativeCS 10-163

EditResistiveSheet 13-38

EditSetup

Analysis module command 15-3

optimization command 16-24

parametric command 16-17

sensitivity command 16-38

statistical command 16-45

EditSink 13-64

EditSkinOp 14-25

EditSolverOnDemandModel 7-74

EditSurfaceMeshSummaryData 18-39

- EditSymmetry 13-40
- EditTangentialHField 13-41
- EditThinLayer 13-41
- EditTorque 21-16
- EditTouching 13-43
- EditTrueSurfOp 14-26
- EditUserDefinedSolution 23-3
- EditVectorPotential 13-44
- EditVoltage 13-64
- EditVoltageAPhi 13-66
- EditVoltageDrop 13-67
- EditVolumeChargeDensity 13-67
- EditWindingGroup 13-67
- EditWithComps 7-74, 7-101, 7-128, 7-156, 7-184
- EditZeroTangentialHField 13-45
- ElectricRuleCheck 26-49
- EnableAutoSave 4-13
- EnableDiffPairs 24-13
- EnableHarmonicForceCalculation 9-11
- EnterComplex 18-43, 19-17
- EnterComplexVector 18-43, 19-17
- EnterCoord 18-44
- EnterEdge 18-45
- EnterLine 18-45, 19-18
- EnterOutputVar 18-46
- EnterPoint 18-46, 19-19
- EnterQty 18-47, 19-19
- EnterScalar 18-48, 19-20
- EnterScalarFunc 18-48, 19-20
- EnterSurf 18-49, 19-21
- EnterVector 18-49, 19-21
- EnterVectorFunc 18-50, 19-22
- EnterVol 18-50, 19-23
- Example Scripts 29-1
- Export 7-83, 7-110, 7-138, 7-171, 7-190, 10-165
- ExportCircuit 15-10
- ExportCitiFile 24-13
- ExportConvergence 9-20
- ExportDataset 6-20
- ExportDXConfigFile, Optimetrics module command 16-9
- ExportFieldPlot 18-51
- ExportFile 27-23
- ExportFullWaveSpice 7-115
- ExportHarmonicTransientForce 9-21
- ExportIcepak, Analysis module command 15-14
- ExportImageToFile 12-36
- ExportMarkerTable 12-45, 18-52
- ExportMaterial 7-31
- ExportMatlab 24-15
- ExportMesh Stats 9-22
- ExportNetlist 26-51
- ExportOptimetricsProfile, Optimetrics module command 16-9
- ExportOptimetricsResults, Optimetrics module command 16-10

ExportOptionsFiles 4-13

ExportParametricResults, Opti-
metrics module command 16-
11

ExportPlotImageToFile 12-41, 18-
54

ExportPlotImageWithViewToFile
18-55

ExportProfile 9-20, 9-23

ExportReport 12-42

ExportScript 7-195

ExportSolnData, Analysis module
command 15-15

ExportSpreadsheet 24-17

ExportSurfaceMeshSummary 18-
56

ExportToFile 12-43, 18-57

ExportTouchstone 24-19

ExportTouchstone2 24-21

ExportTransientData 17-10

Extract 24-23

F

FFTONReport 12-45, 17-11

Field Calculator commands

 DoesNamedExpressionExists
 18-39

Field Overlay module commands,
 GetFieldPlotNames 18-58

field overlay script commands 18-1

Field Overlays module commands

 CalcOp 18-5, 19-5

CalcStack 18-5, 19-7

ChangeGeomSettings 18-14, 19-11

ClcMaterial 18-17, 19-14

ClearAllNamedExpr 18-19, 19-15

CopyNamedExprToStack 18-19, 19-16

CreateFieldPlot 18-20

DeleteFieldPlot 18-37

DeleteNamedExpr 18-38, 19-16

DeleteUneditablePlot 18-38

EditSurfaceMeshSummaryData 18-39

EnterComplex 18-43, 19-17

EnterComplexVector 18-43, 19-17

EnterCoord 18-44

EnterEdge 18-45

EnterLine 18-45, 19-18

EnterOutputVar 18-46

EnterPoint 18-46, 19-19

EnterQty 18-47, 19-19

EnterScalar 18-48, 19-20

EnterScalarFunc 18-48, 19-20

EnterSurf 18-49, 19-21

EnterVector 18-49, 19-21

EnterVectorFunc 18-50, 19-22

EnteVol 18-50, 19-23

ExportFieldPlot 18-51

ExportOnGrid 18-52

ExportSurfaceMeshSummary 18-56

GetFieldFolderNames 18-58

- GetFieldPlotQuantityName 18-59
- GetMeshPlotNames 18-59
- HidePolarPlot 18-62
- HideRadiatedPlotOverlay 18-63
- ModifyFieldPlot 18-60, 18-64
- ReassignFieldPlot 18-67
- RenameFieldPlot 18-69
- RenamePlotFolder 18-70
- SaveFieldsPlots 18-70
- SetFieldPlotSettings 18-71, 18-73
- SetPlotFolderSettings 18-75
- SetPlotsViewSolutionContext 18-79
- UpdateAllFieldsPlots 18-80
- UpdateQuantityFieldsPlots 18-81
- Fields Calculator commands
 - AddNamedExpr 18-3, 19-2
 - CalcOp 18-5, 19-5
 - CalcStack 18-5, 19-7
 - CalculatorWrite 18-7, 19-8
 - ChangeGeomSettings 18-14, 19-11
 - ClcEval 18-14, 19-12
 - ClcMaterial 18-17, 19-14
 - ClcMaterialValue 18-18
 - ClearAllNamedExpr 18-19, 19-15
 - CopyNamedExprToStack 18-19, 19-16
 - DeleteNamedExpr 18-38, 19-16
 - EnterComplex 18-43, 19-17
 - EnterComplexVector 18-43, 19-17
 - EnterCoord 18-44
 - EnterEdge 18-45
 - EnterLine 18-45, 19-18
 - EnterOutputVar 18-46
 - EnterPoint 18-46, 19-19
 - EnterQty 18-47, 19-19
 - EnterScalar 18-48, 19-20
 - EnterScalarFunc 18-48, 19-20
 - EnterSurf 18-49, 19-21
 - EnterVector 18-49, 19-21
 - EnterVectorFunc 18-50, 19-22
 - EnterVol 18-50, 19-23
 - ExportOnGrid 19-23
 - ExportToFile 19-25
- Fields Calculator Script Commands 19-1
- Fields reporter module commands
 - AddMarker[Fields Reporter] 18-3
 - ClearAllMarkers[Fields Reporter] 18-18
 - DeleteMarker[Fields Reporter] 18-37
 - ExportMarkerTable 12-45, 18-52
- Fillet 10-170
- FindElements 26-52
- FitToBorder 26-54
- FlattenGroup 10-171

FlipHorizontal 26-56, 27-24

FlipVertical 26-57, 27-24

G

general method script
 commands 26-31

GenerateHistory 10-172

GenerateVariationData Parametric,
 parametric command 16-18

Get3DComponentDefinitionNames
 10-88

Get3DComponentInstanceNames
 10-89

Get3DComponentMaterialNames
 10-89

Get3DCom-
 ponentMaterialProperties 10-
 90

Get3DComponentParameters 10-
 88

GetActiveCoordinateSystem 10-
 176

GetAct-
 iveCoordin-
 ateSystemTransform 10-177

GetActiveDesign 2-38, 6-21

GetActiveProject 4-14

GetAdaptiveSettings 17-12

GetAllCategories 12-47

GetAllQuantities 12-47

GetAllReportNames 12-46

GetAllSourceMagnitudes 17-12

GetAllSourceModes 17-13

GetAllSourcePhases 17-13

GetAllSources 17-14

GetAntennaParameters 17-14

GetArrayVariables 6-22, 8-22

GetAutoSaveEnabled 4-15

GetAvailableDisplayTypes 12-48

GetAvailableReportTypes 12-49

GetAvailableSolutions 12-49

GetAvailableVariations 17-15

GetBodyNamesByPosition 10-232

GetBoundaries 13-6

GetBoundariesOfType 13-7

GetBoundaryAssignment 13-5

GetBuildDateTimeString 4-15

GetChildNames 6-22

GetChildObject Design 6-23

GetChildTypes 6-24

GetComponentName 25-2

GetComponentPinLocation 26-90

GetCoordinateSystems 10-177

GetCoreLossEffect 13-7

GetCoreLossEffectOnField 13-8

GetDataUnits 12-50

GetDefinitionManager 2-38, 6-24

GetDependentFiles 2-39, 6-25

GetDesign 2-40, 2-40

GetDesigns 4-18, 6-26

GetDesignValidationInfo 9-24

GetDesignVariableNames 12-51

GetDesignVariableUnits 12-51	GetFieldPlotNames, Field Overlay module command 18-58
GetDesignVariableValue 12-52	GetFieldPlotQuantityName 18-59
GetDesignVariationKey 12-53	GetFieldType 17-16
GetDisplacementCurrent 13-8	GetFrequencies 24-24
GetDisplayType 12-54	GetFrequencyCount 24-25
GetDistributedAnalysisMachines 4- 20	GetGeo- metryId- sForAllNetLayerCombinations 9-25
GetDis- trib- utedAna- lysisMachinesForDesignType 4-20	GetGeo- metryId- sForNetLayerCombinations 9- 26
GetDocumentNames 22-5	GetGeometryMode 9-27
GetEddyEffect 13-8	GetGeometryModelerMode 10-239
GetEdgeByPosition 10-233	GetImagDataValues 12-54
GetEdgeIDsFromFace 10-234	GetIncludePortPostProcessing 17-16
GetEdgeIDsFromObject 10-235	GetInstanceID 25-3
GetEditorName [Schematic] 26-90	GetInstanceName 25-3
GetEntityIDsCon- tainedByNamedSelection 10- 235	GetLayerAndNetOfMeshBody 17-17
GetEntityListIDByName 10-236	GetLibraryDirectory 4-22
GetEvaluatedPropertyValue 26-81	GetMatchedObjectName 10-244
GetExcitations 13-9	GetMeshBodiesOnLayerAndNet 17- 19
GetExcitationsOfType 13-9	GetMeshPlotNames 18-59
GetExeDir 4-21	GetModelBoundingBox 10-240
GetFaceArea 10-236	GetModelUnits 10-244
GetFaceByPosition 10-237	GetModule 9-28
GetFaceCenter 10-237	GetMotionSetupNames G
GetFaceIDs 10-239	GetMultipactionBreakdown 17-21
GetFieldFolderNames 18-58	

GetName 2-41, 4-24, 6-28, 9-29, 24-25	GetPortCount 24-26
GetNames 7-84, 7-89, 7-111, 7-139, 7-172, 7-191	GetPortNumber 24-27
GetNetConnections 26-91	GetPostProcSettings 24-28
GetNetworkDataSolution 17-22	GetProcessID 4-25
GetNetworkDataSolutionDefinition 17-23	GetProjectDirectory 4-26
GetNetworkPostprocSetup 17-23	GetProjectList 4-27
GetNumBoundaries 13-10	GetProjectMaterialNames 7-30
GetNumBoundariesOfType 13-11	GetProjects 4-23, 4-26
GetNumExcitations 13-11	GetProperties 6-33, 7-90, 8-23, 26-81, 27-25
GetNumExcitationsOfType 13-12	GetPropertyValue 6-34, 8-24, 12-56, 27-25
GetNumObjects 10-245	GetPropEvaluatedValue 6-30
GetNumPages 26-92	GetPropHost 25-4, 25-4
GetObjectBoundingBox 10-240	GetPropNames 6-31
GetObjectIDByName 10-241	GetPropServerName 25-5
GetObjectName 10-241	GetPropSIValue 6-31
GetObjectNameByFaceID 10-242	GetPropValue Project 6-32
GetObjectsByMaterial 10-242	GetRealDataValues 12-58
GetObjectsIngroup 10-243	GetRegistryInt 4-56
GetObjPath 6-29	GetRegistryString 4-57
GetOperationNames 14-2	GetReportTraceNames 12-58
GetOutputVariableValue 11-5	GetSelections 9-31, 10-245
GetParentDesign 25-4	GetSetupCount 15-16
GetPath 2-42, 6-29	GetSetupCount, Analysis module command 15-16
GetPer-Quant-ityPrimarySweepValues 12-55	GetSetupNames (Optimetrics), Optimetrics module command 16-11
GetPersonalLibDirectory 4-25	GetSetupNamesByType (Optimetrics), Optimetrics module command 16-12

- GetSetups 15-17
 - Analysis module command 15-16
 - GetSolutionContexts 12-59
 - GetSolutionDataPerVariation 12-60
 - GetSolutionType 9-31
 - GetSolverOnDemandData 7-85
 - GetSolverOnDemandModelList 7-85
 - GetSweepCount 15-17
 - GetSweepCount, Analysis module command 15-17
 - GetSweepNames 12-61
 - GetSweeps 15-18
 - Analysis module command 15-18
 - GetSweepUnits 12-62
 - GetSweepValues 12-63
 - GetSymmetryMultiplier G
 - GetSysLibDirectory 4-29
 - GetTempDirectory 4-29
 - GetTerminalExcitationType 17-24
 - GetTool 4-59
 - GetTopDesignList 2-42, 6-35
 - GetTopEntryValue 18-62, 19-27
 - GetTransientSolveTimes 17-24
 - GetUserDefinePartVersion 10-90
 - GetUserLibDirectory 4-30
 - GetUserPosition 10-246
 - GetValidISolutionList 17-25
 - GetVariables 6-37, 8-25
 - GetVariableValue 6-36, 8-26
 - GetVariation 24-29
 - GetVariationVariableValue 9-32
 - GetVersion 4-31
 - GetVertexIDsFromEdge 10-247
 - GetVertexIDsFromFace 10-247
 - GetVertexIDsFromObject 10-248
 - GetVertexPosition 10-248
 - GetVisibleLevels 27-25
 - GridSetup 26-54, 27-26
 - GroupPlotCurvesByGroupingStrategy 12-64
- ## H
- HasSameData 24-29
 - HealObject 10-173
 - HidePolarPlot 18-62
 - HideRadiatedPlotOverlay 18-63
 - hierarchy of variables in HFSS 1-48
- ## I
- Icepak boundary condition commands
 - ImportIDX 4-70
 - Import (3D Modeler command) 10-178
 - ImportANF 4-60, 4-61
 - ImportAutoCAD 4-61
 - ImportAWRMicrowaveOffice 4-62
 - ImportDataset 6-37

ImportDXF 10-180
 ImportEDB 4-63
 ImportExport Tool 4-59
 ImportExtracta 4-64
 ImportGDSII 4-65, 4-65, 10-184
 ImportIDF 4-66, 4-69
 ImportIDX 4-70
 ImportIntoReport 12-64
 ImportIPC 4-72
 ImportODB 4-73
 ImportSetup, Optimetrics module
 command 16-13
 ImportXFL 4-73
 Information Method List 26-85
 InitialMeshSettings 14-28
 Insert3DComponent 10-91
 InsertDesign 2-43, 6-39
 InsertDesignWithWorkflow 2-44, 6-
 40
 InsertNativeComponent 10-93
 InsertSetup
 Analysis module command 15-
 18
 optimization command 16-29
 parametric command 16-18
 sensitivity command 16-42
 statistical command 16-45
 Intersect 10-187
 IronPython 1-2
 Is2D 9-32

Is3D 9-33
 IsDataComplex 12-65
 IsPerQuantityPrimarySweep 12-66
 IsUsed 7-85, 7-90, 7-111, 7-139, 7-172, 7-
 191

L

LaunchJobMonitor 4-31
 LoadNamedExpressions 18-63, 19-27
 LoadSolution 24-31

M

material commands
 AddDefinitionFromBlock 10-209
 AddDefinitionFromLibFile 10-214
 AddMaterial 7-2, 7-14
 CloneMaterial 7-12
 ExportMaterial 7-31
 RemoveMaterial 6-43, 7-32
 Materials Scripting Support 2-22
 Maxwell boundary condition commands
 AssignCurrent 13-50
 AssignFluxTangential_1 13-21
 AssignResistiveSheet 13-26
 AssignThinLayer 13-29
 AssignTouching 13-32
 AssignVectorPotential 13-33
 AssignVoltageAPhi_ 13-56, 13-74
 EditCurrent 13-61
 EditFluxTangential 13-35

- EditResistiveSheet 13-38
- EditThinLayer 13-41
- EditTouching 13-43
- EditVectorPotential 13-44
- EditVoltageAPhi 13-66
- Maxwell definition manager commands
 - ComputeCoreLossCoefficients 7-13
- Maxwell parameter setup commands
 - AddReduceOp 21-2
 - EditReduceOp 21-15
- Mesh Operations module commands
 - AssignCylindricalGapOp 14-4
 - AssignLengthOp 14-10
 - AssignModelResolutionOp 14-12
 - AssignSkinDepthLayerSetting 14-13
 - AssignSkinDepthOp 14-14
 - AssignTrueSurfOp 14-16
 - DeleteOp 14-2
 - EditCylindricalGapOp 14-17
 - EditLengthOp 14-22
 - EditModelResolutionOp 14-24
 - EditSkinOp 14-25
 - EditTrueSurfOp 14-26
 - GetOperationNames 14-2
 - InitialMeshSettings 14-28
 - RenameOp 14-3
- Mesh Operations Module Script Commands 14-1
 - mesh operations script commands 14-2
 - method format, create schematic objects 26-1
- Mirror 10-115
- Model Manager Script Commands 7-91
- Modeler Menu Commands 10-122
- ModifyFieldPlot 18-60, 18-64
- ModifyInceptionParameters 18-66
- ModifyLibraries 7-197
- module script commands
 - boundary/excitation 13-2
 - field overlay 18-1
 - general method 26-31
 - mesh operations 14-2
- Module Setup commands
 - GetSymmetryMultiplier G
 - SetSymmetryMultiplier H
- modules in HFSS scripting 1-48
- Motion Setup commands
 - AssignBand B
 - DeleteMotionSetup A
 - EditMotionSetup E
 - GetMotionSetupNames G
 - ReassignMoving B
- Move 10-117, 27-27

MoveCSToEnd 10-188

MoveEntityToGroup 10-188

MoveFaces 10-189

MovePlotCurveToGroup 12-67

MovePlotCurveToNewGroup 12-67

N

Named Arguments 1-56

NameNets 26-59

NdExplorer

Script Commands 24-1

Network Data Explorer Manager
Commands 7-114

NewProject 4-32

O

oAnsoftApp object 1-48

Object Oriented Property
Scripting 2-1

oDesign object 1-48

oDesktop object 1-48

oEditor object 1-48

OffsetFaces 10-118

oModule object 1-48

Open 24-32

OpenAndConvertProject 4-32

OpenMultipleProjects 4-33

OpenProject 4-34

oProject object 1-48

Optimetrics module commands

DeleteSetups 16-7

DistributeAnalyzeSetup 16-8

ExportDXConfigFile 16-9

ExportOptimetricsProfile 16-9

ExportOptimetricsResults 16-10

ExportParametricResults 16-11

GetSetupNames 16-12

GetSetupNamesByType 16-12

ImportSetup 16-13

RenameSetup 16-14

SolveSetup 15-2, 16-7, 16-15, 16-16

Optimetrics Module Script Commands 16-1

optimization commands

EditSetup 16-24

InsertSetup 16-29

Optimization Script Commands 16-24

Other oEditor Commands 10-207

output variable commands

CreateOutputVariable 11-1

DeleteOutputVariable 11-2

DoesOutputVariableExist 11-3

EditOutputVariable 11-4

GetOutputVariableValue 11-5

Output Variable Script Commands 11-1

P

PageBorders 26-61

- PageSetup 10-249
- Pan 26-61, 27-27
- Parameter Setup commands
 - DeleteAllParameters 21-11
 - EditForce 21-13
 - EditMatrix 21-15
 - EditTorque 21-16
 - ReassignParameter 21-16
 - RenameParameter 21-17
- Parameter setup commands
 - DeleteReduceMatrix 21-11
 - DeleteReduceOp 21-12
 - RenameReduceMatrix 21-17
 - RenameReduceOp 21-18
- parametric commands
 - EditSetup 16-17
 - GenerateVariationData Parametric 16-18
 - InsertSetup 16-18
- Parametric Script Commands 16-16
- Paste 2-45, 4-35, 6-42, 26-62, 27-28
- Paste (Model Editor) 10-119
- Paste (Project Object) 2-45, 4-35, 6-42
- PasteData 26-63
- PasteDesign (Schematic Editor) 26-64
- PasteReports 12-68
- PasteSetup
 - Analysis module command 15-31
 - Optimetrics module command 16-14
- PasteSweep 15-31
- PasteTraces 12-69
- PauseRecording 4-35
- PauseScript 4-36
- Print 4-36
- Project object commands
 - AddDataset 6-4
 - AddMaterial 7-2, 7-14
 - ChangeProperty 27-6
 - CloneMaterial 7-12
 - Close 2-35, 6-15
 - CopyDesign 2-35, 6-15
 - CutDesign 2-36, 6-16
 - DeleteDataset 6-17
 - DeleteDesign 2-37, 6-17
 - EditDataset 6-18, 6-20, 6-37
 - ExportMaterial 7-31
 - GetActiveDesign 2-38, 6-21
 - GetArrayVariables 6-22, 8-22
 - GetChildObject Design 6-23
 - GetDependentFiles 2-39, 6-25
 - GetDesign 2-40
 - GetName 2-41, 6-28
 - GetObjPath 6-29
 - GetPath 2-42, 6-29

- GetProjectMaterialNames 7-30
- GetProperties 6-33, 8-23, 27-25
- GetPropertyValue 6-34, 8-24, 12-56, 27-25
- GetPropEvaluatedValue 6-30
- GetPropSIValue 6-31
- GetPropvalue Project 6-32
- GetTopDesignList 2-42, 6-35
- GetTopEntryValue 18-62, 19-27
- GetVariables 6-37, 8-25
- GetVariableValue 6-36, 8-26
- InsertDesign 2-43, 6-39
- LoadSolution 24-31
- Paste 2-45, 2-45, 4-35, 4-35, 6-42, 6-42
- Redo 2-46, 6-42
- RemoveAllUnusedDefinitions 6-43
- RemoveMaterial 6-43, 7-32
- RemoveUnusedDefinitions 6-44, 7-33
- Rename 2-46, 6-45
- RestoreProjectArchive 2-47, 4-39
- Save 2-48, 6-47
- SaveAs 2-48, 6-47
- SaveAsStandAloneProject 2-50, 6-49
- SaveProjectArchive 2-51, 6-50
- SetActiveDesign 2-52, 6-51
- SetPropertyValue 6-53, 8-26
- SetPropValue Project 6-52
- SetVariableValue 6-54, 8-28
- SimulateAll 2-33, 2-53, 6-6, 6-55
- Undo 2-53, 6-55
- UpdateDefinitions 2-54, 6-56
- Project Object Script Commands 2-30, 6-1
- ProjectSheet 10-191
- property commands
 - ChangeProperty 27-6
 - GetArrayVariables 6-22, 8-22
 - GetProjectMaterialNames 7-30
 - GetProperties 6-33, 8-23, 27-25
 - GetPropertyValue 6-34, 8-24, 12-56, 27-25
 - GetPropEvaluatedValue 6-30
 - GetPropSIValue 6-31
 - GetTopEntryValue 18-62, 19-27
 - GetVariables 6-37, 8-25
 - GetVariableValue 6-36, 8-26
 - SetPropertyValue 6-53, 8-26
 - SetVariableValue 6-54, 8-28
- Property Method List 26-79
- Property Script Commands 8-1
 - Conventions uSed in thie Chapter 8-12
 - Object Script Property Function Summary 8-3
- PurgeHistory 10-192
- PushExcitations 26-65
 - Layout Editor 26-65

Q

QuitApplication 4-37

R

ReassignBoundaries 13-3, 13-12

ReassignBoundary 13-12

ReassignFieldPlot 18-67

ReassignMoving B

ReassignParameter 21-16

Redo 27-29

 design-level command 9-33

 project-level command 2-46, 6-42

RefreshJobMonitor 4-37

ReleaseData 12-69

Remove 7-86, 7-112, 7-139, 7-172, 7-191

RemoveAllUnusedDefinitions 6-43

RemoveAssignmentFromBoundary 13-13

RemoveLevels 27-29

RemoveMaterial 6-43, 7-32

RemovePort 27-29

RemoveScript 7-196

RemoveSolverOnDemandModel 7-87

RemoveUnused 7-87, 7-91, 7-113, 7-141, 7-173, 7-192

RemoveUnusedDefinitions 6-44, 7-33

Rename 2-46, 6-45, 24-33

RenameBoundary 13-14

RenameDesignInstance 9-34

RenameFieldPlot 18-69

RenameOp 14-3

RenameParameter 21-17

RenamePart 10-250

RenamePlotFolder 18-70

RenameReduceMatrix 21-17

RenameReduceOp 21-18

RenameReport 12-70

RenameSetup 15-32

 Analysis module command 15-32

 Optimetrics module command 16-14

RenameTraces 12-70

Renormalize 24-34

Reorder 24-35, 24-36

Reporter editor commands

 AddCartesianLimitLine 12-4

 AddCartesianXMarker 12-5, 12-6

 AddDeltaMarker 12-7

 AddMarker 12-8

 AddMarkerToPlot 18-2

 AddNote 12-8

 AddTraceCharacteristics 12-10

 AddTraces 12-11

 ChangeProperty 12-13

 ClearAllMarkers 12-21

ClearAllTraceCharacteristics 12-22	GetSolutionContexts 12-59, 12-60
CopyReportDefinition 12-24	GroupPlotCurvesByGroupingStrategy 12-64
CopyReportsData 12-23	MovePlotCurveToGroup 12-67
CopyTraceDefinitions 12-24	MovePlotCurveToNewGroup 12-67
CopyTracesData 12-23	PasteReports 12-68
CreateReport 12-26	PasteTraces 12-69
CreateReportFromFile 12-25	RenameReport 12-70
CreateReportFromTemplate 12- 32	RenameTraces 12-70
CreateReportOfAllQuantities 12-33	ResetPlotSettings 12-71
DeleteAllReports 12-34	SavePlotSettingsAsDefault 12-72
DeleteReport 12-35	UnGroupPlotCurvesInGroup 12-77
DeleteTraces 12-35	UpdateAllReports 12-72
ExportImageToFile 12-36	UpdateReports 12-73
ExportPlotImageToFile 12-41, 18-54, 18-55	UpdateTraces 12-73
ExportReport 12-42	UpdateTracesContextAndSweeps 12- 75
ExportToFile 12-44, 12-64	Reporter Editor Script Commands 12-1
FFTONReport 12-45, 17-11	ReprioritizeBoundaries 13-15
GetAllCategories 12-47	Reset 24-37
GetAllQuantities 12-47	ResetAllToTimeZero 15-32
GetAllReportNames 12-46	ResetLogging 4-38
GetAvailableDisplayTypes 12- 48	ResetPlotSettings 12-71
GetAvailableReportTypes 12-49	ResetSetupToTimeZero 15-33
GetAvailableSolutionss 12-49	RestoreWindow 4-40
GetDisplayType 12-54	RestToTimeZero 9-35
GetReportTraceNames 12-58	ResumeRecording 4-40
	RevertAllToInitial 15-33
	RevertSetupToInitial 15-34

RevertSetupToInitialTemperature
15-35

Rotate 10-120, 26-67, 27-30

Running Instance Manager Script
Commands 5-1

RunProgram 4-41

RunScript 4-42

RunScriptWithArguments 4-43

RunToolkit 6-46, 9-56

S

sample scripts

variable helix 29-6

Save 2-48, 6-47, 27-30

SaveAs 2-48, 6-47

SaveAsStandAloneProject 2-50, 6-
49

SaveFieldsPlots 18-70

SaveNamedExpressions 18-73, 19-
28

SavePlotSettingsAsDefault 12-72

Scale 10-121

Schematic Scripting 26-1

Create Method List 26-5

Script and Library Scripts 7-193

script commands

boundary/excitation 13-2

field overlay 18-1

general method script
commands 26-31

mesh operations 14-2

Script Commands for Creating and
Modifying Excitations in
Maxwell 13-45

Script Commands for Creating and
Modifying Boundaries in Max-
well 13-16

Script Commands for Creating and
Modifying Mesh Operations 14-
4

scripts

CPython 1-44

IronPython 1-2

overview 1-1

pausing 1-52

recording 1-53

running 1-51

stopping 1-52

Section 10-193

SelectAll 26-69, 27-31

SelectPage 26-70

SelectScheduler 4-44

SendToBack 26-73, 27-31

sensitivity commands

EditSetup 16-38

InsertSetup 16-42

Sensitivity Script Commands 16-38

SeparateBody 10-194

SetActiveDefinitionEditor 2-52, 6-51

SetActiveDesign 2-52, 6-51

SetActiveEditor 9-35

SetActiveLevel 27-32

SetActiveProject 4-46
SetActiveProjectByPath 4-46
SetAllPortImpedance 24-38
SetConductivityThreshold 9-36
SetFastTransformationForLayoutComponent 9-45, 9-45
SetFieldPlotSettings 18-71, 18-73
SetInitialLevels 27-33
SetLibraryDirectory 4-49
SetMinimumTimeStep 13-74
SetModelUnits 10-195
SetPageData 26-70
SetPermeabilityComputation 13-78
SetPlotFolderSettings 18-75
SetPlotsViewSolutionContext 18-79
SetPortDeembedDistance 24-39
SetPortImpedance 24-39
SetPostProcSettings 24-40
SetProjectDirectory 4-49
SetPropertyValue 6-53, 8-26, 27-33
SetPropValue Project 6-52
SetShowLayoutForLayoutComponent 9-52, 9-53
SetSymmetryMultiplier H
SetTempDirectory 4-51
Setting Numerical Values 1-58
SetupYConnection 13-76
SetVariableValue 6-54, 8-28
SetVisibleLevels 27-34
SetWCS 10-196
SGetAppDesktop 3-1
ShowWindow 10-197
ShowVariableBlock 26-73
SimulateAll 2-33, 2-53, 6-6, 6-55
SimulateLink 9-54
Sleep 4-52
Smooth 24-41
Solutions module commands
 ConstructVariationString 9-6
 DeleteFieldVariation 9-9
 DeleteFullVariation 9-6, 9-10
 DeleteLinkedDataVariation 9-10
 DeleteSolutionVariation 17-7
 ExportTransientData 17-10
 GetAdaptiveSettings 17-12
 GetAllSourceMagnitudes 17-12
 GetAllSourceModes 17-13
 GetAllSourcePhases 17-13
 GetAllSources 17-14
 GetAntennaParameters 17-14
 GetAvailableVariations 17-15
 GetFieldType 17-16
 GetIncludePortPostProcessing 17-16
 GetLayerAndNetOfMeshBody 17-17
 GetMeshBodiesOnLayerAndNet 17-19

- GetMultipactionBreakdown 17-21
- GetNetworkDataSolution 17-22
- GetNetworkDataSolutionDefinition 17-23
- GetNetworkPostprocSetup 17-23
- GetTerminalExcitationType 17-24
- GetTransientSolveTimes 17-24
- GetValidISolutionList 17-25
- TDROnReport 17-25
- Solutions Module commands
 - ExportNetworkData 17-8
- Solutions Module Script Commands 17-1
- Solve 9-55
- SolveAllSetup, Optimetrics module command 16-16
- SolveSetup, Optimetrics module command 16-15
- SortComponents 26-74
- Split 10-197
- statistical commands
 - EditSetup 16-45
 - InsertSetup 16-45
- Statistical Script Commands 16-45
- Stretch 24-43
- Structures 1-30
- SubmitJob 4-52
- Subtract 10-199
- SweepAlongPath 10-98
- SweepAlongVector 10-100
- SweepAroundAxis 10-101
- SweepFacesAlongNormal 10-103, 10-103, 10-200, 10-200
- Sweep-FacesAlongNormalWithAttributes 10-104
- Symbol Editor Scripts 27-1
- Symbol Manager Script Commands 7-118
- T**
- TDROnReport 17-25
- Terminate 24-44
- ThickenSheet 10-202
- ToggleLevel 27-34
- U**
- UDM Structures 1-30
- UncoverFaces 10-203
- Undo 27-35
 - project-level command 2-53, 6-55
- Ungroup 10-205
- UnGroupPlotCurvesInGroup 12-77
- Unite 10-204
- UpdateAllFieldsPlots 18-80
- UpdateAllReports 12-72
- UpdateComponentDefinition 10-106
- UpdateDefinitions, project-level command 2-54, 6-56
- UpdateDynamicLink 7-88

UpdateQuantityFieldsPlots 18-81

UpdateReports 12-73

UpdateTraces 12-73

UpdateTracesContextAndSweeps
12-75

User defined documents module
commands

 GetDocumentNames 22-5

User defined solution module com-
mands

 CreateUserDefinedSolution 23-
 1

 DeleteUserDefinedSolutions 23-
 2

 EditUserDefinedSolution 23-3

User Defined Solutions
Commands 23-1

V

ValidateDesign 9-57

variables

 hierarchy in HFSS 1-48

 used in HFSS scripts 1-48

W

Wire (Schematic Editor) 26-75

WrapSheet 10-206

Z

ZoomArea 26-76, 27-35

ZoomIn 26-77, 27-37

ZoomOut 26-77, 27-37