



POWERING INNOVATION THAT DRIVES HUMAN ADVANCEMENT

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

Twin Builder® 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, contact ANSYS, Inc.

Table of Contents

Table of Contents	Contents-1
1 - Introduction to Twin Builder Scripting	1-1
Overview of Twin Builder Script Variables	1-1
oAnsoftApp	1-2
oDesktop	1-2
oProject	1-2
oDesign	1-3
oEditor	1-3
oModule	1-4
Recording a Script to a File	1-4
Recording a Script to a Project	1-5
Working with Project Scripts	1-5
Stopping Script Recording	1-5
Running a Script	1-5
Running Scripts in an HPC Environment	1-6
Pausing and Resuming a Script	1-7
Stopping a Script	1-7
Modifying a Script for Easier Playback	1-8
Event Callback Scripting	1-8
Simple and Composite Names	1-9
Aborting Scripts	1-10
Executing a Script from Within a Script	1-10
Editing Properties	1-10
Unused Script Commands	1-11
Twin Builder Scripting Conventions	1-11
Syntax Conventions	1-11

Script Command Conventions	1-12
Script Command Name	1-12
2 - Twin Builder and IronPython	2-1
Controlling Program Execution	2-1
Conditional Statements	2-1
For Loop	2-2
While Loop	2-2
Python Operators	2-2
Operator Precedence	2-2
Arithmetic Operators	2-3
Comparison Operators	2-3
Logical Operators	2-4
Python Variables	2-4
Assigning Values to Variables	2-4
Python Lists	2-4
Converting between Data Types	2-5
Recommended Python and IronPython References	2-6
A Sample Twin Builder Script	2-6
3 - Application Object Script Commands	3-1
GetAppDesktop	3-1
4 - Twin Builder Application Object Script Commands	4-1
GetProcessResources	4-1
RecompileLibraries	4-1
ConvertSMLToSMT	4-2
SimulateSMT	4-2
5 - Desktop Object Script Commands	5-1
AddMessage	5-1

ClearMessages	5-2
CloseAllWindows	5-2
CloseProject	5-3
CloseProjectNoForce	5-3
DeleteProject	5-4
EnableAutoSave	5-4
ExportOptionsFiles	5-5
GetActiveProject	5-5
GetAutoSaveEnabled	5-6
GetBuildDateTimeString	5-6
GetCustomMenuSet	5-7
GetDesktopConfiguration	5-9
GetDistributedAnalysisMachines	5-9
GetDistributedAnalysisMachinesForDesignType	5-10
GetExeDir	5-10
GetMessages	5-11
GetLibraryDirectory	5-12
GetPersonalLibDirectory	5-12
GetProcessID	5-13
GetProjectDirectory	5-13
GetProjectList	5-14
GetProjects	5-14
GetSchematicEnvironment	5-14
GetSysLibDirectory	5-15
GetTempDirectory	5-15
GetUserLibDirectory	5-16
GetVersion	5-16

LaunchJobMonitor	5-17
NewProject	5-17
OpenProject	5-18
PauseRecording	5-18
PauseScript	5-19
Print	5-19
QuitApplication	5-20
RefreshJobMonitor	5-20
RestoreWindow	5-21
ResumeRecording	5-21
RunProgram	5-22
RunScript	5-22
RunScriptWithArguments	5-24
SelectScheduler	5-25
SetActiveProject	5-26
SetActiveProjectByPath	5-26
SetCustomMenuSet	5-27
SetDesktopConfiguration	5-27
SetLibraryDirectory	5-28
SetProjectDirectory	5-28
SetSchematicEnvironment	5-29
SetTempDirectory	5-29
ShowDockingWindow	5-30
Sleep	5-30
SubmitJob	5-31
6 - Running Instances Manager Script Commands	6-1
GetAllRunningInstances	6-1

GetRunningInstanceByProcessID	6-1
GetRunningInstanceByProject	6-2
7 - Project Object Script Commands	7-1
AnalyzeAll [Project]	7-1
ClearMessages	7-1
Close [Project object command]	7-2
CopyDesign	7-2
CutDesign	7-3
DeleteDesign	7-3
GetActiveDesign	7-4
GetDefinitionManager	7-4
GetDependentFiles	7-5
Get Design	7-5
GetDesigns	7-6
GetName [Project]	7-6
GetPath [Project]	7-7
GetProperties [Project]	7-7
GetTopDesignList	7-8
InsertDesign	7-8
Paste [Project]	7-9
Redo [Project]	7-9
Rename [Project]	7-10
Save [Project]	7-10
SaveProjectArchive	7-11
SaveAs [Project]	7-11
SetActiveDefinitionEditor	7-12
SetActiveDesign	7-12

SimulateAll	7-13
Undo [Project]	7-13
UpdateDefinitions	7-14
8 - Object Oriented Property Scripting	8-1
Child Modules Available for Twin Builder Design	8-2
Additional details on Solve setup	8-2
9 - Definition Manager Script Commands	9-1
AddMaterial	9-1
EditMaterial	9-2
ExportMaterial	9-3
GetData [Material Manager]	9-4
GetNames [Material Manager]	9-4
GetProperties [Material Manager]	9-5
IsUsed [Material Manager]	9-5
RemoveUnused [Material Manager]	9-6
RemoveMaterial	9-6
Component Manager Script Commands	9-7
Add [Component]	9-7
AddFMUModel [Component]	9-15
AddMotorCADROM [Component]	9-17
AddPythonModel [Component]	9-18
AddTwinModel	9-20
Edit [Component]	9-21
Export [Component]	9-32
ExportAllProjectComps [Component]	9-32
ExportDefs [Component]	9-33

ExportDefinitionArchive [Component]	9-34
GetData [Component]	9-34
GetLibContents [Component]	9-35
GetModelicaObjectList [Component]	9-35
GetNames [Component]	9-36
ImportDefinitionArchive [Component]	9-36
ImportModelsFromFile [Component]	9-37
IsUsed [Component]	9-41
LoadExample [Component]	9-42
ManageAliases [Component]	9-43
Remove [Component]	9-44
RemoveUnused [Component]	9-44
Symbol Manager Script Commands	9-45
Add [Symbol Manager]	9-45
Edit [Symbol Manager]	9-51
EditWithComps [Symbol Manager]	9-55
Export [Symbol Manager]	9-61
GetData [Symbol Manager]	9-61
GetLibraryDefNames [Symbol Manager]	9-62
GetNames [Symbol Manager]	9-62
IsUsed [Symbol Manager]	9-63
Remove [Symbol Manager]	9-63
RemoveUnused [Symbol Manager]	9-64
Script and Library Script Commands	9-64
AddScript	9-65
EditScript	9-65
ExportScript	9-66

GetData [Script Manager]	9-66
GetNames [Script Manager]	9-67
IsUsed [Script Manager]	9-67
RemoveUnused [Script Manager]	9-68
RemoveScript	9-68
ModifyLibraries	9-69
Model Manager Script Commands	9-70
Export [Model Manager]	9-70
ExportAsFMU [Model Manager]	9-70
ExportAsTwin	9-72
ExportTwinModel	9-73
	9-73
GetData [Model Manager]	9-74
GetLibraryDefNames [Model Manager]	9-74
GetModelText [Model Manager]	9-75
GetNames [Model Manager]	9-75
IsUsed [Model Manager]	9-76
Remove [Model Manager]	9-76
RemoveUnused [Model Manager]	9-77
Package Manager Script Commands	9-77
Export [Package Manager]	9-77
GetData [Package Manager]	9-78
GetLibraryDefNames [Package Manager]	9-78
GetNames [Package Manager]	9-79
IsUsed [Package Manager]	9-79
Remove [Package Manager]	9-80
RemoveUnused [Package Manager]	9-80

Network Data Explorer Manager Script Commands	9-81
ExportFullWaveSpice	9-81
ExportNetworkData	9-83
ExportNMFData	9-85
10 - Compliance Script Commands	10-1
Callback Scripting Using Compliance Object	10-1
Definitions	10-1
Compliance Functions	10-2
GetComponentName	10-2
GetEditor	10-2
GetFullComponentName	10-3
GetInstanceID	10-3
GetParentDesign	10-4
GetPropHost	10-4
GetPropServerName	10-5
11 - Property Script Commands	11-1
Conventions Used in this Chapter	11-1
<PropServer> and <PropTab> Names	11-4
ChangeProperty	11-5
Callback Scripting Using PropHost Object	11-14
Definitions	11-14
PropHost Functions	11-16
AddMenuProp	11-16
AddMenuProp2	11-17
AddProp	11-18
AddProp2	11-20
ExecuteScript	11-22

GetApplication	11-23
GetCallBack	11-23
GetChangedProperty	11-24
GetDescription	11-24
GetEditor	11-25
GetEvaluatedText	11-25
GetFileName	11-26
GetHidden	11-26
GetMenuChoices	11-26
GetPropServers	11-27
GetPropTabType	11-27
GetReadOnly	11-28
GetScriptName	11-28
GetTabTypeName	11-29
GetText	11-29
GetValue	11-30
IsValueConstant	11-30
PropertyExists	11-31
RemoveProp	11-31
SetCallback	11-31
SetDescription	11-32
SetHidden	11-32
SetReadOnly	11-33
SetText	11-33
SetValue	11-34
Additional Property Scripting Commands	11-34
GetProperties	11-34

GetPropertyValue	11-35
GetVariables	11-37
GetVariableValue	11-38
SetPropertyValue	11-38
SetVariableValue	11-39
12 - Dataset Script Commands	12-1
AddDataset	12-1
DeleteDataset	12-2
EditDataset	12-3
ImportDataset	12-3
ExportDataset [Dataset]	12-4
13 - Design Object Script Commands	13-1
AddMessage	13-1
AddPortImpedance	13-1
AddPortSource	13-2
AddSdbOutput	13-3
Analyze	13-3
AnalyzeAll [Design]	13-4
AnalyzeInBackground	13-4
AnalyzeUI	13-5
ChangeGPort	13-5
ChangeIPort	13-6
ChangePortImpedanceProperty	13-7
ChangePortSourceProperty	13-8
ChangeProperty [Design Level]	13-8
CompileAsFMU	13-8
CompileAsTwin	13-10

DeleteDesignInstance	13-11
DeleteOutputVariable	13-11
DeletePort	13-12
DeletePortImpedance	13-12
DeletePortSource	13-13
DeleteSolutionVariation	13-13
EditDesignSettings	13-14
ExportNetlist	13-15
GetConfigurableData	13-16
GetData	13-16
Get Design Type	13-16
GetLinkComponents	13-17
GetLinkInstances	13-18
GetModule	13-19
GetName	13-20
GetNoteText	13-21
GetPostProcessingVariables	13-21
GetProperties [Design]	13-22
GetPropHost	13-22
GetResultsDirectory	13-23
GetSdbOutputs	13-23
GetSimulationOptionFilePath	13-24
GetSimulationSettings	13-25
GetSubDesigns	13-27
GetVariables	13-27
GetVariableValue	13-28
ImportSolution	13-28

InsertDesign	13-29
LoadConfiguration	13-30
ManageOutputs	13-31
PushExcitations	13-31
Redo	13-34
RemoveAllSdbOutputs	13-34
RemoveSdbOutput	13-34
RenamePort	13-35
RenameDesignInstance	13-35
ResetSymbolAnimation	13-36
SaveConfiguration	13-36
SetActiveEditor	13-37
SetAssignsDuringSimulation	13-37
SetDCBiasDisplayOptions	13-38
SetEventCallback	13-39
SetLinkStatic	13-40
SetSimulationOptionFilePath	13-40
SetSimulationSettings	13-42
SetVariableValue	13-43
StartAnalysis	13-44
ToggleDCBiasDisplay	13-44
Undo	13-45
UpdateDCBiasDisplay	13-45
UpdateLink	13-46
14 - Design Analysis Setup Script Commands	14-1
AddACAnalysis [Analysis Setup]	14-1
AddAnalysisOptions [Analysis Setup]	14-2

AddDCAnalysis [Analysis Setup]	14-6
AddTransient [Analysis Setup]	14-7
EditACAnalysis [Analysis Setup]	14-8
EditAnalysisOptions [Analysis Setup]	14-9
EditDCAnalysis [Analysis Setup]	14-13
EditTransient [Analysis Setup]	14-14
GetAllOptiSolutionSetups	14-15
GetAllSolutionSetups [Analysis Setup]	14-15
GetNominalSetupNamesForOptiSetup	14-16
GetSDBNamesForOptiSetup	14-17
GetSDBNameForSetupName [Analysis Setup]	14-17
GetSetupInfo [Analysis Setup]	14-18
Remove [Analysis Setup]	14-18
Rename [Analysis Setup]	14-19
SetSetupInfo [Analysis Setup]	14-19
15 - ROM Scripting	15-1
ROM Manager	15-1
GetROMInfo	15-1
ROMBuilderInstances	15-1
GetAllROMs	15-2
CreateROMComponent	15-2
Static ROM Builder	15-3
Build	15-3
Configuration File	15-3
Evaluate	15-4
Configuration File	15-5
Importing a Static ROM Model Into Twin Builder	15-6

Dynamic ROM Builder	15-7
Build	15-7
Configuration File	15-7
Evaluate	15-8
Configuration File	15-9
Importing a Dynamic ROM Model Into Twin Builder	15-9
16 - Output Variable Script Commands	16-1
CreateOutputVariable	16-1
DeleteOutputVariable	16-2
DoesOutputVariableExist	16-2
EditOutputVariable	16-3
GetOutputVariables	16-4
GetOutputVariableValue	16-4
SimValueContext	16-5
17 - Reporter Editor Script Commands	17-1
Report Setup Script Commands	17-1
AddCartesianLimitLine	17-1
AddCartesianXMarker	17-2
AddCartesianYMarker	17-2
AddDeltaMarker	17-3
AddMarker	17-3
AddNote	17-4
AddTraceCharacteristics	17-5
AddTraces	17-5
ChangeProperty [ReportSetup]	17-7
ClearAllMarkers	17-8
ClearAllTraceCharacteristics	17-8

CopyReportData	17-9
CopyReportDefinitions	17-9
CopyTraceDefinitions	17-9
CopyTracesData	17-10
CreateReport	17-10
CreateReportFromFile	17-15
CreateReportFromTemplate	17-15
CreateReportOfAllQuantities	17-16
DeleteAllReports	17-17
DeleteMarker	17-17
DeleteReports	17-18
DeleteTraceCharacteristics	17-18
DeleteTraces	17-19
EditCartesianXMarker	17-19
EditMarker	17-20
ExportImageToFile [Reporter]	17-21
ExportMarkerTable	17-21
ExportPlotImageToFile [Reporter]	17-22
ExportReport	17-23
ExportToFile	17-24
ExportToFile [Reporter]	17-24
ExportUniformPointsToFile	17-25
FFTONReport	17-26
GetAllCategories	17-27
GetAllReportNames	17-28
GetAllQuantities	17-29
GetAvailableDisplayTypes	17-29

GetAvailableReportTypes	17-30
GetAvailableSolutions	17-30
GetDataExpressions	17-31
GetDataUnits	17-31
GetDesignVariableNames	17-32
GetDesignVariableUnits	17-32
GetDesignVariableValue	17-32
GetDesignVariationKey	17-33
GetDisplayType	17-33
GetImagDataValues	17-34
GetPerQuantityPrimarySweepValues	17-35
GetPropertyValue	17-35
GetPropertyValue [ReportSetup]	17-37
GetRealDataValues	17-37
GetReportTraceNames	17-38
GetSolutionContexts	17-38
GetSolutionDataPerVariation	17-39
GetSweepNames	17-40
GetSweepUnits	17-41
GetSweepValues	17-41
GroupPlotCurvesByGroupingStrategy	17-42
ImportIntoReport	17-42
IsDataComplex	17-43
IsPerQuantityPrimarySweep	17-43
MovePlotCurvesToGroup	17-44
MovePlotCurvesToNewGroup	17-45
OpenWindowForAllReports	17-45

PasteReports	17-46
PasteTraces	17-46
Release Data	17-47
RemoveReport	17-47
RenameReport	17-47
RenameTrace	17-48
ResetPlotSettings	17-48
SavePlotSettingsAsDefault	17-49
UnGroupPlotCurvesInGroup	17-49
UpdateTraces	17-50
UpdateTracesContextAndSweeps	17-52
UpdateAllReports	17-53
UpdateReports	17-54
Output Variable Script Commands [Report]	17-54
18 - Optimetrics Module Script Commands	18-1
Conventions Used in this Chapter	18-1
General Commands Recognized by the Optimetrics Module	18-4
CopySetup [Optimetrics]	18-4
DeleteSetups [Optimetrics]	18-5
DistributedAnalyzeSetup	18-5
ExportDXConfigFile	18-6
ExportOptimetricsProfile	18-6
ExportOptimetricsResult	18-7
ExportParametricResults	18-8
GetSetupNames [Optimetrics]	18-8
GetSetupNamesByType [Optimetrics]	18-9
ImportSetup [Optimetrics]	18-9

PasteSetup [Optimetrics]	18-10
RenameSetup [Optimetrics]	18-10
SolveAllSetup	18-11
SolveSetup [Optimetrics]	18-11
Parametric Script Commands	18-12
EditSetup [Parametric]	18-12
GenerateVariationData (Parametric)	18-12
InsertSetup [Parametric]	18-13
Optimization Script Commands	18-15
EditSetup [Optimization]	18-15
InsertSetup [Optimization]	18-19
Sensitivity Script Commands	18-22
EditSetup [Sensitivity]	18-23
InsertSetup [Sensitivity]	18-26
Statistical Script Commands	18-29
EditSetup [Statistical]	18-29
InsertSetup [Statistical]	18-29
19 - Schematic Scripting Commands	19-1
Create Method List [Schematic Editor]	19-1
CreateArc [Schematic]	19-1
CreateCircle [Schematic]	19-3
CreateComponent [Schematic]	19-4
CreateCurve [Schematic]	19-5
CreateEllipse [Schematic]	19-6
CreateGlobalPort [Schematic]	19-7
CreateGround [Schematic]	19-8
CreateImage [Schematic]	19-9

CreateLine [Schematic]	19-11
CreatePagePort [Schematic]	19-12
CreatePlot [Schematic]	19-12
CreateIPort [Schematic]	19-15
CreatePolygon [Schematic]	19-16
CreateRectangle [Schematic]	19-17
CreateText [Schematic]	19-18
CreateWire [Schematic]	19-20
General Method List [Schematic]	19-21
Activate [Schematic]	19-21
AddPinPageConnectors [Schematic]	19-21
AddPinIPorts [Schematic]	19-22
AddPinPageConnectors [Schematic]	19-23
AlignHorizontal [Schematic]	19-23
AlignVertical [Schematic]	19-24
BringToFront [Schematic]	19-25
ChangePinVisibility [Schematic]	19-25
CloseEditor [Schematic]	19-26
Copy [Schematic]	19-26
CopySubdesign [Schematic]	19-27
CreatePage [Schematic]	19-28
Cut [Schematic]	19-28
DeactivateOpen [Schematic]	19-29
DeactivateShort [Schematic]	19-29
Delete [Schematic]	19-30
DeletePage [Schematic]	19-30
ElectricRuleCheck [Schematic]	19-31

ExportImage [Schematic]	19-32
FindElements [Schematic]	19-32
FitToBorder [Schematic]	19-34
FlipHorizontal [Schematic]	19-34
FlipVertical [Schematic]	19-35
GetAllNets [Schematic]	19-36
GridSetup [Schematic]	19-36
Move [Schematic]	19-38
MovePins [Schematic]	19-38
NameNets [Schematic]	19-39
PageBorders [Schematic]	19-40
Pan [Schematic]	19-40
Paste [Schematic]	19-41
RevertToDefaultSymbol [Schematic]	19-41
Rotate [Schematic]	19-42
SelectAll [Schematic]	19-43
SelectElement [Schematic]	19-43
SelectPage [Schematic]	19-44
SendToBack [Schematic]	19-44
SetPageData [Schematic]	19-45
SortComponents	19-47
ZoomArea [Schematic]	19-47
ZoomIn [Schematic]	19-48
ZoomOut [Schematic]	19-48
ZoomPrevious [Schematic]	19-49
ZoomToFit [Schematic]	19-49
Property Script Methods [Schematic]	19-50

ChangeProperty [Schematic]	19-50
GetProperties [Schematic]	19-50
GetPropertyAttribute [Schematic]	19-51
GetPropertyValue [Schematic]	19-52
SetPropertyValue [Schematic]	19-53
Information Method List [Schematic]	19-53
GetCompInstanceFromRefDes [Schematic]	19-53
GetComponentInfo [Schematic]	19-54
GetComponentPins [Schematic]	19-55
GetComponentPinInfo [Schematic]	19-55
GetComponentPinLocation [Schematic]	19-56
GetNumPages [Schematic]	19-57
GetPortInfo [Schematic]	19-57
GetSelections [Schematic]	19-58
GetSignals [Schematic]	19-59
GetWireConnections [Schematic]	19-60
GetWireInfo [Schematic]	19-60
GetWireSegments [Schematic]	19-62
Symbol Editor Script Commands	19-62
AddLevel [Symbol]	19-63
AlignHorizontal [Symbol]	19-63
AlignVertical [Symbol]	19-64
BringToFront [Symbol]	19-65
CloseEditor [Symbol]	19-65
Copy [Symbol]	19-65
CreateArc [Symbol]	19-66
CreateCircle [Symbol]	19-67

CreateImage [Symbol]	19-68
CreateLine [Symbol]	19-69
CreatePin [Symbol]	19-70
CreatePolygon [Symbol]	19-71
CreateRectangle [Symbol]	19-72
CreateText [Symbol]	19-73
Cut [Symbol]	19-74
DisplayPinNames [Symbol]	19-75
Delete [Symbol]	19-76
ExportFile [Symbol]	19-76
GetVisibleLevels [Symbol]	19-77
GridSetup [Symbol]	19-77
FlipHorizontal [Symbol]	19-79
FlipVertical [Symbol]	19-79
Move [Symbol]	19-80
Pan [Symbol]	19-81
Paste [Symbol]	19-81
Redo [Symbol]	19-82
RemoveLevels [Symbol]	19-82
Rotate [Symbol]	19-83
Save [Symbol]	19-84
SelectAll [Symbol]	19-84
SendToBack [Symbol]	19-85
SetActiveLevel [Symbol]	19-85
SetInitialLevels [Symbol]	19-86
SetVisibleLevels [Symbol]	19-86
ToggleLevel [Symbol]	19-87

Undo [Symbol]	19-88
ZoomArea [Symbol]	19-88
ZoomIn [Symbol]	19-89
ZoomOut [Symbol]	19-89
ZoomPrevious [Symbol]	19-90
ZoomToFit [Symbol]	19-91
20 - Twin Builder Result Database (SdbCtrl) Automation Script Commands	20-1
General Functions	20-1
GetLastError (SDB)	20-1
File Control Functions	20-2
OpenSDBFile	20-2
CloseSDBFile	20-2
Data Access Functions	20-2
ReadData	20-3
ReadImagData	20-4
GetAvailableTasks	20-5
GetTaskType	20-5
GetNumberOfRuns	20-6
GetRunNumbers	20-7
GetAvailableAnalysis	20-7
GetAnalysisType	20-8
GetAvailableChannels	20-8
GetAvailableExpChannels	20-9
GetDataType	20-9
GetDataTypeExp	20-11
GetNumberOfSamples	20-11
GetBaseChannelName	20-13

GetEnumString	20-13
GetChannelDim	20-14
GetArrayDim	20-15
GetChannelName	20-16
21 - Twin Builder Mathematical Automation (MathMod) Interface Script Com-	
mands	21-1
Example Scripts	21-1
Error Handling Functions	21-2
GetLastError (MathMod)	21-2
Example: Error Handling	21-2
Base and Reference Channel Functions	21-3
SetFundMethod	21-3
SetImageRefChannel	21-4
SetRealRefChannel	21-4
UnsetRefChannels	21-4
Example: Base Channel with Start and Stop Times	21-5
Example: SetFundMethod for TR Probe and PWR Probe Functions	21-5
Example: Reference Channels for CTRL AC and CTRL ERR Probes	21-5
Channel Create, Access, and Delete Functions	21-6
Calculate	21-6
GetChannel	21-7
MakeConstant	21-7
MakeLinearChannel	21-8
MakeLogChannel	21-8
RemoveChannel	21-8
SetChannel	21-9
Example: Creating Channels from Data Arrays	21-9
Example: Creating Channels from SDB Files	21-10

Example: Creating Channels from Calculate Functions	21-11
Channel Utility Functions	21-12
CalculateCommonBase	21-12
GetChannelLength	21-12
GetChannelMax	21-13
GetChannelMin	21-13
GetMonotony	21-13
Interpolate	21-14
PolarToRect	21-14
RectToPolar	21-15
Channel Calculation Functions	21-15
CalculateCharacteristic	21-15
Differentiation	21-17
FFT	21-17
GetChannelBasedISE	21-19
Integration	21-20
Smooth	21-20
Example: CalculateCharacteristic Function	21-21
Example: Power-of-2 FFT Function	21-22
Example: Integration, Differentiation, and Smooth Functions	21-23
AC Probe Functions	21-23
GetACFrqMax	21-23
GetACFrqMin	21-24
GetACMax	21-24
GetACMin	21-25
GetACRMS	21-25
Example: AC Probe Functions	21-26

CTRL AC Probe Functions	21-27
GetBandwidth	21-27
GetGainCrossover	21-27
GetGainMargin	21-28
GetLowerCutoff	21-29
GetPeakGain	21-30
GetPeakGainFreq	21-30
GetPhaseCrossover	21-31
GetPhaseMargin	21-32
GetUpperCutoff	21-33
Example: CTRL AC Probe Functions	21-33
CTRL ERR Probe Functions	21-34
GetFinalError	21-34
GetIAE	21-35
GetISE	21-35
GetITAE	21-36
GetITSE	21-36
Example: CTRL ERR Probe Functions	21-37
CTRL TR Probe Functions	21-38
GetDeadTime	21-38
GetDelayTime	21-38
GetOvershoot	21-39
GetOvershootTime	21-39
GetRiseTime	21-40
GetSettlingTime	21-40
Example: CTRL TR Probe Functions	21-41
F Probe Functions	21-42

GetValueLogInterpolation	21-42
Example: F Probe Functions	21-43
FFT Probe Functions	21-43
SetupFFT	21-43
GetFFTFreq	21-44
GetFFTImag	21-45
GetFFTMag	21-45
GetFFTPhase	21-46
GetFFTReal	21-46
GetFFTResultCIT	21-47
GetFFTResultCMWI	21-47
GetFFTResultDF	21-48
GetFFTResultIECClassA	21-48
GetFFTResultIECClassB	21-48
GetFFTResultIECClassC	21-49
GetFFTResultIECClassD	21-49
GetFFTResultIECInstallation	21-49
GetFFTResultIECThreePhase	21-50
GetFFTResultIEEE519Current	21-50
GetFFTResultIEEE519Voltage	21-51
GetFFTResultIT	21-51
GetFFTResultKFactor	21-51
GetFFTResultMagnitude	21-52
GetFFTResultPhaseDeg	21-52
GetFFTResultRipple	21-53
GetFFTResultRMS1	21-53
GetFFTResultRMSAC	21-54

GetFFTResultRMSDC	21-54
GetFFTResultTHD	21-55
GetFFTResultTIF	21-55
Example: FFT Probe Functions	21-56
Example: FFT Probe Output Channel Functions	21-57
Example: Harmonic Checking Functions	21-58
Moving Average Functions	21-59
GetSlidingMean	21-59
GetSlidingRMS	21-60
Example: Moving Average Functions	21-60
PULSE Probe Functions	21-61
GetPulseFall9010	21-61
GetPulseFront1090	21-62
GetPulseFront3090	21-62
GetPulseMax	21-63
GetPulseMaxTime	21-63
GetPulseMin	21-64
GetPulseMinTime	21-64
GetPulseTail150	21-65
GetPulseWidth5050	21-65
Example: PULSE Probe Functions	21-65
PWR Probe Functions	21-67
SetupPWR	21-67
GetPWRPhaseDP	21-67
GetPWRPhaseEnergy	21-68
GetPWRPhaseHP	21-69
GetPWRPhaseP	21-69

GetPWRPhaseP1	21-70
GetPWRPhasePF	21-70
GetPWRPhasePF1	21-71
GetPWRPhasePower	21-71
GetPWRPhaseQ1	21-72
GetPWRPhaseS	21-72
GetPWRPhaseS1	21-73
GetPWRSystemDPe	21-73
GetPWRSystemEnergy	21-74
GetPWRSystemHPe	21-74
GetPWRSystemle	21-75
GetPWRSystemle1	21-75
GetPWRSystemP	21-75
GetPWRSystemP1	21-76
GetPWRSystemPF1	21-76
GetPWRSystemPFe	21-77
GetPWRSystemPower	21-77
GetPWRSystemQ1	21-78
GetPWRSystemS1	21-78
GetPWRSystemSe	21-78
GetPWRSystemVe	21-79
GetPWRSystemVe1	21-79
Example: PWR Probe Functions	21-80
T Probe Functions	21-81
GetValue	21-81
Example: T Probe Functions	21-82
TR Probe Functions	21-82

GetCrestFactor	21-82
GetDistortion	21-82
GetFormFactor	21-83
GetFundamentalMag	21-83
GetIntegral	21-84
GetMax	21-84
GetMaxTime	21-85
GetMean	21-85
GetMeanAbs	21-86
GetMin	21-86
GetMinTime	21-87
GetPeakAverage	21-87
GetPeakPeak	21-88
GetRipple	21-88
GetRMS	21-89
GetRMSAC	21-89
Example: TR Probe Functions	21-89
Random Number Functions	21-91
GetUnitRandMinMax	21-91
GetUnitRandMeanTol	21-91
GetNormRandMinMax	21-92
GetNormRandMeanTol	21-92
GetCustomRNDValue	21-93
22 - User Defined Solutions Commands	22-1
DeleteUserDefinedSolutions	22-3
EditUserDefinedSolution	22-3
GetSimulationOptionFilePath	22-5

GetSimulationSettings	22-6
SetSimulationOptionFilePath	22-8
SetSimulationSettings	22-9
23 - Best Practices and Examples	23-1
Best Practices for Recorded Scripts	23-1
Example: Analyze a Transient Solution	23-1
Example: Export Report Data	23-2
Example: Running in an HPC Environment	23-3
Example: Save/Load Custom KRN Files	23-3
Index	Index-1

1 - Introduction to Twin Builder 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.

Ansys Twin Builder can record scripts in IronPython. Additionally, it contains an IronPython command shell for executing scripts.

When running Twin Builder from the command line, you can write scripts in any language that provides Microsoft COM methods.

Note:

The following UI components cannot be scripted because scripting commands are not supported for operations within them.

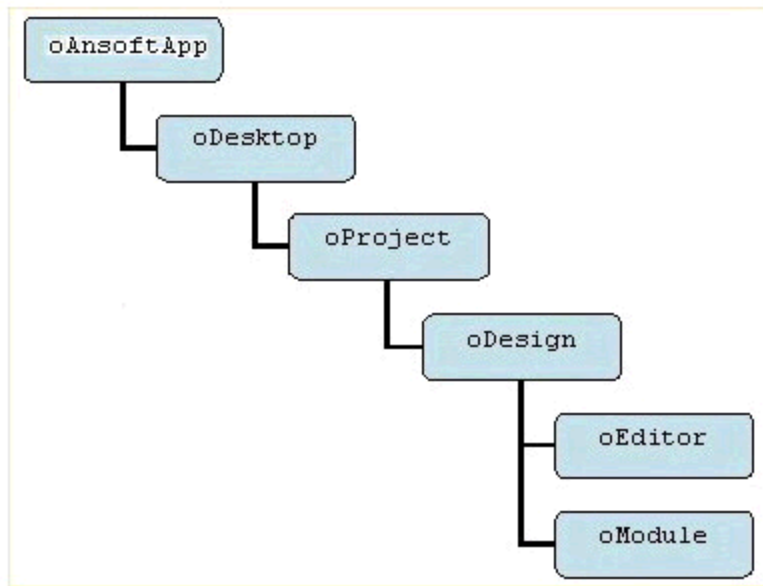
- All toolkits listed under **Twin Builder > Toolkit**
- Device Characterization Wizard
- Model editors
- User Defined Documents (UDDs)

Overview of Twin Builder Script Variables

When you record a Twin Builder script, the beginning of the script looks like the following:

```
oDesktop.RestoreWindow()  
  
oProject = oDesktop.NewProject()  
  
oProject.InsertDesign("Twin Builder", "TwinBuilder1", "", "")  
  
oDesign = oProject.SetActiveDesign("TwinBuilder1")
```

The lines above define the variables used by Twin Builder in the script and assign values to the variables. The variables are used in the following hierarchy:



Class hierarchy of variables.

oAnsoftApp

The `oAnsoftApp` object provides a handle to access the `AnsoftSimplorer` product.

One example of accessing this object is:

Python Example	<pre>oAnsoftApp = CreateObject("Ansoft.ElectronicsDesktop")</pre>

oDesktop

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

One example of accessing this object is:

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

See [Desktop Object Script Commands](#), for details about script commands recognized by the `oDesktop` object.

oProject

The `oProject` object corresponds to one project open in the product. It is used to manipulate the project and its data. Its data includes variables, material definitions and one or more designs. One example of accessing this object is:

Python Example	<pre>oProject = oDesktop.OpenProject("c:\myprojects\test.aedt")</pre>

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

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

oDesign

The `oDesign` object corresponds to an instance of a design in the project. This object is used to manipulate the design and its data. Its data includes variables, modules, and editors.

One example of accessing this object is:

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

See the following chapters for details about the script commands recognized by the `oDesign` object:

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

oEditor

The `oEditor` object corresponds to an editor, such as the Schematic Editor. This object is used to add and modify data in the editor.

One example of accessing this object is:

Python Example	<pre>oEditor = oDesign.SetActiveEditor("SchematicEditor")</pre>

The `AnsoftSimplorer` product scripting supports the following editors:

Editor	Name in Script
Schematic Editor	"Schematic"
Reporter Editor	There is no Reporter editor object in the script. Instead, Reporter editor commands are executed by the Twin Builder design object <code>oDesign</code> .

See the following for details about the script commands recognized by the `oEditor` object.

- [Schematic Scripting Commands](#)

See [Reporter Editor Script Commands](#) for details about reporter editor commands.

oModule

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

An example of accessing this object is:

Python Example	<code>oModule = oDesign.GetModule("Optimetrics")</code>

Editor	Name in Script
Schematic Editor	"Schematic"
Reporter Editor	There is no Reporter editor object in the script. Instead, Reporter editor commands are executed by the Twin Builder design object <code>oDesign</code> .

The `AnsoftSimplorer` product scripting supports the following modules:

Module	Name in Script	Chapter
Output Variable Module	"OutputVariable"	Output Variable Scripting Commands
Optimetrics Module	"Optimetrics"	Optimetrics Script Commands
Corresponds to the Optimetrics branch in the project tree.		
Report Setup Module	"ReportSetup"	Reporter Editor Script Commands

Recording a Script to a File

Once you start to record a script, your subsequent actions are added to the script. Each interface command has one or more associated script commands that are recorded to the script. The script is recorded to a text file in .py(Python) format.

1. Click **Tools>Record Script**.
The **Save As** dialog box appears.
2. Use the file browser to locate the folder in which you want to save the script, such as `C:\ANSYS\Twin Builder\Scripts`, and then double-click the folder's name.
3. Type the name of the script in the **File name** text box.

4. Click **Save**.

The script is saved in the folder you selected by the file name *filename.py*.

5. Perform the steps that you want to record.
6. When you have finished recording the script, click **Stop Script Recording** on the **Tools** menu.

Recording a Script to a Project

1. On the **Tools** menu, click **Record Script to Project**.

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

2. Enter a name for the script, and click **OK** to close the dialog and begin recording.
3. Perform the steps that you want to record.
4. When you have finished recording the script, click **Stop Script Recording** on the **Tools** menu.

The script you recorded is saved to the current project's scripts library. See *Managing Library Contents* for information on working with libraries.

Working with Project Scripts

- You can edit scripts stored in a project by right-clicking the desired script in the **Definitions>Scripts** folder in the Project Manager window; then selecting **Edit Script** from the context menu. The script will open in the Script Editor.
- You can execute a project script, by right-clicking the desired script in the **Definitions>Scripts** folder in the Project Manager window; then selecting **Run Script** from the context menu.

Note:

Project scripts are stored in the project scripts library. See *Managing Library Contents* in the main help for information on working with libraries.

Stopping Script Recording

- Click **Tools>Stop Script Recording**.

Twin Builder stops recording to the script.

Running a Script

1. Click **Tools > Run Script**.

The **Open** dialog box appears.

2. Use the file browser to locate the folder in which you saved the script, and then double-click the folder's name.
3. Type the name of the script in the **File name** text box, or click its name, and then click **Open**.

Twin Builder executes the script.

To supply script arguments when running from **Tools > Run Script**, use the edit field at the bottom of the file selection dialog box.

To run a script from a command line (as described in the Twin Builder Help), use:

-runscriptandexit or **-runscript** arguments to the Twin Builder command line syntax.

You can give **-scriptargs** parameter to the script and specify arguments.

Running Scripts in an HPC Environment

Follow this procedure if you intend to run scripts in a high performance computing (HPC) environment.

1. In the Twin Builder window, click **Tools > Options > HPC and Analysis Options**. The **HPC and Analysis Options** dialog box appears.
2. Select **Design Type > Twin Builder**.
3. In the **Available Configurations** area, select the local configuration. This configuration will have the word **Local** in the **Name** column.
4. Click the **Make Active** button.
5. Click **OK** to save your selection and close the **HPC and Analysis Options** dialog box.
6. In the Twin Builder window, click **Tools > Job Management > Submit Job**. The **Submit Job To** dialog box appears.
7. On the **Analysis Specification** tab, navigate to and open a Twin Builder project in the **Project path** field.
8. At the bottom of the dialog box, click **Import Configuration > Twin Builder - Local**.
9. Click the **Compute Resources** tab.
10. In the **Resource selection** area, double-click the value in the **Tasks** column and add a new value to specify the number of tasks that will simultaneously run.
11. Click the **Export** button to save your settings to an **.areg** file.

Run these scripts to use the settings file created in the procedure above.

Python Example	<pre>projectFileName = "<twinbuilder project filepath>" hpcSettingsFileName = "<.areg filepath>" jobIds = oDesktop.SubmitJob(hpcSettingsFileName, ProjectFileName)</pre>


```
oDesktop.LaunchJobMonitor(projectFileName)
oDesktop.RefreshJobMonitor()
//user can use same RSM settings file for more other projects as well..
projectFileName2 = "<anotherProject>"
oDesktop.SubmitJob(hpcSettingsFileName,ProjectFileName2)
oDesktop.LaunchJobMonitor(projectFileName2)
oDesktop.RefreshJobMonitor()
```

Note:

Refresh job monitor will return status of job. For more details check command's help page.

Run these scripts to use custom save/load .krn (simulation state) files.

Python Example

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

saveKRNFile = "<UserSpecifiedFileNameWithPath>"
// "C:\MyFile.krn"
loadKRNFile = "<UserSpecifiedFileNameWithPath>"
// "C:\MyFile.krn"
ret = oDesign.SetSimulationOptionFilePath(0, saveKRNFile)
ret = oDesign.SetSimulationOptionFilePath(1, loadKRNFile)
...
oDesign.AnalyzeUI("TR",3) // "TR" : setup name
```

Pausing and Resuming a Script

To pause a script during its execution:

Click **Tools>Pause Script**.

To resume a script after pausing it:

Click **Tools>Resume Script**.

Stopping a Script

- On the **Tools** menu, click **Stop Script**.

Twin Builder stops executing the script that has been paused.

Modifying a Script for Easier Playback

Recorded scripts set the `oProject` variable to the return value from `SetActiveProject` with a currently loaded project name, for example "Project1". That means that the script must be played back within Project1 to operate correctly. Alternatively, `oProject` could be set to the active project without specifying a project name.

For example:

Python Example

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

Using the line above, the script can be played back in any project.

Event Callback Scripting

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

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

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

		ComplInstance.GetRunStatus() — Returns the status number of the specified design simulation.
		ComplInstance.Abort() — Aborts the specified design simulation.

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

Simple and Composite Names

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

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

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

column contains the library portion of the composite name, while the **Name** column contains the simple portion of the composite name.

Aborting Scripts

Press ESC to abort a script that is running in Twin Builder. Terminating a script in this manner works for each of the following:

- Scripts in the project library that are run by right-clicking the script icon in the project window and choosing "Run Script"
- Scripts in files that are external to Twin Builder and are run by choosing "Tools/Run Script..."
- Scripts that are specified as callbacks in the Property dialog

Executing a Script from Within a Script

Twin Builder provides a script command that enables you to launch another script from within the script that is being executed:

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

If the full path to the script is not specified, Twin Builder searches for the specified script in the following locations, in this order:

- 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.

- User library directory.

This is the **userlib** subdirectory in the library directory. The library directory can be specified in the **General Options** dialog box (click **Tools>Options>General Options** to open this dialog box) under the **Project Options** tab.

- 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.

- Twin Builder installation directory.

Editing Properties

Any data that is shown in the dockable **Properties** dialog box or in the modal **Properties** pop-up window is called a property. For example, project and local variables are properties. The

FillStyle of a box in the Symbol editor is also a property. See [Property Script Commands](#), for an explanation of how to manipulate properties in a script.

Unused Script Commands

When using the Iron Python console to query for script commands, be aware that not all available commands are intended for use with Twin Builder. See below for a list of script commands present in the system that must not be used with your Twin Builder designs.

- **AddAntennasAtPins**
- **AddTerminatorsAtPins**
- **CreateNPort**

Twin Builder Scripting Conventions

The following sections detail Twin Builder scripting conventions:

- [Syntax Conventions](#)
- [Script Command Conventions](#)

Syntax Conventions

String	A quoted string.
Boolean	A boolean value. Should be set to either <code>true</code> or <code>false</code> (no quotes). Example: <code>"SolveInside:=", true</code>
Double	A double precision value. Example: <code>1.2</code>
Integer	An integer. Example: <code>1</code>
Value	Can be a number, a variable, or a quoted string containing a valid Twin Builder expression. Examples: - <code>"XSize:=", 1</code> - <code>"XSize:=", "3mm"</code> - <code>"XSize:=", Var</code> - <code>"XSize:=", "Simplorer_Var + 10mm"</code>

Script Command Conventions

The majority of this guide lists individual script commands. The following conventions are used to describe them:

Script Command Name

Description of the function of the script command.

UI Access	Lists the interface command, if any, that corresponds to the script command. Menu commands are separated by carets. For example: Twin Builder>Add Solution Setup>Transient.						
Parameters	<table><tr><td><i>Name</i></td><td><i>Type</i></td><td><i>Description</i></td></tr><tr><td>Describes the arguments or information in the syntax description, if an explanation is needed.</td><td>The type of the argument</td><td>A brief description of the argument</td></tr></table>	<i>Name</i>	<i>Type</i>	<i>Description</i>	Describes the arguments or information in the syntax description, if an explanation is needed.	The type of the argument	A brief description of the argument
<i>Name</i>	<i>Type</i>	<i>Description</i>					
Describes the arguments or information in the syntax description, if an explanation is needed.	The type of the argument	A brief description of the argument					
Return Value	Describes the return value, if any.						

Python Syntax	Demonstrates the correct syntax for the command. Caret brackets < > enclose information or arguments that you must enter.
Python Example	Provides a working example of the script command, if needed.

2 - Twin Builder and IronPython

IronPython is an open source implementation of Python for the .NET CLR and Mono, originally created by Jim Hugunin.

IronPython uses the Dynamic Language Runtime, a framework for writing dynamic languages for .NET which originated in IronPython 1. It also runs on Silverlight, a .NET browser plugin that runs on Windows and the Mac (and a Mono port called Moonlight runs on Linux). This means that IronPython can be used for client-side scripting in the browser.

IronPython is a Python compiler. It compiles Python code to in memory bytecode before execution (which can be saved to disk, making binary only distributions possible).

This chapter provides an overview of Twin Builder scripting using IronPython. Information is included on the following topics:

- Python Operators
- Python Variables
- Controlling the Flow of the program
- Datatypes in Python
- Recommended References
- Sample Scripts

Controlling Program Execution

You can use conditional statements to control the flow of a script and you can use iterative statements to loop through a set of commands a certain number of times.

Conditional Statement:

- `if...elif...else`

Iterative Statements:

- `for loop`
- `while loop`

Conditional Statements

When you want to execute a statement or a block of statements based on the satisfaction of certain conditions, use conditional statements.

Python supports the conditional statement 'If', 'Elif', 'Else'

Following is an example:

```
if obj = "Box1":  
  
    <statements to execute>
```

```
elif obj = "Circle1":  
    <statements to execute>  
else:  
    <statements to execute>
```

For Loop

The `For` type of loop allows you to run a group of statements repeatedly. It uses a counter to run statements a specified number of times. The following displays how a for loop statement works in Python

```
for variable in range(start, end):  
    <statements to execute>
```

You can exit early from a `for` loop with the `break` statement.

While Loop

The While loop statement in Python allows you to repeatedly execute a block of statements as long as the given condition stays true. The syntax of a while loop is:

```
while <expression>:  
    <statements to execute>
```

Python Operators

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

See the online *Python User's Guide* for more details.

Operator Precedence

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

This is a list of all the operators, beginning with the highest precedence, to the lowest:

Operator	Description
**	Exponentiation (raise to

	the power)
~ + -	Complement, unary plus and minus (method names for the last two are +@ and -@)
* / % //	Multiply, divide, modulo and floor division
+ -	Addition and subtraction
>> <<	Right and left bitwise shift
&	Bitwise 'AND'
^	Bitwise exclusive 'OR' and regular 'OR'
<= < > >=	Comparison operators
<> == !=	Equality operators
= %= /= //= -= += *= **=	Assignment operators
is is not	Identity operators
in not in	Membership operators
not or and	Logical operators

Arithmetic Operators

Following is a list of Python's arithmetic operators.

Operator	Example
+ Addition	$a + b = 30$
- Subtraction	$a - b = -10$
* Multiplication	$a * b = 200$
/ Division	$b / a = 2$
% Modulus	$b \% a = 0$
** Exponent	$a ** b = 10 \text{ to the power } 20$
// Floor Division	$9 // 2 = 4$ and $9.0 // 2.0 = 4.0$, $-11 // 3 = -4$, $-11.0 // 3 = -4.0$

Comparison Operators

Following is a list of Python's comparison operators.

Symbol	Description
==	Equality
<>	Inequality

!=	Inequality
<	Less than
>	Greater than
<=	Less than or equal to
>=	Greater than or equal to
Is	Object equivalence

Logical Operators

Following is a list of Python's logical operators:

If a=5, b=10

Operator	Description	Example
and	Logical AND: If both the operands are true then makes the condition true.	(a and b) is true.
or	Logical OR: If any of the two operands are non-zero then makes the condition true.	(a or b) is true.
not	Logical NOT: Reverses the logical state of its operand.	Not(a and b) is false.

Python Variables

A Python variable is a placeholder representing information that may change during the time your script is running. Use a variable name in a script to view or modify its value.

Assigning Values to Variables

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

You can directly assign information to it. For example:

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

Python Lists

A Python list is a container that holds a number of objects, in a particular order. It is easy to add and remove items from this list as we deem necessary. We do not have to declare the name of the list variable first, before utilizing it.

To create a list, we put a number of comma separated values in box brackets, for example:

```
testList = ["element0", "element1", 2, 3]
```

The elements in a list need not be of the same data type.

Accessing Items in a list:

To access items in a list, use square brackets to denote the index of the item you are looking to access.

Example:

```
list = ['Twin Builder', 'HFSS', 'Circuit']
```

If we want to access the first element, and let's say store it in a different variable:

```
newVar = list[0]
```

Updating a list:

If you want to change the value of a particular element, we can reassign the value stored in that index.

Example:

If we want to replace 'Circuit' with 'SIwave':

```
list[2] = 'SIwave'
```

Additionally, you can also add elements to the end of the list by using the append method.

Example:

```
list.append('SIwave')
```

Deleting items from a list:

You can make use of the del statement if you know the index of the item you would like to remove from a list.

Example:

```
del list[1];
```

For further information on Lists, see [The Official Python Documentation](#)

Converting between Data Types

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

str Syntax: `str(variablename)`.

Converts `variablename` to a string. For example, it can be used to convert the number 2.5 to a string "2.5".

bool Syntax: `CBool(variablename)`.

Converts `variablename` to a boolean. If `variablename` is "", CBool returns False. Otherwise it returns True.

float Syntax: `float(variablename)`.

Converts `variablename` to a C double number. For example, it can be used to convert the string "2.5" to the number 2.5.

int Syntax: `int(variablename)`.

Converts `variablename` to an integer.

```
class int(x=0)
```

```
class int(x, base=10)
```

Return an integer object constructed from a number or string `x`, or return 0 if no arguments are given. If `x` is a number, it can be a plain integer, a long integer, or a floating point number. If `x` is floating point, the conversion truncates toward zero. If the argument is outside the integer range, the function returns a long object instead.

If `x` is not a number or if `base` is given, then `x` must be a string or Unicode object representing an integer literal in radix `base`. Optionally, the literal can be preceded by + or - (with no space in between) and surrounded by whitespace. A base-`n` literal consists of the digits 0 to `n-1`, with a to z (or A to Z) having values 10 to 35. The default base is 10. The allowed values are 0 and 2-36. Base-2, -8, and -16 literals can be optionally prefixed with 0b/0B, 0o/0O/0, or 0x/0X, as with integer literals in code. Base 0 means to interpret the string exactly as an integer literal, so that the actual base is 2, 8, 10, or 16.

From [The Official Python Documentation](#)

Recommended Python and IronPython References

Python Software Foundation *Beginner's Guide to Python*.

Available <https://wiki.python.org/moin/BeginnersGuide>

The IronPython Tutorial: <http://ironpython-test.readthedocs.io/en/latest/tutorial/>

The Tutorial in Ansys Twin Builder>Help>Twin Builder Contents>Desktop Scripting with IronPython

A Sample Twin Builder Script

Following is an example of a Twin Builder script.

The following example gets information from selected items in the current schematic and displays it in the Message Window.

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

```
oDesktop = oAnsoftApp.GetAppDesktop()
oDesktop.RestoreWindow()
sys = oDesktop.GetSysLibDirectory()
AddInfoMessage("SysLib Directory: " + sys)
user = oDesktop.GetUserLibDirectory()
AddInfoMessage("UserLib Directory: " + user)
per = oDesktop.GetPersonalLibDirectory()
AddInfoMessage("PersonalLib Directory: " + per)
pjt = oDesktop.GetProjectDirectory()
AddInfoMessage("Project Directory: " + pjt)
oProject = oDesktop.GetActiveProject()
oDesign = oProject.GetActiveDesign()
oEditor = oDesign.SetActiveEditor("SchematicEditor")
selectionArray = oEditor.GetSelections()
if len(selectionArray) == 0:
    AddInfoMessage("No selections!")
for i in selectionArray:
    AddInfoMessage("Selected element: " + i)
    compArray = oEditor.GetComponentInfo(i)
    for k in compArray:
        AddInfoMessage("GetComponentInfo: " + k)
    pinArray = oEditor.GetComponentPins(i)
    for j in pinArray:
        AddInfoMessage("GetComponentPins: " + j)
    portArray = oEditor.GetPortInfo(i)
    for l in portArray:
        AddInfoMessage("GetPortInfo: " + l)
```

```
netArray = oEditor.GetWireConnections(i)
for m in netArray:
    AddInfoMessage("GetWireConnections: " + m)
netArray2 = oEditor.GetWireSegments(i)
for n in netArray2:
    AddInfoMessage("GetWireSegments: " + n)
netArray3 = oEditor.GetWireInfo(i)
for o in netArray3:
    AddInfoMessage("GetWireInfo: " + o)
sigs = oEditor.GetSignals(wn)
for ll in sigs:
    AddInfoMessage("Element in array from GetSignals: " + ll)
```

For additional Twin Builder script examples, see ["Best Practices and Examples " on page 23-1](#) .

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	<i>Name</i> None	<i>Type</i>	<i>Description</i>
Return Value	Object		

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

4 - Twin Builder Application Object Script Commands

These commands allow you to access Twin Builder-specific application functionality. The commands are available from the Twin Builder application object through `GetApp("Twin Builder")`.

GetProcessResources

To get an accurate picture of Twin Builder resource usage during script processing.

UI Access	NA		
Parameters	<i>Name</i> <type>	<i>Type</i> String	<i>Description</i> "GDI" or "USER"
Return Value	Integer; the number of a particular type of resource in use by Twin Builder.		

Python Syntax	<code>GetProcessResources (<type>)</code>
Python Example	<code>count = oSimplorerApp.GetProcessResources ("GDI")</code>

RecompileLibraries

To update model libraries to a new version of Twin Builder. The command opens a new project to do its work.

UI Access	Tools>Manage Model Libraries		
Parameters	<i>Name</i> <libname1>, <libname2>, ... <closeProject> <useLog> <logfile name>	<i>Type</i> String Boolean Boolean String	<i>Description</i> Full path of the model or package libraries to be recompiled. Flag that indicates whether to close the newly created project. Flag that indicates whether to write a log. Full path of the log file name.

Return Value	None

Python Syntax	RecompileLibraries ([<libname1>, <libname2>,...], <closeProject>, <useLog>, <logfilename>)
Python Example	<pre>oSimplorerApp = oAnsoftApp.GetApp("Twin Builder") count = oSimplorerApp.RecompileLibraries(["E:\Projects\PersonalLib\foo.asmd", "E:\Projects\PersonalLib\ID1.asmd"], false, true, "e:\compLibs.txt")</pre>

ConvertSMLToSMT

Compiles the incoming SML file and writes a binary SMT file in the same directory

UI Access	NA		
Parameters	<i>Name</i> < filename >	<i>Type</i> String	<i>Description</i> Pathname of the SML File
Return Value	String Name of the SMT file		

Python Syntax	ConvertSMLToSMT(<filename>)
Python Example	See the SimulateSMT command example

SimulateSMT

Simulates an SMT file and writes SDB file of the same name to the same directory. The SMT and SDB are written to the same directory as the SML file. It is necessary to use a full path name

for both the SML and SMT files.

UI Access	NA		
Parameters	<i>Name</i> < filename >	<i>Type</i> String	<i>Description</i> Pathname of the SMT file
Return Value	Boolean result of the simulation		

Python Syntax	SimulateSMT(<filename>)
Python Example	<pre>oSimplorerApp = oAnsoftApp.GetApp("Twin Builder") smtname = oSimplorerApp.ConvertSMLtoSMT ("E:\Projects\TestSMLSimulation\test.sml") ret = oSimplorerApp.SimulateSMT(smtname)</pre>

5 - Desktop Object Script Commands

Python Example	<pre>oAnsoftApp = CreateObject("Ansoft.ElectronicsDesktop") oDesktop = oAnsoftApp.GetAppDesktop() oDesktop.CommandName(<args>)</pre>

AddMessage

Add a message with severity and context to message window.

UI Access	NA		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<projectName>	String	Name of the Project
	<designName>	String	Name of appropriate design, or empty string.
	<severity>	Integer	0 = Informational, 1 = Warning, 2 = Error, 3 = Fatal
	<msg> <category>	String String	The message Optional. The category is created with the message under the design tree node if the category does not exist. If the category already exists, the new message is added to the end of the existing category. It is ignored if the project or design is empty. If missing or empty, the message is added to the Design node in the message tree.
Return Value	None		

Python Syntax	AddMessage(<projectName>, <designName>, <severity>, <msg>, <category>)
Python Example	<pre>oDesktop.AddMessage ("Project1", "TwinBuilder1", 0, "This is a test message", "")</pre>

ClearMessages

Clears messages, optionally specifying severity and design.

UI Access	In Message Manager, right-click the project name and click Clear messages for Project#		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	project	<string>	project name, an empty string clears all project messages.
	design	<string>	design name; ignored if project name is empty; an empty string clears messages in all designs.
	severity	<int>	0 - clear all info messages 1 - clear all info and warning messages 2- clear all info, warning and error messages 3 - clear all messages included info, warning, error, and fatal-error
Return Value	None		

Python Syntax	ClearMessages(<project>, <design>, <severity>)
Python Example	<code>oDesktop.ClearMessages("", "", 3)</code>

CloseAllWindows

Closes all MDI child windows on the desktop

UI Access	From main menu, Window>CloseAll.		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	None		
Return Value	None		

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

CloseProject

Close a named project currently open in the Desktop. Changes to the project will not be saved. Do a **Save** or **SaveAs** before closing the project if you want to save changes.

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

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

CloseProjectNoForce

Close a named project currently open in the Desktop, unless a simulation is running. Changes to the project will not be saved. Do a **Save** or **SaveAs** before closing the project if you want 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	<i>Name</i> <ProjectName>	<i>Type</i> String	<i>Description</i> 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>

DeleteProject

Delete a project from disk

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

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

EnableAutoSave

Enable or disable autosave feature

UI Access	NA		
Parameters	<i>Name</i> <enable>	<i>Type</i> Boolean	<i>Description</i> True enables autosave and False disables it.
Return Value	None		

Python	EnableAutoSave(<Enable>)

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

ExportOptionsFiles

Copies the options config files to the *DestinationDirectory*.

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

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

GetActiveProject

Obtain the object for the project that is currently active in the Desktop.

UI Access	NA		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	None		
Return Value	Object, the project that is currently active in the desktop		

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

GetAutoSaveEnabled

Checks to see if the autosave feature is enabled

UI Access	NA		
Parameters	<i>Name</i> None	<i>Type</i>	<i>Description</i>
Return Value	Boolean True if autosave is enabled		

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

GetBuildDateTimeString

To get the build date and time of the product

UI Access	NA		
Parameters	<i>Name</i> None	<i>Type</i>	<i>Description</i>
Return Value	String The build date and time. Format: year-month-day hour:minute:second, e.g., 2011-01-18 21:59:33		

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

GetCustomMenuSet

Returns the name of the currently-selected value in **Tools > Options > General Options > General > Desktop Configuration > Custom Menu Set**.

UI Access	N/A		
Parameters	<i>Name</i> None	<i>Type</i>	<i>Description</i>
Return Value	String for current custom menu set. For example: 'Default', 'Twin Builder'.		

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

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	<i>Name</i> <type>	<i>Type</i> String	<i>Description</i> 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	<code>GetDefaultUnit(<type>)</code>
Python Example	<code>oDesktop.GetDefaultUnit("Length")</code>

GetDesktopConfiguration

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

UI Access	NA		
Parameters	<i>Name</i> None	<i>Type</i>	<i>Description</i>
Return Value	String for 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 methods.

UI Access	NA		
Parameters	<i>Name</i> None	<i>Type</i>	<i>Description</i>
Return Value	Returns a collection of names of machines used for distributed analysis		

Python Syntax	GetDistributedAnalysisMachines()
Python Example	<pre>for machine in oDesktop.GetDistributedAnalysisMachines(): AddInfoMessage(str(machine))</pre>

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

GetMessages

Collects the messages from a specified project and design.

UI Access	NA		
Parameters	<i>Name</i> <ProjectName>	<i>Type</i> String	<i>Description</i> 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 is info and above 1 is warning and above 2 is error and fatal 3 is fatal only (rarely used)
Return Value	A simple array of strings.		

Python Syntax	<code>GetMessages (<ProjectName>, <DesignName>, <Severity>)</code>
Python Example	<pre>Messages = oDesktop.GetMessages ("MyProject", "TwinBuilder1,1)</pre>

GetLibraryDirectory

Get the path to the `SysLib` directory.

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

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

GetPersonalLibDirectory

Get the path to the `PersonalLib` directory.

UI Access	NA		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	None		
Return Value	String		
	The path to the <code>PersonalLib</code> directory		

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

GetProcessID

Returns the process ID of ansysedt.exe.

UI Access	NA		
Parameters	<i>Name</i> None	<i>Type</i>	<i>Description</i>
Return Value	String returns the process ID of ansysedt.exe, such as 12716		

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

GetProjectDirectory

Gets the path to the Project directory.

UI Access	NA		
Parameters	<i>Name</i> None	<i>Type</i>	<i>Description</i>
Return Value	String The path to the Project directory.		

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

GetProjectList

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

UI Access	NA		
Parameters	<i>Name</i> None	<i>Type</i>	<i>Description</i>
Return Value	An array of strings, the names of all open projects in the Desktop.		

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

GetProjects

Return a list of all the projects that are currently open in the Desktop.

UI Access	NA		
Parameters	<i>Name</i> None	<i>Type</i>	<i>Description</i>
Return Value	Returns a collection containing objects for all open projects in the Desktop		

Python Syntax	GetProjects()
Python Example	<pre>for pjt in oDesktop.GetProjects(): AddInfoMessage(str(pjt.GetName()))</pre>

GetSchematicEnvironment

Returns the value of the current schematic environment.

UI Access	N/A
Parameters	None.
Return Value	Integer • 0 - Circuit. • 1 - Twin Builder. • 2 - Maxwell circuit.

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

GetSysLibDirectory

Get the path to the `SysLib` directory.

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

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

GetTempDirectory

Gets the path to the `Temp` directory.

UI Access	NA		
Parameters	<i>Name</i> None	<i>Type</i>	<i>Description</i>
Return Value	String The path to the Temp directory.		

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

GetUserLibDirectory

Gets the path to the UserLib directory.

UI Access	NA		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	None		
Return Value	String		
	The path to the <code>UserLib</code> directory.		

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

GetVersion

Returns a string representing the version.

UI Access	NA		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	None		
Return Value	String The version string of the product.		

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

LaunchJobMonitor

For use in starting job monitoring. This brings up the Monitor Job dialog.

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

Python Syntax	LaunchJobMonitor()		
Python Example	oDesktop.LaunchJobMonitor("C:\\projects\\basic.aedt")		

NewProject

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

UI Access	File>New		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>

	None
Return Value	Object, the project that is added.

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

OpenProject

Load a project from disk. The opened project will become the active project.

UI Access	File>Open		
Parameters	<i>Name</i> <FileName>	<i>Type</i> String	<i>Description</i> Full path of the project to open
Return Value	Object, the project that is opened		

Python Syntax	OpenProject(<Filename>)
Python Example	<pre>oDesktop.OpenProject("C:/Projects/MyProject.aedt")</pre>

PauseRecording

Temporarily stop script recording.

UI Access	NA		
Parameters	<i>Name</i> None	<i>Type</i>	<i>Description</i>
Return Value	None		

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

PauseScript

Pause the execution of the script and pop up a message. The script execution will not resume until you choose **Tools > Resume Script**.

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

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

Print

Prints the contents of the active view window.

UI Access	File>Print		
Parameters	<i>Name</i> None	<i>Type</i>	<i>Description</i>
Return Value	None		

Python Syntax	Print()
Python	<code>oDesktop.Print()</code>

Example	

QuitApplication

Exits the desktop.

UI Access	File>Exit		
Parameters	<i>Name</i> None	<i>Type</i>	<i>Description</i>
Return Value	None		

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

RefreshJobMonitor

For use in monitoring a job.

UI Access	Monitor Jobs		
Parameters	<i>Name</i> None	<i>Type</i>	<i>Description</i>
Return Value	A string specifying the job state. The result could be one 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>

RestoreWindow

Restores a minimized Twin Builder window

UI Access	NA		
Parameters	<i>Name</i> None	<i>Type</i>	<i>Description</i>
Return Value	None		

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

ResumeRecording

Resume recording a script.

UI Access	NA		
Parameters	<i>Name</i> None	<i>Type</i>	<i>Description</i>
Return Value	None		

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

RunProgram

Runs an external program.

UI Access	NA		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<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.
Return Value	<ArgArray>	Array of Strings	Arguments to pass to the program. If no arguments, pass in <code>None</code>
	None		

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

RunScript

Launches another script from within the currently executing script.

UI Access	Tools>Run Script

Parameters	<i>Name</i> <ScriptPath>	<i>Type</i> String	<i>Description</i> 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 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. Twin Builder installation directory.
	Return Value	Long	the return code for the script method.

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

RunScriptWithArguments

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

UI Access	NA		
Parameters	<i>Name</i> <ScriptPath>	<i>Type</i> String	<i>Description</i> 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: directory "userlib". The UserLib directory can be specified in Tools>Options>General Options on the 'Project Options' tab. - System library directory: directory "syslib". The SysLib directory can be specified in Tools>Options>General Options on the 'Project Options' tab. - 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

Select 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, and if unable to get the information, it then displays the **Select Scheduler** dialog box, waiting for you to complete the settings.

UI Access	Select Scheduler		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	option: String for the scheduler to select. (Not case sensitive.)	string	blank 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.)
	address (optional)	string	String specifying the IP address or host-name of the head node (e.g. for Windows HPC), 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(<i>option, address, username, forcePasswordEntry</i>)

Python Example	<pre>result = oDesktop.SelectScheduler("Windows HPC", "headnode.win.example.com")</pre>

SetActiveProject

Specify the name of the project that should become active in the desktop. Return that project.

UI Access	NA		
Parameters	<i>Name</i> <ProjectName>	<i>Type</i> String	<i>Description</i> 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

Specify the name of the project that should become active in the desktop. Return that project. If you have 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	NA		
Parameters	<i>Name</i> <ProjectName>	<i>Type</i> String	<i>Description</i> The full pathname of the project already in the Desktop that is to be activated.
Return Value	Object, the project that is activated.		

Python	SetActiveProjectByPath(<i>ProjectName</i>)

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

SetCustomMenuSet

Sets the value of **CustomMenuSet** in **Tools > Options > General Options > General > Desktop Configuration > Custom Menu Set**.

UI Access	N/A		
Parameters	<i>Name</i> <CustomMenuSet>	<i>Type</i> String	<i>Description</i> The name of the selected custom menu set.
Return Value	None		

Python Syntax	SetCustomMenuSet(< CustomMenuSet >)
Python Example	<pre>oDesktop.SetCustomMenuSet("Twin Builder")</pre>

SetDesktopConfiguration

Sets the value of the Desktop configuration in **Tools > Options > General Options > General > Desktop Configuration**.

UI Access	N/A		
Parameters	<i>Name</i> < <i>ConfigName</i> >	<i>Type</i> String	<i>Description</i> Name of selected Desktop configuration.
Return Value	None		

Python	SetDesktopConfiguration (< ConfigName >)

Syntax	
Python Example	<code>oDesktop.SetDesktopConfiguration("Twin Builder")</code>

SetLibraryDirectory

Set the path to the SysLib directory.

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

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

SetProjectDirectory

Set the path to the Project directory

UI Access	NA		
Parameters	<i>Name</i> <DirectoryPath>	<i>Type</i> String	<i>Description</i> The path to the Project directory. This should be writeable.
Return Value	None		

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

SetSchematicEnvironment

Sets the Desktop schematic environment.

UI Access	N/A		
Parameters	<i>Name</i> <schEnv Enum>	<i>Type</i> Integer	<i>Description</i> 0 - Circuit. 1 - Twin Builder. 2 - Maxwell circuit.
Return Value	None		

Python Syntax	SetSchematicEnvironment (<schEnvEnum>)
Python Example	oDesktop. SetSchematicEnvironment (1)

SetTempDirectory

Sets the path to the Temp directory.

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

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

ShowDockingWindow

Shows or hides a docking window.

UI Access	Right click docking window then select Show or Hide .		
Parameters	<i>Name</i> <windowName>	<i>Type</i> String	<i>Description</i> 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 Twin Builder for the specified number of milliseconds, up to 60,000 milliseconds (1 minute).

UI Access	NA		
Parameters	<i>Name</i> <TimeInMilliseconds>	<i>Type</i> Integer	<i>Description</i> 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

To submit 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	Submit Job		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	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(<i>settingsPath</i> , <i>projectPath</i> , <i>design</i> , <i>setup</i>)
Python Example	<pre> jobIDs = oDesktop.SubmitJob("C:\\hpc-settings\\Submit_ Job_Settings.ared", "C:\\projects\\basic.aedt") moreIDs = oDesktop.SubmitJob("C:\\hpc-settings\\Submit_ Job_Settings.ared", "C:\\projects\\basic.aedt", "Design1", "Setup1") </pre>

6 - 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	obj = oRunningInstances.GetAllRunningInstances()

GetRunningInstanceByProcessID

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

UI Access	NA		
Parameters	<i>Name</i> <processID>	<i>Type</i> Integer	<i>Description</i> Process ID

Return Value	String of the returned instance.

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

GetRunningInstanceByProject

Returns the instance of Ansys Electronics Desktop that contains a specified open project.

UI Access	NA		
Parameters	<i>Name</i> <projectNameOrPath>	<i>Type</i> String	<i>Description</i> Project name or full file path.
Return Value	String of the returned instance.		

Python Syntax	GetRunningInstanceByProject(<projectNameOrPath>)
Python Example	<pre>obj = oRunningInstances.GetRunningInstanceByProject('my_project')</pre>

7 - Project Object Script Commands

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

```
oProject.CommandName(<args>)
```

Many of the following commands take design names as arguments or return design names. For Twin Builder, the name is made up of the top-level design name and, if the design is not top level, the **InstanceName** for each parent in order from the top level design down to the design in question, separated by periods ("."). To find the **InstanceName** for a subdesign, navigate to its parent, then double click on the component instance in the schematic that represents the design, and go to the **Parameters** tab. The **InstanceName** property is toward the top of the list. Note: the **InstanceName** is part of the string used to identify the design in the project tree.

The format of the name string is:

```
<TopDesignName>::<InstanceName>.<InstanceName>.....<InstanceName>
```

where *TopDesignName* is the name of the top level design, and *InstanceName* is the value of the **InstanceName** property of the component instance representing the appropriate subdesign, with the last *InstanceName* in the list for the desired design.

AnalyzeAll [Project]

Simulates all setups including optmetrics setups. The UI waits until simulation is finished before continuing with the script.

UI Access	Project>Analyze All		
Parameters	<i>Name</i> None	<i>Type</i>	<i>Description</i>
Return Value	None		

Python Syntax	AnalyzeAll()
Python Example	oProject.AnalyzeAll()

ClearMessages

Clears the Message Window

UI Access	NA		
Parameters	<i>Name</i> None	<i>Type</i>	<i>Description</i>
Return Value	None		

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

Close [Project object command]

Closes the project. Unsaved changes will be lost.

UI Access	NA		
Parameters	<i>Name</i> None	<i>Type</i>	<i>Description</i>
Return Value	None		

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

CopyDesign

Copies a design

UI Access	Edit>Copy		
Parameters	<i>Name</i> DesignName	<i>Type</i> String	<i>Description</i> Name of the design
Return Value	Value		

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

CutDesign

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

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

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

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.

DeleteDesign

Deletes a specified design in the project

UI Access	Edit>Delete		
Parameters	<i>Name</i> <DesignName>	<i>Type</i> String	<i>Description</i> The name of the design
Return Value	None		

Python Syntax	DeleteDesign (<DesignName>)

Python Example	<code>oProject.DeleteDesign ("SimplorerDesign2")</code>

GetActiveDesign

Returns the design in the active project

UI Access	NA		
Parameters	<i>Name</i> None	<i>Type</i>	<i>Description</i>
Return Value	The active design		

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

Note:

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

GetDefinitionManager

Obtain the `DefinitionManager` object.

UI Access	NA		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	None		
Return Value	Object for the <code>DefinitionManager</code>		

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

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

GetDependentFiles

Provides a list of the external files referenced in the project, including characteristic (for example, MDX) and coupled project files; does not include those in <libdir>/Data directories.

UI Access	NA		
Parameters	<i>Name</i> None	<i>Type</i>	<i>Description</i>
Return Value	The list of referenced files.		

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

Get Design

Returns the specified design.

UI Access	NA		
Parameters	<i>Name</i> <DesignName>	<i>Type</i> String	<i>Description</i> Name of the design to return
Return Value	The specified design		

Python Syntax	<code>GetDesign(<DesignName>)</code>
Python Example	<code>oDesign = oProject.GetDesign ("SimplorerDesign1")</code>

GetDesigns

Obtain objects for all the designs in the project

UI Access	NA		
Parameters	<i>Name</i> None	<i>Type</i>	<i>Description</i>
Return Value	A collection containing objects for all designs in the project		

Python Syntax	GetDesigns()
Python Example	<pre>for dsn in oProject.GetDesigns(): AddInfoMessage(dsn.GetName())</pre>

GetName [Project]

Obtain the project name

UI Access	NA		
Parameters	<i>Name</i> None	<i>Type</i>	<i>Description</i>
Return Value	String, the project name not including the path or extension.		

Python Syntax	GetName()
Python Example	<pre>AddInfoMessage (str(oProject.GetName()))</pre>

GetPath [Project]

Returns the location of the project on disk

UI Access	NA		
Parameters	<i>Name</i> None	<i>Type</i>	<i>Description</i>
Return Value	String The path to the project, which does not include the project name.		

Python Syntax	GetPath()
Python Example	AddInfoMessage (str (oProject.GetPath()))

GetProperties [Project]

Gets a list of all the properties belonging to a specific PropServer (the owner of the properties, or the list containing them) and PropTab (basically, the type or category of property). This command can be executed by the oProject, oDesign, or oEditor objects. (This command can be used for project properties when it's invoked from oProject.) The PropServer and PropTab argument choices are listed below.

UI Access	NA		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<PropTab>	String	"ProjectVariableTab"
	<PropServer>	String	"ProjectVariables"
Return Value	Variant array of strings – the names of the properties belonging to the prop server.		

Python Syntax	GetProperties(<PropTab>, <PropServer>)
Python	all_props = oProject.GetProperties("ProjectVariableTab",

Example	"ProjectVariables")

GetTopDesignList

Returns a list of the names of the top-level designs.

UI Access	NA		
Parameters	<i>Name</i> None	<i>Type</i>	<i>Description</i>
Return Value	An array of strings that are the names of the top-level designs.		

Python Syntax	GetTopDesignList()
Python Example	<pre>name_list = oProject.GetTopDesignList() for i in nameList: AddInfoMessage(str(i))</pre>

InsertDesign

Adds a subcircuit to the current design. If the named design already exists, a dialog is brought up allowing the choice of adding an instance of the existing design, or an instance of a new design which is a copy of the existing one. Note that adding a subdesign to a design that is itself used more than once will add instances of the new subdesign in the same measure.

UI Access	Twin Builder>SubCircuit>Add SubCircuit		
Parameters	<i>Name</i> <designName>	<i>Type</i> String	<i>Description</i> The name of the new design.
Return Value	The added design object.		

Python	InsertDesign("", <designName>, "", "")

Syntax	
Python Example	<code>oDesign2 =oDesign.InsertDesign("", "AmpChannel", "", "")</code>

Paste [Project]

Pastes a design in the active project.

UI Access	Edit>Paste		
Parameters	<i>Name</i> None	<i>Type</i>	<i>Description</i>
Return Value	None		

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

Redo [Project]

Reapplies the last project-level command.

UI Access	Edit>Redo		
Parameters	<i>Name</i> None	<i>Type</i>	<i>Description</i>
Return Value	None		

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

Rename [Project]

Renames the project and saves it. Similar to `SaveAs`, except `SaveAs` requires a path.

UI Access	Edit>Rename		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<NewName>	String	The desired name of the project. The path is optional.
	<OverWriteOk>	Boolean	Whether to overwrite the same named file on disk if it exists. True to overwrite.
Return Value	None		

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

Save [Project]

Saves the active project.

UI Access	File>Save		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	None		
Return Value	None		

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

SaveProjectArchive

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

UI Access	File>Archive		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<archiveFilePath>	String	Path to archived file
	<IncludeExternalFiles>	Boolean	Whether to include external files associated with the project, True or False
	<IncludeResultsFiles>	Boolean	Whether to include simulation files associated with the project, True or False
	<AdditionalFiles>	Array	Whether to include additional specified files
	<ArchiveNotes>	String	A descriptive string about the archive
Return Value	None		

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

SaveAs [Project]

Saves the project under a new name. Similar to `Rename` except `SaveAs` requires a path.

UI Access	File>Save As		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<NewName>	String	The desired name of the project, with directory and extension.
	<OverWriteOK>	Boolean	Whether to overwrite the same-named file on disk, if it exists. True to overwrite.
Return Value	None		

Python Syntax	SaveAs <NewName> <OverWriteOK>
Python Example	<code>oProject.SaveAs ("D:/projects/project1.aedt", true)</code>

SetActiveDefinitionEditor

Obtain a particular definition editor.

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

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

SetActiveDesign

Sets a new design to be the active design. Specify the name of the design that should become active and return that design.

UI Access	NA

Parameters	<i>Name</i> <DesignName>	<i>Type</i> String	<i>Description</i> Name of the design to set as the active design.
Return Value	Object for the specified design .		

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

SimulateAll

Simulate all the setups in all the designs in the project. Script processing only continues when all analyses are finished.

UI Access	NA		
Parameters	<i>Name</i> None	<i>Type</i>	<i>Description</i>
Return Value	None		

Python Syntax	SimulateAll()
Python Example	oProject.SimulateAll()

Undo [Project]

Undo the last project level command.

UI Access	Edit>Undo		
Parameters	<i>Name</i> None	<i>Type</i>	<i>Description</i>

Return Value	None

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

UpdateDefinitions

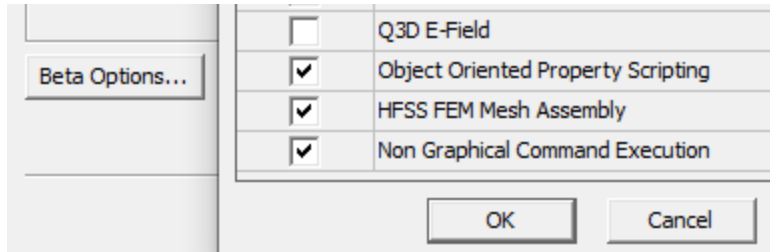
Updates all definitions. Warning messages are sent to the message window when definitions can't be found, informational messages when definitions are updated.

UI Access	The script method performs the same as if you ran Tools/ProjectTools/Update Definitions (the dialog default is to list "newer" definitions), then clicked Select All and then Update .		
Parameters	<i>Name</i> None	<i>Type</i>	<i>Description</i>
Return Value	None		

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

8 - Object Oriented Property Scripting

Object oriented property scripting is a Beta feature. Select **Tools > Options > General Options > Desktop Configuration**, click **Beta Options...** and select the **Object Oriented Property Scripting** check box.



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, such as name of a design, solve setup, boundary, and so on.
- **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 Twin Builder design object has these child types. The following examples show how the commands run in the **Tools > Open Command Window** for IronPython.

```
>>> project = oDesktop.GetActiveProject()
>>> design = project.GetActiveDesign()
>>> design.GetChildTypes()

['Module', '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 Twin Builder design object has these children.

```
>>> design.GetChildNames('Module')
```

```
['Analysis', 'Optimetrics', 'Results']  
>>> design.GetChildNames('Design Data')  
['Variables', 'Design Settings']
```

- **GetChildObject**
 - Input: String -- Object path. The path may include multiple generations.
 - Return the child object if found.
- **GetPropNames**
 - Input: [BOOL - IncludeReadOnly] -- default to true.
 - Return an array of the object's properties.
- **GetPropValue**
 - Input: String – Property Path. The path may include multiple generations.
 - Return the property value as VARIANT.
- **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.

Child Modules Available for Twin Builder Design

- Analysis, Optimetrics, Results

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  
("TwinBuilder1")  
  
>>> a = d.GetChildObject("Analysis").GetChildNames  
()  
  
>>> setups = d.GetChildObject("analysis")
```

The children of this module script object in a TwinBuilder design is categorized into different solve setup types, such as “Transient”, “AC”, and “DC”.

9 - Definition Manager Script Commands

Python Example	<pre>oProject = oDesktop.SetActiveProject("Project1") oDefinitionManager = oProject.GetDefinitionManager()</pre>

[AddMaterial](#)

[EditMaterial](#)

[ExportMaterial](#)

[GetData](#)

[GetNames](#)

[GetProperties](#)

[IsUsed](#)

[RemoveUnused](#)

[RemoveMaterial](#)

The following Definition Manager Script Command sets are available:

[Component Manager Script Commands](#)

[Symbol Manager Script Commands](#)

[Material Manager Script Commands](#)

[Script and Library Script Commands](#)

[Model Manager Script Commands](#)

[Package Manager Script Commands](#)

[NdExplorer Manager Script Commands](#)

AddMaterial

Add Material command in the material editor

UI Access	NA		
Parameters	<i>Name</i> <MatProperty>	<i>Type</i> Type	<i>Description</i> Different properties of the material.

	<PropertyName> String Should be one of the following: "permittivity", "permeability", "conductivity" "dielectric_loss_tangent", "magnetic_loss_tangent", "saturation_mag", "lande_g_factor", "delta_H" <Unit> String Possible values: delta_H: "Oe" saturation_mag: "Gauss", "uGauss", "Tesla", "uTesla" other properties: "" (empty string)
Return Value	None

Python Syntax	AddMaterial ["NAME:<MaterialName>", <MatProperty>, <MatProperty>, ...]
Python Example	oDefinitionManager.AddMaterial ["permittivity:=", "2.2", "0.002"]

EditMaterial

Modifies an existing material

UI Access	View/Edit Materials command in the material editor		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<OriginalName>	String	Name of the material before editing
	<NewName>	String	New name for the material
Return Value	None		

Python Syntax	EditMaterial (<OriginalName>, ["NAME:<NewName>", <MatProperty>, <MatProperty>, ...])
Python Example	<pre>oDefinitionManager.EditMaterial("alumina_92pct", ["NAME:alumina_92pct", "CoordinateSystemType:=", "Cartesian", "BulkOrSurfaceType:=" , 1, ["NAME:PhysicsTypes", "set:=" , ["Electromagnetic","Thermal","Structur], "permittivity:=" , "9.3", "dielectric_loss_tangent:=", "0.008", "thermal_conductivity:=", "26", "mass_density:=" , "3720", "specific_heat:=" , "790", "youngs_modulus:=" , "2670000000000", "poissons_ratio:=" , "0.26", "thermal_expansion_coeffcient:=", "7.2e-006"])</pre>

ExportMaterial

Exports a local material to a library

UI Access	Export to Library command in the material editor		
Parameters	<i>Name</i> <ExportData>	<i>Type</i> Array	<i>Description</i> Array("NAME:<LibraryName>",

	<MaterialName>, <MaterialName>, ...)
Return Value	None

Python Syntax	ExportMaterial (<ExportData>, <Library location>)
Python Example	<pre>oDefinitionManager.ExportMaterial (["NAME:mo0907b", _ "Material1", "Material2", "Material3"], "UserLib")</pre>

GetData [Material Manager]

Gets data that describes the material definition

UI Access	NA		
Parameters	<i>Name</i> <SimpleName>	<i>Type</i> String	<i>Description</i> Simple name of the definition to edit.
Return Value	<DefinitionData> This is an array of strings with basic package data		

Python Syntax	GetData(<SimpleName>)
Python Example	<pre>dataArray = oMaterialManager.GetData ("mymaterial")</pre>

GetNames [Material Manager]

Returns an array of names of materials (used and unused) in the project.

UI Access	NA		
Parameters	Name	Type	Description
	None		

Return Value	An array of strings

Python Syntax	GetNames()
Python Example	<code>materialnames = oMaterialManager.GetNames()</code>

GetProperties [Material Manager]

To obtain a list of the definitions in the specified library.

UI Access	NA		
Parameters	<i>Name</i> <materialname>	<i>Type</i> String	<i>Description</i> The material name.
Return Value	Array of strings; the properties of the material.		

Python Syntax	GetProperties (<materialname>)
Python Example	<code>oMaterialManager.GetProperties("Gold")</code>

IsUsed [Material Manager]

Used to determine if a material is used in the design.

UI Access	NA		
Parameters	<i>Name</i> <ComboName>	<i>Type</i> String	<i>Description</i> Composite name of the material.
Return Value	Boolean True if the specified material is used in the design		

Python Syntax	IsUsed(<ComboName>)
Python Example	<pre>isused = oMaterialManager.IsUsed("mylib:mymaterial")</pre>

RemoveUnused [Material Manager]

Removes materials that are not used in the design.

UI Access	Project->Remove Unused Definitions is similar.		
Parameters	<i>Name</i> None	<i>Type</i>	<i>Description</i>
Return Value	Boolean True if one or more materials are removed.		

Python Syntax	RemoveUnused()
Python Example	<pre>thereWereExtras = oMaterialManager.RemoveUnused()</pre>

RemoveMaterial

Removes a material from a library

UI Access	Remove Material(s) command in the material editor		
Parameters	<i>Name</i> <MaterialName> <IsProjectMaterial> <LibraryName>	<i>Type</i> String Boolean String	<i>Description</i> Name of the material to be removed. If true, Twin Builder assumes the material is a project material. In this case, the last two parameters will be ignored. The name of the user or personal library

	<LibraryLocation> String where the material resides. Should be "UserLib" or "PersonalLib".
Return Value	None

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

Component Manager Script Commands

Python Example	<pre>oDefinitionManager = oProject.GetDefinitionManager() oComponentManager = oDefinitionManager.GetManager("Component")</pre>

Add [Component]

Add a component

UI Access	Tools > Edit Libraries > Components > Add Component		
Parameters	<i>Name</i> <ComponentName> <ComponentInfo>	<i>Type</i> String Array	<i>Description</i> Simple name of the component 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>	Integer	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: Circuit -- 1 binary, 1 decimal, 0x1 System -- 10 binary, 2 decimal, 0x2 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>	Integer	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	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	One of "Electrical", "Magnetic", "Fluidic", "Translational", "Translational_V", "Rotational", "Rotational_V", ""Radiant", "Thermal", or <VHDLPackageName> VHDL package name in the form <Library>.<Package>
<VHDLPackageName>	String	
<Library>	String	Name of the VHDL Library
<Package>	String	Name of the VHDL Package
<VariableInfo>	Array	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, RHD - readonly, hidden, & has description parameter
<CheckBoxInfo>	Array	Array(<string>, // name <FlagLetters>, <string>, // description

		"CB:=", <string>, // optional - script for call back
<ButtonInfo>	Array	<bool>) // value: true or false Array(<string>, // name <FlagLetters>, <string>, // description <string>, // button title <string>, // extra text <ClientID>, "ButtonPropClientData:= ", <ClientDataArray>) Specifies Button Prop Client. 0 - unknown, ButtonPropClientData array will be empty 1 - Netlist Prop Client 2 - not used 3 - File Name Prop Client <ClientID> is 0 or 1: empty array Array() <ClientID> is 3: Array("InternalFormatText:=", "<prefix><RelativePath>") "<Project>", "<PersonalLib>", "<User-Lib>", or "<SysLib>" Relative path to file from <prefix> Array(<string>, // name <FlagLetters>, <string>, // description "CB:=", <string>, // optional - script for call back <string>) // value: a text string
<ClientID>	Integer	
<ClientDataArray>	Varies with <ClientID>	
<prefix>	String	
<RelativePath>	String	
<TextInfo>	Array	

<NumberInfo>	Array	Array(<string>, // name <FlagLetters>, <string>, // description "CB:=", <string>, // optional - script for call back <real>, // value: a number <string>) // units
<SeparatorInfo>	Array	Array(<string>, // name <FlagLetters>, <string>, // description "CB:=", <string>, // optional - script for call back <string>) // value: a text string
<ValueInfo>	Array	Array(<string>, // name <FlagLetters>, <string>, // description "CB:=", <string>, // optional - script for call back <string>) // value: a number, variable or expression
<MenuPropInfo>	Array	Array(<string>, // name <FlagLetters>, <string>, // description <string>, // menu choices - separated by commas <int>) // 0 based index of current menu choice
<VPointInfo>	Array	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	Array(<string>, // name
		<FlagLetters>,
		<string>, // description
		"CB:=", <string>, // optional - script for call back
		<real>, // x value
		<real>) // y value
<QuantityPropInfo>	Array	Array(<string>, // name
		<FlagLetters>,
		<string>, // description
		<string>, // value
		<TypeString>,
		<TypeStringDependentInfo>)
<TypeString>	String	"Across", "Through", or "Free"
<TypeStringDependentInfo>	Varies	If <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"

	If <TypeString> is "Across" or "Through": <int>, // terminal 1 <int> // terminal 2
Return Value	Type: 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.

Python Syntax	<pre>Add [("NAME:<ComponentName>", "Info:=", <ComponentInfo>, "RefBase:=", <string>, // reference designator "NumParts:=", <int>, // parts per component "OriginalComponent:=", <string> "Terminal:=", <TerminalInfo>, "Terminal:=", <TerminalInfo>, ... The remaining parameters are optional. ["NAME:Parameters", // any combo of the following "VariableProp:=", <VariableInfo>, "CheckboxProp:=", <CheckBoxInfo>, "ButtonProp:=", <ButtonInfo>, "TextProp:=", <TextInfo>, "NumberProp:=", <NumberInfo>, "SeparatorProp:=", <SeparatorInfo>, "ValueProp:=", <ValueInfo>, "MenuProp:=", <MenuInfo>], ["NAME:Properties", Any combination of the following: "CheckboxProp:=", <CheckBoxInfo>, "TextProp:=", <TextInfo></pre>

	<pre> "NumberProp:=", <NumberInfo>, "SeparatorProp:=", <SeparatorInfo>, "ValueProp:=", <ValueInfo>, "MenuProp:=", <MenuInfo>, "VPointProp:=", <VPointInfo>, "PointProp:=", <PointInfo>], ["Quantities", "QuantityProp:=", <QuantityPropInfo>...], ["NAME:CosimDefinitions", <CosimDefInfo>, <CosimDefInfo>...]] </pre>
Python Example	<pre> oComponentManager.Add(["NAME:Component", "Info:=", ["Type:=", 0, "NumTerminals:=", 0, "DataSource:=", "", "ModifiedOn:=", 1467910752, "Manufacturer:=", "", "Symbol:=", "Component", "ModelNames:=", "", "Footprint:=", "", "Description:=", "", "InfoTopic:=", "", "InfoHelpFile:=", "", "IconFile:=", "", "Library:=", "" </pre>

	<pre>"OriginalLocation:=", "Project", "IEEE:=", "", "Author:=", "", "OriginalAuthor:=", "", "CreationDate:=", 1467910746, "ExampleFile:=", ""], "Refbase:=", "U", "NumParts:=", 1, "ModSinceLib:=", True, "CompExtID:=", 2])</pre>

AddFMUModel [Component]

It imports FMU model and creates component.

UI Access	Twin Builder>SubCircuit>Add FMU Component...		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<FileName>	String	The file path and name of the FMU model file.
	<ShowInternalOutputs>	Boolean	It controls whether to show internal or local output variables in component or not.
	<ShowHierarchyLevel>	Integer	It controls which hierarchy level interfaces to expose in the component. For example: sub1.sub2.width (hierarchy level = 2) Sub1.length (hierarchy level = 1)
	<AdditionalNames>	String	Default: 1 Comma separated interface names that should be included. For example: "sub1.sub2.width, sub1.length"
	<ExcludeNames>	String	Comma separated interface names that

	should be excluded. For example: "sub1.sub2.width, sub1.length" <UpdateFMUOptions> String Provide existing FMU component name to update the existing FMU component with new options or FMU file. <ModelName> String Provide unique name to use for imported FMU component.
Return Value	String Component Name

Python Syntax	AddFMUModel (<FileName>, ["NAME":Options", "ShowFMUVariablesOptions:=", <ShowInternalOutputs>, <ShowHierarchyLevel>, <AdditionalNames>, <ExcludeNames>]])
Python Example	<pre>oDefinitionManager = oProject.GetDefinitionManager() oComponentManager = oDefinitionManager.GetManager("Component") With default value of flags, which is false for both flag oComponentManager.AddFMUModel("C:\...\Add.fmu", ["NAME:Options"]) OR oComponentManager.AddFMUModel("C:\...\Add.fmu", ["NAME:Options", "ShowFMUVariablesOptions:=", [true, false]]) OR oComponentManager.AddFMUModel("C:\...\Excavator.fmu", ["NAME:Options", "ShowFMUVariablesOptions:=", [true, true]])</pre>

	<pre> OR oComponentManager.AddFMUModel("D:\\Temp\\FMU_Import_ Test\\FMUModel_TwinBuilder2.fmu", ["NAME:Options", "FMI Type:=" , [1], "ShowFMUVariablesOptions:=", [False,1,"",""], "UpdateFMUOptions:=" , [""], "ModelName:=" , ["FMUModel_TwinBuilder3"]]) </pre>

AddMotorCADROM [Component]

Import, create, or update a Motor-CAD ROM component.

UI Access	Twin Builder > Add Component > Add Motor-CAD ROM Component...		
Parameters	<i>Name</i> <ComponentName>	<i>Type</i> String	<i>Description</i> Component or model name.
	<MotorCADResultsFolder>	String	Motor CAD Results folder path which will be used by toolkit to generate model and create component.
	<Options>	Type MotorCAD ROM Options.	Options to create component.
Return Value	<i>Name</i>	<i>Type</i> String	<i>Description</i> Component name.

Type	“Name:Options”, “ShowPin:=”, <Array of Interfaces>, “Type:=”, [0],

MotorCAD ROM Options:	"ModelType:=", [0]		
	<i>Name</i> ShowPin	<i>Type</i> Array of interfaces	<i>Description</i> Interface names in array; for example, name1 , name2 , and so on.
	Type	Integer	0: Thermal
	ModelType	Integer	0: SML

Python Syntax	AddMotorCADROM(<Component Name>, <Motor-CAD Results directory>, <Options>)
Python Example	<pre>oDefinitionManager = oProject.GetDefinitionManager() oComponentManager = oDefinitionManager.GetManager ("Component") oComponentManager.AddMotorCADROM("MotorCADROM", "D:\\...\\e8_with SlotWJ", ["NAME:Options", "ShowPin:=" , ["Loss_Armature_Copper_Loss", "Loss_Rotor_Copper_Loss", "T_ Endcap_F"], "Type:=" , [0], "ModelType:=" , [0]])</pre>

AddPythonModel [Component]

Imports and creates a new Python component or updates an existing one.

UI Access	Twin Builder > Add Component > Add Python Component

Parameters	<i>Name</i> <ComponentName>	<i>Type</i> String	<i>Description</i> Component or model name.
	<Python File Name>	String	Python file path which will be imported to create component.
	<Interfaces>	Array of type Python Component Interface .	Interfaces of component.
Return Value	String Component name.		
Array of type Python Component Interface	Name:Interfaces, "<Name of Interface>", <Structure of type Python Component Interface >		

Python Component Interface		
<i>Name</i> <Type>	<i>Type</i> Integer	<i>Description</i> 0: Real 1:String
<Value>	String	Real value ("0.000") String value ("\"stringValue\"")
<Direction>	Integer	0: Input (Real or String) 1: Output (Real only)
<Description>	String	Description of interface
<Show Pin>	Boolean	True or False (true or false)

Python Syntax	AddPythonModel(<Component Name>, <Python filename>, <Array Of Interfaces>) <array of Interfaces>: ["Name:Interfaces", <Interface1 Name> , <Python Component Interface for Interface1>, ...] <Python component interface>: [<Type>, <Value>, <Direction>, <Description>, <Show Pin>]
Python Example	<pre> oDefinitionManager = oProject.GetDefinitionManager() oComponentManager = oDefinitionManager.GetManager("Component") #No interfaces oComponentManager.AddPythonModel ("MyPythonComp", "C:\...\PythonExample.py", </pre>

	<pre>["NAME:Interfaces"]) # or # Some interfaces oComponentManager.AddPythonModel ("MyPythonComp", "C:\...\PythonExample.py", ["NAME:Interfaces", "x1:=" , [0, "0.000", 0, "", True], "x2:=" , [0, "0.000", 1, "Test", True], "x3:=" , [1, "\"Test\"", 0, "", False]]) #string value with escape characters "\"Test\" (comment)</pre>

AddTwinModel

Add a .twin model to a design.

UI Access	Twin Builder>SubCircuit>Add Twin Component		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<FileName>	String	The file path and name of the .twin model file.
	<ComponentName>	String	The component name to update.
Return Value	String: The created component name		

Python Syntax	<pre>oComponentManager.AddTwinModel(<FileName>, ["NAME:Options", "ComponentName:=", [<ComponentName>]])</pre>
Python Example	Example1: To create new component name, with unique name (check return argument): <pre>oComponentManager.AddTwinModel ("C:\\<filepath>\\<SomeModel>.Twin", ["NAME:Options", "ComponentName:=" , [""]</pre>

	<pre>]) Example2: To update TwinModel1 component from selected file: oComponentManager.AddTwinModel ("C:\\<filepath>\\<SomeModel>.Twin", ["NAME:Options", "ComponentName:=" , ["TwinModel1"]]) </pre>

Edit [Component]

Modifies an existing component

UI Access	Tools > Edit Configured Libraries > Components > Edit Component		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<ComponentName>	String	Composite name of the component to edit.
	<NewComponentName>	String	New simple name for the component
	<ComponentInfo>	Array	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>	Integer	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: Circuit -- 1 binary, 1 decimal, 0x1 System -- 10 binary, 2 decimal, 0x2 Nexxim -- 100 binary, 4 decimal, 0x4 SIwaveDeNovo -- 1000 binary, 8 decimal, 0x8 Twin Builder -- 10000 binary, 16 decimal, 0x10 MaxwellCircuit -- 100000 binary, 32 decimal, 0x20
<ModifiedOnInfo>	Integer	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	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	<i>Twin Builder choices:</i> "Electrical", "Magnetic", "Fluidic", "Translational", "Translational_V", "Rotational", "Rotational_V", ""Radiant", "Thermal", or <VHDLPackageName>
<VHDLPackageName>	String	VHDL package name in the form <Library>.<Package>.
<Library>	String	Name of the VHDL library
<Package>	String	Name of the VHDL package.
<VariableInfo>	Array	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, RHD - readonly, hidden, & has description parameter
<CheckBoxInfo>	Array	Array(<string>, // name
		<FlagLetters>, <string>, // description "CB:=", <string>, // optional - script for call back <bool>) // value: true or false
<ButtonInfo>	Array	Array(<string>, // name

			<FlagLetters>, <string>, // description <string>, // button title <string>, // extra text <ClientID>, "ButtonPropClientData:= ", <ClientDataArray>)
	<ClientID>	Integer	Specifies Button Prop Client. 0 - unknown, "ButtonPropClientData array will be empty 1 - Netlist Prop Client 2 - not used 3 - File Name Prop Client <ClientID> is 0 or 1: Empty Array() <ClientID> is 3: Array("InternalFormatText:=", "<prefix><RelativePath>")
	<ClientDataArray>	Varies	"<Project>", "<PersonalLib>", "<User- Lib>", or "<SysLib>" Relative path to file from <prefix>. Array(<string>, // name <FlagLetters>, <string>, // description "CB:=", <string>, // optional - script for call back <string>) // value: a text string
	<prefix>	String	Array(<string>, // name
	<RelativePath>	String	<FlagLetters>, <string>, // description "CB:=", <string>, // optional - script for call back
	<TextInfo>	Array	<string>) // value: a text string
	<NumberInfo>	Array	Array(<string>, // name <FlagLetters>, <string>, // description "CB:=", <string>, // optional -

		script for call back <real>, // value: a number <string>) // units Array(<string>, // name <FlagLetters>, <string>, // description "CB:=", <string>, // optional - script for call back
<SeparatorInfo>	Array	
<ValueInfo>	Array	<string>) // value: a text string Array(<string>, // name <FlagLetters>, <string>, // description "CB:=", <string>, // optional - script for call back
<MenuPropInfo>	Array	<string>) // value: a number, variable or expression Array(<string>, // name <FlagLetters>, <string>, // description <string>, // menu choices - separated by commas <int>) // 0 based index of current menu choice
<VPointInfo>	Array	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	Array(<string>, // name

		<FlagLetters>, <string>, // description "CB:=", <string>, // optional - script for call back <real>, // x value <real>) // y value Array(<string>, // name <FlagLetters>, <string>, // description <string>, // value <TypeString>, <TypeStringDependentInfo>) "Across", "Through", or "Free" <TypeString> String <TypeStringDependentInfo>: If <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" If <TypeString> is "Across" or "Through" <int>, // terminal 1 <int> // terminal 2
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.

**Python
Syntax**

```

Edit <ComponentName>,
["NAME:<NewComponentName>",
  "Info:=", <ComponentInfo>,
  "RefBase:=", <string>, // reference designator
  "NumParts:=", <int>, // parts per component
  "OriginalComponent:=", <string>
  "Terminal:=", <TerminalInfo>,
  "Terminal:=", <TerminalInfo>, ...
  #The remaining parameters are optional
  ["NAME:Parameters", // any combo of the following
    "VariableProp:=", <VariableInfo>,
    "CheckboxProp:=", <CheckBoxInfo>,
    "ButtonProp:=", <ButtonInfo>,
    "TextProp:=", <TextInfo>,
    "NumberProp:=", <NumberInfo>,
    "SeparatorProp:=", <SeparatorInfo>,
    "ValueProp:=", <ValueInfo>,
    "MenuProp:=", <MenuInfo>],
  ["NAME:Properties", # any combo of the following
    "CheckboxProp:=", <CheckBoxInfo>,
    "TextProp:=", <TextInfo>,
    "NumberProp:=", <NumberInfo>,
    "SeparatorProp:=", <SeparatorInfo>,
    "ValueProp:=", <ValueInfo>,
    "MenuProp:=", <MenuInfo>),

```

	<pre> "VPointProp:=", <VPointInfo>, "PointProp:=", <PointInfo>), ["Quantities", "QuantityProp:=", <QuantityPropInfo>...], ["NAME:CosimDefinitions", <CosimDefInfo>, <CosimDefInfo>...]) </pre>
Python Example	<pre> name = oComponentManager.Edit ("Simplorer Circuit Elements\BJTs:Level01_NPN", _ ["NAME:Level01_NPN", "Info:=", ["Type:=", 4294901764,_ "NumTerminals:=", 3, "DataSource:=", "Ansoft built-in component",_ "ModifiedOn:=", 1152722112, "Manufacturer:=", "", _ "Symbol:=", "nexx_bjt_npn", "Footprint:=", "", _ "Description:=", "BJT, GP, NPN", "InfoTopic:=", "NXBJT1.htm", _ "InfoHelpFile:=", "nexximcomponents.chm", "IconFile:=", "bjtsn.bmp", _ "Library:=", "Nexxim Circuit Elements\BJTs",_ "OriginalLocation:=", "SysLibrary ", "Author:=", "", _ "OriginalAuthor:=", "", "CreationDate:=", 1152722102], _ "Refbase:=", "Q", "NumParts:=", 1, "Terminal:=", ["collector", _ "collector", "A", false, 6, 0, "", "Electrical"], _ "Terminal:=", ["base", "base", "A", false, _ 7, 0, "", "Electrical"], "Terminal:=", ["emitter", _ "emitter", "A", false, 8, 0, "", "Electrical"], _ ["NAME:Parameters", "TextProp:=", ["LabelID", _ "HD", "Property string for netlist ID", _ "Q@ID"], "TextProp:=", ["MOD", "D", _ </pre>

	<pre> "Name of model data reference", "required"], _ "VariableProp:=", ["AREA", "D", _ "Emitter area multiplying factor, which affects currents, resistances, and capacitances", "1"], _ "VariableProp:=", ["AREAB", "D", "Base AREA", _ "1"], "VariableProp:=", ["AREAC", "D", "Collector AREA", _ "1"], "VariableProp:=", ["DTEMP", "D", _ "The difference between element and circuit temperature (deg Cel)", _ "0"], "VariableProp:=", ["M", "D", _ "Multiplier factor to simulate multiple BJTs in parallel", _ "1"], "ButtonProp:=", ["NexximNetlist", "HD", "", _ "Q@ID %0 %1 %2 *MOD(@MOD) *AREA (AREA=@AREA) "& _ " *AREAB (AREAB=@AREAB) *AREAC (AREAC=@ " & _ "AREAC) *DTEMP (DTEMP=@DTEMP) *M (M=@M) ", _ "Q@ID %0 %1 %2 *MOD(@MOD) " & "*AREA (AREA=@AREA) *AREAB (AREAB=@AREAB) *AREAC (AREAC=@ " & _ "AREAC) *DTEMP (DTEMP=@DTEMP) *M (M=@M) ", 1, _ "ButtonPropClientData:=", [], "TextProp:=", ["ModelName", "HD", "", "Q"]]]) </pre>
Python Example 2	<pre> name2 = oComponentManager.Edit ("MyComponent",_ (["NAME:MyOtherComponent","Info:=", ["Type:=", 4294901767, _ "NumTerminals:=", 2, "DataSource:=", "", _ </pre>

```
"ModifiedOn:=", 1071096503, "Manufacturer:=", "Ansoft", _  
  
"Symbol:=", "bendo", "Footprint:=", "BENDO", _  
  
"Description:=", "", "InfoTopic:=", "", _  
  
"InfoHelpFile:=", "", "IconFile:=", "", _  
  
"LibraryName:=", "", "OriginalLocation:=", "Project", _  
  
"Author:=", "", "OriginalAuthor:=", "", _  
  
"CreationDate:= ", 1147460679], "Refbase:=", "U", _  
  
"NumParts:=", 1, "OriginalComponent:=", "", _  
  
"Terminal:=", ["n1", "n1", "A", false, 0, 0, "", _  
  
"Electrical"], "Terminal:=", ["n2", "n2", "A", _  
  
false, 1, 0, "", "Electrical"], ["NAME:Parameters", _  
  
"MenuProp:=", ["CoSimulator", "D", "", _
```

```

"Default,Custom,Netlist", 0], "ButtonProp:=", ["CosimDefin-
ition", _

"D", "", "", "Edit", 0, "ButtonPropClientData:=", []], _

["NAME:CosimDefinitions", ["NAME:CosimDefinition", _

"CosimulatorType:=", 0, "CosimDefName:=", "Planar EM", _

"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, _

```

	<pre>"CosimDefName:=", "Netlist", "IsDefinition:=", true, _ "NetlistString:=", ""]]])</pre>

Export [Component]

Export component(s) to a library

UI Access	Tools > Edit Configured Libraries > Components > Export to Library		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<LibraryName>	String	Name of the library
	<ComponentName>	String	Composite name of the component to export.
	<LibraryLocation>	String	Location of the library in <LibraryName>. Can be any of the following: "PersonalLib", or "UserLib"
Return Value	None		

Python Syntax	Export(["NAME:<LibraryName>", <ComponentName>, <ComponentName>...], <LibraryLocation>)
Python Example	<pre>oComponentManager.Export(["NAME:mylib", "Simpleorer Circuit Elements\BJTs:Level01_NPN"], "PersonalLib")</pre>

ExportAllProjectComps [Component]

To export all local components and their local reference symbols, models and packages to a set of libraries. Can be used as part of a script to convert model files to usable libraries.

UI Access	NA

Parameters	<i>Name</i> <LibraryName>	<i>Type</i> String	<i>Description</i> The base name of the set of libraries, one for each type of definition, to be exported
	<toPersonalLib>	Boolean	True for export to <code>PersonalLib</code> , false to export to <code>userLib</code>
Return Value	The added design object		

Python Syntax	<code>ExportAllProjectComps(<libraryName>, <toPersonalLib>)</code>
Python Example	<pre>oComponentManager = oDefinitionManager.GetManager("Component") ret2 = oComponentManager.ExportAllProjectComps ("exportTest1", True)</pre>

ExportDefs [Component]

Export definitions(s) to a library

UI Access	NA		
Parameters	<i>Name</i> <LibraryName>	<i>Type</i> String	<i>Description</i> Name of the library to which the definition should be written
	<LibraryLocation>	String	Location of the library in <LibraryName>. Can be "PersonalLib" or "UserLib"
	<ComboName>	String	Composite name of the definition to export.
	<Type>	String	is one of: "Components", "Symbols", "Models", "Packages", "Materials", "Scripts"
Return Value	None		

Python Syntax	<code>ExportDefs ([[<LibraryName>, <LibraryLocation>, <ComboName>, <Type>]], [<LibraryName>, <LibraryLocation>, <ComboName>, <Type>]]*)</code>
Python Example	<pre>oComponentManager.ExportDefs ([["mylib", "PersonalLibrary", "simmodel", "Models"]])</pre>

ExportDefinitionArchive [Component]

Packages up the specified definitions of the specified types, along with references and all related data files, including DLLs

UI Access	Tools>Export Definition Archive...		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<archiveName>	String	The name of the archive that will be created from the specified definitions
	<Deftype>	String	The type of definition. One of: "Component", "Model", "Symbol", "Package", "Script", "Material"
	<Name1>, <name2>	String	The full name of the definition, <i>lib-name:defname</i> , where <i>libname</i> is the library name and path from one of the library directories, and <i>defname</i> is the name of the definition. For a project definition, <i>libname</i> would be empty and the colon can be omitted.
Return Value	Boolean TRUE on success, FALSE on failure		

Python Syntax	ExportDefinitionArchive(<archiveName> [[<deftype>, <name1>,< name2>, ...], [<deftype>, <name1>, <name2>, ...]])
Python Example	<pre>ret = oComponentManager.ExportDefinitionArchive ("abc.definition.zip", ["Component", "Myvhdl:RLC", "Myvhdl:Res"])</pre>

GetData [Component]

Gets data that describes the definition.

UI Access	NA		
Parameters	<i>Name</i> <DefinitionName>	<i>Type</i> String	<i>Description</i> Composite name of the definition to edit.
Return Value	<DefinitionData> This is an array of data for the definition.		

Python Syntax	GetData(<DefinitionName>)
Python Example	<pre>compData = oComponentManager.GetData("Level01_NPN")</pre>

GetLibContents [Component]

Gets the names of the definitions contained in the specified library.

UI Access	NA		
Parameters	<i>Name</i> <filename>	<i>Type</i> String	<i>Description</i> The name of the library file, of any type.
Return Value	An array of definition names.		

Python Syntax	GetLibContents(<filename>)
Python Example	<pre>comps = oComponentManager.GetLibContents ("C:\\Program Files\\ANSYS Inc\\v252\\AnsysEM\\userlib\\mylib.aclb") for i in comps: AddInfoMessage(str(i))</pre>

GetModelicaObjectList [Component]

Getting Modelica model, clas,s or block names from a given .mo file.

UI Access	NA		
Parameters	<i>Name</i> None	<i>Type</i>	<i>Description</i>
Return Value	Multi dimensional array of object names		

Python Syntax	GetModelicaObjectList(<fileName>)
Python Example	<pre>oComponentManager.GetModelicaObjectList ("C:\tmp\MyModel.mo")</pre>

GetNames [Component]

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.

UI Access	NA		
Parameters	<i>Name</i> None	<i>Type</i>	<i>Description</i>
Return Value	An array of strings		

Python Syntax	GetNames()
Python Example	<pre>componentNames = oComponentManager.GetNames()</pre>

ImportDefinitionArchive [Component]

Import a package of definition libraries and related files.

UI Access	Tools>Import Definition Archive...		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<archiveName>	String	The name of the archive that will be read in.
	<personalOrUser>	Boolean	TRUE to install in personalLib , FALSE for userlib .

	<path> String The directory path under or in which to place the definitions from the archive. <projdir> String The directory path under or in which to place the definitions from the archive.
Return Value	Boolean TRUE on success, FALSE on failure

Python Syntax	<code>ImportDefinitionArchive (<archiveName> <personalOrUser>, <path>, <projdir>)</code>
Python Example	<code>oComponentManager.ImportDefinitionArchive ("abc.definition.zip", TRUE, "VHDLtest", "")</code>

ImportModelsFromFile [Component]

Imports simulation models from a file.

UI Access	Tools>Import Simulation Models		
Parameters	<i>Name</i> <modelFileFullPath>	<i>Type</i> String	<i>Description</i> The full path to the file with the model data.
	<Options> 1.<mode>	Integer	A value specifying the type of model: 1 = Twin Builder VHDL, SML, SPICE (PSPICE or SPICE3), Modelica, or DLL data 2 = SPICE (HSPICE) data 4 = IBIS model data 5 = IBIS EBD model data
	Note: Following Options only		

	when mode is 1. 2.<Models> Complex Model Array 3.<Components> Complex Component 4.<ModelicaOptions> Array Complex Modelica Option Array
Return Value	None on success. Error string otherwise.

Complex Model Array	Complex Array	Complex Array Structure
1 st element	String	"Name:Models"
For each model: Next Element <name of model>	string	"<modelName>:=" replace with actual model name, it must match with definition name in imported model file
Next Element <Options for model> Only 1 element in Array Import	Array <1 Element Array> bool	true
Repeat ...		

Complex Component Array	Complex Array	Complex Array Structure
1 st element	String	"Name:Components"
For each component:		
Next Element <name of Component/model>	string	"<modelName>:=" replace with actual model name, it must match with definition name in imported model file
Next Element <Options for Component>	Array	
6 element in Array:	<6 Element Array>:	
Import	Bool	True
Override	Bool	True
Symbol Name	String	<modelName> or any existing compatible library symbol name
Show pin description	Bool	True
SDB outputs/ Result Outputs	String	Comma separated list of names
Non-conservative Pins/ Symbol pins	String	"<property1>,<property2>..." Same as above
Repeat ...		

Complex Modelica Options Array	Complex Array	Complex Array Structure
1 st element	String	"Name:ModelicaOptions"
For each models:		
Next Element <name of model>	string	"<modelName>:=" replace with actual model name, it must match with definition name in imported model file
Next Element <Options for Modelica model>	Array	
5 element in Array:	<5 Element Array>:	
Type	Integer	0:Normal 1: Use external Solver
ShowInternalVariables	Bool	True/False to show/hide
ShowHierarchyLevel	Integer	0: top Level properties 1: top and one level below N: top and N level below hierarchy will be shown
<use below option to make specific selection that not covered under above general selection>		
Select additional interfaces	String	Comma separated list of names "<property1>,<property2>..."
Exclude Interfaces	String	Same as above
Repeat ...		

Python Syntax	<pre> ImportModelsFromFile (<modelFileFullPath>, ["NAME:Options", "Mode:=", <mode>]) </pre>
Python Example	<pre> oComponentManager.ImportModelsFromFile ("<Path>/BjT.vhd <path>/MSL/Modelica/package.mo;Modelica.Blocks.Examples.PID_Controller", </pre>

mple	<pre> ["NAME:Options", "Mode:=" , 1, ["NAME:ModelicaOptions", "PID_Controller:=" , [0,False,1,"PI.Nd,PI.Ni,PI.Td,PI.Ti,PI.controlError,PI.controllerType,PI.k,P.kFF,P l.wd,PI.wp,PI.xd_start,PI.xi_start,inertia2.J,inertia2.a","driveAngle"]], ["NAME:Models", "bjt:=" , [True], "PID_Controller:=" , [True]], ["NAME:Components", "bjt:=", [True,True,"bjt",True,"ctrl,rb","vf,vt"], "PID_Controller:=", [True,True,"PID_Controller",True,"inertia2_a,inertia1_J","PI_k,PI_kFF"]]]) </pre>

IsUsed [Component]

Used to determine if a component is used in the design.

UI Access	NA		
Parameters	<i>Name</i> <ComponentName>	<i>Type</i> String	<i>Description</i> Name of the Component
Return Value	Boolean True if the specified component is used in the design		

Python Syntax	IsUsed(<ComponentName>)		
Python Example	<pre>IsUsed = oComponentManager.IsUsed("MyComponent")</pre>		

LoadExample [Component]

Loads a component example file.

UI Access	Right click on a component in the project window and select Load Example.		
Parameters	<i>Name</i> <uniqueComponentName>	<i>Type</i> String	<i>Description</i> The unique name for a component (<libraryName>:<definitionName>) that has an example specified for it <libraryName> String The path from the library directory down to the library, not including the file extension <definitionName> String The name of the component in the library
Return Value	Boolean True if the component has an example, false otherwise.		

Python Syntax	LoadExample(<uniqueComponentName>)		
Python Example	<pre>oDefinitionMgr = oProject.GetDefinitionManager()</pre>		


```

oCompMgr = oDefinitionMgr.GetManager("Component")

compname = "Simplorer Elements\Basic Elements\States:STATE_
01"

success = oCompMgr.LoadExample(compname)

AddInfoMessage(str(success) + compname)

```

ManageAliases [Component]

Add, update or remove library aliases.

UI Access	Tools>Manage Library Aliases		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<action>	String	The action to be performed. One of "add", "remove" or "update".
	<shortcut>	String	The shorthand name to be expanded, or empty. An empty string in this argument will cause a prefix to be added..
	<libpath>	String	The expanded path to be substituted for the shorthand, or used as a prefix, depending on the previous argument.
	<saveToRegistry>	Boolean	True/false, whether to save to the registry. For continued use of the alias change, use True
Return Value	Boolean TRUE on success, FALSE on failure		

Python Syntax	ManageAliases (<action>, <shortcut>, <libpath>, <saveToRegistry>)
Python Example	<pre> oComponentManager.ManageAliases ("add", "spice", "Simplorer Elements\Spice", True) </pre>

Remove [Component]

Remove a component from a library

UI Access	Tools > Edit Configured Libraries > Components > Remove Component		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<ComponentName>	String	Composite name of the component to remove.
	<IsProjectComponent>	Boolean	
	<LibraryName>	String	Name of the library.
	<LibraryLocation>	String	Location of the library in <LibraryName>. Can be any of the following: "Project", "PersonalLib", or "UserLib"
Return Value	None		

Python Syntax	Remove (<ComponentName>, <IsProjectComponent>, <LibraryName>, <LibraryLocation>)
Python Example	<pre>oComponentManager.Remove ("Simplorer Circuit Elements\BJTs:Level01_NPN", _ true, "Project")</pre>

RemoveUnused [Component]

Removes components that are not used in the design.

UI Access	Project->Remove Unused Definitions is similar but operates slightly different and does not record script commands.
Parameters	<i>Name</i> None <i>Type</i> <i>Description</i>
Return Value	Boolean True if one or more components are removed.

Python	RemoveUnused()

Syntax	
Python Example	<code>removedDefs = oComponentManager.RemoveUnused()</code>

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 of an unused component is not unusable until after the component itself is removed using the Component Manager Remove script.

Symbol Manager Script Commands

```
oDefinitionManager = oProject.GetDefinitionManager()  
oSymbolManager = oDefinitionManager.GetManager("Symbol")
```

Add [Symbol Manager]

Add a symbol

UI Access	Tools > Edit Configured Libraries > Symbols > Add Symbol		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<SymbolName> <ModifiedOnInfo>	String Integer	Simple name of the symbol being added. 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	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	Type: <string>	N : normal pin I : input pin O : output pin
<OptionalPinInfo>	Boolean	Specify both or neither:	<bool>, // true if name is to be shown <bool>, // true if number is to be shown
<PropDisplayMapInfo>	Array	Array("NAME:PropDisplayMap",	<PropDisplayInfo>, <PropDisplayInfo>,...)
<PropDisplayInfo>	Array	<NameString>, Array	(<DisplayTypeInfo>, <DisplayLocationInfo>, <int>, // optional, level number <TextInfo>) <NameString> Type: <string> PropertyName:=, where PropertyName is the name of the property to be displayed.
<DisplayTypeInfo>	Integer	Type: <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> Integer	Type: <int> 0 : Left 1 : Top 2 : Right 3 : Bottom 4 : Center 5 : Custom placement
	<GraphicsDataInfo> Array	Array("NAME:Graphics", One or more of the following <RectInfo>, <CircleInfo>, <ArcInfo>, <LineInfo>, <PolygonInfo>, <TextInfo>, <ImageInfo>)
	<RectInfo> Array	"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>	Array	"Circle:=", Array(<real>, // line width <int>, // fill pattern <int>, // color <real>, // x position of center <real>, // y position of center < real>) // radius
<ArcInfo>	Array	"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>	Array	"Line:=", Array(<real>, // line width <int>, // line pattern <int>, // color <PointInfo>, // must specify at least 2 points <PointInfo>...)
<PointInfo>	Real	<real>, // x position <real> // y position
<PolygonInfo>	Array	"Polygon:=", Array(<real>, // line width <int>, // fill pattern <int>, // color <PointInfo>, // must specify at least 3 points <PointInfo>...)
<TextInfo>	Array	"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 Type: <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
	<Justification>	Integer
	<ImageInfo>	Array
	<ImageData>	String
		"Image:=", Array(<RectInfo>, <ImageData>, <bool>) // is mirrored <string> File path. <int> 0 : use the file path and link to it 1 : ignore file path and use next parameter <string> External data, only present if preceding <int> is 1.
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.	

Python Syntax	<pre>Add(["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</pre>
Python Example	<pre>oDefinitionManager = oProject.GetDefinitionManager() oSymbolManager = oDefinitionManager.GetManager("Symbol") oSymbolManager.Add(["NAME:Symbol", "ModTime:=" , 1469209994, "Library:=" , "", "ModSinceLib:=" , False, "LibLocation:=" , "Project", "HighestLevel:=" , 1, "Normalize:=" , True, "InitialLevels:=" , [0,1], ["NAME:Graphics"]])</pre>

Edit [Symbol Manager]

Edit an existing symbol

UI Access	NA		
Parameters	<i>Name</i> <SymbolName>	<i>Type</i> String	<i>Description</i> Composite name of the symbol being edited.
	<NewSymbolName>	String	new simple name for the symbol
	<ModifiedOnInfo>	Integer	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	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>	Boolean	// Specify both or neither <bool>, // true if name is to be shown

		<code><bool>, // true if number is to be shown</code> <code>Array("NAME:PropDisplayMap",</code> <code><PropDisplayInfo>,</code> <code><PropDisplayInfo>,...)</code> <code><NameString>, Array</code> <code>(<DisplayTypeInfo>,</code> <code><DisplayLocationInfo>,</code> <code><int>, // optional, level number</code> <code><TextInfo>)</code> <code>// PropertyName:=, where</code> <code>PropertyName is the name of</code> <code>// the property to be displayed</code> <code>// 0 : No display</code> <code>// 1 : Display name only</code> <code>// 2 : Display value only</code> <code>// 3 : Display both name and value</code> <code>// 4: Display evaluated value only</code> <code>// 5: Display both name and evaluated value</code> <code>// 0 : Left</code> <code>// 1 : Top</code> <code>// 2 : Right</code> <code>// 3 : Bottom</code> <code>// 4 : Center</code> <code>// 5 : Custom placement</code> <code>Array("NAME:Graphics",</code> <code>// one or more of the following</code> <code><RectInfo>,</code> <code><CircleInfo>,</code> <code><ArcInfo>,</code> <code><LineInfo>,</code>
<code><PropDisplayMapInfo></code>	Array	
<code><PropDisplayInfo></code>	Array	
<code><NameString></code>	String	
<code><DisplayTypeInfo></code>	Integer	
<code><DisplayLocationInfo></code>	Integer	
<code><GraphicsDataInfo></code>	Array	

		<PolygonInfo>, <TextInfo>, <ImageInfo>) "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
<RectInfo>	Array	
<CircleInfo>	Array	"Circle:=", Array(<real>, // line width <int>, // fill pattern <int>, // color <real>, // x position of center <real>, // y position of center < real>) // radius
<ArcInfo>	Array	"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>	Array	"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 "Polygon:=", Array(<real>, // line width <int>, // fill pattern <int>, // color <PointInfo>, // must specify at least 3 points
<PolygonInfo>	Array	
<TextInfo>	Array	<PointInfo>...) "Text:=", Array(<real>, // x position <real>, // y position <real>, // angle, in radians <Justification>, <bool>, // is plotter font <string>, // font name <int>, // color
<Justification>	Integer	<string>) // text string // 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>	Array	"Image:=", Array(<RectInfo>, <ImageData>, <bool>) // is mirrored
<ImageData>		// 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
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.

Python Syntax	<pre>Edit (<SymbolName>, ["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</pre>
Python Example	None

EditWithComps [Symbol Manager]

Edit a list of existing symbols

UI Access	NA		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<SymbolName>	String	composite name of the symbol being edited
	<NewSymbolName>	String	new simple name for the symbol

	<ModifiedOnInfo>	Integer	An integer that corresponds to the number of seconds that have elapsed
	<PinDefInfo>	Array	since 00:00 hours, Jan 1, 1970 UTC from the system clock. 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>	Boolean	// Specify both or neither <bool>, // true if name is to be shown <bool>, // true if number is to be shown
	<PropDisplayMapInfo>	Array	Array("NAME:PropDisplayMap", <PropDisplayInfo>, <PropDisplayInfo>,...)
	<PropDisplayInfo>	Array	<NameString>, Array (<DisplayTypeInfo>, <DisplayLocationInfo>,

		<int>, // optional, level number
<NameString>	String	<TextInfo>) // PropertyName:=, where PropertyName is the name of
<DisplayTypeInfo>	Integer	// the property to be displayed // 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>	Integer	// 0 : Left // 1 : Top // 2 : Right // 3 : Bottom // 4 : Center // 5 : Custom placement
<GraphicsDataInfo>	Array	Array("NAME:Graphics", // one or more of the following <RectInfo>, <CircleInfo>, <ArcInfo>, <LineInfo>, <PolygonInfo>, <TextInfo>, <ImageInfo>)
<RectInfo>	Array	"Rect:=", Array(<real>, // line width <int>, // fill pattern <int>, // color <real>, // angle, in radians

		<code><real>, // x position of center</code> <code><real>, // y position of center</code> <code><real>, // width</code> <code>< real>) // height</code> <code>"Circle:=", Array(<real>, // line width</code> <code><int>, // fill pattern</code> <code><int>, // color</code> <code><real>, // x position of center</code> <code><real>, // y position of center</code>
<code><CircleInfo></code>	Array	
		<code>< real>) // radius</code> <code>"Arc:=", Array(<real>, // line width</code> <code><int>, // line pattern</code> <code><int>, // color</code> <code><real>, // x position of center</code> <code><real>, // y position of center</code> <code>< real>, // radius</code> <code><real>, // start angle, in radians</code> <code><end>) // end angle, in radians</code> <code>"Line:=", Array(<real>, // line width</code> <code><int>, // line pattern</code> <code><int>, // color</code> <code><PointInfo>, // must specify at least 2</code> <code>points</code> <code><PointInfo>...)</code> <code><PointInfo>:</code> <code><real>, // x position</code> <code><real> // y position</code> <code>"Polygon:=", Array(<real>, // line width</code> <code><int>, // fill pattern</code> <code><int>, // color</code>
<code><ArcInfo></code>	Array	
<code><LineInfo></code>	Array	
<code><PolygonInfo></code>	Array	

	<TextInfo> Array <Justification> Integer <ImageInfo> Array <ImageData>	<PointInfo>, // must specify at least 3 points <PointInfo>...) "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 // 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 "Image:=", Array(<RectInfo>, <ImageData>, <bool>) // is mirrored // 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
Return Value	String	

	<pre>// 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.</pre>

Python Syntax	<pre>Edit (<SymbolName>, ["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 [<ListOfComponentNames>]) #Component names</pre>
Python Example	<pre>oSymbolManager.EditWithComps ("MySymbol",_ ("Nexxim Circuit Elements\Distributed\Distributed:bendo",_ ["NAME:bendo new name", "ModTime:=", 1152722165, _ "Library:=", "Nexxim Circuit Elements\Distributed\Distributed", _ "LibLocation:=", "SysLibrary", ["NAME:PinDef", _ "Pin:=", ["n1",-0.00254,0.00254,0,"N",0,0,False,0,_ True,"",False,False]],_ ["NAME:PinDef","Pin:=", ["n2",0.00254,-0.00254,_ 1.5708,"N",0,0,False,0,True,"",False,False]], _ ["NAME:Graphics","Polygon:=",_ [0, 0, 12566272, 0, 0.00381, 0, 0.00127, 0.00127, _ 0.00127, 0.00127, 0, 0.00381, 0, 0.00381, _</pre>

	<pre> 0.002032, 0.00127, 0.00381]], _ "Line:=", [0,1, 12566272, -0.00254, 0.00254, 0, 0.00254]], _ "Line:=", [0,1, 12566272, 0.00254, -0.00254, 0.00254, 0]], _ ["NAME:PropDisplayMap","W:=", [3,5,0, _ "Text:=", [2.1684E-019,0.00504119,0,1,5,False,"Arial", _ 0,"W=***"], ["MY_COMP"]]]])) </pre>

Export [Symbol Manager]

Exports symbol(s) to a library

UI Access	Tools > Edit Configured Libraries > Symbols > Export to Library		
Parameters	Name	Type	Description
	<LibraryName>	String	Name of the library.
	<SymbolName>	String	Composite name of symbol to export.
	<LibraryLocation>	String	Location of the library in <LibraryName>. Value can be: "Project", "PersonalLib", or "UserLib"
Return Value	None		

Python Syntax	<pre> Export Array("NAME:<LibraryName>",<SymbolName>, <SymbolName>...), <LibraryLocation> </pre>
Python Example	<pre> oSymbolManager.Export_ (["NAME Nexxim Circuit Elements\Distributed\Distributed:_ bendo:mylib","mySymbol"], "PersonalLib") </pre>

GetData [Symbol Manager]

UI Access	NA

Parameters	<table><tr><th><i>Name</i></th><th><i>Type</i></th><th><i>Description</i></th></tr><tr><td><DefinitionName></td><td>String</td><td>Composite name of the definition to edit.</td></tr></table>	<i>Name</i>	<i>Type</i>	<i>Description</i>	<DefinitionName>	String	Composite name of the definition to edit.
<i>Name</i>	<i>Type</i>	<i>Description</i>					
<DefinitionName>	String	Composite name of the definition to edit.					
Return Value	<DefinitionData> This is an array of data for the definition.						

Python Syntax	GetData(<DefinitionName>)
Python Example	<pre>symbolData = oSymbolManager.GetData("Nexxim Circuit Elements\Hspice:nexx_bjt_npn")</pre>

GetLibraryDefNames [Symbol Manager]

To obtain a list of the definitions in the specified library

UI Access	NA		
Parameters	<i>Name</i> <libname>	<i>Type</i> String	<i>Description</i> The library name, with path, starting from syslib, userlib or personalLib
Return Value	Array of strings, the composite names of definitions in the library.		

Python Syntax	GetLibraryDefNames (<libname>)
Python Example	<pre>oSymbolManager.GetLibraryDefNames("Simplorer Elements\Basic Elements\Circuit\Passive Elements")</pre>

GetNames [Symbol Manager]

UI Access	NA		
Parameters	Name	Type	Description
	None		

Return Value	An array of strings

Python Syntax	GetNames()
Python Example	<code>symbolNames = oSymbolManager.GetNames()</code>

IsUsed [Symbol Manager]

UI Access	NA		
Parameters	<i>Name</i> <SymbolName>	<i>Type</i> String	<i>Description</i> Name of the symbol
Return Value	Boolean True if the specified symbol is used in the design		

Python Syntax	IsUsed(<SymbolName>)
Python Example	<code>IsUsed = oSymbolManager.IsUsed("MySymbol")</code>

Remove [Symbol Manager]

Removes a symbol from a library

UI Access	Tools > Edit Configured Libraries > Symbols > Remove Symbol		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<SymbolName>	String	Composite name of the symbol to remove.
	<IsProjectSymbol>	Boolean	
	<LibraryName>	String	Name of the library
	<LibraryLocation>	String	Location of the library in <LibraryName>. One of "Project", "PersonalLib" or "UserLib"
Return Value	None		

Python Syntax	Remove (<SymbolName>, <IsProjectSymbol>, <LibraryName>, <LibraryLocation>)
Python Example	<pre>oSymbolManager.Remove("Nexxim Circuit Elements \Distributed\Distributed:bendo", true, "Project")</pre>

RemoveUnused [Symbol Manager]

UI Access	Project->Remove Unused Definitions is similar but operates slightly different and does not record script commands.		
Parameters	<i>Name</i> None	<i>Type</i>	<i>Description</i>
Return Value	Boolean True if one or more symbols are removed.		

Python Syntax	RemoveUnused()
Python Example	<pre>removedDefs = oSymbolManager.RemoveUnused()</pre>

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.

Script and Library Script Commands

```
oDefinitionManager = oProject.GetDefinitionManager()
```

```
oScriptManager = oDefinitionManager.GetManager("Script")
```

AddScript

Add Script in the script definition manager

UI Access	NA		
Parameters	<i>Name</i> <AddScriptArray>	<i>Type</i> Array	<i>Description</i> Array("NAME<ScriptName>", <string>, <ScriptLanguage>, <string> ("script language") <ScriptText>, <string> // text of script
Return Value	None		

Python Syntax	AddScript ([<AddScriptArray>])
Python Example	<pre>oDefinitionManager.AddScript (["NAME:MyScript", _ "ScriptLang:=", "python", _ "ScriptText:=", "MsgBox(" & Chr(34) & "Hello World" & Chr(34) & ")"])</pre>

EditScript

Edit Script in the script definition manager

UI Access	NA		
Parameters	<i>Name</i> <OriginalName> <NewName> <ScriptText>	<i>Type</i> String String	<i>Description</i> Name of the script before editing New name for the script Text of script
Return Value	None		

Python Syntax	EditScript(<OriginalName>,["NAME:<NewName>", < ScriptText >])
Python Example	<pre>oDefinitionManager.EditScript ("myscript", ["NAME:myscript", "ScriptLang:=", _ "python", "ScriptText:=", _ "MsgBox(" & Chr(34) & "Hello Again" & Chr(34) & ")"])</pre>

ExportScript

Export to Library in the script definition manager

UI Access	NA									
Parameters	<table><tr><td><i>Name</i></td><td><i>Type</i></td><td><i>Description</i></td></tr><tr><td><ExportData></td><td>Array</td><td>Array</td></tr><tr><td></td><td></td><td>("NAME:<LibraryName>",<ScriptName>,<ScriptName>,...)</td></tr></table>	<i>Name</i>	<i>Type</i>	<i>Description</i>	<ExportData>	Array	Array			("NAME:<LibraryName>",<ScriptName>,<ScriptName>,...)
<i>Name</i>	<i>Type</i>	<i>Description</i>								
<ExportData>	Array	Array								
		("NAME:<LibraryName>",<ScriptName>,<ScriptName>,...)								
Return Value	None									

Python Syntax	ExportScript(<ExportData>,<Library location>)
Python Example	<pre>oProject.ExportComponent (["NAME:mylib", "myscript"], "PersonalLib")</pre>

GetData [Script Manager]

Gets data that describes the script definition.

UI Access	NA		
Parameters	<i>Name</i> <SimpleName>	<i>Type</i> String	<i>Description</i> Simple name of the script to edit.

Return Value	<DefinitionData> This is an array of strings with basic script data.

Python Syntax	GetData(<SimpleName>)
Python Example	<pre>dataArray = oScriptManager.GetData ("myscript")</pre>

GetNames [Script Manager]

Returns an array of names of scripts (used and unused) in the project.

UI Access	NA		
Parameters	<i>Name</i> None	<i>Type</i>	<i>Description</i>
Return Value	An array of strings		

Python Syntax	GetNames()
Python Example	<pre>scriptnames = oScriptManager.GetNames()</pre>

IsUsed [Script Manager]

Used to determine if a script is used in the design.

UI Access	NA		
Parameters	<i>Name</i> <ComboName>	<i>Type</i> String	<i>Description</i> Composite name of the material.
Return Value	Boolean True if the specified script is used in the design		

Python Syntax	IsUsed(<ComboName>)
Python Example	<pre>isused = oScriptManager.IsUsed("mylib:myscript")</pre>

RemoveUnused [Script Manager]

Removes scripts that are not used in the design.

UI Access	Project->Remove Unused Definitions is similar.		
Parameters	<i>Name</i> None	<i>Type</i>	<i>Description</i>
Return Value	Boolean True if one or more scripts are removed.		

Python Syntax	RemoveUnused()
Python Example	<pre>thereWereExtras = oScriptManager.RemoveUnused()</pre>

RemoveScript

Remove Script in the script definition manager

UI Access	NA		
Parameters	<i>Name</i> <ScriptName> <IsProjectScript> <LibraryName> <LibraryLocation>	<i>Type</i> String Boolean String String	<i>Description</i> Name of the script True if Script is a project script Name of the Library Location of the Library
Return Value	None		

Python Syntax	<code>RemoveScript (<ScriptName>,<IsProjectScript>, <LibraryName>,<LibraryLocation>)</code>
Python Example	<pre>oDefinitionManager.RemoveScript("myscript", true, "Local", "Project")</pre>

ModifyLibraries

Configure Libraries on the Tools menu

UI Access	NA		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<DesignName>	String	Name of the design
	<ConfigLibArray>	Array	Array("NAME:<LibraryType>,<ConfiguredLib>,<ConfiguredLib>,...),...
	<ConfiguredLib>	-	Blank to leave unchanged
	<DefinitionType>	Array	Array("<libraryname >","<libraryname>"),...)
Return Value	None		

Python Syntax	<code>ModifyLibraries (<DesignName>,<ConfigLibArray>)</code>
Python Example	<pre>oDefinitionManager.ModifyLibraries("MyCircuit", _ ["NAME:PersonalLib"], _ ["NAME:UserLib"], _ ["NAME:SystemLib"], _ "Symbols:=", ["Circuit Elements", "Symbols", _ "ParamExtraElements\PE_Symbols", _</pre>

	"Vendor Elements\Nonlinear"]])

Model Manager Script Commands

```
oDefinitionManager = oProject.GetDefinitionManager()
```

```
oModelManager = oDefinitionManager.GetManager("SimModel")
```

Export [Model Manager]

Export model(s) to a library

UI Access	Tools > Edit Configured Libraries > Models > Export to Library		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<LibraryName>	String	Name of the library to which the definition is written
	<ComboName> <LibraryLocation>	String String	Composite name of the model to export. Location of the library in <LibraryName>. Can be either "PersonalLib", or "UserLib"
Return Value	None		

Python Syntax	Export(["NAME:<LibraryName>", <ComboName>, <ComboName>...], <LibraryLocation>)
Python Example	oModelManager.Export(["NAME:mylib", "model1", "model2"])

ExportAsFMU [Model Manager]

Export the model as FMU. If successful, returns 1, otherwise 0

UI Access	NA		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<Model Name>	String	Model Name under Project Definitions

	<FileName> String Save as (FMU) file name with path <FMI version> String "1.0" or "2.0" (if empty it will extract available FMU based on compiled interfaces) <FMI Type> String "Model Exchange" or "Co-Simulation" or "" [Note: If the <FMI Type> is either "" or not specified, the type used will be chosen automatically based on the model type. If the model uses an external solver, then Co-Simulation is used; otherwise, Model Exchange is used. <Modelica Package Location> String Leave it empty or use package file name if model is from a package and if it is different from the original location. <Target Platform> String "Windows64" or "Windows32" or "". If empty or omitted, "Windows64" is the default.
Return Value	Integer 1: success , 0: fail

Python Syntax	ExportAsFMU (<ModelName>, <FileName>, <FMI Version>, <FMI Type>, < Modelica Package Location>)
Python Example	<pre>oDefinitionManager = oProject.GetDefinitionManager() oSimModelManager = oDefinitionManager.GetManager("SimModel") If the model is from a library in the project. oSimModelManager.ExportAsFMU ("LibA:Add", "filePath/fileName.fmu") oSimModelManager.ExportAsFMU("Add", "filePath/fileName.fmu", "1.0", "", "", "Windows32") oSimModelManager.ExportAsFMU("Add", "filePath/fileName.fmu", "2.0", "", "", "Windows64") If the model is not from a library in the project. oSimModelManager.ExportAsFMU("Add", "filePath/fileName.fmu") If the package location changed. oSimModelManager.ExportAsFMU("Add", "filePath/fileName.fmu",</pre>

	<pre>"2.0", "", "filePath/Package.mo") If you want to export for "Co-Simulation" or "Model Exchange" oSimModelManager.ExportAsFMU ("modelName","filePath/fileName.fmu", "2.0" , "Co-Simulation", "", "") oSimModelManager.ExportAsFMU ("modelName","filePath/fileName.fmu", "2.0" , "Model Exchange", "") oSimModelManager.ExportAsFMU ("modelName","filePath/fileName.fmu", "1.0" , "Co-Simulation", "") oSimModelManager.ExportAsFMU ("modelName","filePath/fileName.fmu", "1.0" , "Model Exchange", "")</pre>

ExportAsTwin

Export the selected twin model definition as a **.twin** file.

UI Access	This command is available only through script and is deprecated. For UI access, see ExportTwinModel .		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<ModelName>	String	The name of the model.
	<FileNameWithPath>	String	Output pathname with .twin file name.
	"Other"	String	Product key; only "other" is supported.
	<UseRuntimeLicense>	String	1 - Runtime license. 0 - Deployment license.
	<NumberOfDeployments>	String	Number of twins for deployment.
Return Value	Integer: 1 for success, 0 for failure		

Python Syntax	<pre>oSimModelManager.ExportAsTwin(<ModelName>,<FileNameWithPath>)</pre>

Python Example	<pre>oSimModelManager.ExportAsTwin ("TwinModel", "C:/<someFilePath>/<FileName>.twin", "other", "0", " 100")</pre>

ExportTwinModel

Export the selected twin model definition as a Twin or FMU file.

UI Access	Right-click model in Project Manager Definitions > Models and select Export as Twin .		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<ModelName>	String	The name of the model.
	<FileNameWithPath>	String	Output file pathname with target extension.
	<ExportTarget>	String	"twin", "fmu".
	<optarg1>	String	Optional argument (only for twin).
	<optarg2>	String	Optional argument (only for twin).
	<optarg3>	String	Optional argument (only for twin).
Return Value	Integer: 1 for success, 0 for failure		

Optional args (for twin)	Description	
optarg1	Product key.	Only "other" is supported.
optarg2	License types.	"0" - Unlicensed twin at runtime; requires deployment license to export. "1" - Licensed twin; requires runtime license. Default.
optarg3	Number of deployments.	Number of deployments. (>0) default=1.

Python	<pre>oSimModelManager.ExportTwinModel(<ModelName>, <FileNameWithPath>, <ExportTarget> [, <optarg1> [, <optarg2> [, <optarg3>]]])</pre>

Syntax	
Python Example	<pre>oSimModelManager.ExportAsTwin ("TwinModel", "C:/<someFilePath>/<FileName>.twin", "twin", "other", "0", "50")</pre>

GetData [Model Manager]

Gets data that describes the model definition

UI Access	NA		
Parameters	<i>Name</i> <ComboName>	<i>Type</i> String	<i>Description</i> Composite name of the definition to edit.
Return Value	<DefinitionData> This is an array of strings with basic model data		

Python Syntax	GetData(<ComboName>)
Python Example	<pre>dataArray = oModelManager.GetData ("mylib:mymodel")</pre>

GetLibraryDefNames [Model Manager]

Returns an array of names of the models in the library.

UI Access	NA		
Parameters	<i>Name</i> <LibraryName>	<i>Type</i> String	<i>Description</i> The name of the library
Return Value	An array of strings		

Python Syntax	GetLibraryDefNames (<LibraryName>)

Python Example	<code>libnames = oModelManager.GetLibraryDefNames ("mylib")</code>

GetModelText [Model Manager]

To view text of a model.

UI Access	NA		
Parameters	<i>Name</i> None	<i>Type</i>	<i>Description</i>
Return Value	Array of strings (Model Text)		

Python Syntax	<code>GetModelText(definition name)</code>
Python Example	<code>oSimModelManager.GetModelText ("libraryName:modelName")</code>

GetNames [Model Manager]

Returns an array of names of models (used and unused) in the project.

UI Access	NA		
Parameters	<i>Name</i> None	<i>Type</i>	<i>Description</i>
Return Value	An array of strings		

Python Syntax	<code>GetNames()</code>
Python Example	<code>modelnames = oModelManager.GetNames ()</code>

IsUsed [Model Manager]

Used to determine if a model is used in the design

UI Access	NA		
Parameters	<i>Name</i> <ComboName>	<i>Type</i> String	<i>Description</i> Composite name of the model.
Return Value	Boolean True if the specified model is used in the design		

Python Syntax	IsUsed(<ComboName>)
Python Example	<pre>isused = oModelManager.IsUsed ("mylib:mymodel")</pre>

Remove [Model Manager]

Remove a model from a library.

UI Access	Tools > Edit Configured Libraries > Models > Remove Model(s)		
Parameters	<i>Name</i> <ModelName> <IsLocal> <LibraryName> <LibraryLocation>	<i>Type</i> String Boolean String String	<i>Description</i> Composite name of the model. True, means remove from the project; False, means remove from library on disk. Name of the library. Location of the library in <LibraryName>. Can be any of the following: "Project", "PersonalLib", or "UserLib"
Return Value	None		

Python Syntax	Remove (<ModelName>, <IsLocal>, <LibraryName>, <LibraryLocation>)
Python Example	<pre>oModelManager.Export ([</pre>

	<code>"NAME:mylib", "model1", "model2"])</code>

RemoveUnused [Model Manager]

Removes models that are not used in the design.

UI Access	Project->Remove Unused Definitions is similar.		
Parameters	<i>Name</i> None	<i>Type</i>	<i>Description</i>
Return Value	Boolean True if one or more models are removed.		

Python Syntax	<code>RemoveUnused()</code>
Python Example	<code>thereWereExtras = oModelManager.RemoveUnused()</code>

Package Manager Script Commands

```
oDefinitionManager = oProject.GetDefinitionManager()
```

```
oPackageManager = oDefinitionManager.GetManager("SimPackage")
```

Export [Package Manager]

Export package(s) to a library

UI Access	Tools > Edit Configured Libraries > Packages > Export to Library		
Parameters	<i>Name</i> <LibraryName>	<i>Type</i> String	<i>Description</i> Name of the library to which the definition is written
	<ComboName>	String	Composite name of the package to export.
	<LibraryLocation>	String	Location of the library in <LibraryName>.

	Can be either "PersonalLib", or "UserLib"
Return Value	None

Python Syntax	Export(["NAME:<LibraryName>", <ComboName>, <ComboName>...], <LibraryLocation>)
Python Example	<pre>oPackageManager.Export (["NAME:mylib", "package1", "package2"])</pre>

GetData [Package Manager]

Gets data that describes the package definition

UI Access	NA		
Parameters	<i>Name</i> <ComboName>	<i>Type</i> String	<i>Description</i> Composite name of the definition to edit.
Return Value	<DefinitionData> This is an array of strings with basic package data		

Python Syntax	GetData(<ComboName>)
Python Example	<pre>dataArray = oPackageManager.GetData ("mylib:mypackage")</pre>

GetLibraryDefNames [Package Manager]

Returns an array of names of the packages in the library

UI Access	NA		
Parameters	<i>Name</i> <LibraryName>	<i>Type</i> String	<i>Description</i> The name of the library

Return Value	An array of strings

Python Syntax	GetLibraryDefNames (<LibraryName>)
Python Example	<pre>libnames = oPackageManager.GetLibraryDefNames("mylib")</pre>

GetNames [Package Manager]

Returns an array of names of packages (used and unused) in the project

UI Access	NA		
Parameters	<i>Name</i> None	<i>Type</i>	<i>Description</i>
Return Value	An array of strings		

Python Syntax	GetNames()
Python Example	<pre>pkgnames = oPackageManager.GetNames()</pre>

IsUsed [Package Manager]

Used to determine if a package is used in the design.

UI Access	NA		
Parameters	<i>Name</i> <ComboName>	<i>Type</i> String	<i>Description</i> Composite name of the package.
Return Value	Boolean True if the specified package is used in the design		

Python Syntax	<code>IsUsed(<ComboName>)</code>
Python Example	<code>isused = oPackageManager.IsUsed("mylib:mypackage")</code>

Remove [Package Manager]

Remove a package from a library.

UI Access	Tools > Edit Configured Libraries > Packages > Remove Package(s)		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<PackageName>	String	Composite name of the package.
	<IsLocal>	Boolean	True, means remove from the project; False, means remove from library on disk.
	<LibraryName>	String	Name of the library.
	<LibraryLocation>	String	Location of the library in <LibraryName>. Can be any of the following: "Project", "PersonalLib", or "UserLib"
Return Value	None		

Python Syntax	<code>Remove(<PackageName>, <IsLocal>, <LibraryName>, <LibraryLocation>)</code>
Python Example	<code>oPackageManager.Remove("SimPackage", true, "", "")</code>

RemoveUnused [Package Manager]

Removes packages that are not used in the design.

UI Access	Project->Remove Unused Definitions is similar.		
Parameters	<i>Name</i> None	<i>Type</i>	<i>Description</i>
Return Value	Boolean True if one or more packages are removed.		

Python Syntax	<code>RemoveUnused()</code>
Python Example	<code>thereWereExtras = oPackageManager.RemoveUnused()</code>

Network Data Explorer Manager Script Commands

```
oDefinitionManager = oProject.GetDefinitionManager()
```

```
oNdExplorerManager = oDefinitionManager.GetManager("NdExplorer")
```

ExportFullWaveSpice

Export FullWaveSpice data in a format of your choice

UI Access	In the ndExplorer dialog box: File>Export MacroModel>Broadband (SYZ, FWS...)		
Parameters	<i>Name</i> None	<i>Type</i>	<i>Description</i>
Return Value	None		

Python Syntax	<pre> ExportFullWaveSpice "", // Empty string. 1, //the number 1. "Name", // 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"), </pre>

```

// Optional; if none defined all frequencies are used
Array("NAME:SpiceData",

// Spice export options object
"SpiceType:=", "Simplorer",

// SpiceType can be "PSPice", "HSpice", "Spectre", "SSS", "Simplorer", "Touch-
Stone1.0", "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"
"ColumnFittingType:=", "Column",

// Column FittingType can be "Column", "Entry"
"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,

```


	<pre> // 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) </pre>
Python Example	<pre> oDefinitionManager = oProject.GetDefinitionManager() oNdExplorerManager = oDefinitionManager.GetManager ("NdExplorer") oNdExplorerManager.ExportFullWaveSpice ("",1, "C:\Program Files\ANSYS Inc\v252\AnsysEM\Examples\Circuit\ Signal Integrity\Transient_Decoup_EMC.aedt","", _ ["NAME:Frequencies"],["NAME:Spicedata", _ "SpiceType:=", "Simplorer", "CreateNPortModel:=", true, _ "ExportDirectory:=", "e:\tmp", "ExportSpiceFileName:=", "TestNDE.sml", "FullWaveSpiceFileName:=", "TestNDE.sml"]]) </pre>

ExportNetworkData

Export the solution in a format of your choice (Citifile, Spreadsheet, Matlab, Touchstone, Neutral)

UI Access	In the ndExplorer dialog: File > Export SYZ Data		
Parameters	<i>Name</i> See Syntax	<i>Type</i>	<i>Description</i>
Return Value	None		

**Python
Syntax**

```
ExportNetworkData(  
    "",  
    # Empty string.  
    1,  
    #The number 1.  
    "Name",  
    # This is the full path of the file from which the solution is loaded.  
    "ExportFile",  
    # full path of file to export to  
    "variation",  
    # Pick a particular variation. Leave blank if no variation  
    ["NAME:Frequencies"],  
    # optional, if none defined all frequencies are used  
    ["NAME:Options"],  
    # Export options object  
    "DataTypes:=", ["S"],  
    # DataTypes can be "S", "Y", "Z", "G", and "Z0", for S , Y, Z matrix, Gamma and  
    # Z0 (zero)  
    "DisplayFormat:=", "MA",  
    # DisplayFormat "MA", "RI", "DB"  
    "FileType:=", "",  
    # Export File Type:  
    2 - Spreadsheet(*.tab)  
    3 - Touchstone(*.sNp)  
    4 - Citifile(*.cit)  
    6 - Neutral format(*.nmf)  
    7 - Matlab format(*.m)  
    "Renormalize:=", false,
```

	<pre># Renormalize true/false "RefImpedance:=", 50, # Reference Impedance "Precision:=", 8, # Number of digits Precision "CreateNPortModel:=", true # Create a model based on the exported file true/false])</pre>
Python Example	<pre>oDefinitionManager = oProject.GetDefinitionManager() oNdExplorerManager.ExportNetworkData ("",1,_ "C:\Program Files\ANSYS Inc\v252\AnsysEM\Examples\Circuit\ Signal Integrity\Transient_Decoup_EMC.aedt", "", _ "E:\tmp\testSnP.s11p", "", ["NAME:Frequencies"], ["NAME:Options", _ "DataTypes:=",["S"], "FileType:=", 3])</pre>

ExportNMFData

Export the solution in NMF format

UI Access	In the ndExplorer dialog: File > Export SYZ Data		
Parameters	<i>Name</i> None	<i>Type</i>	<i>Description</i>
Return Value	None		

Python Syntax	<pre>ExportNMFData("", # Empty string. 1,</pre>

```
#The number 1.
"Name",

# This is the full path of the file from which the solution is loaded
"ExportFile",

# full path of file to export to
["NAME:Frequencies"],

#optional, if none defined all frequencies are used
["NAME:NMFOptions",

#Export NMF options object
"DataTypes:=", ["S"],

#DataTypes can be "S", "Y", "Z", "G", and "Z0", for S, Y, Z matrix, Gamma
and Z0 (zero)
"DisplayFormat:=", "MA",

#DisplayFormat "MA", "RI", "DB"
"FileType:=", "",

# Export File Type
2 - Spreadsheet(*.tab)
3 - Touchstone(*.sNp)
4 - Citifile(*.cit)
6 - Neutral format(*.nmf)
7 - Matlab format(*.m)
"Renormalize:=", false,

#Renormalize true/false
"RefImpedance:=", 50,

# Reference Impedance
"Precision:=", 8,

# Number of digits Precision
"Variables:=", ["FF", "cap", "Rs"]

# Array of variables
```

	<pre> "Variations:=", ["", "", ""] # Array of variations to export solutions for ["NAME:ConstantVars"] #Array of variables that are constant, can be empty ["NAME: DependentVars"] #Array of variables that are dependent, can be empty "MatrixSize:=", 2, #Matrix size, optional (used in nmf file header) "CreateNPortModel:=", true #Create a model based on the exported file true/false]) </pre>
Python Example	<pre> oTool = oDesktop.GetTool("NdExplorer") oTool.ExportNetworkData("", True, "C:/.../100OHM.S2P", "\$SYSLIB/Test.s2p", "", ["NAME:Frequencies", 5000000000, 10000000000, 100000000000, 250000000000, 500000000000, 750000000000, 1000000000000], ["NAME:TOptions", "DataTypes:=", ["S"], </pre>

	<pre>"DisplayFormat:=" , "DB", "FileType:=" , 3, "Renormalize:=" , True, "RefImpedance:=" , 50, "Precision:=" , 6, "UseMultipleCores:=" , False, "NumberOfCores:=" , 1, "Comments:=" , True, "Noise:=" , False])</pre>

10 - CompInstance Script Commands

This command gets the component instance object from the instance name of a component.

Python Syntax	<code>inst = oEditor.GetCompInstanceFromInstanceName("R1")</code>
Python Example	<pre>Dim inst Set inst = oEditor.GetCompInstanceFromInstanceName("R1")</pre>

Many commands related to **CompInstances** are available through the **Design** and **PropHost** objects.

Python Syntax	<code>oDesign2 = inst.GetParentDesign()</code>
Python Example	<pre>Dim oDesign2 Set oDesign2 = inst.GetParentDesign</pre>

Callback Scripting Using CompInstance Object

You can obtain an interface to a **CompInstance** in a schematic by calling `oEditor.GetCompInstanceFromInstanceName(name)`. For more information see [Schematic Scripting](#). This interface is also available as a **CompInstance** object in [CompInstance event callbacks](#), such as placing a component in a 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"

Related Topics

[CompInstance Functions](#)

CompInstance Functions

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

- [GetComponentName](#)
- [GetEditor](#)
- [GetFullComponentName](#)
- [GetInstanceID](#)
- [GetParentDesign](#)
- [GetPropHost](#)
- [GetPropServerName](#)

GetComponentName

Returns the name of the component corresponding to this `CompInstance`

UI Access	NA		
Parameters	<i>Name</i> None	<i>Type</i>	<i>Description</i>
Return Value	String		

Python Syntax	<code>GetComponentName()</code>
Python Example	<pre>name = CompInstance.GetComponentName()</pre>

GetEditor

Returns an interface to the editor requested — if the `compInstance` is contained within that type of editor. Common editor names are "SchematicEditor" and "Layout"

UI Access	NA		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>

	<code><editorName></code> String Name of the editor (Ex: "SchematicEditor" or "Layout")
Return Value	Interface to Editor Requested

Python Syntax	<code>GetEditor(<editorName>)</code>
Python Example	<code>oLayout2 = PropHost.GetEditor("Layout")</code>

GetFullComponentName

Returns the composite name of the component corresponding to this CompInstance, e.g., "Simplorer Elements\Basic Elements\Circuit\Passive Elements:R"

UI Access	NA		
Parameters	<i>Name</i> None	<i>Type</i>	<i>Description</i>
Return Value	String		

Python Syntax	<code>GetFullComponentName()</code>
Python Example	<code>name = CompInstance.GetFullComponentName()</code>

GetInstanceID

Returns the instanceID of the CompInstance

UI Access	NA		
Parameters	<i>Name</i> None	<i>Type</i>	<i>Description</i>
Return Value	String		

Python Syntax	GetInstanceID()
Python Example	<code>id = CompInstance.GetInstanceID()</code>

GetParentDesign

Returns an interface to the compInstance's parent design

UI Access	NA		
Parameters	<i>Name</i> None	<i>Type</i>	<i>Description</i>
Return Value	Returns the Interface to Design		

Python Syntax	GetParentDesign()
Python Example	<code>oDesign2 = CompInstance.GetParentDesign()</code>

GetPropHost

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

UI Access	NA		
Parameters	<i>Name</i> None	<i>Type</i>	<i>Description</i>
Return Value	Returns interface to PropHost		

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

Example	

*Returns the interface to the properties of the compInstance.

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

GetPropServerName

Returns the PropServerName of the Component corresponding to this CompInstance

UI Access	NA		
Parameters	<i>Name</i> None	<i>Type</i>	<i>Description</i>
Return Value	String		

Python Syntax	GetPropServerName()
Python Example	<pre>name = CompInstance.GetPropServerName()</pre>

11 - Property Script Commands

```
oEditor = oDesign.SetActiveEditor("SchematicEditor")
```

```
oEditor.<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 Twin Builder 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 Twin Builder solution.

`<Calc>`

Type: `<string>`

An expression that is composed of a basic solution quantity and an output variable.

`<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.

`<PropServer>` and `<PropTab>` Names

Project:

Project Variables:

`<PropServer>` "ProjectVariables"

`<PropTab>` "ProjectVariableTab"

AnsoftSimplorer Design:

Local Variables:

`<PropServer>` "LocalVariables"

`<PropTab>` "LocalVariableTab"

AnsoftSimplorer Modules:

<PropServer>

Format is: <ModuleName>:<ItemName>, where <ItemName> is the boundary name, solution setup name, etc., depending on which module is being edited.

Example: <PropServer> for the boundary "PerfE1" is

"BoundarySetup:PerfE1"

<PropTab>

Analysis module: "SimplorerTab"

Optimetrics module: "OptimetricsTab"

Solutions module: *Does not support properties.*

Operation on an object:

<PropServer>

Format is <ObjName>:<OperationName>:<int>

Concatenation of object name, operation name, and the index of the operation.

Example: "Box2:CreateBox:2" refers to the second

"CreateBox" command in Box2's history.

<PropTab> Geometry3DCmdTab"

ChangeProperty

Changes to properties are scripted using the **ChangeProperty** command. This command can be executed by **oEditor** to change editor properties, by **oDesign** to change design level properties, and by **oProject** to change project level properties. The command can be used to create, edit, and/or remove properties. In Twin Builder, only Variable and Separator properties can be deleted.

Note:

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	NA	
Parameters	Name <PropTabArray>	Type Description 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>,

	<pre> "Callback:=", <string>, "NewRowPosition:=", <int>, "ReadOnly:=", <bool>, "Hidden:=", <bool>, <PropTypeSpecificArgs> OR (for PropDisplays only) Array ("Name:<PropName>", <PropDisplayData>) </pre>
<pre> <PropDisplayData> <PropDisplayAttributes> <PropDisplayNewAttributes> </pre>	for adding, changing, deleting PropDisplays for changing PropDisplays only 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. <pre> "NewName:=", <string>, "NewFormat:=", <PropDisplayType>, "NewStyle:=", <string> </pre>
<pre> <PropDisplayAttributes> </pre>	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>, "Style:=", <string> Identifies the format of PropDisplay. "Name " "Value " "NameAndValue " "EvaluatedValue " "NameAndEvaluatedValue " Identifies where PropDisplay is located with respect to object "Left " "Top " "Right " "Bottom " "Custom " Identifies the type of property when a new property is added. In Twin Builder, only separator properties and variable properties can be added. "SeparatorProp " "VariableProp " "TextProp "
<PropDisplayType>	
<PropDisplayLocation>	
<PropType>	

	"NumberProp" "ValueProp" "CheckboxProp" "MenuProp" "PointProp" "VPointProp" "ButtonProp"
<NewName>	Specify the new name of a property if the property's name is being edited. In Twin Builder, the name can only be changed for separators and variables.
<Description>	Specify a description of the property. In Twin Builder, the description can only be changed for separators and variables.
<NewRowPosition>	Used to reorder rows in the Property dialog box. In Twin Builder, this only applies to the Project>Project Variables panel and the Twin Builder>Design Properties 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 Twin Builder, 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 Twin Builder, 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>

	<pre> Material Button: "Material:=", <string> Color Button: "R:=", <int>, "G:=", <int>, "B:=", <int> Transparency Button: "Value:=", <double> Array("NAME:Optimization", "Included:=", <bool>, "Min:=", <value>, "Max:=", <value>) Array("NAME:Tuning", "Included:=", <bool>, "Step:=", <value>, "Min:=", <value>, "Max:=", <value>) Array("NAME:Sensitivity", "Included:=", <bool>, "Min:=", <value>, "Max:=", <value>, "IDisp:=", <value>) Array("NAME:Statistical", "Included:=", <bool>, "Dist:=", <Distribution>, "StdD:=", <value>, "Min:=", <value>, "Max:=", <value>, "Tol:=", <string>) Value should be "Gaussian" or "Uniform" Standard deviation. Low cut-off for the distribution Tolerance for uniform distributions. Format is "<int>%". </pre>
<OptimizationFlagsArray>	
<Tuning flagsArray>	
<SensitivityFlagsArray>	
<StatisticsFlagsArray>	
<Distribution>	
<StdD>	
<Min>	
<Tol>	

	Example: "20%".
Return Value	None

Python Syntax	ChangeProperty (["Name:AllTabs", <PropTabArray>, <PropTabArray>, ...])
Python Example	<pre> oDesign = oProject.SetActiveDesign("TwinBuilder1") oDesign.ChangeProperty(["NAME:AllTabs", ["NAME:LocalVariableTab", ["NAME:PropServers", "Instance:0;TwinBuilder1"], ["NAME:NewProps", ["NAME:NewVar1", "PropType:=", "VariableProp", "UserDef:=", True, "Value:=", "100ohm"]]]]) oDesign.ChangeProperty(</pre>

```
[
  "NAME:AllTabs",
  [
    "NAME:LocalVariableTab",
    [
      "NAME:PropServers",
      "Instance:0;TwinBuilder1"
    ],
    [
      "NAME:ChangedProps",
      [
        "NAME:NewVar1",
        "Value:=", "200ohm"
      ]
    ]
  ]
])
oProject.ChangeProperty(
  [
    "NAME:AllTabs",
    [
      "NAME:ProjectVariableTab",
      [
        "NAME:PropServers",
        "ProjectVariables"
      ],
      [
        "NAME:NewProps",
```



```
[
  "NAME:$Var1",
  "PropType:=", "VariableProp",
  "UserDef:=", True,
  "Value:=", "90deg"
]
],
[
  "NAME:ChangedProps",
  [
    "NAME:$Var1",
    "Value:=", "90deg"
  ]
]
]
])
oProject.ChangeProperty(
[
  "NAME:AllTabs",
  [
    "NAME:ProjectVariableTab",
    [
```

```
"NAME:PropServers",  
  
"ProjectVariables"  
  
],  
  
[  
"NAME:ChangedProps",  
  
[  
  
"NAME:$Var1",  
  
"Value:=", "180deg"  
]  
  
]  
  
]  
])
```

Callback Scripting Using PropHost Object

Definitions

<propName> = text string

<value> = double

<valueText> = text string

<fileName> = full path file name

<choices> = string containing menu choices separated by commas

<initialChoice> = string containing initial choice for menu; must be one of the <choices>

<scriptName> = string containing name of script stored in project

<bool> is 1 for true or 0 for false

<propType> Property Description

0	Text
1	Menu
2	Checkbox
3	Variable
4	VPoint
5	V3DPoint
6	Number
7	Point
8	Value
9	Button
10	Separator
11	Netlist
12	FileName

<tabType>	Description	Objects with this tab type
1	kPassedParameterTab	Instances of designs, components, geometric objects
2	kDefinitionParameterTab	Definitions of designs, components
3	kLocalVariableTab	Definitions of designs, components
4	kProjectVariableTab	Projects
5	kConstantsTab	Projects
6	kBaseElementTab	Geometric objects
7	kComponentTab	Designs, components
8	kPropertyTab	
9	kHarmonicaTab	
10	kSymphonyTab	
11	kEnsembleTab	
12	kHfssTab	
13	kOptimetricsTab	
14	kAltraSimTab	
15	kReport3DTab	report3d

<tabType>	Description	Objects with this tab type
16	kFieldsPostProcessorTab	fieldspostprocessor
17	kMeshSetupTab	manual meshing setup
18	kRadFieldSetupTab	radiation field geometry setup
19	kGeometry3DAttributeTab	Geometry3D
20	kGeometry3DCmdTab	Geometry3D
21	kGeometry3DPolylineTab	Geometry3D
22	kGeometry3DCSTab	Geometry3D
23	kGeometry3DPlaneTab	Geometry3D
24	kGeometry3DPointTab	Geometry3D
25	kGeometry3DListTab	Geometry3D

PropHost Functions

Following are commands that can be used to manipulate properties from a script.

AddMenuProp

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

UI Access	NA		
Parameters	<i>Name</i> <tabType>	<i>Type</i> Integer	<i>Description</i> One of the following, where tab titles are shown in parentheses: 1 for PassedParameterTab ("Parameter Values") 2 for DefinitionParameterTab (Parameter Defaults") 3 for LocalVariableTab ("Variables" or "Local Variables") 4 for ProjectVariableTab ("Project variables") 5 for ConstantsTab ("Constants") 6 for BaseElementTab ("Symbol" or "Footprint")

			7 for ComponentTab ("General")
			28 for CustomTab ("Intrinsic Variables")
			39500 for Quantities ("Quantities")
			39506 for Signals ("Signals")
	<propName>	String	Name of the new menu property
	<choices>	String	A comma-separated list of menu choice strings.
	<initialChoice>	String	One of the strings in the list of choices.
Return Value	None		

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

AddMenuProp2

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

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

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

AddProp

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

UI Access	NA		
Parameters	<i>Name</i> <tabType>	<i>Type</i> Integer	<i>Description</i> One of the following, where tab titles are shown in parentheses: 1 for PassedParameterTab ("Parameter Values") 2 for DefinitionParameterTab (Parameter Defaults) 3 for LocalVariableTab ("Variables" or "Local Variables") 4 for ProjectVariableTab ("Project variables")

		5 for ConstantsTab ("Constants") 6 for BaseElementTab ("Symbol" or "Footprint") 7 for ComponentTab ("General") 28 for CustomTab ("Intrinsic Variables") 39500 for Quantities ("Quantities") 39506 for Signals ("Signals")
<propType>	Integer	One of the following to add new definition properties (e.g., to component, design or project): 0 for TextProp (any string) 2 for CheckboxProp ("true" or "false" values) 3 for VariableProp (useful for Project and Local Variables, expression) 4 for VPointProp (x and y values, which can be expressions) 5 for V3DPointProp (x, y and z values, which can be expressions) 6 for NumberProp (simple number, no expression) 7 for ColorProp (takes an RGB value) 8 for PointProp (x and y values, no expressions) 11 for SeparatorProp (used for property list organization, the value is the separator "title") 13 for FileNameProp (special button property for filenames) 39506 for GenericProp (for generics in Twin Builder, SML and VHDL)

	39500 for QuantityProp (for Twin Builder quantities, SML and VHDL) 39504 for SignalProp (for VHDL signals in Twin Builder) 39502 for VariablePropNU (like VariableProp, but it provides access to setting NetlistUnits in the Component Definition dialog box for Ansys Electronics Desktop. For Twin Builder, use GenericProp in DefinitionParametersTab, QuantityProp in iQuantities, and signalProp in Signals) <propName> String Name of the new menu property. <valueText> String The property value		
Return Value	None		

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

AddProp2

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

UI Access	NA		
Parameters	<i>Name</i> <tabType>	<i>Type</i> Type	<i>Description</i> 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") CustomTab ("Intrinsic Variables") Quantities ("Quantities") Signals ("Signals")
<propType>	String	One of the following to add new definition properties (e.g., to component, design or project): TextProp (any string) CheckboxProp ("true" or "false" values) VariableProp (useful for Project and Local Variables, expression) VPointProp (x and y values, which can be expressions) V3DPointProp (x, y and z values, which can be expressions) NumberProp (simple number, no expression) ColorProp (takes an RGB value) PointProp (x and y values, no expressions) SeparatorProp (used for property list organization, the value is the separator "title") FileNameProp (special button property for filenames) GenericProp (for generics in Twin

	Builder, SML and VHDL) QuantityProp (for Twin Builder quantities, SML and VHDL) SignalProp (for VHDL signals in Twin Builder) VariablePropNU (like VariableProp, but it provides access to setting NetlistUnits in the Component Definition dialog box of the Ansys Electronics Desktop application. For Twin Builder, use GenericProp in DefinitionParametersTab, QuantityProp in Quantities, and SignalProp in Signals) <propName> String Name of the new menu property. <valueText> String The property value		
Return Value	None		

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

ExecuteScript

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

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

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

GetApplication

Returns the Application object

UI Access	NA		
Parameters	<i>Name</i> None	<i>Type</i>	<i>Description</i>
Return Value	Object The Application		

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

GetCallback

Finds named property and returns name of Callback script.

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

Python Syntax	GetCallback(<propName>)

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

GetChangedProperty

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

UI Access	NA		
Parameters	<i>Name</i> None	<i>Type</i>	<i>Description</i>
Return Value	String Returns "C" if the script was a Callback associated with the property named "C" and the script was called in response to the property "C" changing value.		

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

GetDescription

Finds named property and returns description string

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

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

Example	

GetEditor

To obtain a specific editor object for the current PropHost, if applicable

UI Access	NA		
Parameters	<i>Name</i> <editorName:=>	<i>Type</i> String	<i>Description</i> One of "SchematicEditor" or "Layout" for circuit editing
Return Value	String		

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

GetEvaluatedText

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

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

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

GetFileName

Finds the named buttonProp and if it is a FileName buttonProp, it returns the file's full path name; otherwise it returns an empty string .

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

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

GetHidden

Finds named property and returns its Hidden flag.

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

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

GetMenuChoices

Returns the choices in the named menu property.

UI Access	NA		
Parameters	<i>Name</i> <propname>	<i>Type</i> String	<i>Description</i> The name of the property.
Return Value	Array of strings; the choices in the menu.		

Python Syntax	GetMenuChoices(<propname>)		
Python Example	<pre>choices = prophost.GetMenuChoices("SimulatorModel")</pre>		

GetPropServers

Returns array of objects that have properties showing on tabTypeName.

UI Access	NA		
Parameters	<i>Name</i> None	<i>Type</i>	<i>Description</i>
Return Value	Array of propserver names.		

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

GetPropTabType

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

UI Access	NA		

Parameters	<i>Name</i> <propName>	<i>Type</i> String	<i>Description</i> Name of the property
Return Value	Returns string		

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

GetReadOnly

Finds named property and returns its ReadOnly flag.

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

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

GetScriptName

Returns the name of the currently running script.

UI Access	NA		
Parameters	<i>Name</i> None	<i>Type</i>	<i>Description</i>
Return Value	String; the script name.		

Python Syntax	GetScriptName()
Python Example	<code>script = prophost.GetScriptName()</code>

GetTabTypeName

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

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

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

GetText

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

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

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

GetValue

Obtains the interpolated channel value at the base time (Base)

UI Access	NA		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<ChannelName>	String	The name of an existing input channel.
	<Base>	Double	The value of the base channel at which to interpolate.
	<OutputName>	Variant	Optional name of an output channel to create with values vs. the base channel. Any existing channel with the same name is overwritten. If not supplied, only the final value is returned from the function call.
Return Value	Double Returns the interpolated channel value.		

IsValueConstant

Determines whether the specified property is a constant or not.

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

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

PropertyExists

Finds named property and returns its property type.

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

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

RemoveProp

Removes the named property from whichever tab it is found.

UI Access	NA		
Parameters	<i>Name</i> <propName>	<i>Type</i> String	<i>Description</i> Name of the property
Return Value	None		

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

SetCallback

Finds named property and sets its Callback script.

UI Access	NA		

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

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

SetDescription

Finds named property and sets its description text.

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

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

SetHidden

Finds named property and sets its Hidden flag.

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

	<code><hidden></code> Boolean True or 1, false or 0.
Return Value	None

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

SetReadOnly

Finds named property and sets its ReadOnly flag.

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

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

SetText

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

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

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

SetValue

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

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

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

Additional Property Scripting Commands

Following are other commands that can be used to manipulate properties from a script.

GetProperties

Gets a list of all properties belonging to a specific PropServer and PropTab. This function is available with the Project, Design or Editor objects, including definition editors.

UI Access	NA		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<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") CustomTab ("Intrinsic Variables") Quantities ("Quantities") Signals ("Signals")
	<propServer> String	An object identifier, generally returned from another script method, such as Comp-Inst@R;2;3
Return Value	Array of strings, the names of the appropriate properties.	

Python Syntax	GetProperties(<propTab>, <propServer>)
Python Example	<pre> selectionArray = oEditor.GetSelections() for k in selectionArray: props = oEditor.GetProperties("PassedParameterTab", k) ... </pre>

GetPropertyValue

Gets the value of a single property belonging to a specific serverName and PropTab. This function is available with the Project, Design or Editor objects, including definition editors.

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

UI Access	NA		
Parameters	<i>Name</i> <propTab>	<i>Type</i> String	<i>Description</i> 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") CustomTab ("Intrinsic Variables") Quantities ("Quantities") Signals ("Signals")
	<propServer>	String	An object identifier, generally returned from another script method, such as Comp-Inst@R;2;3
	<propName>	String	Name of the property.
Return Value	String, the value of the property.		

Python Syntax	GetPropertyValue (<propTab>, <serverName>, <propName>)
Python Example	<pre>selectionArray = oEditor.GetSelections() for k in selectionArray: val = oEditor.GetPropertyValue("PassedParameterTab", k, "R")</pre>

	...

Python Syntax	GetPropertyValue ([<array>])
Python Example	<pre>oModule = oDesign.GetModule("ReportSetup") oModule.GetPropertyValue("Trace", "S Parameter Plot 1:dB(S(1,1))", "Y Axis")</pre>

Example: (Query frequency and excitation)

```
freq = oDesign.GetPropertyValue("General", "AnalysisSetup:Setup1", "Adaptive Freq")
```

```
sigType = oDesign.GetPropertyValue("Q3D", "BoundarySetup:via_1", "Type")
```

```
sourceType = oDesign.GetPropertyValue("Q3D", "BoundarySetup:Source1", "Type")
```

Example: (Query Trace values in a specified report)

```
oModule = oDesign.GetModule("ReportSetup")
```

```
oModule.GetPropertyValue("Trace", "S Parameter Plot 1:dB(S(1,1))", "Y
Axis")
```

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	NA						
Parameters	<table><tr><th><i>Name</i></th><th><i>Type</i></th><th><i>Description</i></th></tr><tr><td>None</td><td></td><td></td></tr></table>	<i>Name</i>	<i>Type</i>	<i>Description</i>	None		
<i>Name</i>	<i>Type</i>	<i>Description</i>					
None							
Return Value	Variant array of strings – the names of the variables.						

Python Syntax	GetVariables ()
Python Example	<pre>project_var_array = oProject.GetVariables() local_var_array = oDesign.GetVariables()</pre>

GetVariableValue

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

UI Access	NA		
Parameters	<i>Name</i> <VarName>	<i>Type</i> String	<i>Description</i> Name of the variable to access
Return Value	String A string representing 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.

UI Access	NA		
Parameters	<i>Name</i> <propTab>	<i>Type</i> String	<i>Description</i> 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") CustomTab ("Intrinsic Variables") Quantities ("Quantities") Signals ("Signals")
	<propServer>	String	An object identifier, generally returned from another script method, such as Comp-Inst@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	<pre>selectionArray=oEditor.GetSelections() for k in selectionArray: ...#filter for resistors oEditor.SetPropertyValue("PassedParameterTab", k, "R", "2200") ...</pre>

SetVariableValue

Sets the value of a variable. To set the value of a Project variable, execute this command using `oProject`. To set the value of a local variable, use `oDesign`.

UI Access	NA		
Parameters	<i>Name</i> <VarValue>	<i>Type</i> Value	<i>Description</i> New value for the variable.
Return Value	None		

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

12 - Dataset Script Commands

Python Example	<pre>oProject = oDesktop.SetActiveProject("Project1") oProject.CommandName (<args>)</pre>

AddDataset

Adds a dataset

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

Python Syntax	AddDataset <DatasetDataArray>
Python Example	<pre>oProject.AddDataset (["NAME:\$ds1", ["NAME:Coordinates", [</pre>

	<pre>"NAME:Coordinate", "X:=", 2, "Y:=", 4], ["NAME:Coordinate", "X:=", 6, "Y:=", 8]]])</pre>

DeleteDataset

Deletes the specified dataset

UI Access	Remove command in the Datasets dialog box.		
Parameters	<i>Name</i> <DatasetName>	<i>Type</i> String	<i>Description</i> Name of the dataset found in the project
Return Value	None		

Python Syntax	DeleteDataset (<DatasetName>)
Python Example	<code>oProject.DeleteDataset ("ds1")</code>

EditDataset

Modifies a dataset. When a dataset is modified, its name as well as its data can be changed

UI Access	Edit command in the Datasets dialog box		
Parameters	<i>Name</i> <OriginalName>	<i>Type</i> String	<i>Description</i> Name of the dataset before editing.
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>

ImportDataset

Imports a dataset.

UI Access	Import command in the Datasets dialog box.		
Parameters	<i>Name</i> <datasetFileFullPath>	<i>Type</i> String	<i>Description</i> The full path to the file containing the dataset values.
Return Value	None		

Python Syntax	ImportDataset (<datasetFileFullPath>)
Python Example	<pre>oProject.ImportDataset ("e:\tmp\dsdata.txt")</pre>

ExportDataset [Dataset]

Exports a dataset.

UI Access	Symbol>Export File		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<type> <filename>	String String	Dataset name. Path, name, and extension for the file to be saved.
Return Value	None		

Python Syntax	ExportFile(<type>, <filename>)
Python Example	<pre>oProject = oDesktop.SetActiveProject("DatasetProject") oProject.ExportDataset("\$CH1", "C:/SimplorerStuff/CSV_Files/exportTest.tab")</pre>

13 - Design Object Script Commands

Design object commands should be executed by the oDesign object:

oDesign.CommandName (<args>)

where CommandName is used to access any of the Twin Builder design methods.

AddMessage

To add a message for the design.

UI Access	NA		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<severity>	String	One of "Error", "Warning" or "Info". Anything other than the first two is treated as "Info"
	<msg>	String	The message for the message window
	<gotoID>	String	The instance name of a component instance in the design
Return Value	None		

Python Syntax	AddMessage (<severity>, <msg>, <gotoID>)
Python Example	oDesign.AddMessage("Warning", "Test Message", "R1")

AddPortImpedance

Adds an impedance to the named port, removing any prior impedance

UI Access	NA		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<port name>	String	Name of the port
	<impedance component defname>	String	Simple name of the component, such as "R"
	<impedance component libname>	String	The library in which the component is found

Return Value	Boolean TRUE if the impedance is added successfully, FALSE otherwise (e.g., if the component is inappropriate or not found)

Python Syntax	AddPortImpedance (<port name>, <impedance component defname>, <impedance component libname>)
Python Example	<pre>ret = oDesign.AddPortImpedance "P1", "R", "Simplorer Elements\Basic Elements\Circuit\Passive Elements"</pre>

AddPortSource

Adds a source to the named port, removing any prior source.

UI Access	NA		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<port name>	String	Name of the port
	<source component defname>	String	Simple name of the component, such as "R".
	<source component libname>	String	The library in which the component is found.
Return Value	Boolean		
	TRUE if the source is added successfully, FALSE otherwise (e.g., if the component is inappropriate or not found).		

Python Syntax	AddPortSource (<port name>, <source component defname>, <source component libname>)
Python Example	<pre>ret = oDesign.AddPortSource "P1", "E", "Simplorer Elements\Basic Elements\Circuit\Sources"</pre>

AddSdbOutput

Add an SDB output

UI Access	NA		
Parameters	<i>Name</i> <outputName>	<i>Type</i> String	<i>Description</i> The name of the output, as seen in the output dialog
Return Value	None		

Python Syntax	AddSdbOutput (<outputName>)		
Python Example	<pre>oProject.AddSdbOutput ("R1.V")</pre>		

Analyze

Simulates using the named Twin Builder solution setup. The UI waits until simulation is finished before continuing with the script.

UI Access	NA		
Parameters	<i>Name</i> <setup>	<i>Type</i> String	<i>Description</i> Setup name
Return Value	Integer 0: successful completion -1: simulation error		

Python Syntax	Analyze (<setup>)		
Python Example	<pre>oDesign.Analyze ("TR1")</pre>		

AnalyzeAll [Design]

Simulates all Twin Builder 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	Right-click on design in Project Manager and select Analyze All from context menu		
Parameters	<i>Name</i> None	<i>Type</i>	<i>Description</i>
Return Value	None		

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

AnalyzeInBackground

Simulates using the named Twin Builder solution setup. The script continues running while the simulation progresses in the background. It is an alias of AnalyzeUI command

It helps running different design simulation in parallel.

UI Access	NA		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<setupName>	String	The name of the analysis to be simulated
	<id>	Integer	An ID for the analysis - should be different for each simulation that may otherwise overlap.
Return Value	Integer		
	0: successful completion		
	-1: simulation error		

Python Syntax	AnalyzeInBackground (<setupName>,<id>)

Python Example	<code>oDesign.AnalyzeInBackground("TR", 3)</code>

AnalyzeUI

Simulates using the named Twin Builder solution setup. The script continues running while the simulation progresses in the background.

UI Access	NA		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<setupName>	String	The name of the analysis to be simulated
	<id>	Integer	An ID for the analysis - should be different for each simulation that may otherwise overlap.
Return Value	Integer 0: successful completion -1: simulation error		

Python Syntax	<code>AnalyzeUI (<setupName>,<id>)</code>
Python Example	<code>oDesign.AnalyzeUI ("TR", 3)</code>

ChangeGPort

Changes a global port name

UI Access	NA		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<oldname>	String	Current name for this port
	<newname>	String	Desired name for this port
Return Value	Boolean success or failure. Failure is returned if there is no port named oldname, or		

	if the new name is invalid.

Python Syntax	ChangeGPort (<oldname>,<newname>)
Python Example	oDesign.ChangeGPort ("G_0", "Ggg")

ChangelPort

To make changes to an interface port, including name, domain, etc. An array of ports can be changed by using bus syntax in the name fields and specifying semicolon-separated values for each of the other fields in the same number.

UI Access	NA		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<oldname>	String	Current Name for this port
	<newname>	String	Desired name for this port
	<domain>	String	one of "Conservative", "Parametric", "Quantity"
	<type>	String	The domain for the port, or a semicolon-separated set of domain names for a bus port. For Conservative domain, one of "Electrical", "Magnetic", "Fluidic", "Translational", "Translational_Velocity", "Rotational", "Rotational_Velocity", "Radiant", "Thermal", "Compressible_Fluidic", or a custom nature; for Quantity and Parametric domains, only "Real" is accepted. The nature, if conservative, or data type if nonconservative, or a semicolon-separated set.
	<dir>	String	one of "In", "Out" or "InOut"
	<unconnected action>	String	The direction, applicable only for nonconservative ports, or a semicolon-separated set. one of "Flag as error", "Unique net", "Grounded", "No net".

	<value> String The action to be taken when the port is unconnected, applicable only for conservative ports; or a semicolon-separated set. The port's value, applicable to non-conservative output ports; or a semicolon-separated set.
Return Value	None

Python Syntax	<code>ChangePort (<oldname>, <newname>, <domain>, <type>, <dir>, <unconnected action>, <value>)</code>
Python Example	<pre>ChangeIPort ("Port1", "Port1", "Conservative", "Fluidic", "", "Flag as error", "")</pre>

ChangePortImpedanceProperty

Changes a property of a port impedance.

UI Access	Right click on an impedance in the project window and choose Properties.		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<port name>	String	Name of the port
	<prop name>	String	Name of the Impedance property
	<prop value>	String	New value for the impedance property.
Return Value	Boolean		
	TRUE if the property is found and changed, FALSE otherwise.		

Python Syntax	<code>ChangePortImpedanceProperty (<port name>, <prop name>, <prop value>)</code>
Python Example	<code>oDesign.ChangePortImpedanceProperty ("P1", "R", "10000")</code>

ChangePortSourceProperty

Changes a property of a port source

UI Access	Right click on a source in the project window and choose Properties.		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<port name>	String	Name of the port
	<prop name>	String	Name of the source property
	<prop value>	String	New value for the source property
Return Value	Boolean		
	TRUE if the property is found and changed, FALSE otherwise.		

Python Syntax	ChangePortSourceProperty (<port name>, <prop name>, <prop value>)
Python Example	oDesign.ChangePortSourceProperty ("P1", "AMPL", "125")

ChangeProperty [Design Level]

See [ChangeProperty](#) in [Property Script Commands](#) for details.

CompileAsFMU

Compiles a Twin Builder design from a Subsheet as a Tool Coupling model. You can export this Tool Coupling model as an FMU and import it into any external tool that supports FMI co-simulation version 2.0. For the simulation, the external tool is the primary solver, while the Twin Builder solver in stand-alone mode is the secondary solver. Ansys Electronics Desktop installation is required to run the simulation on the target machine.

UI Access	N/A		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<modeleame>	String	Model name under project model defin-

	<code><comp settings></code> Array <code><setting></code> Pair <code><name></code> String <code><value></code> String	itions. Array of compiler settings. Single named compiler setting: "<code><name>:=</code>" , "<code><value></code>". Supported compilation parameters: "IncludeDep", "ConfigFile". UseRuntimeLic = "false" – Creates a co-simulation FMU that does not require a license at runtime. "true" – Creates a co-simulation FMU that requires a license at runtime IncludeDep = "0" – Automatic search for dependent model DLLs and model data files to be copied into the FMU file. "1" – Use the config file to specify additional model and data dependencies to be copied into the FMU file. ConfigFile = "<code><filename></code>" See <i>Creating and Exporting Tool Coupling FMU Models</i> in the Twin Builder help for syntax details of the config file. If left empty, Toolcoupling.cfg from your config folder is used. AddLinux = "false" – No Linux binaries are included in the FMU. "true" – Linux binaries and model libraries are included in the FMU. LinuxBinDir = <code><dirname></code> - Location of the Linux solver binaries. Those binaries are available in a installation separate package. LinuxUserLibDir = <code><dirname></code> - Location of the Linux binaries for models from UserLib. LinuxPersLibDir = <code><dirname></code> - Location of the Linux binaries for models from PersonalLib.
Return Value	None.	

Python Syntax	CompileAsFMU (<modelname >, [<setting> [, <setting>]...])
Python Example	<pre>oDesign = oProject.SetActiveDesign ("TwinBuilder1::SubSheet1") oDesign.CompileAsFMU ("FMUModel", ["UseRuntimeLic:=" , "false", "IncludeDep:=" , "1", "ConfigFile:=" , "", "AddLinux:=" , "false", "LinuxBinDr:=" , "", "LinuxUserLibDr:=" , "", "LinuxPersLibDr:=" , ""])</pre>

CompileAsTwin

Compiles the selected object as a Twin model.

UI Access	Right-click subcircuit instance and select Compile as Twin Model		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<CompName>	String	The name of the Twin component.
	<Option1: solver method>	String	"1" for ADAMS , "2" for BDF
	<Option2: Step size>	String	Step size in seconds.
	<Option3: Relative tolerance>	String	Relative tolerance
	<Option4: Absolute tolerance>	String	Absolute tolerance
Return Value	String; The created component name		

Python Syntax	<code>CompileAsTwin(<CompName>, [<Options1>, <Options2>, <Option3>, <Option4>])</code>
Python Example	<pre>oDesign = oProject.SetActiveDesign ("TwinBuilder1::SubSheet1") oDesign.CompileAsTwin("TwinModel", ["1", "0.006", "1e-4", "1e-12"])</pre>

DeleteDesignInstance

Delete the design instance

UI Access	NA		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<InstanceName>	String	This is the name seen in the project window. In the following example, "U1" is the InstanceName for a subdesign in "TwinBuilder1"
Return Value	None		

Python Syntax	<code>DeleteDesignInstance (<InstanceName>)</code>
Python Example	<code>oDesign.DeleteDesignInstance ("TwinBuilder1:U1")</code>

DeleteOutputVariable

Deletes an existing output variable. The variable can only be deleted if it is not in use by any traces.

UI Access	Twin Builder>Results>Output Variables, dialog Delete Button		
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>

DeletePort

Delete a port

UI Access	NA		
Parameters	<i>Name</i> <PortName>	<i>Type</i> String	<i>Description</i> Name of the port to be deleted
Return Value	None		

Python Syntax	DeletePort (<PortName>)
Python Example	<pre>oDesign.DeletePort ("Port1")</pre>

DeletePortImpedance

Deletes an impedance from the named port that references it

UI Access	Right click on an impedance in the project window and choose Delete		
Parameters	<i>Name</i> <portname>	<i>Type</i> String	<i>Description</i> Name of the port
Return Value	Boolean TRUE if the impedance is found and deleted, FALSE otherwise		

Python Syntax	DeletePortImpedance (<port name>)
Python Example	<code>oDesign.DeletePortImpedance ("P1")</code>

DeletePortSource

Deletes a source from the named port that references it.

UI Access	Right click on a source in the project window and choose Delete.		
Parameters	<i>Name</i> <port name>	<i>Type</i> String	<i>Description</i> Name of the port
Return Value	Boolean TRUE if the source is found and deleted, FALSE otherwise		

Python Syntax	DeletePortSource (<port name>)
Python Example	<code>oDesign.DeletePortSource ("P1")</code>

DeleteSolutionVariation

Deletes the selected solution variation(s)

UI Access	Right-click on Results in Project Manager and select Browse Solutions from context menu. On the Browse tab, select the variation(s) you wish to delete and click the Delete button		
Parameters	<i>Name</i> None	<i>Type</i>	<i>Description</i>
Return Value	None		

Python Syntax	DeleteSolutionVariation ()
Python Example	oDesign.DeleteSolutionVariation ([["Param1", "TR", "TR"]])

EditDesignSettings

UI Access	Twin Builder>Design Settings...		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	Simulation Options	Boolean	Automatically sort block diagram before simulation
		Boolean	Enable symbol animation, output property display update
		Integer	Number of significant digits to display
	Simulation State (KRN)	Boolean	Save at end of simulation
		Boolean	Ask for filename
		Boolean	Load at start of simulation
		Boolean	Ask for filename
	Initial Values	Boolean	Save at end of simulation
		Boolean	Ask for filename
	Statistical Package	Boolean	Enable
		Boolean	Specify initial seed value
		Integer	Seed value
Return Value	None		

Use: To set the design settings.

Command: Twin Builder>Design Settings

Syntax: EditDesignSettings <OverrideArray>

Return Value: None

Parameters: <OverrideArray>

Python	EditDesignSettings ([<SelectionsArray>])

Syntax	
Python Example	<pre> oDesign.EditDesignSettings(["NAME:DesignSettings", ["NAME:SimSettings", "AutoSaveState:=" , True], ["NAME:StatisticalPackage", "Enable:=" , True, "UserSpecifySeed:=" , True, "Seed:=" , 1]]) </pre>

ExportNetlist

Exports a netlist for the named analysis setup to the specified file

UI Access	NA		
Parameters	<i>Name</i> <setupName>	<i>Type</i> String	<i>Description</i> The name of an analysis setup in the design, such as "TR"
	<filename>	String	The name of the netlist file. The path to the file must exist – the command will not create directories.
Return Value	None		

Python Syntax	ExportNetlist (<setupName>, <filename>)

Python Example	<code>oDesign.ExportNetlist ("TR", "c:\tmp\netlist.txt")</code>

GetConfigurableData

Note:

This command is for internal Ansys use only.

Python Syntax	<code>GetConfigurableData()</code>
Python Example	<code>oDesign.GetConfigurableData()</code>

GetData

Note:

This command is for internal Ansys use only.

Python Syntax	<code>GetData()</code>
Python Example	<code>oDesign.GetData()</code>

Get Design Type

Returns the specified design.

UI Access	NA
Parameters	NA
Return Value	String indicating the design type of the active design ("Circuit Design", "Cir-

	cuit Netlist", "HFSS 3D Layout Design", "HFSS", "HFSS-IE", "Icepak", "Maxwell 2D", "Maxwell 3D", "Q2D Extractor", "Q3D Extractor", "RMxpert", "Twin Builder")

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

GetLinkComponents

To obtain a list of coupling components of the specified coupling type in the project.

UI Access	NA		
Parameters	<i>Name</i> <type>	<i>Type</i> String	<i>Description</i> The type of coupling. The following types can be updated from the original source, including terminals and properties: "FluentModel" for Fluent components, "SimulinkData" for Simulink components, "MxTranTranData" for Maxwell transient cosimulation components, "MaxwellData" for Maxwell dynamic inductance, dynamic capacitance and state space components, "RMxpertData" for RMxpert dynamic components, "SimpQ3DDData" for Q3D equivalent circuit and statespace components, "IcepakData" for Icepak components, "MechanicalData" for Mechanical components,

	"HFSSData" for HFSS components, "SIWaveData" for SIwave components, "MBDDData" for RBD components The following types perform file existence check only - it is not possible to reobtain terminal and parameter information through script: "MathCadData" for mathCCAD components, "MaxwellECEDData" for Maxwell equivalent circuit components
Return Value	An array of strings, the IDs of the components of the specified coupling type.

Python Syntax	GetLinkComponents (<type>)
Python Example	<pre>comps = oDesign.GetLinkComponents("FluentModel")</pre>

GetLinkInstances

To obtain a list of component instances of the specified coupling type in the design.

UI Access	NA		
Parameters	<i>Name</i> <type>	<i>Type</i> String	<i>Description</i> The type of coupling. The following types can be updated from the original source, including terminals and properties: "FluentModel" for Fluent components, "SimulinkData" for Simulink components, "MxTranTranData" for Maxwell transient cosimulation components, "MaxwellData" for Maxwell dynamic

	inductance, dynamic capacitance and state space components, "RMxpvtData" for RMxpvt dynamic components, "SimpQ3DDData" for Q3D equivalent circuit and statespace components, "IcepakData" for Icepak components, "MechanicalData" for Mechanical components, "HFSSData" for HFSS components, "SIWaveData" for SIwave components, "MBDDData" for RBD components The following types perform file existence check only - it is not possible to reobtain terminal and parameter information through script: "MathCadData" for mathCCAD components, "MaxwellECEDData" for Maxwell equivalent circuit components
Return Value	An array of strings, the IDs of the component instances of the specified coupling type.

Python Syntax	GetLinkComponents (<type>)
Python Example	<pre>insts = oDesign.GetLinkinstances("MxTranTranData")</pre>

GetModule

Returns the IDispatch for the specified module.

UI Access	NA

Parameters	<i>Name</i> <modulename> <i>Type</i> String <i>Description</i> Currently the only module name supported is "SimSetup"
Return Value	Module IDispatch

Python Syntax	GetModule (<modulename>)
Python Example	<pre>oModule = oDesign.GetModule("SimSetup")</pre>

GetName

Returns the name of the Design.

UI Access	NA
Parameters	<i>Name</i> None <i>Type</i> <i>Description</i>
Return Value	String, the hierarchical name of the design, with the following format: <id0>;<name0>[/<id1>;<name1>]*, where the id and name for a design at any level of hierarchy can be seen as the values of the ID and Representation properties on the General tab when the design is selected in the project window. The first set of id and name identifies the top level design, the next set identifies the design at the next level of hierarchy, and the remaining sets identify further levels of hierarchy. Note that two instances of the same design will have different ID values, so that the hierarchical name is specific to the actual design instance responding to the script method.

Python Syntax	GetName()
Python Example	<pre>dname = oDesign.GetName()</pre>

GetNoteText

Return the text of the note added to a design in the project tree.

Stops a task from running. Call this method only once for each task and subtask. In a script, there should be the same number of BeginTask calls as **EndTask** calls.

UI Access	NA		
Parameters	<i>Name</i> None	<i>Type</i>	<i>Description</i>
Return Value	Returns text of the design note		

Python Syntax	GetNoteText()
Python Example	<code>progMon.EndTask(True)</code>

GetPostProcessingVariables

Return the list of post processing variables.

UI Access	Right click on the created document in the Project Manager under Results>Documents and click Delete .						
Parameters	<table><tr><td><i>Name</i></td><td><i>Type</i></td><td><i>Description</i></td></tr><tr><td>None</td><td></td><td></td></tr></table>	<i>Name</i>	<i>Type</i>	<i>Description</i>	None		
<i>Name</i>	<i>Type</i>	<i>Description</i>					
None							
Return Value	Returns list of variables						

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

GetProperties [Design]

Gets a list of all the properties belonging to a specific PropServer (the owner of the properties, or the list containing them) and PropTab (basically, the type or category of property). This command can be executed by the oProject, oDesign, or oEditor objects. (This command can be used for design properties when it's invoked from oDesign.) The PropServer and PropTab argument choices are listed below.

UI Access	NA		
Parameters	<i>Name</i> <PropTab>	<i>Type</i> String	<i>Description</i> One of five choices, "LocalVariableTab", "DefinitionParameterTab", "ComponentTab", "Quantities", "Signals"
	<PropServer>	String	One of five choices corresponding to the appropriate proptabs, "LocalVariables", "DefinitionParameters", "Propertyts", "Quantities", "Signals"
Return Value	Variant array of strings – the names of the properties belonging to the prop server.		

Python Syntax	GetProperties (<PropTab>, <PropServer>)
Python Example	<pre>all_props = oDesign.GetProperties("BaseElementTab", "rect_1")</pre>

GetPropHost

Obtains the PropHost object for managing design properties.

UI Access	NA		
Parameters	<i>Name</i> None	<i>Type</i>	<i>Description</i>
Return Value	The PropHost object for the design.		

Python Syntax	GetPropHost()
Python Example	<code>oPropHost = oDesign.GetPropHost()</code>

GetResultsDirectory

Obtains the PropHost object for managing design properties.

UI Access	NA		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<setupName:=>	String	The name of the setup applicable to the results.
	<variation:=>	String	The variation applicable to the results (which can be seen with the Show Profile command).
	<finalOrTemp:=>	Boolean	Whether to return the final, saved, results directory (true), or the temporary, unsaved, directory (false).
Return Value	The results directory, either temp or final.		

Python Syntax	GetResultsDirectory (<setupname>, <variation>, <finalOrTemp>)
Python Example	<code>dir = oDesign.GetResultsDirectory("TR", "", false)</code>

GetSdbOutputs

Obtains the properties specified for SDB output in the Output dialog

UI Access	NA		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>

	None
Return Value	Array of strings, the output property names.

Python Syntax	GetSdbOutputs()
Python Example	<code>outputs = oDesign. GetSdbOutputs()</code>

GetSimulationOptionFilePath

Retrieves simulation settings for the current design. Select options as described below to extract the current value.

UI Access	NA		
Parameters	<i>Name</i> <option>	<i>Type</i> Integer	<i>Description</i> Choices are: 0 – Save simulation state (KRN). 1 – Load simulation state (KRN). 2 – Save Initial values.
Return Value	value	string	Return value is the user-specified file path of the selected option. It will return a file name if the selected option was supposed to use the user-specified file; otherwise, it will return an empty file path. For auto or ask file name option value, it will return empty. If empty, it failed to retrieve file name for the option.

Python Syntax	<code>fileName = GetSimulationOptionFilePath(<option>)</code>

Python Example	<pre> oProject = oDesktop.GetActiveProject() oDesign = oProject.GetActiveDesign() //return value is filename. // empty return value indicates option could be set to use auto(default) or ask before simulation. // it will return value only if option was set to use user specified filepath saveKRNFileName = oDesign.GetSimulationOptionFilePath(0) loadKRNFileName = oDesign.GetSimulationOptionFilePath(1) </pre>

Note:
 To set KRN/initial values file settings from a user-specified location, use the **SetSimulationOptionFilePath** command.

GetSimulationSettings

Retrieves simulation settings for the current design.

UI Access	NA		
Parameters	<i>Name</i> <option>	<i>Type</i> Integer	<i>Description</i> Choices are: 0 – Save simulation state (KRN) status. 1 – Load simulation state (KRN) status. 2 – Save Initial values status. 3 – Automatically sort block diagram before simulation status. 4 – Status of symbol animation and output property display update. 5 – Significant number of digits to display in output property display. 6 – Status of statistical package use. 7 – Is seed specified by user for statistical package.

		8 – Get seed value.
Return Value	value	integer
		For options 0, 1, or 2, the return value will be: 0 – No 1 – Auto 2 – Ask 3 – specify file by user For options 3, 4, 6, or 7, the return value will be: 0 – False 1 – True For option 5, the return value will be the number of significant digits. For option 8, the return value will be the seed value.

Python Syntax	<code>returnVal=GetSimulationSettings(<option>)</code>
Python Example	<pre> oProject = oDesktop.GetActiveProject() oDesign = oProject.GetActiveDesign() // return value meanings: 0 (No), 1 (Auto), 2 (Ask), 3 (specify file by user) saveKRNOptVal = oDesign.GetSimulationSettings(0) //save KRN Option status loadKRNOptVal = oDesign.GetSimulationSettings(1) //load KRN Option status //return value will be 0, 1 sortBlock = oDesign.GetSimulationSettings(3) // auto- matically sort blocks animateSym = oDesign.GetSimulationSettings(4) //Symbol Anim- ation status </pre>

Note:

To *get* KRN/initial values file settings from a user-specified location, use the **GetSimulationOptionFilePath** command.

To *set* KRN/initial values file settings from a user-specified location, use the **SetSimulationOptionFilePath** command.

GetSubDesigns

Returns a container object with objects for all subdesigns of the current design (one level).

UI Access	NA		
Parameters	<i>Name</i> None	<i>Type</i>	<i>Description</i>
Return Value	Container object of the design objects.		

Python Syntax	GetSubDesigns()
Python Example	<pre> subs = oDesign.GetSubDesigns() for sub in subs: AddInfoMessage(str(sub.GetName())) </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	NA		
Parameters	<i>Name</i> None	<i>Type</i>	<i>Description</i>
Return Value	Variant array of strings – the names of the variables.		

Python Syntax	GetVariables ()
Python Example	<pre>project_var_array = oProject.GetVariables() local_var_array = oDesign.GetVariables()</pre>

GetVariableValue

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

UI Access	NA		
Parameters	<i>Name</i> <VarName>	<i>Type</i> String	<i>Description</i> Name of the variable to access
Return Value	String A string representing the value of the variable.		

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

ImportSolution

Imports an SDB, mdx, or other supported file format for plotting results.

UI Access	NA		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>

	<filename> String The full path to the file. Supported formats are: .sdb Twin Builder solution data .mdx, .mda Twin Builder characteristic format .mdb Microsoft Access .xls Microsoft Excel (version 3 through version 2003). .txt text file .csv Comma-separated value .out Maxwell SPICE (read-only - reads data inside the KW_DATA section) .cfg Comtrade (IEEE Std C37-111-1999) .dat TEK Oscilloscope .tab tab delimited data files
Return Value	Array of setup names in the imported solution.

Python Syntax	<code>ImportSolution (<filename>)</code>
Python Example	<code>setups = oDesign.ImportSolution("c:\SDBs\Design1.sdb")</code>

InsertDesign

Adds a subcircuit to the current design. If the named design already exists, a dialog is brought up allowing the choice of adding an instance of the existing design, or an instance of a new design which is a copy of the existing one. Note that adding a subdesign to a design that is itself used more than once will add instances of the new subdesign in the same measure.

UI Access	Twin Builder>SubCircuit>Add SubCircuit		
Parameters	<i>Name</i> <designName>	<i>Type</i> String	<i>Description</i> The name of the new design.
Return Value	The added design object.		

Python Syntax	<code>InsertDesign("", <designName>, "", "")</code>
Python Example	<code>oDesign2 =oDesign.InsertDesign("", "AmpChannel", "", "")</code>

LoadConfiguration

Loads model choices from a file and applies them to the current design and its descendents. Also can add or change design and project variables.

UI Access	Simplorer Circuit/Load Configuration		
Parameters	<i>Name</i> <Filename>	<i>Type</i> String	<i>Description</i> Pathname of model configuration file.
Return Value	Boolean TRUE unless there is a file problem.		

Python Syntax	<code>LoadConfiguration(<filename>)</code>
Python Example	<pre>ret = oDesign.LoadConfiguration ("E:\Projects\subvars.mcfg") where the contents of a typical .mcfg file might be similar to the following: R5.SimulatorModel=R R6.SimulatorModel=R R7.SimulatorModel=R R8.SimulatorModel=R aa=4 bb=if(aa < 5, 1, 0) \$cc=6</pre>

ManageOutputs

To add and remove specified SDB outputs

UI Access	Twin Builder>Output Dialog		
Parameters	<i>Name</i> <Quantity>	<i>Type</i> String	<i>Description</i> The name of the output, as seen in the output dialog.
Return Value	Boolean TRUE for success, FALSE when one or more quantities cannot be added or removed. If a quantity is specified for adding but is already present, it is ignored and will not cause a FALSE return.		

Python Syntax	ManageOutputs(["NAME:Added", <quantity>, ...], ["NAME:Removed", <quantity>, ...])		
Python Example	<pre>oDesign.ManageOutputs(["NAME:Added", "net_2.V", "R1.R"], ["NAME:Removed", "E1.V", "R1.V"])</pre>		

PushExcitations

Generate an excitation file that can be imported back into HFSS and SIwave

UI Access	Right-click on the SIwave component>Push Excitations or HFSS component>Push Excitations		
Parameters	<i>Name</i> <SolutionName> <ComponentID> <ExcludedPorts>	<i>Type</i> String String Array of	<i>Description</i> Solution name Component ID Excluded ports

<SelectedPorts>	Strings Array of Strings	Selected ports
<ExcludedFrequencies>	Array of doubles	Excluded frequencies
<SelectedFrequencies>	Array of doubles	Selected frequencies
<StartTimeUnits>	String	Start time units
<StartTime>	double	Start time
<StopTimeUnits>	String	Stop time units
<StopTime>	double	Stop time
<MaxFreq>	double	Maximum frequency (in Hz)
<UseDefaultFFTCheck>	Boolean	Use default values for: Noise threshold - 40 (dB) Integration type - Fourier Integration Windowing type - Rectangular Kaiser parameter - 0 Adjust coherent gain - false
<EnableNoiseThreshold>	Boolean	0 for disable; 1 for enable
<NoiseThreshold>	double	Noise threshold (dB)
<IntegrationType>	integer	Integration type 0 for Fourier 1 for FFT 2 for Fourier Transform
<WindowType>	integer	Windowing type 0 for Rectangular 1 for Bartlett 2 for Blackman 3 for Hamming 4 for Hanning 5 for Kaiser 6 for Welch

	7 for Weber 8 for Lanzcos
<KaiserParameter>	double Kaiser parameter
<CoherentGainCheck>	Boolean Coherent gain check
Return Value	None

Python Syntax	<pre>SimplePushExcitations(<Solution name>, <ComponentID>, [<ExcludedPorts>], [<SelectedPorts>], [<ExcludedFrequencies>], [<SelectedFrequencies>], <StartTimeUnits>, <StartTime>, <StopTimeUnits>, <StopTime>, <MaxFreq>, <UseDefaultFFTCheck>, <EnableNoiseThreshold>, <NoiseThreshold>, <IntegrationType>, <WindowType>, <KaiserParameter>, <CoherentGainCheck>)</pre>
Python Example	<pre>oProject = oDesktop.SetActiveProject("Simplorer_HFSS_Example") oDesign = oProject.SetActiveDesign("TwinBuilder1") oDesign.Analyze("TR") oDesign.SimplePushExcitations("TR", "coax_bend_fws", ["WavePort2_1"], ["WavePort1_1"], [0, 10000000, 20000000], _ [30000000, 40000000, 50000000, 60000000], "ns", 0, "ns", 30, 5000000000, 0, 1, 50, 1, 1, 0, 1) or oProject = oDesktop.SetActiveProject("Simplorer_SIwave_ Example") oDesign = oProject.SetActiveDesign("Simplorer_TR") oDesign.Analyze("TR") oDesign.SimplePushExcitations("TR", "PCB_Simplorer0_1", [], ["CK_in", "CK_out", "EK_in", "EK_out", "Gate_in", "Gate_out"], [], [0, 2000000, 4000000, 6000000, 8000000, 10000000, 12000000, 14000000], "ns", 0, "ms", 0.02, 5000000000, 1, 1, 40, 0, 0, 0, 0)</pre>

Redo

Reapplies the last design-level command.

UI Access	Edit>Redo		
Parameters	<i>Name</i> None	<i>Type</i>	<i>Description</i>
Return Value	None		

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

RemoveAllSdbOutputs

Clear all SDB outputs.

UI Access	NA		
Parameters	<i>Name</i> None	<i>Type</i>	<i>Description</i>
Return Value	None		

Python Syntax	RemoveAllSdbOutputs()
Python Example	<code>oDesign.RemoveAllSdbOutputs()</code>

RemoveSdbOutput

Remove an SDB output

UI Access	NA		
Parameters	<i>Name</i> <outputName>	<i>Type</i> String	<i>Description</i> The name of the output, as seen in the output dialog.
Return Value	None		

Python Syntax	RemoveSdbOutput <outputName>		
Python Example	oDesign.RemoveSdbOutput "U1.R1.V"		

RenamePort

To change the name of an interface port.

UI Access	NA		
Parameters	<i>Name</i> <oldName> <newName>	<i>Type</i> String String	<i>Description</i> The current name of the port to be renamed The new name for the port
Return Value	None		

Python Syntax	RenamePort (<oldName>, <newName>)		
Python Example	oDesign.RenamePort ("Port1", "Vout")		

RenameDesignInstance

Renames a design.

UI Access	Right-click a design instance in the project tree, and then click Rename on the shortcut menu.

Parameters	<i>Name</i> <OldName>	<i>Type</i> String	<i>Description</i> 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 ("TwinBuilder1", "LowPowerAmplifier")</code>

ResetSymbolAnimation

To reset the symbol levels to default.

UI Access	Twin Builder>More Actions> Reset Symbol Animation		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	None		
Return Value	None		

Python Syntax	<code>ResetSymbolAnimation()</code>
Python Example	<code>count = oDesign.ResetSymbolAnimation()</code>

SaveConfiguration

Saves current model choices for this design and its descendents.

UI Access	Twin Builder>More Actions>Save Configuration

Parameters	<i>Name</i> <Filename>	<i>Type</i> String	<i>Description</i> Pathname of model configuration file.
Return Value	Boolean TRUE unless there is a file problem.		

Python Syntax	SaveConfiguration(<filename>)		
Python Example	<pre>ret = oDesign.SaveConfiguration ("E:\Projects\subvars.mcfg")</pre>		

SetActiveEditor

Sets the active editor to be the schematic editor.

UI Access	NA		
Parameters	<i>Name</i> <editorName>	<i>Type</i> String	<i>Description</i> "SchematicEditor"
Return Value	None		

Python Syntax	SetActiveEditor (<editorName>)		
Python Example	<pre>oEditor = oDesign.SetActiveEditor("SchematicEditor")</pre>		

SetAssignsDuringSimulation

Make changes to various design property values during simulation.

UI Access	NA		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>

	Note: The single Array argument has three optional Array subelements. If no changes are being made in one of them, it can be removed. For each of the subarrays, only name/value pairs that are changed need to be present. All other values will remain unchanged. <table><tr><td><setName:=></td><td>String</td><td>One of the simulation setup keywords, such as Tend, Fend, etc.</td></tr><tr><td><setVal:=></td><td>Double</td><td>A value appropriate to the setup variable.</td></tr><tr><td><optName:=></td><td>String</td><td>An option keyword, particular to the type of simulation (TR, AC or DC), or general options.</td></tr><tr><td><optVal:=></td><td>Double</td><td>A value appropriate to the option variable.</td></tr><tr><td><mvalName:=></td><td>String</td><td>A model property name, such as "R1.R"</td></tr><tr><td><mvalVal:=></td><td>Double</td><td>A value appropriate to the model property.</td></tr></table>	<setName:=>	String	One of the simulation setup keywords, such as Tend, Fend, etc.	<setVal:=>	Double	A value appropriate to the setup variable.	<optName:=>	String	An option keyword, particular to the type of simulation (TR, AC or DC), or general options.	<optVal:=>	Double	A value appropriate to the option variable.	<mvalName:=>	String	A model property name, such as "R1.R"	<mvalVal:=>	Double	A value appropriate to the model property.
<setName:=>	String	One of the simulation setup keywords, such as Tend, Fend, etc.																	
<setVal:=>	Double	A value appropriate to the setup variable.																	
<optName:=>	String	An option keyword, particular to the type of simulation (TR, AC or DC), or general options.																	
<optVal:=>	Double	A value appropriate to the option variable.																	
<mvalName:=>	String	A model property name, such as "R1.R"																	
<mvalVal:=>	Double	A value appropriate to the model property.																	
Return Value	None																		

Python Syntax	<code>SetAssignsDuringSimulation ([["NAME:SetVals", <setName:=>, <setVal> [,<setName:=>, <setVal>]], ["NAME:Optionvals", <optName:=>, <optVal> [,<optName:=>, <optVal>]], ["NAME:ModelVals", <mvalName:=>, <mvalVal> [,<mvalName:=>, <mvalVal>]]])</code>
Python Example	<pre>setups=oDesign.SetAssignsDuringSimulation([["Name:SetVals", "Tend",0.05], ["Name:OptionVals", "Iteratmax", 20, "VEmax", 0.003], ["Name:ModelVals", "R1.R", 2200])</pre>

SetDCBiasDisplayOptions

Sets common DC bias display options.

UI Access	Twin Builder>View DC Bias Values>Display Options...

Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<showAcross>	Boolean	Sets display of net across values
	<showAcrossAll>	Boolean	Sets range of net across values to display, TRUE meaning all, FALSE only those selected as output
	<showThrough>	Boolean	Sets display of component instance through values
	<showThroughAll>	Boolean	Sets range of component instance through values to display, TRUE meaning all, FALSE only those selected as output
Return Value	None		

Python Syntax	SetDCBiasDisplayOptions (<showAcross>, <showAcrossAll>, <showThrough>, <showThroughAll>)
Python Example	oDesign.SetDCBiasDisplayOptions(True, True, False, False)

SetEventCallback

To set an event callback script.

UI Access	Project > Event Callbacks		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<type>	String	One of "BeforeSim", "AfterSim" or "After-PlaceComp".
	<scriptname>	String	The name of a script in the project.
Return Value	None		

Python Syntax	SetEventCallback (<type>, <scriptname>)
Python	count = oDesign.SetEventCallback("BeforeSim", "EventCB1")

Example	

SetLinkStatic

Sets the dynamic/static attribute of the specified coupling component instance.

UI Access	NA		
Parameters	Name	Type	Description
	<instID> <val>	String Boolean	The ID of the coupling component instance. TRUE for static, or FALSE for dynamic.
Return Value	Boolean TRUE for success, FALSE for failure.		

Python Syntax	SetLinkStatic(<instID>,<val>)
Python Example	<pre> insts=oDesign.GetLinkInstances("FluentModel") for i in insts: ret = oDesign.SetLinkStatic(i, true) </pre>

SetSimulationOptionFilePath

Sets simulation settings for the current design. Select options as described below to set the current value.

UI Access	NA		
Parameters	Name	Type	Description
	<option>	Integer	Choices are: 0 – Save simulation state (KRN). 1 – Load simulation state (KRN). 2 – Save Initial values.

	value	string	Return value will set selected option to use file specified by this value.
Return Value	ret	integer	0 – Fail 1 – Success

Python Syntax	<code>ret = SetSimulationOptionFilePath(<option>, <FileName>)</code>
Python Example	<pre> oProject = oDesktop.GetActiveProject() oDesign = oProject.GetActiveDesign() //<option> 0: save krn, 1:load krn //return value is 0: fail, 1:success. saveKRNFile = "<UserSpecifiedFileNameWithPath>" // "C:\MyFile.krn" loadKRNFile = "<UserSpecifiedFileNameWithPath>" // "C:\MyFile.krn" ret = oDesign.SetSimulationOptionFilePath(0, saveKRNFile) ret = oDesign.SetSimulationOptionFilePath(1, loadKRNFile) </pre>

Note:

To set KRN/initial values to auto/ask location, use the **SetSimulationSettings** command.

SetSimulationSettings

Retrieves simulation settings for the current design.

UI Access	NA		
Parameters	<i>Name</i> <option>	<i>Type</i> Integer	<i>Description</i> Choices are: 0 – Save simulation state (KRN) status. 1 – Load simulation state (KRN) status. 2 – Save Initial values status. 3 – Automatically sort block diagram before simulation status. 4 – Status of symbol animation and output property display update. 5 – Significant number of digits to display in output property display. 6 – Status of statistical package use. 7 – Is seed specified for statistical package. 8 – Get seed value.
	<value>	Integer	For options 0, 1, or 2, the return value will be: 0 – No 1 – Auto 2 – Ask For options 3, 4, 6, or 7, the return value will be: 0 – False 1 – True

			For option 5, command will set the number of significant digits. For option 8, command will set the seed value.
Return Value	ret	integer	1 – Success 0 – Fail

Python Syntax	returnVal=SetSimulationSettings(<option>, <value>)
Python Example	<pre> oProject = oDesktop.GetActiveProject() oDesign = oProject.GetActiveDesign() //<value> meanings: 0 (No), 1 (Auto), 2 (Ask) //<value> = 1(save KRN at default location (auto)) ret = oDesign.SetSimulationSettings(0,1) //<value> = 2 (Ask for filename before simulation) ret = oDesign.SetSimulationSettings(1,2) // <value>will be 0, 1 ret = oDesign.SetSimulationSettings(3, 0) // disable auto- matically sort blocks ret = oDesign.SetSimulationSettings(4, 1) // enable Symbol Animation </pre>

Note:

To *get* KRN/initial values file settings from a user-specified location, use the **GetSimulationOptionFilePath** command.

To *set* KRN/initial values file settings from a user-specified location, use the **SetSimulationOptionFilePath** command.

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	NA		
Parameters	<i>Name</i> <VarValue>	<i>Type</i> Value	<i>Description</i> New value for the variable.
Return Value	None		

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

StartAnalysis

Simulates all Twin Builder solution setups except Optimetrics setups. The UI waits until simulation is finished before continuing with the script.

UI Access	NA		
Parameters	<i>Name</i> None	<i>Type</i>	<i>Description</i>
Return Value	None, but script error on unsuccessful simulation.		

Python Syntax	StartAnalysis()		
Python Example	<code>oDesign.StartAnalysis()</code>		

ToggleDCBiasDisplay

Toggles the display of DC Bias values.

UI Access	Twin Builder>View DC Bias Values>Show DC Bias		
Parameters	<i>Name</i> None	<i>Type</i>	<i>Description</i>

Return Value	None

Python Syntax	ToggleDCBiasDisplay()
Python Example	<code>oDesign.ToggleDCBiasDisplay()</code>

Undo

Cancels the last design-level command.

UI Access	Edit>Undo		
Parameters	<i>Name</i> None	<i>Type</i>	<i>Description</i>
Return Value	None		

Python Syntax	Undo()
Python Example	<code>oDesign.Undo()</code>

UpdateDCBiasDisplay

Refreshes the display of DC Bias values

UI Access	Twin Builder>View DC Bias Values>Update		
Parameters	<i>Name</i> None	<i>Type</i>	<i>Description</i>
Return Value	None		

Python	UpdateDCBiasDisplay()

Syntax	
Python Example	<code>oDesign.UpdateDCBiasDisplay()</code>

UpdateLink

Updates the specified coupling component and all instances of it (in any design).

UI Access	NA		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<compid>	String	The id of the coupling component. The entire component is updated for most supported coupling types. File existence only is checked for Maxwell equivalent circuit components.
Return Value	Boolean TRUE for success, FALSE for failure		

Python Syntax	<code>UpdateLink (<compid>)</code>
Python Example	<pre>comps = oDesign.GetLinkComponents("FluentModel") for i in comps: ret = oDesign.UpdateLink(i)</pre>

14 - Design Analysis Setup Script Commands

Python Example	<pre>oDesign = oProject.SetActiveDesign("TwinBuilder1") oModule = oDesign.GetModule("SimSetup") oModule.CommandName(<args>)</pre>

AddACAnalysis [Analysis Setup]

Adds an AC analysis setup to the currently active design.

UI Access	Twin Builder>Add Solution Setup>AC		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<Options>	String	The name of the solution options to use for this setup (optional, default is no special options)
	<enabled>	Integer	1 for enabled, 0 for disabled (optional, default is enabled)
	<continue>	Boolean	True for Continue to Solve enabled, false for disabled (optional, default is disabled)
	<setupName>	String	The name for this setup
	<sweepType>	Integer	1 for linear sweep, 2 for decadic – other values are unsupported
	<startFreq>	String	The starting frequency, greater or equal to 0 and less than stopFreq
	<stopFreq>	String	The ending frequency, greater or equal to 0 and more than startFreq
	<step>	String	For a linear sweep, the frequency amount for each step, or for a decadic sweep, the number of points per decade
Return Value	None		

Python Syntax	<pre>AddACAnalysis [("NAME:SimSetup", ["OptionName:=", <options>,"AnalysisEnabled:=", <enabled>," EnableContSolve:=", <continue>,"</pre>

	"Name:=", <setupName>, "LinearFrequencyData:=", Array(<sweepType>, <startFreq>, <stopFreq>, <step>)]
Python Example	<pre>oModule.AddACAnalysis [("NAME:SimSetup", "Name:=", "AC", "LinearFrequencyData:=", [(1, "1Hz", "1kHz", "10Hz"))]</pre>

AddAnalysisOptions [Analysis Setup]

Adds a named set of analysis options to the currently active design.

UI Access	Right-click Analysis in the Project Manager pane and select Add Solution Options .		
Parameters	<i>Name</i> <optionsName> <digCosimInput> <integMethodTR> <ldfErrorTR> <maxIterationsTR> <maxlerrorTR> <maxVerrorTR>	<i>Type</i> String Boolean Integer Double Integer String String	<i>Description</i> The name of this set of analysis options <i>True</i> to pause before the start of simulation to allow third party digital simulator setup, <i>false</i> otherwise, default = <i>false</i> Use 0 for Euler, 1 for Adaptive Trapezoid-Euler, and 3 for Trapezoid, default = 1 Local truncation error for TR simulation, in percent, default = 1 The maximum number of iterations per time step for TR simulation, default = 40 The maximum current error allowed for TR simulation. The default value is 1 <i>m</i> , where <i>m</i> stands for milli (10 ⁻³). The maximum voltage error allowed for TR simulation. The default value is 1 <i>m</i> , where <i>m</i> stands for milli (10 ⁻³).

		3).
<manualRelToITR>	Boolean	Use true to be able to specify reitol, false to use an automatic value, default = false.
<relToITR>	Double	The relative tolerance in percent for TR simulation, default = 0.1
<A/DsynchTypeTR>	Integer	Analog/digital synchronization choice. Use 0 for Hybrid, 1 for Adaptive and 2 for Conservative. Default = 0
<advStepModeTR>	Integer	Specify 1 to use advanced step mode, 0 otherwise. Default = 1
<numEqualStepsTR>	Integer	Number of equal steps, default = 0
<stepAccelDampingTR>	Double	The percentage damping for step acceleration, default = 0
<autoPivotTR>	Boolean	Use true for an automatic pivoting strategy, false for manual. Default = true
<pivotFactor>	Double	Performance factor, between 0 and 1 to specify relative amount of pivoting. Default = 0.1
<samanskiiAmountTR>	Integer	Specify the number of iterations after which to update the Jacobian. Default = 1
<MSlibCmd>	String	ModelSim command to use for lib, default = "vlib work"
<MScompileCmd>	String	The ModelSim command to use for compile, default = "nvcom - bindAtCompile \$(SMLName)_ EDSCoSim_Netlist.vhd"
<MSloadCmd>	String	The ModelSim command to use for load, default = "vsim -t 1fs work.top"
<maxErrorAC>	String	The max error for AC simulation, default = "1n"
<maxIterationsAC>	Integer	The max number of iterations per frequency point for AC simulation, default = 40
<maxIterationsDC>	Integer	The max number of iterations for DC simulation, default = 50
<maxRelaxationDC>	Integer	The max number of relaxations for DC simulation, default = 10
<maxErrorDC>	String	The max error for DC simulation,

	<ambientTemp>	Double	default = "1m" Specify ambient temperature in Celsius, default = 27
	<changeLimitFlag>	Integer	Flag to control damping heuristics (limiting node potential change per iteration), 0 indicates no damping, 1 specifies absolute damping, 2 specifies absolute damping for initial simulation only, 3 specifies relative damping, and 4 specifies relative damping for initial simulation only. Default = 0
	<absLimit>	Double	Absolute limiting value for node potential change, default = 0
	<relLimit>	Double	Relative limiting value for node potential change, default = 0
	<startDataCollection>	Double	Time or frequency at which to start saving data, default = 0
	< stopDataCollection >	Double	Time or frequency at which to stop saving data, default = -1, which indicated the natural end of simulation.
	<dataReductionType>	Integer	Data reduction type, use 0 for no data reduction, 1 to collect data every N steps, 2 to save data at fixed time or frequency intervals, 3 to save data when it has changed by a relative amount from its previous value. Default = 0
	<drNumStepsValue>	Integer	For data reduction type 1, this is the number N. Default = 1
	<drStepwidthValue>	Double	For data reduction type 2, this is the time or frequency interval. Default = 0
	<drPercentChangeValue>	Double	For data reduction type 3, this is the relative change, in percent to govern data output. Default = 0
	<header>	String	SML text, often using "#include <file-name>", to bring external data into the simulation. Default = ""
Return Value	None		

**Python
Syntax**

```

AddAnalysisOptions Array("NAME:SimSetup",
"Name:=", <optionsName>,
"StopForUserInput:=", <digCosimInput>,
Array("NAME:ModifiedOptions",
"ts_intmethod:=", <integMethodTR>,
"ts_ldferror:=", <ldfErrorTR>,
"ts_maxiter:=", <maxIterationsTR>,
"ts_iemaxerror:=", <maxIerrorTR>,
"ts_vemaxerror:=", <maxVerrorTR>,
"ts_reltolflag:=", <manulalRelTolTR>,
"ts_reltol:=", <relTolTR>,
"ts_adsync:=", <A/DsynchTypeTR>,
"ts_advstepmode:=", <advStepModeTR>,
"ts_equalsteps:=", <numEqualStepsTR>,
"ts_stepacceldamping:=", <stepAccelDampingTR>,
"ts_perffactorflag:=", <autoPivotTR>,
"ts_performancefactor:=", <pivotFactor>,
"ts_samanskiifactor:=", <samanskiiAmountTR>,
"ts_ModelSimLib:=", <MSlibCmd>,
"ts_ModelSimCompile:=", <MScompileCmd>,
"ts_ModelSimLoad:=", <MSloadCmd>,
"ac_errormax:=", <maxErrorAC>,
"ac_maxiter:=", <maxIterationsAC>,
"dc_maxiter:=", <maxIterationsDC>,
"dc_maxrelax:=", <maxRelaxationDC>,
"dc_maxerror:=", <maxErrorDC>,
"gen_ambientTemp:=", <ambientTemp>,
"gen_limitnodechange:=", <changeLimitFlag>,
"gen_AbsLimitValue:=", <absLimit>,

```

	<pre>"gen_RelLimitValue:=", <relLimit>, "gen_dataStart:=", <startDataCollection>, "gen_dataEnd:=", < startDataCollection >, "gen_reduce:=", <dataReductionType>, "gen_stepno:=", <drNumStepsValue>, "gen_stepwidth:=", <drStepwidthValue>, "gen_steprelchange:=", <drPercentChangeValue>, "sml_Header:=", <header>)</pre>
Python Example	<pre>oModule.AddAnalysisOptions(["NAME:SimSetup", "DataBlockID:=",7, "SimSetupID:=", -1, "OptionName:=", "", "AnalysisEnabled:=",1, "Name:=", "Options", "StopForUserInput:=",False, ["NAME:ModifiedOptions"]])</pre>

AddDCAnalysis [Analysis Setup]

Adds a DC analysis setup to the currently active design.

UI Access	Twin Builder>Add Solution Setup>DC		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>

	<Options>	String	The name of the solution options to use for this setup (optional, default is no special options)
	<enabled>	Integer	1 for enabled, 0 for disabled (optional, default is enabled)
Return Value	None		

Python Syntax	AddDCAnalysis ["NAME:SimSetup", ["OptionName:=", <options>,"] ["AnalysisEnabled:=", <enabled>,"] "Name:=", <setupName>]		
Python Example	<pre>oModule.AddDCAnalysis [("NAME:SimSetup", "Name:=", "DC")]</pre>		

AddTransient [Analysis Setup]

Adds a transient analysis setup to the currently active design

UI Access	Twin Builder>Add Solution Setup>Transient		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<Options>	String	The name of the solution options to use for this setup (optional, default is no special options)
	<enabled>	Integer	1 for enabled, 0 for disabled (optional, default is enabled)
	<continue>	Boolean	True for Continue to Solve enabled, false for disabled (optional, default is disabled)
	<setupName>	String	The name for this setup
	<Tend>	String	The simulation stop time, greater than 0
	<Hmin>	String	The minimum timestep, greater than 0 and less than Hmax
	<Hmax>	String	The maximum timestep, greater than 0, more than Hmin and less than Tend
	<useInitialValues>	Boolean	True to load an initial values file, false otherwise (the initialValuesFile argument required if true – default is false)
	<initialvaluesFile>	String	The name of the AWS file to load at simulation startup

Return Value	None

Python Syntax	AddTransient (["NAME:SimSetup", ["OptionName:=", <options>,] ["AnalysisEnabled:=", <enabled>,] ["EnableContSolve:=", <continue>,] "Name:=", <setupName>, "TransientData:=", [<Tend>, <Hmin>, <Hmax> [, <useInitialValues>, <initialvaluesFile>]]])
Python Example	<pre>oModule.AddTransient (["NAME:SimSetup", "OptionName:=", "Options1", "Name:=", "TR1", "TransientData:=", ["40ms", "10us", "1ms"]])</pre>

EditACAnalysis [Analysis Setup]

Edits an AC analysis setup of the currently active design

UI Access	Right-click on the desired analysis icon in Project Manager and select Edit from the context menu		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<acName>	String	The name of the solution setup to modify. Note that setupName must be specified, and should be the same as acName if a name change is not wanted.
	<options>	String	The name of the solution options to use for this setup (optional, default is no special options)
	<enabled>	Integer	1 for enabled, 0 for disabled (optional, default is enabled)
	<continue>	Boolean	True for Continue to Solve enabled, false for disabled (optional, default is disabled)
	<setupName>	String	The (new) name for this setup, required
	<sweepType>	Integer	1 for linear sweep, 2 for decadic – other values are unsupported
	<startFreq>	String	The starting frequency, greater or equal to 0 and less than stopFreq
	<stopFreq>	String	The ending frequency, greater or equal to 0 and more than startFreq

	<step>	String	For a linear sweep, the frequency amount for each step, or for a decadic sweep, the number of points per decade
Return Value	None		

Python Syntax	EditACAnalysis(<acName>, ["NAME:SimSetup", ["OptionName:=", <options>],[["AnalysisEnabled:=", <enabled>],[["EnableContSolve:=", <continue>],["Name:=", <setupName>,"LinearFrequencyData:=", [<sweepType>, <startFreq>, <stopFreq>, <step>]]])		
Python Example	oModule.EditACAnalysis ("AC", ["NAME:SimSetup", "Name:=", "AC", "LinearFrequencyData:=", [1, "100Hz", "100kHz", "100Hz"]])		

EditAnalysisOptions [Analysis Setup]

Edits the analysis options of the currently active design

UI Access	Right-click on the desired analysis options icon in Project Manager and select Edit from the context menu.		
Parameters	<i>Name</i> <optionsName>	<i>Type</i> String	<i>Description</i> The name of the set of analysis options to edit
	<digCosimInput>	Boolean	<i>True</i> to pause before the start of simulation to allow third party digital simulator setup, <i>false</i> otherwise, default = <i>false</i> ;
	<integMethodTR>	Integer	Use 0 for Euler, 1 for Adaptive Trapezoid-Euler, and 3 for Trapezoid, default = 1.
	<ldfErrorTR>	Double	Local truncation error for TR simulation, in percent, default = 1
	<maxIterationsTR>	Integer	The maximum number of iterations per time step for TR simulation, default = 40
	<maxlerrorTR>	String	The maximum current error allowed for TR simulation, default = "1m"

<maxVerrorTR>	String	The maximum voltage error allowed for TR simulation, default = "1m"
<manualRelTolTR>	Boolean	Use true to be able to specify reltol, false to use an automatic value, default = false
<relTolTR>	Double	The relative tolerance in percent for TR simulation, default = 0.1
<A/DsynchTypeTR>	Integer	Analog/digital synchronization choice. Use 0 for Hybrid, 1 for Adaptive and 2 for Conservative. Default = 0
<advStepModeTR>	Integer	Specify 1 to use advanced step mode, 0 otherwise. Default = 1.
<numEqualStepsTR>	Integer	Number of equal steps, default = 0
<stepAccelDampingTR>	Double	The percentage damping for step acceleration, default = 0
<autoPivotTR>	Boolean	Use true for an automatic pivoting strategy, false for manual. Default = true
<pivotFactor>	Double	Performance factor, between 0 and 1 to specify relative amount of pivoting. Default = 0.1
<samanskiiAmountTR>	Double	Specify the number of iterations after which to update the Jacobian. Default = 1
<MSlibCmd>	String	The ModelSim command to use for lib, default = "vlib work"
<MScompileCmd>	String	The ModelSim command to use for compile, default = "nvcom -bindAtCompile \$(SMLName)_EDSCoSim_Netlist.vhd"
<MSloadCmd>	String	The ModelSim command to use for load, default = "vsim -t 1fs work.top"
<maxErrorAC>	String	The max error for AC simulation, default = "1n"
<maxIterationsAC>	Integer	The max number of iterations per frequency point for AC simulation, default = 40
<maxIterationsDC>	Integer	The max number of iterations for DC simulation, default = 50
<maxRelaxationDC>	Integer	The max number of relaxations for DC simulation, default = 10

	<maxErrorDC>	String	The max error for DC simulation, default = "1m"
	<ambientTemp>	Double	Specify ambient temperature in Celsius, default = 27
	<changeLimitFlag>	Integer	Flag to control damping heuristics (limiting node potential change per iteration), 0 indicates no damping, 1 specifies absolute damping, 2 specifies absolute damping for initial simulation only, 3 specifies relative damping, and 4 specifies relative damping for initial simulation only. Default = 0
	<absLimit>	Double	Absolute limiting value for node potential change, default = 0
	<relLimit>	Double	Relative limiting value for node potential change, default = 0
	<startDataCollection>	Double	Time or frequency at which to start saving data, default = 0
	<stopDataCollection>	Double	Time or frequency at which to stop saving data, default = -1, which indicated the natural end of simulation
	<dataReductionType>	Integer	Data reduction type, use 0 for no data reduction, 1 to collect data every N steps, 2 to save data at fixed time or frequency intervals, 3 to save data when it has changed by a relative amount from its previous value. Default = 0
	<drNumStepsValue>	Integer	For data reduction type 1, this is the number N. Default = 1
	<drStepwidthValue>	Double	For data reduction type 2, this is the time or frequency interval. Default = 0
	<drPercentChangeValue>	Double	For data reduction type 3, this is the relative change, in percent to govern data output. Default = 0
	<header>	String	SML text, often using "#include <filename>", to bring external data into the simulation. Default = "".
Return Value	None		

**Python
Syntax**

```

AddAnalysisOptions ([ "NAME:SimSetup",
"Name:=", <optionsName>,
"StopForUserInput:=", <digCosimInput>,
[ "NAME:ModifiedOptions",
"ts_intmethod:=", <integMethodTR>,
"ts_ldferror:=", <ldfErrorTR>,
"ts_maxiter:=", <maxIterationsTR>,
"ts_iemaxerror:=", <maxIerrorTR>,
"ts_vemaxerror:=", <maxVerrorTR>,
"ts_reltolflag:=", <manulalRelTolTR>,
"ts_reltol:=", <relTolTR>,
"ts_adsync:=", <A/DsynchTypeTR>,
"ts_advstepmode:=", <advStepModeTR>,
"ts_equalsteps:=", <numEqualStepsTR>,
"ts_stepacceldamping:=", <stepAccelDampingTR>,
"ts_perffactorflag:=", <autoPivotTR>,
"ts_performancefactor:=", <pivotFactor>,
"ts_samanskiifactor:=", <samanskiiAmountTR>,
"ts_ModelSimLib:=", <MSlibCmd>,
"ts_ModelSimCompile:=", <MScompileCmd>,
"ts_ModelSimLoad:=", <MSloadCmd>,
"ac_errormax:=", <maxErrorAC>,
"ac_maxiter:=", <maxIterationsAC>,
"dc_maxiter:=", <maxIterationsDC>,
"dc_maxrelax:=", <maxRelaxationDC>,
"dc_maxerror:=", <maxErrorDC>,
"gen_ambientTemp:=", <ambientTemp>,
"gen_limitnodechange:=", <changeLimitFlag>,
"gen_AbsLimitValue:=", <absLimit>,

```

	<pre> "gen_RelLimitValue:=", <relLimit>, "gen_dataStart:=", <startDataCollection>, "gen_dataEnd:=", < startDataCollection >, "gen_reduce:=", <dataReductionType>, "gen_stepno:=", <drNumStepsValue>, "gen_stepwidth:=", <drStepwidthValue>, "gen_steprelchange:=", <drPercentChangeValue>, "sml_Header:=", <header>]])) </pre>
Python Example	<pre> oModule.EditAnalysisOptions ("Options", ["NAME:SimSetup", "Name:=", "Options", ["NAME:ModifiedOptions","ts_maxiter:=", 50]]) </pre>

EditDCAnalysis [Analysis Setup]

Edits a DC analysis setup of the currently active design

UI Access	Right-click on the desired analysis icon in Project Manager and select Edit from the context menu		
Parameters	<i>Name</i> <dcName>	<i>Type</i> String	<i>Description</i> The name of the solution setup to modify. Note that setupName must be specified, and should be the same as dcName if a name change is not wanted
	<options>	String	The name of the solution options to use for this setup (optional, default is no special options)
	<enabled>	Integer	1 for enabled, 0 for disabled (optional, default is enabled)
Return Value	None		

Python Syntax	<pre> EditDCAnalysis (<dcName>,"NAME:SimSetup", ["OptionName:=", </pre>

	<options>,"["AnalysisEnabled:=", <enabled>,"Name:=", <setupName>"])
Python Example	oModule.EditDCAnalysis ("DC", ["NAME:SimSetup", "AnalysisEnabled:=", 0, "Name:=", "DC"])

EditTransient [Analysis Setup]

Edits a transient analysis setup of the currently active design

UI Access	Right-click on the desired analysis icon in Project Manager and select Edit from the context menu		
Parameters	<i>Name</i> <trName>	<i>Type</i> String	<i>Description</i> The name of the solution setup to modify. Note that setupName must be specified, and should be the same as trName if a name change is not wanted.
	<options>	String	The name of the solution options to use for this setup (optional, default is no special options)
	<enabled>	Integer	1 for enabled, 0 for disabled (optional, default is enabled)
	<continue>	Boolean	True for Continue to Solve enabled, false for disabled (optional, default is disabled)
	<setupName>	String	The name for this setup
	<Tend>	String	The simulation stop time, greater than 0
	<Hmin>	String	The minimum timestep, greater than 0 and less than Hmax
	<Hmax>	String	The maximum timestep, greater than 0, more than Hmin and less than Tend
	<useInitialValues>	Boolean	True to load an initial values file, false otherwise. To change from a previous setting, both this flag and a string (possibly empty) for initialValuesFile are required
	<initialvaluesFile>	String	The name of the AWS file to load at simulation startup
Return Value	None		

Python Syntax	<code>EditTransient (<trName>, ["NAME:SimSetup", ["OptionName:=", <options>.,] ["AnalysisEnabled:=", <enabled>.,] ["EnableContSolve:=", <continue>.,] "Name:=", <setupName>, "TransientData:=", [<Tend>, <Hmin>, <Hmax> [, <useInitialValues>, <initialvaluesFile>]]])</code>
Python Example	<code>oModule.EditTransient ("TR", ["NAME:SimSetup", "AnalysisEnabled:=", 0, "Name:=", "TR"])</code>

GetAllOptiSolutionSetups

Obtain the names of all the Optimetrics setups in the design.

UI Access	NA		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	None		
Return Value	Array of strings The Optimetrics setup names.		

Python Syntax	<code>GetAllOptiSolutionSetups()</code>
Python Example	<pre> optisetups = oModule.GetAllOptiSolutionSetups() for val in optisetups: AddInfoMessage (val) </pre>

GetAllSolutionSetups [Analysis Setup]

Obtain the names of all solution setups in the design

UI Access	NA		
Parameters	Name	Type	Description
	None		

Return Value	Array of strings, the solution setup names

Python Syntax	<code>GetAllSolutionSetups()</code>
Python Example	<pre>solsetups = oModule.GetAllSolutionSetups() for val in solsetups: AddInfoMessage(val)</pre>

GetNominalSetupNamesForOptiSetup

Obtain the nominal setup names for the specified Optimetrics setup.

UI Access	NA		
Parameters	<i>Name</i> <optiName>	<i>Type</i> String	<i>Description</i> The name of the Optimetrics setup as seen in the Project Window.
Return Value	Array of strings, the setup names that have been selected for the specified Optimetrics setup.		

Python Syntax	<code>GetNominalSetupNameForOptiSetup (<optiName>)</code>
Python Example	<pre>nomNames = oModule.GetNominalSetupNamesForOptiSetup ("parametricSetup1") for val in nomNames: AddInfoMessage(str(val)) sdbnames = oModule.GetSDBNamesForOptiSetup ("ParametricSetup1", val) for sdb in sdbnames: AddInfoMessage(str(sdb))</pre>

GetSDBNamesForOptiSetup

Obtain the SDB filenames for the specified nominal analysis of an Optimetrics setup.

UI Access	NA		
Parameters	<i>Name</i> <optiName>	<i>Type</i> String	<i>Description</i> The name of the Optimetrics setup as seen in the Project Window.
	<nomSetupiName>	String	The name of the nominal setup as seen in the Optimetrics setup General tab.
Return Value	Array of strings, the SDB filenames containing data for the specified Optimetrics setup.		

Python Syntax	GetSDBNameForOptiSetup (<optiName><nomSetupiName>)
Python Example	<pre>sdbnames = oModule.GetSDBNamesForOptiSetup ("ParametricSetup1", "TR") for val in sdbnames: AddInfoMessage (str (val))</pre>

GetSDBNameForSetupName [Analysis Setup]

Obtain the name of the SDB file for the given setup, often so that `SdbCtrl` functions can be used to then extract data from the SDB file.

UI Access	NA		
Parameters	<i>Name</i> <setupName>	<i>Type</i> String	<i>Description</i> The name of the setup in the Twin Builder Design.
Return Value	String, the file name of the SDB file created when the named setup is analyzed.		

Python Syntax	GetSDBNameForSetupName (<setupName>)
Python Example	<code>oModule.GetSDBNameForSetupName ("TR")</code>

GetSetupInfo [Analysis Setup]

It returns setup information for given setup name and property name.

UI Access	NA		
Parameters	<i>Name</i> <setupName> <i>Type</i> String <i>Description</i> Name of available setup. For Transient analysis type setup "Tend", "Hmin", "Hmax" For AC analysis type setup "FEnd", "FStart", "Fstep"		
Return Value	String Property Value in String		

Python Syntax	GetSetupInfo (<setupName>,<PropertyName>)
Python Example	<pre>oModule = oDesign.GetModule("SimSetup") oModule.GetSetupInfo("TR", "Tend")</pre>

Remove [Analysis Setup]

To remove an unnecessary simulation setup from the design.

UI Access	Right-click on the desired setup in Project Manager and select Delete from the context menu.		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>

	<setup name>	String	The name of the setup to be deleted
Return Value	None		

Python Syntax	Remove (<setupName>)		
Python Example	<code>oModule.Remove("TR")</code>		

Rename [Analysis Setup]

Rename analysis setups.

UI Access	Right-click on the desired analysis icon in Project Manager and select Rename from the context menu.		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<oldname>	String	Name of the Analysis to rename.
	<newname>	String	New name of that Analysis.
Return Value	None		

Python Syntax	Rename (<oldname> <newname>)		
Python Example	<code>oModule.Rename("AC", "AC1")</code>		

SetSetupInfo [Analysis Setup]

Set setup information for given setup name and property name.

Other commands like EditTransient, EditACAnalysis or EditAnalysisOptions can also be used

UI Access	NA		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>

	<setupName> String Name of available setup. <PropertyName> String For Transient analysis type setup "Tend", "Hmin", "Hmax" For AC analysis type setup "FEnd", "FStart", "Fstep" <PropertyValue> String "40ms", "1hour", "10ns" "10khz", "20Mhz"
Return Value	Integer 1: Success; 0: Failure

Python Syntax	GetSetupInfo (<setupName>, <PropertyName>, <PropertyValue>)
Python Example	<pre>oModule = oDesign.GetModule("SimSetup") oModule.SetSetupInfo("TR", "Tend", "50ms")</pre>

15 - ROM Scripting

This section describes how to build ROM models for dynamic and static ROM Builders with the scripting interface.

You can build, evaluate, and export these ROM models as FMUs to Twin Builder.

This functionality generally requires an input JSON configuration file that details various settings necessary for building, evaluating, and exporting the model. Examples of various configuration files and scripts appear in your "<Installation_Directory>**ANSYS Inc**v252\AnsysEM\Examples\Twin Builder\Applications\Scripting" directory.

ROM Manager

You can use ROM manager to create instance handles to dynamic and static ROM builders. You can also see what ROM models were built and choose the models to export to Twin Builder.

GetROMInfo

Retrieves important model information about the ROM. The attributes retrieved are the model complexity, relative error and maximum absolute error.

The first argument is a complete path to the **.dyn** file (Dynamic ROM).

You can also pass in a value from the array returned by **GetAllROMs()** function as the first argument if you don't wish to type in the full path. Both methods are illustrated below.

Note: This function is currently supported only for Dynamic ROMs.

Python Syntax	GetROMInfo(<i>model name</i>)
Python Example	<pre>romMgr = oDesign.GetROMManager() #Method 1 romMgr.GetROMInfo("F:/Testing/StaticRom.dyn") # Dynamic ROM #Method 2 arr = romMgr.GetAllROMs() romMgr.GetROMInfo(arr[0])</pre>

ROMBuilderInstances

Get instance handles to the static and dynamic ROM builder.

Python Syntax	GetROMManager()
Python Example	<pre>romMgr = oDesign.GetROMManager()</pre>
Python Syntax	GetDynamicROMBuilder()
Python Example	<pre>Dyn = romMgr.GetDynamicROMBuilder()</pre>
Python Syntax	GetStaticROMBuilder()
Python Example	<pre>Static = romMgr.GetStaticROMBuilder()</pre>

GetAllROMs

Returns an array of all the ROM models built. The type of ROM is also specified.

Python Syntax	GetAllROMs ()
Python Example	<pre>romMgr = oDesign.GetROMManager() arr = romMgr.GetAllROMs()</pre>

CreateROMComponent

Creates and imports the ROM Component into Twin Builder. Use this function to create static and dynamic ROM components. The first argument is a complete path to the **.dyn** file (for a dynamic ROM) or the **.rom** file (for a static ROM). The second argument is the name of the component inside Twin Builder.

You can also pass in a value from the array returned by the [GetAllROMs\(\)](#) function as the first argument if you don't want to type in the full path. Both methods are illustrated below.

Python Syntax	CreateROMComponent (<modelName>, <componentName>)
Python Example	<pre>romMgr = oDesign.GetROMManager() #Method 1 romMgr.CreateROMComponent ("F:/Testing/StaticRom.rom", "staticabc")# StaticROM romMgr.CreateROMComponent</pre>

	<pre>("F:/Testing/StaticRom.dyn", "dynabc") # Dynamic ROM #Method 2 arr = romMgr.GetAllROMs() romMgr.CreateROMComponent (arr[0], "dynabc")</pre>

Note:

Only models created in the current Twin Builder session with the Build command can be imported as components.

Static ROM Builder

See the topics in this section for information on working with static ROM models.

Build

The Build command generates a static ROM model (**.rom** file). You must give an input argument which specifies the full path to the configuration file to the function.

The **.rom** file will be created based on a set of snapshots and a **.doe** file (list of parameters values associated to the snapshots).

Python Syntax	Build (<ConfigurationFile>)
Python Example	<pre>romMgr=d.GetROMManager() srom=romMgr.GetStaticROMBuilder() srom.Build("F:/Testing/ SROMbuild.conf")</pre>

Configuration File

The configuration file is a JSON file which details the parameters you can tune to build the model. See "**ANSYS Inc\252\AnsysEM\Examples\Twin Builder\Applications\Scripting\Ex7_StaticROMScripting\SROM_build.conf**" in your local installation directory for an example.

The configuration file contains the following attributes :

- **doe_file** – The **doe.csv** file,
- **snapshots_directory** – The snapshots directory associated with the **doe.csv** file,
- **write_rom_to** – The output ROM in **write_rom_to**.

Example of SROMbuild.conf

```
{
  "SROMbuild":
  {
    "write_rom_to" : "StaticRom.rom",
    "doe_file" : " D: /TestCases/Ex1/doe.csv",
    "snapshots_directory": "D: /TestCases/Ex1/snapshots/"
  }
}
```

To use fewer learning points

All the points in the **Doe.csv** file are considered learning points, and the number of modes is automatically determined by the Leave One Out process. You may change the **doe.csv** file to keep only the points you want to use for learning.

To fix the number of modes

Use the **basis_dimension** keyword in the JSON file as in the following example.

```
{
  "SROMbuild":
  {
    "write_rom_to" : "StaticRom.rom",
    "doe_file" : " D: /TestCases/Ex1/doe.csv",
    "snapshots_directory": "D: /TestCases/Ex1/snapshots/",
    "basis_dimension" :5
  }
}
```

Evaluate

Once a static ROM model (**.rom** file) has been created using the Build command, you can evaluate the generated model. A configuration file is necessary to perform this step. An input argument which specifies the full path to the configuration file must be given.

Python Syntax	Evaluate(<configurationFile>)
Python	romMgr=d.GetROMManager()

Example	<pre>srom=romMgr.GetStaticROMBuilder() srom.Evaluate("F:/Testing/ SROMEvaluate1.conf")</pre>

Configuration File

The configuration file is a JSON file which details the parameters you can tune to evaluate the model. Refer to "**ANSYS Inc\vt252\AnsysEM\Examples\Twin Builder\Applications\Scripting\Ex7_StaticROMScripting\SROM_Evaluate1.conf**" in your local installation directory for an example.

There are two ways to enter the input parameters values. You can specify them in the configuration file, or you can specify them in a separate CSV file. To input the parameters in the configuration file, use the keyword **input_parameters_values**. To input parameters in a CSV file, use the keyword **input_param_CSV_file**. See an example of the data file below (**data.csv**).

The resulting snapshot files are stored in the files given in the keyword **output_snapshots_filenames**.

Example of SROMEvaluate.conf

```
{
  "SROMEvaluate":
  {
    "read_rom_from" : "StaticRom.rom",

    // use one of the 2 following keywords :
    "input_parameters_values" : [ [6,420],[8,450]],
    //"input_param_CSV_file" : "data.csv",

    "output_snapshots_filenames" : ["res1_data.bin","res2_
data.bin"]
  }
}
```

Example of data.csv (two input parameters and two points evaluated)

```
6.0;420.0
8.0;450.0
```

To force the evaluation of the ROM by taking into account less modes

Use the **basis_dimension** keyword in the JSON file as in the following example.

```
{
    "SROMevaluate":
    {
        "read_rom_from" : "StaticRom.rom",

        // use one of the 2 following keywords :
        "input_parameters_values" : [ [6,420],[8,450]],
        //"input_param_CSV_file" : "data.txt",

        "output_snapshots_filenames" : ["res1_data.bin","res2_
data.bin"],
        "basis_dimension" :3 //optional : to reduce the number of
modes
    }
}
```

Importing a Static ROM Model Into Twin Builder

Once you have created a ROM model(.rom file) with the Build command, you can subsequently import that model into Twin Builder with the CreateROMComponent function. For a static ROM, a configuration file named **SROM_props.conf** is required. This file must be placed in the same folder where your model .rom file is located. This file is used to define the input pins names. Refer to "**\\ANSYS Inc\\v252\\AnsysEM\\Examples\\Twin Builder\\Applications\\Scripting\\Ex7_StaticROMScripting\\SROM_props.conf**" in your local installation directory for an example.

```
{
    "inputNames" : [ "inlet-velocity", "inlet-temperature" ], //to
update
}
```


Note: The static ROM component imported into Twin Builder will contain:

- The geometry if the geometry file ("**point.bin**") is present in the same directory as the rom file
- The settings file ("**settings.json**") if it is present in the same directory as the rom file
- The views definition only if both the geometry file ("**point.bin**") and the views definition file ("**pointView.json**") are present in the same directory as the rom file
- The definition of some scalar output pins if the "**outputDefinition.json**" file is present in the same directory as the rom file. Otherwise the output pins will be restricted to the mode coefficients output pins.

Dynamic ROM Builder

See the topics in this section for information on working with dynamic ROM models. Note that the dynamic model scripting has been developed yet only for the scalar cases (no input reduction, neither output reduction)

Build

The Build command generates a dynamic ROM model (**.dyn** file). You must give an input argument which specifies the full path to the configuration file to the function.

Python Syntax	Build (<ConfigurationFile>)
Python Example	<pre>romMgr=d.GetROMManager() dyn=romMgr.GetDynamicROMBuilder() dyn.Build("F:/Testing/ dynarom_build.conf")</pre>

Configuration File

The configuration file is a JSON file which details the parameters you can tune to build the model. See "**\\ANSYS Inc\\v252\\AnsysEM\\Examples\\Twin Builder\\Applications\\Scripting\\Ex7_DynamicROMScripting\\dynarom_build.conf**" in your local installation directory for an example.

- **exc_filenames** – Should contain a list of excitation files: one file per scenario. Each excitation file is a CSV file containing the time history of the input data.

- **output_filenames** – Should contain a list of outputs files: one file per scenario. Each output file is a CSV file containing the time history of the output data.

Note:

exc_filenames and **output_filenames** are CSV files with a ; or a , as separator. The first column is the time column. Then, there are as many columns as the number of excitations (respectively outputs) in the dynamic ROM model. The number of lines is the number of time steps plus the header line. A scenario is defined as a transient simulation of any solver where the input data is stored in the excitation file and the outputs data is stored in the corresponding outputs file.

- **write_model_to** – Contains the name and path of the file where the dynamic model will be stored.
- **write_model_output_to_directory** – Path to the directory where the learning results are stored . There will be as many CSV files as the number of scenarios, each containing the time history of the ROM-predicted outputs data for the corresponding excitation file.

Note:

For **write_model_output_to_directory** and **write_model_to**, note that the combination of **Write_model_output_to_directory** and **write_model_to** must point to a valid path. Relative paths may also be provided. For example:

Write_model_output_to_directory: "F:/Testing/"

Write_model_to: "model.dyn"

The combination would be **F:/Testing/model.dyn**.

- **settings** – Contains two fields to set the precision: **relative_l2_precision** and **absolute_max_precision**. To use only one of these accuracy types, set the other one to **0**. If using both, the learning step ends when either of them is satisfied. The definition of accuracy is detailed in **Appendix B: ROM Accuracy** of the Dynamic ROM Builder documentation.

Evaluate

Once a dynamic ROM model (.dyn file) has been created using the Build command, you can evaluate the generated model. A configuration file is necessary to perform this step. An input argument which specifies the full path to the configuration file must be given.

Refer to "**\\ANSYS Inc\\v252\\AnsysEM\\Examples\\Twin Builder\\Applications\\Scripting\\Ex7_DynamicROMScripting\\dynarom_evaluate.conf**" in your local installation directory for an example.

**Python
Syntax**

Evaluate(<configurationFile>)

**Python
Example**

```
romMgr=d.GetROMManager()
dyn=romMgr.GetDynamicROMBuilder()
dyn.Evaluate("F:/Testing/ dynarom_evaluate1.conf")
```

Configuration File

The configuration file is a JSON file which details the parameters you can tune to build the model. Refer to "**ANSYS Inc\v252\AnsysEM\Examples\Twin Builder\Applications\Scripting\Ex7_DynamicROMScripting\dynarom_evaluate.conf**" in your local installation directory for an example.

- **dynarom_filename** – Contains the name and corresponding path) of the dynamic ROM model to be evaluated.
- **exc_filenames** – Contains one or several excitation files, depending on whether you want to evaluate the ROM on one or several scenarios.
- **write_model_output_to_directory** – Path to the directory where the results are stored. There will be as many CSV files as the number of scenarios, each containing the time history of the ROM-predicted outputs data for the corresponding excitation file.

Importing a Dynamic ROM Model Into Twin Builder

Once you have created a ROM model (.dyn file) with the Build command, you can subsequently import that model into Twin Builder with the CreateROMComponent function. Unlike exporting a static ROM component, no specific configuration file is necessary for exporting a dynamic ROM component.

16 - Output Variable Script Commands

```
oModule = oDesign.GetModule("OutputVariable")

oModule.CommandName <args>
```

CreateOutputVariable

Creates and 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 can not be cyclic. For example, $A = 2 \cdot B$, $B = 3 \cdot A$ is not allowed.

UI Access	Twin Builder>Results>Output Variables		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<OutputVarName>	String	Name of the output variable
	<Expression>	Value	Value to assign to the variable
	<SolutionName>	String	The name of the solution as seen in the output variable UI.
	<reportTypeName> >	String	The name of the report type as seen in the output variable UI.
Return Value	<simValueCtxt>	Variant	Context for which the output variable expression is being evaluated.
	None		

Python Syntax	CreateOutputVariable (<OutputVarName>, <Expression>, <Solution Name>, <reportTypeName>, <simValueCtxt>)
Python Example	<pre>oModule = oDesign.GetModule("OutputVariable") oModule.CreateOutputVariable ("test", "normalize(R1_0.V)", — "TR", "Standard", ["NAME:Context", "SimValueContext:=", _ [1, 0, 2, 0, false, false, -1, 1, 0, 1, 1, "", 0, 0]])</pre>

DeleteOutputVariable

Deletes an existing output variable. The variable can only be deleted if it is not in use by any traces.

UI Access	Twin Builder>Results>Output Variables , dialog Delete Button		
Parameters	<i>Name</i> <OutputVarName>	<i>Type</i> String	<i>Description</i> Name of the output variable
Return Value	None		

Python Syntax	DeleteOutputVariable (<OutputVarName>)		
Python Example	<pre>oModule = oDesign.GetModule("OutputVariable") oModule.DeleteOutputVariable ("testNew")</pre>		

DoesOutputVariableExist

Verify that a named output variable exists

UI Access	NA		
Parameters	<i>Name</i> <outputVariableName>	<i>Type</i> String	<i>Description</i> Text string of the output variable name
Return Value	Boolean True if the variable exists. False otherwise.		

Python Syntax	DoesOutputVariableExist(<outputVariableName>)		
Python Example	<pre>oProject = oDesktop.GetActiveProject() oDesign = oProject.GetActiveDesign() oModule = oDesign.GetModule("OutputVariable")</pre>		

```
oModule.DoesOutputVariableExist("MyTestVar")
```

EditOutputVariable

Changes the name or expression of an existing output variable

UI Access	NA		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<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"
	<ReportTypeName>	String	The name of the report type as seen in the output variable UI
	<simValueCtxt>	Variant	Context for which the output variable expression is being evaluated Array("Context:=", <Context>)
Return Value	None		

Python Syntax	EditOutputVariable (<OrigVarName>, <NewExpression>, <NewVarName>, <SolutionName>, <reportTypeName>, <simValueCtxt>)
Python Example	<pre>oModule = oDesign.GetModule("OutputVariable") oModule.EditOutputVariable ("test", "normalize(R1_0.V)", "testNew", "TR", "Standard", [])</pre>

Note:

Provide empty quotation marks "" as the `NewVarName` or `NewExpression` if it should not be changed.

GetOutputVariables

Gets a list of output variables.

UI Access	NA		
Parameters	<i>Name</i> None	<i>Type</i>	<i>Description</i>
Return Value	An array of output variable names.		

Python Syntax	GetOutputVariables()
Python Example	<pre>vec = oModule.GetOutputVariables()</pre>

GetOutputVariableValue

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.

UI Access	NA		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<OutputVarName>	String	Name of the output variable.
	<IntrinsicVariation>	String	A set of intrinsic variable value pairs to use when evaluating the output expression. Example: "Freq='20GHz' Theta='20deg' Phi='30deg' in Twin Builder "" in Q3D Extractor
	<SolutionName>	String	Name of the solution as listed in the output variable UI.

	<ReportTypeName> String The name of the report type as seen in the output variable UI. <SimValueContext> Array Context for which the output variable expression is being evaluated. This can be empty if there is no context.
Return Value	Double value of the output variable .

Python Syntax	GetOutputVariableValue(<OutputVarName>, <IntrinsicVariation>, <SolutionName>, <ReportTypeName>, <SimValueContext>)
Python Example	<pre>Val = oModule.GetOutputVariableValue("test","Freq = '20Ghz' Theta='20deg' Phi='30deg'", "TR", "Standard", [])</pre>

SimValueContext

SimValueContext holds context information for a trace, and describes how data for a trace should be extracted from the simulation. SimValueContext contains a list of 14 required initial values:

```
SimValueContext (
Domain ID, Calculation Type, Number of Cycles, Rise Time,
Step, Impulse, Context ID, Window Width,
Window Type, TDR Kaiser Parameter, Hold Time, DeviceName,
TDR Step Time, DR Maximum Time )
```

For example, the following indicates a trace in the Time Domain, Standard Calculation with the number of cycles being 2:

```
"SimValueContext:=", Array(1, 0, 2, 0, false, false, -1, 1, 0, 1, 1,
"", 0, 0)
```

Additional, context-specific values may follow the required values, as described in subsection 15 below.

1. Domain ID

No Domain	0
Time Domain	1
Spectrum Domain	2
Sweep Domain	3
Device Domain	4
SinglePt Domain	5
LoadPull Domain	6
Transient Domain	7
Budget Domain	8
NetworkFunction Domain	9
Oscillator Domain	55802
Noise Domain	55803
Transfer Function Domain	55807
Time Frequency Domain	55808
Transient Time Domain	55809
Periodic AC Domain	55818
UI Domain	55819
Eye Measurement Domain	55823
Initial Response Domain	55824
Peak Distortion Domain	55825

2. Calculation Type

Standard Calculation	0
Device2_DCIV	1
Device3_DCIV_Output	2
Device3_DCIV_Input	3
Device3_DCIV_Transfer	4
Device3_DCIV_Reverse	5
Device2_ACLoad	6
Device3_ACLoad_Output	7
Device3_ACLoad_Input	8
Device3_ACLoad_Transfer	9
Device3_ACLoad_Reverse	10
Constellation	11
EyeDiagram	12

FreeX (Statistic Report) 13

3. **Number of Cycles** — Used in Time Domain in HarmonicBalance analysis.
4. **Rise Time** — Not used by Nexxim.
5. **Step** — Not used by Nexxim.
6. **Impulse** — Not used by Nexxim.
7. **Context ID** — Not used by Nexxim.
8. **Window Width** — Not used by Nexxim.
9. **Window Type** — Not used by Nexxim.
10. **TDR Kaiser Parameter** — Not used by Nexxim.
11. **Hold Time** — Not used by Nexxim.
12. **DeviceName** — Not used by Nexxim.
13. **TDR Step Time** — Not used by Nexxim.
14. **TDR Maximum Time** — Not used by Nexxim.
15. **Context-specific values** — Used in Time Domain in HarmonicBalance analysis.

Context-specific values are entered in the format "key, true/false, keyvalue", where:

- **"key"** is the name of the key being set.
- **"true/false"** indicates whether the key is a string value.
- **"keyvalue"** is the value of the key.
- The order of the context keys is not significant.
- Context keys have software defaults that will be used if not provided in the script.

a. Plotting Range for Time domain in Transient and QuickEye analysis:

Description	Key Name	Is a string?	Key Value
Start Time	WS	False	0ns
Stop Time	WE	False	10ns
Minimum Time	WM	False	0ns

Maximum Time	WN	False	10ns
Is Thinning Enabled?	DE	False	0
Dy/dx Tolerance	DT	False	0.001
Number of points	DP	False	20000000

b. Transient report context for Spectral domain in Transient analysis:

Description	Key Name	Is a string?	Key Value
Start Time	TS	False	0ns
Stop Time	TE	False	10ns
Max Harmonics	MH	False	100
Window type	WT	False	0
Width Percentage	WW	False	100
Kaiser Parameter	KP	False	0
Adjust Coherent Gain	CG	False	0

Window Type ID

Rectangular	0
Bartlett	1
Blackman	2
Hamming	3
Hanning	4
Kaiser	5
Welch	6
Weber	7
Lanzcos	8

c. Eyeprobe index context for UI domain, Time domain, Eye Measurement domain in VerifEye and QuickEye analysis:

Description	Key Name	Is a string?	Key Value
Eyeprobe compinst ID	PCID	False	0

d. Eyesource index context for Initial Response domain and Peak Distortion domain in VerifEye and QuickEye analysis:

Description	Key Name	Is a string?	Key Value
Eyesource compinst ID	SCID	False	0

e. UI domain context in VerifEye and QuickEye analysis:

Description	Key Name	Is a string?	Key Value
Use midpoint?	MIDPOINT	False	0 - Don't use midpoint. 1 - Use midpoint of amplitude. 2 - Use midpoint of UI.
Minimum latch overdrive	MLO	False	0

f. Distribution Context for UI Domain in VerifEye and QuickEye analysis:

Description	Key Name	Is a string?	Key Value
Use distribution?	USE_DIST	False	0 - No 1 - Yes
Distribution type	DIST	False	0 - Receiver Jitter 1 - Receiver Noise 2 - User Defined

Receiver Jitter Parameters

Description	Key Name	Is a string?	Key Value
DLL standard deviation	DSD	False	0
Distribution type	DIST	False	0
DLL taps	DMN	False	0
Static Offset	SOFF	False	0
Number of Gaussian data sets	NUMG	False	0
Gaussian std deviation	GS0,GS1...	False	0
Offset mean	GM1,GM1...	False	0
Number of Uniform data sets	NUMU	False	0
Uniform width	UW0,UW1...	False	0
Uniform mean	UM1,UM1...	False	0

Receiver Noise Parameters

Description	Key Name	Is a string?	Key Value
Number of Gaussian data sets	NUMG	False	0
Gaussian std deviation	GS0,GS1...	False	0
Number of Uniform data sets	NUMU	False	0
Uniform width	UW0,UW1...	False	0

User Defined Parameters

Description	Key Name	Is a string?	Key Value
Number of XY data pairs	NUMG	False	0
X data	X0,X1,X2...	False	0
Y data	Y0,Y1,Y2...	False	0
Cutoff probability	CP	False	0

17 - Reporter Editor Script Commands

```
oDesign = oProject.SetActiveDesign("SimplorerDesign1")
```

```
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: Twin Builder supports Reporter scripting. When you execute **Tools>Record Script**, Twin Builder operations performed in the Reporter are automatically recorded.

Report Setup Script Commands

AddCartesianLimitLine

UI Access	Report2D>Add Limit Line		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<ReportName>	String	Name of Report
	<XValues>	Real	Array of X coordinate values
	<XUnits>	String	Unit of measure for X values
	<YValues>	Real	Array of Y coordinate values
	<YUnits>	String	Unit of measure for Y values
	<YAxisName>	Real	Name of the Y axis associated with the limit line
Return Value	None		

Python Syntax	AddCartesianLimitLine <ReportName>, [{"NAME:CartesianLimitLine", [{"NAME:XValues",<XValues>}], "XUnits:=", "<XUnits>", [{"NAME:YValues", <YValues>}], "YUnits:=", "<YUnits>", "YAxis:=", "<YAxisName>")]
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],</pre>

	"YUnits:=", "mV", "YAxis:=", "Y1"])

AddCartesianXMarker

Adds a marker to a report on the X axis

UI Access	Report2D>Marker>Add X Marker		
Parameters	<i>Name</i> <ReportName>	<i>Type</i> String	<i>Description</i> Name of Report
Return Value	None		

Python Syntax	AddCartesianXMarker("<ReportName>", "<MXn>", <Xcoord>)
Python Example	oModule.AddCartesianXMarker ("XY Plot 1", "MX1", 0)

AddCartesianYMarker

Adds a marker to a report on the Y axis.

Command: Report2D>Marker>Add Y Marker

Syntax: AddCartesianYMarker <ReportName>, <MarkerID>, <Ycoord>

Return Value: None

Parameters: <ReportName>

Type: <string>

Name of Report.

<MarkerID>

Type: <string>

ID of the marker, for example: "Y1".

<YCoord>

Type: <real>

Y location for the marker.

Python Syntax	AddCartesianYMarker("<ReportName>", "<Yn>", <Ycoord>)
Python Example	oModule.AddCartesianYMarker ("XY Plot 1", "Y1", 0)

AddDeltaMarker

Add delta markers to a plot.

UI Access	Report2D>Marker>Add Delta Marker		
Parameters	<i>Name</i> <ReportName> <MXn> <XCoord>	<i>Type</i> String String Real	<i>Description</i> Name of Report Name and <i>n</i> for the x marker X location for the marker
Return Value	None		

Python Syntax	AddDeltaMarker("<ReportName>", "<mn>", "<traceID"> "<xcoord>" "<mn>", "<traceID"> "<xcoord>")
Python Example	<pre>oModule.AddDeltaMarker("XY Plot 1", "m3", "GS1.VAL : TR4 : Cartesian", "1826000000ns", "m4","GS1.VAL : TR4 : Cartesian", "2096000000ns")</pre>

AddMarker

Adds a marker to a trace on a report.

UI Access	Report2D>Marker>Add Marker		
Parameters	<i>Name</i> <ReportName> <MXn> <XCoord>	<i>Type</i> String String Real	<i>Description</i> Name of Report Name and <i>n</i> for the x marker. X location for the marker
Return Value	None		

Python Syntax	AddMarker("<ReportName>", "<mn>", "<traceID>" "<xcoord>")
Python Example	oModule.AddMarker("XY Plot 1", "m5", "GS1.VAL : TR4 : Cartesian", "3.615999999999997s")

AddNote

Adds a note at a specified location to a given report.

UI Access	Right-click on the plot and select Add Note		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<ReportName>	String	Name of report
	<NoteDataArray>	Array	Array("NAME:<NoteDataName>", <NoteArray>)
	<NoteDataName> <NoteArray>	String Array	Array("NAME:<NoteDataSourceName>", "SourceName:=", <SourceName>, "HaveDefaultPos:=", <boolean>, "DefaultXPos:=", <XPos>, "DefaultYPos:=", <YPos>, "String:=", <Note>))
Return Value	None		

Python Syntax	AddNote (<ReportName> <NoteDataArray>)
Python Example	oModule = oDesign.GetModule("ReportSetup") oModule.AddNote ("XY Plot1", ["NAME:NoteDataSource", ["NAME:NoteDataSource", "SourceName:=", "Note1", "HaveDefaultPos:=", true, "DefaultXPos:=", 1996, "DefaultYPos:=", _ 3177, "String:=", "This is a

	note"]]

AddTraceCharacteristics

To add characteristics to traces.

UI Access	Report Type>Trace Characteristics>Add		
Parameters	<i>Name</i> <Datatypes>	<i>Type</i> Varies	<i>Description</i> Add the Datatypes
Return Value	None		

Python Syntax	AddTraceCharacteristics (<data types>)		
Python Example	<pre>oModule.AddTraceCharacteristics ("XY Plot 1", "pulsemin", [], ["Full"])</pre>		

AddTraces

Creates a new trace and adds it to the specified report

UI Access	Modify Report>Add Trace		
Parameters	<i>Name</i> <ReportName> <SolutionName> <ContextArray>	<i>Type</i> String String Array of Strings	<i>Description</i> Name of Report Name of the solution as listed in the Modify Report dialog box. Context for which the expression is being evaluated. This can be an empty string if there is no context. Array("Domain:=", <DomainType>) <DomainType> ex. "Sweep" or "Time" Array("Context:=", <GeometryType>)

	<GeometryType> ex. "Infinite Spheren", "Spheren", "Polylinen" <FamiliesArray> Array of Strings Contains sweep definitions for the report. Array("<VariableName>:= ", <ValueArray>) <ValueArray> Array("All") or Array("Value1", "Value2", ... "Valuen") examples of <VariableName> "Freq", "Theta", "Distance" <ReportDataArray> Array of Strings This array contains the report quantity and X, Y, and (Z) axis definitions. Array("X Component:=", <VariableName>, "Y Component:=", <VariableName> <ReportQuantityArray>) <ReportQuantityArray> ex. Array("dB(S(Port1, Port1))")
Return Value	None

Python Syntax	Add Traces (<ReportName>, <SolutionName> ,<ContextArray >,<FamiliesArray>, <ReportDataArray>)
Python Example	<pre>oModule.AddTraces ("XY Plot1", "Setup1 : Sweep1", _ ["Domain:=", "Time", "HoldTime:=", 1, "RiseTime:=", 0, _ "StepTime:=", 6.24999373748E-012, "Step:=", false, _ "WindowWidth:=", 1, _ "WindowType:=", 0, "KaiserParameter:=", 1, _ "MaximumTime:=", 6.2437437437444E-009], _ ["Time:=", ["All"], "OverridingValues:=", ["0s", _</pre>

	<pre>"6.24999373748188e-012s", ...]], ["X Component:=", "Time",_ "Y Component:=", ["TDRZ (WavePort1)"]], [])</pre>

ChangeProperty [ReportSetup]

Change the properties for a Report.

Command: Select a Report and change properties.

Syntax: ChangeProperty <array>

Return Value: None

Parameters: Varies, depending on the properties associated with the select object command.

Python Syntax	ChangeProperty ([<array>])
Python Example	<pre>oModule = oDesign.GetModule("ReportSetup") oModule.ChangeProperty(["NAME:AllTabs", ["NAME:General", ["NAME:PropServers", "XY Plot 1:General"], ["NAME:ChangedProps", ["NAME:Use Scientific Notation", "Value:=" , True</pre>

	]
	]
	]
	])

ClearAllMarkers

Clears all markers from a report

UI Access	Report2d>Markers>ClearAllMarkers		
Parameters	<i>Name</i> <ReportName>	<i>Type</i> String	<i>Description</i> Name of Report
Return Value	None		

Python Syntax	ClearAllMarkers("<ReportName>")
Python Example	oModule.ClearAllMarkers ("XY Plot 1")

ClearAllTraceCharacteristics

Clear all trace characteristics from a report

UI Access	Report_Type>Trace Characteristics>Clear All		
Parameters	<i>Name</i> <plot name>	<i>Type</i> String	<i>Description</i> Name of the plot
Return Value	None		

Python Syntax	ClearAllTraceCharacteristics (<plot name>)
Python Example	oModule.ClearAllTraceCharacteristics("XY Plot 1")

CopyReportData

Copy all data corresponding to the specified reports

UI Access	Select a report in the Project tree, right-click and select Copy Data		
Parameters	<i>Name</i> <ReportsArray>	<i>Type</i> Array of strings	<i>Description</i> Names of reports from which to copy data
Return Value	None		

Python Syntax	CopyReportData (<ReportsArray>)		
Python Example	oModule.CopyReportData (["Transmission", "Reflection"])		

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	<i>Name</i> <ReportsArray>	<i>Type</i> Array of strings	<i>Description</i> Names of reports from which to copy the definitions
Return Value	None		

Python Syntax	CopyReportDefinitions (<ReportsArray>)		
Python Example	oModule.CopyReportDefinitions (["Transmission", "Reflection"])		

CopyTraceDefinitions

Copy trace definitions for a paste operation

UI Access	Select a trace in the Project tree, right-click and select Copy Definition		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<ReportName> <TracesArray>	String Array of Strings	Name of Report Trace definitions to copy
Return Value	None		

Python Syntax	CopyTraceDefinitions(<ReportName>, <TracesArray>)		
Python Example	<pre>oModule.CopyTraceDefinitions ("Transmission", ["mag (S (Port1, Port2)) "])</pre>		

CopyTracesData

Copy trace data for a paste operation

UI Access	Select a trace in the Project tree, right-click and select Copy Data		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<ReportName> <TracesArray>	String Array of Strings	Name of the Report 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>		

CreateReport

Creates a new report and adds it to the **Results** branch in the project tree

UI Access	Twin Builder>Results>Create Standard Report		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<Report Name>	String	Name of Report.
	<ReportType>	String	Possible values are: "Standard" and "BodeNyquist"
	<DisplayType>	String	For "Standard", set to one of the following: Rectangular Plot", "Polar Plot", "Radiation Pattern", "Data Table", "3D Rectangular Plot", "3D Polar Plot", "Rectangular Stacked Plot", or "Digital Plot". For "BodeNyquist", set to one of the following: "Bode Plot", "Nyquist Plot"
	<SolutionName>	String	Name of the setup for the traces to be added, as selected in the Modify Report dialog box, e.g., "TR".
	<ContextArray>	Array	<pre>Array("Name:Context", ["OptiSetup=", <Opti- metricsSetupName>,"SimValueCon- text=", Array(<domain>, 0, 0, 2, 0, false, false, -1, 1, 0, 1, 1, "", 0, 0 [<Attributes>]))</pre>
	<OptimetricsSetupName>	String	The name of the Optimetrics setup, e.g., "ParametricSetup1" [Note: OptiSetup name/value pair included only if trace context is Optimetrics]
	<domain>	Integer	Use 1 for TR analysis time domain or for DC, 3 for AC analysis frequency domain, and 2 for TR analysis spectral domain; a number is necessary for Optimetrics, but is ignored.
	<Attributes>	Pairs of Strings	When Spectral is selected, from the Reporter dialog: "TS", <StartTime> "TE", <StopTime>

		"MH", <MaxHarmonics> when Spectral is selected, from the Advanced button (Windowing) dialog: "WT", <WindowType> (Rectangular=0, Bartlett=1, Blackman=2, Hamming=3, Hanning=4, Kaiser=5, Welch=6, Weber=7, Lanczos=8) "WW", <WidthPercentage> (not interactively settable, defaults to 100) "KP", <Kaiser Param> "CG", <Correct Gain> (0 -> false, 1 -> true) "TH", <Noise Threshold> "UF", <UseFourier> (0 -> false, 1 -> true) // from Reporter dialog when modifying a datatable "LastOnly", <Show Last Value Only> (NumericPlot when true, 0 is false, 1 is true)
<FamiliesArray>	Array	Array("<VariableName>:= ", Array(<ValueArray>))
<VariableName>	String	The name of a swept variable, generally "Time" or "F"
<ValueArray>	String	Either "All" or "Value1", "Value2", ..., "Valuen", which are individual values for the variable
<ReportDataArray>	Array	For "Rectangular Plot", "Data Table", "Rectangular Stacked Plot", or "Digital Plot": <pre>Array("X Component:=", <VariableName>, "Y Component:=", Array (<TraceName>))</pre>

	For Polar plots: <pre>Array("Polar Component:=", Array(<VariableName>))</pre> For Nyquist plots: <pre>Array("Response Component:=", Array(<TraceName>[/<Trace- Name>]))</pre> For Radiation Pattern plots: <pre>Array("Ang Component:=", <VariableName>, "Mag Component:=", Array (<TraceName>))</pre> For 3D Rectangular plots: <pre>Array("X Component:=", <VariableName>, "Y Component:=", <VariableName>, "Z Component:=", Array (<TraceName>))</pre> For 3D Polar plots: <pre>Array("Phi Component:=", <VariableName>, "Theta Component:=", <VariableName>, "Mag Component:=", Array (<TraceName>))</pre>		
	<TraceName>	String	The name of a quantity or expression to plot
	<ExtendedInfo>	Array	Unused currently – use an empty array
	Return Value	None	

Python Syntax	CreateReport (<ReportName>, <ReportType>, <DisplayType>, <SolutionName>, <ContextArray>, <FamiliesArray>, <ReportDataArray>, <ExtendedInfo>)

**Python
Example**

Three examples are given below:(nominal, Optimetrics, spectral)

Nominal Example:

```
oModule.CreateReport( "XY Plot 2", "Standard", _
"Rectangular Plot", "TR", ["NAME:Context", _
"SimValueContext:=", [] 1, 0, 2, 0, false, _
false, -1, 1, 0, 1, 1, "", 0, 0]], ["Time:=", _
["All"], "aaa:=", ["Nominal"]], _
["X Component:=", "Time", "Y Component:=", _
["E1.I"]][[]])
```

Optimetrics Example:

```
oModule.CreateReport ("XY Plot 3", "Standard", _
"Rectangular Plot", "TR", ["NAME:Context", _
"OptiSetup:=", "ParametricSetup1", "SimValueContext:=", _
[1, 0, 2, 0, false, false, 0, 1, _
0, 1, 1, "", 0, 0]], ["Time:=", ["All"], _
"aaa:=", ["Nominal"]], ["X Component:=", _
"Time", "Y Component:=", ["E1.I"]], [])
```

Spectral Example:

```
oModule.CreateReport ("XY Plot 4", "Standard", _
"Rectangular Plot", "TR", ["NAME:Context", _
"SimValueContext:=", [ 2, 0, 2, 0, false, false, _
-1, 1, 0, 1, 1, "", 0, 0, "CG", false, "0", "KP", _
false, "0", "MH", false, "100", "TE", false, "40ms", _
"TH", false, "40", "TS", false, "0ns", "UF", false, _
"0", "WT", false, "0", "WW", false, "100"]], _
["Spectrum:=", ["All"], "aaa:=", _
["Nominal"]], ["X Component:=", "Spectrum", _
"Y Component:=", ["mag(E1.I)"]], [])
```


CreateReportFromFile

Creates a new report from an .rdat file.

Command: [product]>Results>CreateReportFromFile

Syntax: CreateReportFromFile "<FilePathName>"

Return Value: None

Parameters: <FilePathName>

Type: <string>

Path to .rdat file.

Python Syntax	CreateReportFromFile "<FilePathName>"
Python Example	<pre>oModule = oDesign.GetModule("ReportSetup") oModule.CreateReportFromFile ("C:/Users/MyDir/Documents/Return_Loss.rdat")</pre>

CreateReportFromTemplate

Create a report from a saved template

UI Access	Twin Builder>Results>Report Templates>PersonalLib><TemplateName>		
Parameters	<i>Name</i> <TemplatePath>	<i>Type</i> String	<i>Description</i> Path to report template
Return Value	None		

Python Syntax	CreateReportFromTemplate("<TemplatePath>")
Python Example	<pre>oProject = oDesktop.SetActiveProject("3PhaseRectifier") oDesign = oProject.SetActiveDesign("3PhaseRectifier")</pre>

	<pre>oModule = oDesign.GetModule("ReportSetup") oModule.CreateReportFromTemplate ("C:\MySimplorerProjects\PersonalLib\" _ & "ReportTemplates\TestTemplate.rpt")</pre>

CreateReportOfAllQuantities

Create a report including all quantities in a category. Cannot create a report with expressions

UI Access	NA		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	reportTypeArg		Report type name as input parameter
	displayTypeArg		Display type name as input parameter
	solutionNameArg		Solution name as input parameter
	simValueCtxtArg		a context name, or array of string that encoded the context(I).
	categoryNameArg		a category name as input parameter
	pointSetArg		Array of strings(II)
	commonComponentsOfTracesArg		Array of strings(III)
	extTraceInfoArg		Array of strings(IV)
Return Value	Report of all quantities in a category		

Python Syntax	CreateReportOfAllQuantities(reportNameArg, reportTypeArg, displayTypeArg, solutionNameArg, simValueCtxtArg, categoryNameArg, pointSetArg, commonComponentsOfTracesArg, extTraceInfoArg)
Python Example	<pre>solutions= oModule.CreateReportOfAllQuantities("Smith Chart all", "Modal Solution Data", "Smith Chart", "Setup1 : LastAdaptive", [], "S Parameter", ["Freq:=", ["All"], "offset:=", ["All"], "a:=",</pre>

```
[ "Nominal" ], "b:=", [ "Nominal" ] ], [], [])
```

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	<i>Name</i> None	<i>Type</i>	<i>Description</i>
Return Value	None		

Python Syntax	DeleteAllReports()
Python Example	oModule.DeleteAllReports()

DeleteMarker

Use: Deletes the specified marker.

Command: **[product]>Fields>Fields>Marker>Delete Marker**

Syntax: DeleteMarker <MarkerName>

Return Value: None

Parameters: <MarkerName>

Type: <string>

Name of the marker.

Example:

```
oModule = oDesign.GetModule("FieldsReporter")
```

```
oModule.DeleteMarker "m1"
```

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	<i>Name</i> <ReportNameArray>	<i>Type</i> Array of Strings	<i>Description</i> Array of all the report names that are to be deleted
Return Value	None		

Python Syntax	DeleteReports(<ReportNameArray>)		
Python Example	<code>oModule.DeleteReports (["Rept2DRectFreq"])</code>		

DeleteTraceCharacteristics

Delete the selected trace characteristic(s) from a report

UI Access	NA		
Parameters	<i>Name</i> <plot name> <characteristic name>	<i>Type</i> String String	<i>Description</i> Name of the plot Name of the characteristic
Return Value	None		

Python Syntax	DeleteTraceCharacteristics (<plot name> [<characteristic name>,])		
Python Example	<code>oModule.DeleteTraceCharacteristics ("XY Plot 1", ["pulsemin"])</code>		

DeleteTraces

Deletes an existing trace or traces

UI Access	Right-click the Trace to delete in the project tree, and then click Delete on the shortcut menu		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<TraceSelectionArray>	Array of Strings	Array("<ReportName>:=", <TracesArray>, <TracesArray>,...)
	<ReportName>	String	Name of Report
	<TracesArray>	Array of Strings	Contains the traces to delete within a report
	<Trace>	String	Array(<Trace>, <Trace>, ...) A specific trace that you want to delete
Return Value	None		

Python Syntax	DeleteTraces(<TraceSelectionArray>)
Python Example	<pre>oModule.DeleteTraces (["XY Plot 1:=", ["dB(S(LumpPort1,LumpPort1))"], "XY Plot 2:=", ["Mag_E"]])</pre>

EditCartesianXMarker

To change the X value of an X marker in a plot

UI Access	NA		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<plotName>	String	The name of the plot
	<markerName>	String	The name of the marker
	<xValue>	Real	The new X value for the marker
Return Value	None		

Python Syntax	EditCartesianXMarker(<plotName>, <markerName>, <xValue>)
Python Example	ret =oModule.EditCartesianXMarker ("XY Plot1", "MX1", "0")

EditMarker

To change the X value of a trace marker in a plot.

UI Access	NA		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<plotName>	String	The name of the plot
	<markerName>	String	The name of the marker
	<traceName>	String	Given by <expression> : <solution setup name> : <coordinate system type>, where <i>expression</i> is the name of the trace as seen in the plot legend, <i>solution setup name</i> is the name of an analysis setup (often "TR" or "AC"), and <i>coordinate system type</i> is one of Cartesian Polar Magnitude Phase Histogram
	<xValue>	Real	The new X value for the marker
Return Value	None		

Python Syntax	EditMarker(<plotName>, <markerName>, <traceName>, <xValue>)
Python Example	ret =oModule.EditMarker ("XY Plot1", "m1", "R2.V : TR : Cartesian", "0")

ExportImageToFile [Reporter]

Export an image in a specified format.

Command: None.

Syntax: ExportImageToFile(<ReportName>,<exportFileName>, <width>, <height>)

Parameters: <ReportName>

Type: <string>

Name of the image file.

<ExportFileName>

Type: <String>

Full path of the exported image file name; with extension of jpg, gif, tiff, tif, bmp, or wrl.

<width>

Type: <int>

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>

Type: <int>

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.

For the default, use 0, 0. For higher resolution, set the desired <width> and <height>, for example for 8k export as 7680, 4320:

Python Syntax	ExportModellImageToFile (["<path>/<imageName>.<formatsuffix>" 0,0, [(<SaveImageParams>)])
Python Example	oModule.ExportImageToFile(" Plot 1", "C:/Users/kmchrist/Documents/Plot 1.jpg", 0, 0)
Python Example, High Resolution	oModule.ExportImageToFile(" Plot 1", "C:/Users/kmchrist/Documents/Plot 1.jpg", 7680, 4320)

ExportMarkerTable

The documented command is applicable for Q3D Extractor.

*Use:*Exports the marker table to a .csv or .tab file.

Command:[product]>Fields>Fields>Marker>ExportMarkerTable

Syntax:ExportMarkerTable <pathandfilename>

Return Value:none

Parameters:<pathandfilename>

Example:

```
oModule = oDesign.GetModule("FieldsReporter")
oModule.ExportMarkerTable("C:/work/FieldMarkerTable.csv")
```

ExportPlotImageToFile [Reporter]

Use: Create field plot exports of existing field plots from a given view points, and with the model being auto-sized automatically for each view.

Command: None.

Syntax: ExportPlotImageToFile(<FileName>, "", <plotItemName>, <setViewTopDownDirectionByRelativeCS>)

Return Value: An image file.

Parameters: <FileName>

Type: <string>

Full path plus file name.

""

Type: <EmptyString>

<PlotItemName>

Type: <string>

Name of fields to plot.

<SetViewTopDownDirectionByRelativeCS>

Type: <string>

Name of relative coordinate system to use for the field plot.

Example:

```
oModule.ExportPlotImageToFile "C:\TestEPITF1.jpg", "", "Mag_E1",
"Global"
```

```
oModule.ExportPlotImageToFile "C:\TestEPITF2.jpg", "", "Mag_E2",  
"RelativeCS1"  
  
oModule.ExportPlotImageToFile "C:\TestEPITF3.jpg", "", "Mag_E3",  
"RelativeCS2"
```

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](#).

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	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<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	ExportReport(<ReportName>, <FileName>)
Python Example	<pre>oDesign.ExportReport("Plot1", "c: eport1.dat")</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.

Command: Right-click on report name in the Project tree and select Export Data.

Syntax: ExportToFile <ReportName>, <FileName>

Return Value: None

Parameters: <ReportName>

Type: string

<Filename>

Type: string

.txt Post processor format file

.csv Comma-delimited data file

.tab Tab-separated file

.dat Ansys plot data file

Example: Set oModule = oDesign.GetModule('ReportSetup')

```
oModule.ExportToFile('Plot 1','c:\report1.dat')
```

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	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<ReportName>	String	Name of the report
	<FileName>	String	Path and File Name

	Supported formats <table> <tr> <td>.txt</td><td>Post processor format file</td></tr> <tr> <td>.csv</td><td>Comma-delimited data file</td></tr> <tr> <td>.tab</td><td>Tab-separated file</td></tr> <tr> <td>.dat</td><td>Ansys plot data file</td></tr> <tr> <td>.rdat</td><td>Ansys report data file</td></tr> </table>	.txt	Post processor format file	.csv	Comma-delimited data file	.tab	Tab-separated file	.dat	Ansys plot data file	.rdat	Ansys report data file
.txt	Post processor format file										
.csv	Comma-delimited data file										
.tab	Tab-separated file										
.dat	Ansys plot data file										
.rdat	Ansys report data file										
Return Value	None										

Python Syntax	ExportToFile (<ReportName>, <FileName>)
Python Example	<pre>oModule.ExportToFile("Plot1", "c:\report1.dat")</pre>

ExportUniformPointsToFile

From a data table or plot, generates text format, comma delimited, tab delimited, .dat, or .rdat type output files with uniformly spaced point data.

UI Access	Right-click on a 2D or data table report name in the Project tree and select Export Data		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<ReportName>	String	The name of the report
	<FileName>	String	Path and file name, with appropriate extension defining type of export.
			.txt Post processor format file
			.csv Comma-delimited data file
			.tab Tab-separated file
			.dat Ansys plot data file
			.rdat Ansys report data file
	<start>	String	Start value, including units
<stop>	String	Stop value, including units	
<step>	String	Step value, including units	
Return Value	None		

Python Syntax	ExportToFile(<ReportName>, <FileName>, <start>, <stop>, <step>)
Python Example	oModule.ExportUniformPointsToFile ("Rectangular_ Plot0", "E:/testOpti5.csv", "0ms", "40ms", "0.1ms")

FFTONReport

Use: Perform an FFT on a selected report.

Command: **[product]>Results>Perform FFT on Report**

Syntax: FFTOnReport "<plotName>", <FFTWindowType>, "<function>"

Return Value: Creates a plot named FFT "PlotName"

Parameters: <PlotName>

Type: string

<FFTWindowType>

Type: string

Rectangular, Tri, Van Hann, Hamming, Blackman,Lanczos, Weber, Welch.

<function>

Type: string

<none>, ang_deg, ang_rad, arg, cang_deg, cang_rad, dB, dB1 normalize, dB20normalize, dBc, im, mag, normalize, re.

Example:

```
Dim oAnsoftApp
```

```
Dim oDesktop
```

```
Dim oProject
```

```
Dim oDesign
```

```
Dim oEditor
```

```
Dim oModule
```

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



```

Set oDesktop = oAnsoftApp.GetAppDesktop()
oDesktop.RestoreWindow
Set oProject = oDesktop.SetActiveProject("OptimTee")
Set oDesign = oProject.SetActiveDesign("TeeModel")
Set oModule = oDesign.GetModule("Solutions")
oModule.FFTOnReport "XY Plot 1", "Rectangular", "dB"

```

For Q3D Extractor the details for FFTOnReports are as follows:

Use: Perform an FFT on a selected report.

Command: Q3D Extractor or 2D Extractor>Results>Perform FFT on Report

Syntax: "<plotName>", <FFTWindowType>, "<function>"

Return Value: Creates a plot named FFT "Plot Name".

Parameters: <PlotName>

Type: string

<FFTWindowType>

Type: string

Rectangular, Tri, Van Hann, Hamming, Blackman, Lanczos, Weber, Welch.

<Function>

Type: string

<none>, ang_deg, ang_rad, arg, cang_deg, cang_rad, dB, dB1 normalize, dB20normalize, dBc, im, mag, normalize, re.

Example:

```
oModule.FFTOnReport "XY Plot 1", "Rectangular", "dB"
```

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	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<reportTypeArg> <displayTypeArg> <solutionNameArg> <simValueCtxtArg> categoryName_ array		Report type name as input parameter Display type name as input parameter Solution name as input parameter A context name, or array of strings that encode the contexts(I) Output parameter for returning category names
Return Value	Array of text strings		

Python Syntax	GetAllCategories(reportTypeArg,displayTypeArg, solutionNameArg, simValueCtxtArg, categoryName_array)
Python Example	<pre>categories= oModule.GetAllCategories("Far Fields", "Rectangular Plot", "Setup1 : LastAdaptive", "Infinite Sphere1")</pre>

GetAllReportNames

Gets the names of existing reports in a design

UI Access	NA		
Parameters	<i>Name</i> None	<i>Type</i>	<i>Description</i>
Return Value	Array of report names		

Python Syntax	GetAllReportNames()
Python Example	<pre>oReportModule = oDesign.GetModule("ReportSetup") names = oReportModule.GetAllReportNames() for name in names: AddInfoMessage(str(name))</pre>

GetAllQuantities

Gets all available quantity names in category, returned as an array of text strings.

UI Access	NA		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<reportTypeArg>		Report type name as input parameter
	<displayTypeArg>		Display type name as input parameter
	<solutionNameArg>		Solution name as input parameter
	<simValueCtxtArg>		A context name, or array of string that encoded the contexts(l).
	<categoryNameArg>		A category name as input parameter. a category name returned in GetAllCategories() or "Variables", or "Output Variables"
	<quantityName_array>		Output parameter for returning quantity names found in a category.
Return Value	Array of text strings		

Python Syntax	GetAllQuantities(reportTypeArg, displayTypeArg, solutionNameArg, simValueCtxtArg, categoryNameArg, quantityName_array)
Python Example	<pre>quantities= oModule.GetAllQuantities ("Far Fields", "Rectangular Plot", "Setup1 : LastAdaptive", "Infinite Sphere1", "Gain")</pre>

GetAvailableDisplayTypes

All supported display types in report type as an array of text strings.

UI Access	NA		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	< reportTypeArg >		Report type name as input parameter
	<displayType_array>		Output parameter for returning display types

Return Value	Array of text strings

Python Syntax	GetAvailableDisplayTypes(reportTypeArg,displayType_array)
Python Example	<pre>displayTypes = oModule .GetAvailableDisplayTypes("Far Fields")</pre>

GetAvailableReportTypes

Get all available report types in the current Design as an array of text strings

UI Access	NA		
Parameters	<i>Name</i> reportType_array	<i>Type</i> Array	<i>Description</i> Output parameter for returning report types
Return Value	Array of text strings		

Python Syntax	GetAvailableReportTypes([out, retval] VARIANT* reportType_array)
Python Example	<pre>reportTypes = oModule.GetAvailableReportTypes()</pre>

GetAvailableSolutions

Get all available solutions in report type as an array of text strings

UI Access	NA		
Parameters	<i>Name</i> <reportTypeArg> solution_array	<i>Type</i>	<i>Description</i> Report type name as input parameter. Output parameter for returning solution names.
Return Value	Array of text strings		

Python Syntax	<code>GetAvailableSolutions(reportTypeArg,solution_array)</code>
Python Example	<code>solutions=oModule.GetAvailableSolutions("Far Fields")</code>

GetDataExpressions

Returns data expressions.

UI Access	NA
Parameters	NA
Return Value	Array of text strings

Python Syntax	<code>GetDataExpressions()</code>
Python Example	<code>expressions = oModule.GetDataExpressions()</code>

GetDataUnits

Returns text string containing units.

UI Access	NA		
Parameters	<i>Name</i> <expressionString>	<i>Type</i> String	<i>Description</i> Can be returned from GetDataExpressions()
Return Value	Text string of units; empty if no units		

Python Syntax	<code>GetDataUnits(<expressionString>)</code>
Python Example	<code>oModule.GetDataUnits(expressions)</code>

GetDesignVariableNames

Returns array of strings containing design variable names.

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.

UI Access	NA		
Parameters	<i>Name</i> <varName>	<i>Type</i> String	<i>Description</i> 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.

UI Access	NA		
Parameters	<i>Name</i> <varName>	<i>Type</i> String	<i>Description</i> 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.

UI Access	NA
Parameters	NA
Return Value	String containing Variation Key

Python Syntax	GetDesignVariationKey()
Python Example	<pre>key = GetDesignVariationKey()</pre>

GetDisplayType

Get the display type of a report

UI Access	NA		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>

	<ReportName> String Report name
Return Value	Report <displaytype> of a report. <DisplayType> <i>Type: <string></i> If ReportType is "Standard", then returns one of the following: "Rectangular Plot", "Polar Plot", "Radiation Pattern", "Data Table", "3D Rectangular Plot", or "3D Polar Plot", "Rectangular Stacked Plot", "Bode Plot", "Rectangular Nyquist Plot", "Nyquist Plot", "Digital Plot".

Python Syntax	GetDisplayType("<reportName>")
Python Example	GetDisplayType("Standard")

GetImagDataValues

Returns array of imaginary data values.

UI Access	NA		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<expressionString>	String	Can be returned from GetDataExpressions()
	<siValue>	Boolean	True to return SI-value; False to return with units returned in GetSweepUnits() .
Return Value	Array of doubles		

Python Syntax	GetImagDataValues(<expressionString>,<siValue>)
Python Example	imaginaryvalues = oModule.GetImagDataValues('expression',1)

GetPerQuantityPrimarySweepValues

Returns per quantity primary sweep values.

UI Access	NA		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<expressionString>	String	Can be returned from GetDataExpressions() .
	<siValue>	Boolean	True to return SI-value; False to return by GetSweepUnits() .
Return Value	Array of doubles if IsPerQuantityPrimarySweep() returned True; error if returned False		

Python Syntax	GetPerQuantitySweepValues(<expressionString>, <siValue>)
Python Example	<pre>sweepvalues = oModule.GetPerQuantitySweepValues ('0.111,0.201,0.345,0.231', 1)</pre>

GetPropertyValue

Gets the value of a single property belonging to a specific serverName and PropTab. This function is available with the Project, Design or Editor objects, including definition editors.

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

UI Access	NA		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<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") CustomTab ("Intrinsic Variables") Quantities ("Quantities") Signals ("Signals")
	<propServer> String <propName> String	An object identifier, generally returned from another script method, such as Comp-Inst@R;2;3 Name of the property.
Return Value	String, the value of the property.	

Python Syntax	GetPropertyValue (<propTab>, <serverName>, <propName>)
Python Example	<pre>selectionArray = oEditor.GetSelections() for k in selectionArray: val = oEditor.GetPropertyValue("PassedParameterTab", k, "R") ...</pre>

Python Syntax	GetPropertyValue ([<array>])
Python Example	<pre>oModule = oDesign.GetModule("ReportSetup") oModule.GetPropertyValue("Trace", "S Parameter Plot 1:dB(S(1,1))", "Y Axis")</pre>

Example: (Query frequency and excitation)

```

freq = oDesign.GetPropertyValue("General", "AnalysisSetup:Setup1", "Adaptive Freq")
sigType = oDesign.GetPropertyValue("Q3D", "BoundarySetup:via_1", "Type")
sourceType = oDesign.GetPropertyValue("Q3D", "BoundarySetup:Source1", "Type")

```

Example: (Query Trace values in a specified report)

```

oModule = oDesign.GetModule("ReportSetup")

oModule.GetPropertyValue("Trace", "S Parameter Plot 1:dB(S(1,1))", "Y
Axis")

```

GetPropertyValue [ReportSetup]

Find the properties for a Report.

Command: Select a Report and find properties.

Syntax: `GetPropertyValue(tabName, serverName, propName)`

Return Value: Report properties.

Parameters: Varies, depending on the properties associated with the select report command. To find the syntax, it may help to record script creating a report from the user interface to get some sample of the parameter format.

Python Syntax	<code>GetPropertyValue ([<array>])</code>
Python Example	<pre> oModule = oDesign.GetModule("ReportSetup") oModule.GetPropertyValue("Trace", "S Parameter Plot 1:dB(S(1,1))", "Y Axis") </pre>

GetRealDataValues

Returns array of real data values.

UI Access	NA		
Parameters	<i>Name</i> <expressionString>	<i>Type</i> String	<i>Description</i> Can be returned from GetDataExpressions() True to return SI-value; False to return with units returned in GetSweepUnits() .
	<siValue>	Boolean	
Return Value	Array of doubles		

Python Syntax	<code>GetRealDataValues(<expressionString>,<siValue>)</code>
Python Example	<pre>realvalues = oModule.GetRealDataValues('expression',1)</pre>

GetReportTraceNames

Use: Gets the names of existing trace names in a plot.

Syntax: `GetReportTraceNames("<plot name>")`

Return Value: Array of trace names for a named plot.

Parameters: <PlotName>

Type: <string>

Name of plot.

Python Syntax	<code>GetReportTraceNames()</code>
Python Example	<pre>oReportModule = oDesign.GetModule("ReportSetup") names = oReportModule.GetReportTraceNames("SParameter Plot 1") for name in names: AddInfoMessage(str(name))</pre>

GetSolutionContexts

Get all available solution context names in a solution as an array of text strings.

UI Access	NA		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<reportTypeArg>		Report type name as input parameter.
	<displayTypeArg>		Display type name as input parameter.
	<solutionNameArg>		Solution name as input parameter.
	<contextName_ array>		Output parameter for returning context names.
Return Value	Array of text strings		

Python Syntax	<code>GetSolutionContexts(reportTypeArg,displayTypeArg, solutionNameArg, contextName_array)</code>
Python Example	<pre> contexts= oModule.GetSolutionContexts ("Far Fields", "Rectangular Plot", "Setup1: LastAdaptive") </pre>

GetSolutionDataPerVariation

Obtains solution data for a given report type and solution. You must have already run a simulation.

UI Access	N/A		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<reportTypeArg>	String	Report type name as input parameter.
	<solutionNameArg>	String	Solution name as input parameter.
	<simValueCtxtArg>	Structured Array	Same as ContextArray values created in the relevant CreateReport script.
	<familiesArg>	Array of Strings	Same as FamiliesArray values created in the relevant CreateReport script.
	<expressionArg>	String or Array of Strings	Text string or array of text strings; valid expression, may validate it as the data-table Y-component.
Return Value	ARRAY of ISolutionDataResultComInterface objects, containing: • GetSweepNames() • GetSweepUnits() • GetSweepValues() • IsPerQuantityPrimarySweep() • GetDataExpressions() – should be the same as <expressionArg> • GetPerQuantityPrimarySweepValues()		

- [IsDataComplex\(\)](#)
- [GetDataUnits\(\)](#)
- [GetRealDataValues\(\)](#)
- [GetImagDataValues\(\)](#)
- [ReleaseData\(\)](#)
- [GetDesignVariableNames\(\)](#)
- [GetDesignVariableUnits\(\)](#)
- [GetDesignVariableValue\(\)](#)
- [GetDesignVariationKey\(\)](#)

Note:

- This command is *not* recordable from the UI, but its parameters are similar to [CreateReport](#), so you may record a CreateReport script to get the parameter values.
- For the returned ISolutionDataResultComInterface object, some of its functions have an optional boolean parameter: SValue. SValue defaults to True. When the pass in value is True, return data values will be in Standard International values; when False, return data values will be in the current units.

Example: Freq Sweep with [1GHz, 2GHz,3GHz], GetSweepUnits("Freq") return "GHz"; GetSweepValues("Freq", True) return [1000000,2000000,3000000]; GetSweepValues("Freq", False) return [1,2,3].

Python Syntax	GetSolutionDataPerVariation(reportTypeArg, solutionNameArg, simValueCtxtArg, familiesArg, expressionArg)
Python Example	<pre>oModule = oDesign.GetModule("ReportSetup") arr = oModule.GetSolutionDataPerVariation('Modal Solution Data', 'Setup1 : Sweep', ['Domain:=', 'Sweep'], ['Freq:=', ['All'], 'offset:=', ['All']], ['S (Port1,Port1)', 'dB(S(Port1,Port3))'])</pre>

GetSweepNames

Returns array of text strings containing primary sweep name(s).

UI Access	NA
Parameters	NA

Return Value	Array of text strings

Python Syntax	GetSweepNames()
Python Example	<pre>sweepnames = oModule.GetSweepNames()</pre>

GetSweepUnits

Returns text string containing units.

UI Access	NA		
Parameters	<i>Name</i> <sweepName>	<i>Type</i> String	<i>Description</i> 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.

UI Access	NA		
Parameters	<i>Name</i> <sweepName> <siValue>	<i>Type</i> String Boolean	<i>Description</i> Primary sweep name 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

Use: Groups curves in a Stacked Plot automatically based on a curve grouping strategy.

Command: **Group Curves By...**

Syntax: GroupPlotCurvesByGroupingStrategy "<reportName>", "<curve_grouping_strategy>")

Return Value: None

Parameters: <ReportName>

Type: <string>

Report name.

<curve_grouping_strategy>

Type: <string>

Single, By Trace, By Units

Python Syntax	GroupPlotCurvesByGroupingStrategy "<reportName>", "<curve_grouping_strategy>")
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 Export Data .		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<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.

UI Access	NA		
Parameters	<i>Name</i> <expressionString>	<i>Type</i> String	<i>Description</i> Can be returned from GetDataExpressions().
Return Value	Boolean (True if expression is Complex data; False if not)		

Python Syntax	IsDataComplex(<expressionString>)
Python Example	oModule.IsDataComplex(' .001, .234, .455, .434')

IsPerQuantityPrimarySweep

Returns whether data expressions have different primary sweep values.

UI Access	NA
Parameters	NA
Return Value	Boolean (True if data expressions have different primary sweep values)

Python Syntax	IsPerQuantityPrimarySweep()
Python Example	<code>oModule.IsPerQuantityPrimarySweep()</code>

MovePlotCurvesToGroup

Use: 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.

Command: Move Plot Curves to Group

Syntax: `MovePlotCurvesToGroup("<reportName>",["<curve1-name>", "<curve2-name>",...], "<stack-name>")`

Return Value: None

Parameters: <ReportName>

Type: <string>

Report name.

`<curveN_name>`

Type: <string>

Trace name.

`<"stack-name">`

Type: <string>

Stack name.

Python Syntax	<code>oModule.MovePlotCurvesToGroup("<report-name>", ["<curve1-name>", "<curve2-name>",...], "<stack-name>")</code>
Python Example	<code>oModule.MovePlotCurvesToGroup("XY Stacked Plot 1", ["R2.V : TR", "R2.I : TR"], "Stack 2")</code>

MovePlotCurvesToNewGroup

Use: In a Stacked Plot move curve(s) from its stack(s) to anew stack. Here term ‘group’ is synonymous to ‘stack’ in the context of Cartesian stacked plot.

Command: Move Plot Curves to New Group

Syntax: MovePlotCurvestoNewGroup("<reportName>",["<curve1-name>", "<curve2-name>",...], "<stack-name>")

Return Value: None

Parameters: <ReportName>

Type: <string>

Report name.

<curveN_name>

Type: <string>

Trace name.

<"stack-name">

Type: <string>

Stack name.

Python Syntax	<code>oModule.MovePlotCurvesToNewGroup("<report-name>", ["<curve1-name>", "<curve2-name>",...], "<stack-name>")</code>
Python Example	<code>oModule.MovePlotCurvesToNewGroup("XY Stacked Plot 1", ["R2.V : TR", "R2.I : TR"], "Stack 2")</code>

OpenWindowForAllReports

To open a window for each report in the design.

UI Access	NA		
Parameters	<i>Name</i> None	<i>Type</i>	<i>Description</i>
Return Value	None		

Python Syntax	OpenWindowForAllReports()
Python Example	<code>Module.OpenWindowForAllReports()</code>

PasteReports

Paste copied reports to results in the current project.

UI Access	Edit>Paste		
Parameters	<i>Name</i> None	<i>Type</i>	<i>Description</i>
Return Value	None		

Python Syntax	PasteReports ()
Python Example	<code>oModule.PasteReports()</code>

PasteTraces

To paste copied traces to a named plot.

UI Access	Paste		
Parameters	<i>Name</i> <plotName>	<i>Type</i> String	<i>Description</i> Name of plot
Return Value	None		

Python Syntax	PasteTraces ("<plotName>")
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.

UI Access	NA
Parameters	NA
Return Value	NA

Python Syntax	ReleaseData()
Python Example	<code>oModule.ReleaseData()</code>

RemoveReport

Deletes an existing report.

UI Access	Right-click the report to delete in the project tree, and then click Delete on the shortcut menu.		
Parameters	<i>Name</i> <ReportName>	<i>Type</i> String	<i>Description</i> The name of the report to delete
Return Value	None		

Python Syntax	RemoveReport (<ReportName>)
Python Example	<code>oDesign.RemoveReport("XY Plot 1")</code>

RenameReport

Renames an existing report.

UI Access	Select a report on the Project tree, right-click and select Rename		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<OldReportName>	String	Old Report Name
	<NewReportName>	String	New Report Name
Return Value	None		

Python Syntax	RenameReport (<OldReportName>, <NewReportName>)		
Python Example	oModule.RenameReport ("XY Plot1", "Reflection")		

RenameTrace

To rename a trace in a plot

UI Access	NA		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<plotName>	String	Name of plot.
	<traceID>	String	Name of Trace
	<newName>	String	New trace name.
Return Value	None		

Python Syntax	RenameTrace ("<plotName>" "<traceID>" "<newName>")		
Python Example	oModule.RenameTrace ("XY Plot1", "dB (S (WavePort1, WavePort1)) 1", _ "Port1dbS")		

ResetPlotSettings

Resets plot settings to defaults.

Command: Right-click on plot **Edit>Reset Plot Settings**

Syntax: ResetPlotSettings ("PlotName")

Return Value: None

Parameters: <PlotName>

Type: <string>

Name of plot.

Python Syntax	<code>ResetPlotSettings("<PlotName>")</code>
Python Example	<pre>oModule = oDesign.GetModule("ReportSetup") oModule.ResetPlotSettings("Differential S-parameters")</pre>

SavePlotSettingsAsDefault

Saves report plot settings as default.

UI Access	Report Templates>Save Settings as Default		
Parameters	<i>Name</i> <Plot Name>	<i>Type</i> String	<i>Description</i> Name of plot to use for plot defaults.
Return Value	A new report		

Python Syntax	<code>SavePlotSettingsAsDefault("<PlotName>")</code>
Python Example	<code>oModule.SavePlotSettingsAsDefault ("XY Plot1")</code>

UnGroupPlotCurvesInGroup

Use: From a Stacked Plot ungroup curves in a stack, that is, curves in that stack are ungrouped and are shown in their own stacks.

Command: **UnGroup Plot Curves in ...**

Syntax: `oModule.UngroupPlotCurvesInGroup("<report-name>", "<stack-name>")`

Return Value: None

Parameters: <ReportName>

Type: <string>

Report name.

<"stack-name">

Type: <string>

Stack name.

Python Syntax	<code>oModule.MovePlotCurvesToGroup("<report-name>", ["<curve1-name>", "<curve2-name>",...], "<stack-name>")</code>
Python Example	<code>oModule.UngroupPlotCurvesInGroup("S Parameter Plot 3", "Stack 1")</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	Report dialog, Apply Traces button		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<ReportName>	String	Name of Report.
	<SolutionName>	String	Name of the solution as listed in the Modify Report dialog box. For example: "Setup1 : Last Adaptive"
	<ContextArray>	Array of Strings	Context for which the expression is being evaluated. This can be an empty string if there is no context.
	<Domain>	Array	Array("Domain:=", <DomainType>) <DomainType> ex. "Sweep" or "Time"
	<Context>	Array	Array("Context:=", <GeometryType>) <GeometryType> ex. "Infinite Spheren", "Spheren", "Polylinen"
	<FamiliesArray>	Array of strings	Contains sweep definitions for the report.

	<ValueArray> Array <ReportDataArray> Array of Strings <ReportQuantityArray> Array Array() Empty Array	Array("<VariableName>:= ", <ValueArray>) Array("All") or Array("Value1", "Value2", ... "Valuen") examples of <VariableName> "Freq", "Theta", "Distance" This array contains the report quantity and X, Y, and (Z) axis definitions. Array("X Component:=", <VariableName>, "Y Component:=", <VariableName> <ReportQuantityArray>) ex. Array("dB(S(Port1, Port1))") Denotes the end of the UpdateTraces Command
Return Value	None	

Python Syntax	UpdateTraces ("<plotName>" ["<TraceDef>"] [])
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"]], [</pre>

	<pre>"Spectrum:=", ["All"]], ["X Component:=", "Spectrum", "Y Component:=", ["mag (NEG1.VAL) "]], [])</pre>

UpdateTracesContextAndSweeps

Use this command to edit sweeps and context of multiple traces without affecting their component expressions.

UI Access	Modify Report with multiple traces selected.		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<ReportName>	String	Name of Report.
	Array(<traceIDs>)	String	Name of Trace
	<traceID>		
	<SolutionName>	String	Name of the solution as listed in the Modify Report dialog box.
			For example: "Setup1 : Last Adaptive"
	<ContextArray>	String	Context for which the expression is being evaluated. This can be an empty string if there is no context.
			ex. "Sweep" or "Time"
	Array<pointSet>	String	Point set for the selected traces, for example, X and Y values for the plot.
Return Value	None		

Python Syntax	UpdateTracesContextandSweeps()
Python	oModule.UpdateTracesContextAndSweeps_

Example	<pre>("Active S Parameter Quick Report", _ ["dB(ActiveS(Port1:1))", "dB(ActiveS(Port2:1))"], _ "Setup1 : Sweep1", [], _ ["Freq:=", _ ["9GHz", "9.05GHz", "9.1GHz", "9.15GHz", "9.2GHz", _ "9.25GHz", "9.3GHz", "9.35GHz", _ "9.4GHz", "9.45GHz", "9.5GHz", "9.55GHz", _ "9.6GHz", "9.65GHz", "9.7GHz", _ "9.75GHz", "9.8GHz", "9.85GHz", "9.9GHz", "9.95GHz", "10GHz"], _ "offset:=", ["All"]])</pre>

UpdateAllReports

Updates all reports in the **Results** branch in the project tree.

UI Access	Twin Builder>Results>Update All Reports		
Parameters	<i>Name</i> None	<i>Type</i>	<i>Description</i>
Return Value	None		

Python Syntax	UpdateAllReports ()
Python Example	<pre>oModule = oDesign.GetModule("ReportSetup") oModule.UpdateAllReports()</pre>

UpdateReports

Updates the specified reports in the **Results** branch in the project tree.

UI Access	NA		
Parameters	<i>Name</i> <plotname>	<i>Type</i> String	<i>Description</i> Name of plot
Return Value	None		

Python Syntax	UpdateReports(["<plotname>"])		
Python Example	oModule.UpdateReports (["XY Plot 1", "XY Plot 4"])		

Output Variable Script Commands [Report]

Scripting commands for Output Variables are documented in [Output Variable Script Commands](#).

18 - Optometrics Module Script Commands

```
oModule = oDesign.GetModule("Optometrics")
```

```
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 Twin Builder 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 Twin Builder solution.

<Calc>

Type: <string>

An expression that is composed of a basic solution quantity and an output variable.

<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.

General Commands Recognized by the Optimetrics Module

CopySetup [Optimetrics]

Copy the Optimetrics setup

UI Access	NA		
Parameters	<i>Name</i> <SetupName>	<i>Type</i> String	<i>Description</i> Name of the setup
Return Value	None		

Python Syntax	CopySetup (<SetupName>)
Python Example	<code>oModule.CopySetup ("OptimizationSetup1")</code>

DeleteSetups [Optimetrics]

Deletes the specified Optimetrics setups

UI Access	Right-click the setup in the project tree, and then click Delete on the shortcut menu		
Parameters	<i>Name</i> <NameArray>	<i>Type</i> Array of Strings	<i>Description</i> 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	<i>Name</i> <ParametricSetupName>	<i>Type</i> String	<i>Description</i> Name of the Setup
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	<i>Name</i> <SetupName>	<i>Type</i> String	<i>Description</i> Must be one of existing DesignExplorer setup names
	<FileName>	String	Must be a valid file path and name
Return Value	None		

Python Syntax	ExportDXConfigFile (<SetupName>, <FileName>)
Python Example	<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	<i>Name</i> <SetupName>	<i>Type</i> String	<i>Description</i> 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

	[profileNum]	String	extension of csv, tab, dat, or txt 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	<i>Name</i> <SetupName>	<i>Type</i> String	<i>Description</i> 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	oModule.ExportOptimetricsResult (		

Example	<code>"StatisticalSetup1", "c:/exportdir/test.csv", false)</code>

ExportParametricResults

Export existing Parametric results.

UI Access	Right click on the desired Parametric setup in the project tree and choose View Analysis Result... On the Post Analysis Display dialog box, click the Result tab, then select Table view, and click on the Export button.		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<SetupName>	String	Must be one of the existing Parametric setup names
	<FileName>	String	Must be a valid file path and name with extension of csv, tab, dat or txt
	<bOutputUnits>	Boolean	If set to true, values will be printed with units
Return Value	None		

Python Syntax	<code>ExportParametricResults (<SetupName>, <FileName>, <bOutputUnits>)</code>
Python Example	<code>oModule.ExportParametricResults ("ParametricSetup1", "c:/exportdir/test.csv", False)</code>

GetSetupNames [Optimetrics]

Gets a list of Optimetrics setup names

UI Access	NA		
Parameters	<i>Name</i> None	<i>Type</i>	<i>Description</i>
Return Value	IAnsoftCollectionObj – a collection of Optimetrics setup names		

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

Python Example	<pre>oModule = oDesign.GetModule("Optimetrics") setupNames = oModule.GetSetupNames()</pre>

GetSetupNamesByType [Optimetrics]

Gets a list of Optimetrics setup names by type.

UI Access	NA		
Parameters	<i>Name</i> <Optimetrics type>	<i>Type</i> String	<i>Description</i> 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 [Optimetrics]

Import an Optimetric setup from a file.

UI Access	NA		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<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	<i>Name</i> <SetupName>	<i>Type</i> String	<i>Description</i> 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	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<OldName>	String	The name that needs to be replaced
	<NewName>	String	Replacement name
Return Value	None		

Python Syntax	RenameSetup (<OldName> <NewName>)		
Python Example	oModule.RenameSetup ("OptimizationSetup1" "MyOptimization")		

SolveAllSetup

Solves all Optimetrics setups

UI Access	Right-click on Optimetrics in Project Manager and select Analyze>All from context menu		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	None		
Return Value	None		

Python Syntax	SolveAllSetup()		
Python Example	oModule.SolveAllSetup()		

SolveSetup [Optimetrics]

Solves the specified Optimetrics setup.

UI Access	Right-click the setup in the project tree, and then click Analyze on the shortcut menu.		

Parameters	<i>Name</i> <SetupName>	<i>Type</i> String	<i>Description</i> Name of the setup to be solved
Return Value	None		

Python Syntax	SolveSetup (<SetupName>)		
Python Example	oModule.SolveSetup ("OptimizationSetup1")		

Parametric Script Commands

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	<i>Name</i> <SetupName>	<i>Type</i> String	<i>Description</i> Name of the Setup
Return Value	None		

Python Syntax	EditSetup (<SetupName>, <ParametricParams>)		
Python Example	See EditSetup [Optimization]		

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.

Example:

```
oModule.GenerateVariationData "ParametricSetup1"
```

InsertSetup [Parametric]

Inserts a new setup.

UI Access	Right-click the Optimetrics folder in the project tree, and then click Add> Parametric on the shortcut menu.		
Parameters	<i>Name</i> <Parametric Params>	<i>Type</i> Array	<i>Description</i> Array("NAME:<SetupName>", "SaveFields:=", <SaveField>, <StartingPoint>, "Sim. Setups:=", <SimSetups>, <SweepDefs>, <SweepOps>, Array("NAME:Goals", Array("NAME:Goal", <OptiGoalSpec>), ... Array("NAME:Goal", <OptiGoalSpec>))
	<SetupName>	String	Name of the parametric setup.
	<SimSetups>	Array of Strings	An array of Twin Builder solution setup names.
	<SweepDefs>	Array	Array("NAME:Sweeps", Array("NAME:SweepDefinition", "Variable:=", <VarName>, "Data:=", <SweepData>, "Synchronize:=", <SyncNum>), ... Array("NAME:SweepDefinition", "Variable:=", <VarName>, "Data:=", <SweepData>, "Synchronize:=", <SyncNum>))

	<SweepData>	String	"<SweepType>, <StartV>, <StopV>, <StepV>"
	<SweepType>	String	The type of sweep data.
	<SyncNum>	Integer	SweepDatas with the same value are synchronized.
	<SweepOps>		Array("NAME:Sweep Operations", "<OpType>:=", Array(<VarValue>, ..., <VarValue>), ... <OpType>:=, Array(<VarValue>, ..., <VarValue>))
	<OpType>	String	The sweep operation type.
Return Value	None		

Python Syntax	InsertSetup ("OptiParametric", <ParametricParams>)
Python Example	<pre>oModule.InsertSetup("OptiParametric", ["NAME:ParametricSetup1", _ "SaveFields:=", true, _ ["NAME:StartingPoint"], _ "Sim. Setups:=", ["Setup1"], _ ["NAME:Sweeps", _ ["NAME:SweepDefinition", _ "Variable:=", "\$width", _ "Data:=", "LIN 12mm 17mm 2.5mm", _ "OffsetF1:=", false, _ "Synchronize:=", 0], ["NAME:SweepDefinition", _ "Variable:=", "\$length", _ "Data:=", "LIN 8mm 12mm 2mm", _ "OffsetF1:=", false, _</pre>

	<pre>"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"]]]]]))</pre>

Optimization Script Commands

EditSetup [Optimization]

Modifies an existing optimization setup

UI Access	Right-click the setup in the project tree, and then click Properties on the short-cut menu		
Parameters	<i>Name</i> <SetupName>	<i>Type</i> String	<i>Description</i> Name of the Setup

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;]"
```

	<pre>]], "Acceptable_Cost:=" , 0, "Noise:=", 0.0001, "UpdateDesign:=", False, "UpdateIteration:=", 5, "KeepReportAxis:=", True, "UpdateDesignWhenDone:=", True]) </pre>

InsertSetup [Optimization]

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	<i>Name</i> <OptimizationParams>	<i>Type</i> Array	<i>Description</i> Array("NAME:<SetupName>", "SaveFields:=", <SaveField>, <StartingPoint>, "Optimizer:=", <Optimizer>, "MaxIterations:=", <MaxIter>, "PriorPSetup:=", <PriorSetup>, "PreSolvePSetup:=", <Preceed>, <OptimizationVars>, <Constraint>,

		<pre> Array("NAME:Goals", Array ("NAME:Goal", <OptiGoalSpec>, <OptimizationGoalSpec>), ... Array("NAME:Goal", <OptiGoalSpec>, <OptimizationGoalSpec>)), "Acceptable_Cost:=", <AcceptableCost>, "Noise:=", <Noise>, "UpdateDesignWhenDone:=", <UpdateDesign" 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>
<OptimizationVars>	Array	
<MinStepV>	VarValue	The minimum step of the variable.
<MaxStepV>	VarValue	The maximum step of the variable.
<AcceptableCost>	Double	The acceptable cost value for the optimizer to stop.
<Noise>	Double	The noise of the design.
<UpdateDesign>	Boolean	Specifies whether or not to apply the optimal variation to the design after the optimization is done.
<OptimizationGoalSpec>	Array	<pre> "Condition:=", <OptimizationCond>, Array("NAME:GoalValue", "GoalValeType:=", <GoalValueType>, "Format:=", <GoalValueFormat>, "bG:=", </pre>

	<OptimizationCond> String <GoalValueType> String <GoalValueFormat> String <GoalValue> String	Array("v:=", <GoalValue>)), "Weight:=", <Weight>) Either "<=", "==" , or ">=" Either "Independent" or "Dependent" Either "Real/Imag" or "Mag/Ang". Value in string. Value can be a real number, complex number, or expres- sion.
Return Value	None	

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:=", _</pre>	

```
"1.3mm"]], _  
["NAME:LCS"], _  
["NAME:Goals", _  
["NAME:Goal", _  
"Solution:=", "Setup1 : LastAdaptive", _  
"Calculation:=", "reflect", _  
"Context:=", "", _  
["NAME:Ranges", _  
"Range:=", ["Var:=", "Freq", _  
"Type:=", "s", _  
"Start:=", "8GHz", "Stop:=", "8GHz"]], _  
"Condition:=", "<=", _  
["NAME:GoalValue", _  
"GoalValueType:=", "Independent", _  
"Format:=", "Real/Imag", _  
"bG:=", ["v:=", "[0.0001]"]], _  
"Weight:=", "[1]"]],  
"Acceptable_Cost:=", 0.0002, _  
"Noise:=", 0.0001, _  
"UpdateDesign:=", true, _  
"UpdateIteration:=", 5, _  
"KeepReportAxis:=", true, _  
"UpdateDesignWhenDone:=", true)
```

Sensitivity Script Commands

EditSetup [Sensitivity]

Modifies an existing sensitivity setup

UI Access	Right-click the setup in the project tree, and then click Properties on the short-cut menu		
Parameters	<i>Name</i> <SetupName>	<i>Type</i> String	<i>Description</i> 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",</pre>

```
[  
  
  "NAME:AnalysisStopOptions",  
  
  "StopForNumIteration:=", True,  
  "StopForElapsTime:=", False,  
  "StopForSlowImprovement:=", False,  
  
  "StopForGrdTolerance:=" , False,  
  
  "MaxNumIteration:=", 1001,  
  
  "MaxSolTimeInSec:=", 3600,  
  
  "RelGradientTolerance:=", 0,  
  
  "MinNumIteration:=", 10  
  
],  
  
"CostFuncNormType:=", "L2",  
  
"PriorPSetup:=", "",  
  
"PreSolvePSetup:=", True,  
  
[  
  
  "NAME:Variables"  
  
],  
  
[  
  
  "NAME:LCS"
```

```
],  
  
[  
  
"NAME:Goals",  
  
[  
  
"NAME:Goal",  
  
"ReportType:=", "Standard",  
  
"Solution:=", "TR3",  
  
[  
  
"NAME:SimValueContext",  
  
"SimValueContext:=", [1,0,2,0,False,False,-  
1,1,0,1,1,"",0,0]  
  
],  
  
"Calculation:=", "mag(DIFF1.VAL)",  
  
"Name:=", "DIFF1.VAL",  
  
[  
  
"NAME:Ranges",  
"Range:=", [  
"Var:=", "Time","Type:=", "a"]  
  
],
```

```
"Condition:=", "==" ,  
  
[  
  "NAME:GoalValue",  
  
  "GoalValueType:=", "Independent",  
  
  "Format:=", "Real/Imag",  
  
  "bG:=", ["v:=", "[1;]"]  
  
],  
  "Weight:=", "[1;]"  
  
]  
  
],  
  "Acceptable_Cost:=", 0,  
  "Noise:=", 0.0001,  
  "UpdateDesign:=", False,  
  "UpdateIteration:=", 5,  
  
  "KeepReportAxis:=", True,  
  
  "UpdateDesignWhenDone:=", True  
  
])
```

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.

	<i>Name</i>	<i>Type</i>	<i>Description</i>
Parameters	<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>)), "Master Goal:=". <MasterGoalID>, "MasterError:=", <MasterError>)
	<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 master goal. Index starts from zero.
	<MasterError>	Double	Error associated with the master goal.
Return Value	None		

Python Syntax	InsertSetup ("OptiSensitivity", <SensitivityParams>)
Python Example	<pre>oModule.InsertSetup("OptiSensitivity", _ ["NAME:SensitivitySetup1",_ "SaveFields:=", true,_</pre>

```

["NAME:StartingPoint"], _
"MaxIterations:=", 20, _
"PriorPSetup:=", "", _
"PreSolvePSetup:=", true, _
["NAME:Variables"], _
["NAME:LCS"], _
"NAME:Goals", _
["NAME:Goal", _
"Solution:=", "Setup1 : LastAdaptive", _
"Calculation:=", "returnloss", _
"Context:=", "", _
["NAME:Ranges", _
"Range:=", ["Var:=", "Freq", " _
Type:=", "s", _
"Start:=", "8GHz", "Stop:=", "8GHz"]]], _
["NAME:Goal", _
"Solution:=", "Setup1 : LastAdaptive", _
"Calculation:=", "reflect", _
"Context:=", "", _
["NAME:Ranges", _
"Range:=", ["Var:=", "Freq", _
"Type:=", "s", _
"Start:=", "8GHz", "Stop:=", "8GHz"]]]], _
"Master Goal:=", 1, _
"MasterError:=", 0.001])

```


Statistical Script Commands

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	<i>Name</i> <SetupName>	<i>Type</i> String	<i>Description</i> Name of the Setup
Return Value	None		

Python Syntax	EditSetup (<SetupName>, <StatisticalParams>)		
Python Example	See <code>EditSetup [Optimization]</code>		

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	<i>Name</i> <StatisticalParams>	<i>Type</i> Array	<i>Description</i> Array ("NAME:<SetupName>", "SaveFields:=", <SaveField>, <StartingPoint>, "MaxIterations:=", <MaxIter>, "PriorPSetup:=", <PriorSetup>, "PreSolvePSetup:=", <Preceed>, <StatisticalVars>,"

	<pre> Array("NAME:Goals", Array ("NAME:Goal", <OptiGoalSpec>), ..., Array ("NAME:Goal", <OptiGoalSpec>))) 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>))) </pre>
	<StatisticalVars> Array
	<DistType> String Distribution can be "Gaussian" or "Uniform".
	<Tolerance> VarValue The tolerance for the variable when distribution is Uniform
	<StdD> VarValue The standard deviation for the variable when distribution is Gaussian.
	<MinCutoff> Double The minimum cut-off for the variable when distribution is Gaussian.
	<MaxCutoff> Double The maximum cut-off for the variable when distribution is Gaussian.
Return Value	None

Python Syntax	InsertSetup ("OptiStatistical", <StatisticalParams>)
Python Example	oModule.InsertSetup (

```

"OptiStatistical", _
["NAME:StatisticalSetup1", _
"SaveFields:=", true, _
["NAME:StartingPoint"],_
"MaxIterations:=", 50,_
"PriorPSetup:=", "", _
["NAME:Variables"], _
["NAME:Goals", _
["NAME:Goal", _
"Solution:=", "Setup1 : LastAdaptive", _
"Calculation:=", "returnloss", _
"Context:=", "", _
["NAME:Ranges", _
"Range:=", ["Var:=", "Freq", _
"Type:=", "s",_
"Start:=", "8GHz", "Stop:=", "8GHz"]]],_
["NAME:Goal",_
"Solution:=", "Setup1 : LastAdaptive",_
"Calculation:=", "reflect",_
"Context:=", "", _
["NAME:Ranges",_
"Range:=", ["Var:=", "Freq", "Type:=",_
"s", "Start:=", "8GHz", "Stop:=", "8GHz"]]]])

```


19 - Schematic Scripting Commands

The Schematic scripting interface is a set of commands that match the data changing methods available in the UI of the Schematic editor and the Symbol Editor, plus some selection and query methods. Examples of the commands available through the scripting interface include: adding items, removing items, and modifying items on the Schematic. Identifying objects on a 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:Sub-ComponentIndex". 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.

Create Method List [Schematic Editor]

This section lists the scripts that are available in the Schematic editor scripting interface to create and change data. In the following 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. However, if your layout grid is expressed in mils, then 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:

X or Y position in meters = desired_position x (0.00254 / 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:

X or Y position in meters = 500 x (0.00254 / 20)

CreateArc [Schematic]

Create an arc

UI Access	Draw>Primitive>Arc

Parameters	Name <ArcData>	Type Array	Description 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
	<Attributes>	Array	"Id:=", int), // [Opt=New id] Id for this item Array("NAME:Attributes", _ "Page:=", int) _ // [Opt=1] Page number (one-based) [out,retval] string id)
Return Value	String Unique id		

Python Syntax	CreateArc()
Python Example	<pre>oEditor.CreateArc (["NAME:ArcData", _ "X:=", -0.004318, "Y:=", -0.00127, _ "Radius:=", 0.00297299377732279, _ "StartAng:=", 1.9195673303788, _ "EndAng:=", 3.32144615338227, _ "Id:=", 10], _</pre>

```
[ "NAME:Attributes", "Page:=", 1 ] )
```

CreateCircle [Schematic]

Create a circle

UI Access	Draw>Primitive>Circle		
Parameters	<i>Name</i> <CircleData>	<i>Type</i> Array	<i>Description</i> 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
	<Attributes>	Array	"Id:=", int), // [Opt=New id] Id for this item Array ("NAME:Attributes", _ "Page:=", int) _ // [Opt=1] Page number (one-based) [out,retval] string id)
Return Value	String Unique id		

Python	CreateCircle()

Syntax	
Python Example	<pre>oEditor.CreateCircle(["NAME:CircleData", _ "X:=", -0.004572, "Y:=", -0.000508, _ "Radius:=", 0.001778, "Id:=", 12], _ ["NAME:Attributes", "Page:=", 1])</pre>

CreateComponent [Schematic]

Create a component of a given type

UI Access	NA		
Parameters	<i>Name</i> <CompProps>	<i>Type</i> Array	<i>Description</i> Array("NAME:ComponentProps", _ "Name:=", string, _ // Component name (CAP, etc) "Id:=", string), // Component id 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)
	<Attributes>	Array	
Return Value	String Unique id		

Python Syntax	CreateComponent()

Python Example	<pre>oEditor.CreateComponent(["NAME:ComponentProps", "Name:=", _ "Simplorer Elements\Basic Elements\Circuit\PassiveElements:C", _ "Id:=", "1"], ["NAME:Attributes", "Page:=", _ 1, "X:=", -0.01524, "Y:=", 0.00508, "Angle:=", _ 0, "Flip:=", false])</pre>

CreateCurve [Schematic]

Creates a Bezier curve.

UI Access	Draw > Primitive > Curve		
Parameters	<i>Name</i> <curve data>	<i>Type</i> Array	<i>Description</i> ("NAME:CurveData", "Points:=", Array(), //Control Points of curve "LineWidth:=", 0, //Linewidth of curve "Color:=", 0, //Color "Id:=", 1)
	<attributes>	Array	Array("NAME:Attributes", _ "Page:=", int) _ // [Opt=1] Page number (one-based)
Return Value	<i>Name</i> <id>	<i>Type</i> String	<i>Description</i> Unique ID of this object. Use it to perform actions (such as select and move) on object.

Python Syntax	oEditor.CreateCurve([<CurveData>],[<attributes>])
Python Example	<pre>curveObjId = oEditor.CreateCurve(["NAME:CurveData", "Points:=", ["(-0.149860, 0.233680)", "(-0.139700,</pre>

	<pre>0.254000) "," (-0.121920, 0.238760) "," (-0.147320, 0.226060) "," (-0.139700, 0.218440) "," (-0.121920, 0.220980)"] , "LineWidth:=" , 0, "Color:=" , 0, "Id:=" , 7], ["NAME:Attributes", "Page:=" , 1])</pre>

CreateEllipse [Schematic]

Creates an ellipse.

UI Access	Draw > Primitive > Ellipse		
Parameters	<i>Name</i> <EllipseData>	<i>Type</i> Array	<i>Description</i> ("NAME:CurveData", "X1:=" , -0.14986, // x position of upper left "Y1:=" , 0.27432, // y position of upper left "X2:=" , -0.10668, // x position of lower right "Y2:=" , 0.254, // y position of lower right "LineWidth:=" , 0, // line width "BorderColor:=" , 0, // line color "Fill:=" , 0, // fill pattern "Color:=" , 0, // fill color "Id:=" , 17 // id)
	<attributes>	Array	Array("NAME:Attributes", _ "Page:=" , int) _ // [Opt=1] Page number (one-based)
Return Value	<i>Name</i> <id>	<i>Type</i> String	<i>Description</i> Unique ID of this object. Use it to perform actions (such as select and move) on object.

Python Syntax	oEditor.CreateEllipse([<EllipseData>],[<attributes>])
Python Example	<pre> ellipseObjID = oEditor.CreateEllipse(["NAME:EllipseData", "X1:=" , -0.14986, "Y1:=" , 0.27432, "X2:=" , -0.10668, "Y2:=" , 0.254, "LineWidth:=" , 0, "BorderColor:=" , 0, "Fill:=" , 0, "Color:=" , 0, "Id:=" , 17], ["NAME:Attributes", "Page:=" , 1]) </pre>

CreateGlobalPort [Schematic]

Create a global port and enable placement on the schematic

UI Access	Draw>Global Port		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<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

Python Syntax	CreateGlobalPort(["NAME:GlobalPortProps", "Name:=", <portname>, "Id:=", <id>], ["NAME:Attributes", "Page:=", <pagenumber>, "X:=", <xval>, "Y:=", <yval>, "Angle:=", <angle>, [or "Degrees:=", <degrees> ,] "Flip:=", <flip>])
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]

Create a ground

UI Access	Draw>Ground		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<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

Python Syntax	CreateGround()
Python Example	<pre>oEditor.CreateGround (["NAME:GroundProps", "Id:=", 8), ["NAME:Attributes", "Page:=", 1, "X:=", -0.04064, "Y:=", -0.00254, "Angle:=", 0, "Flip:=", false])</pre>

CreateImage [Schematic]

Create picture from a BMP or GIF file

UI Access	Draw>Primitive>Image		
Parameters	<i>Name</i> <ImageData>	<i>Type</i> Array	<i>Description</i> 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)
Return Value	String Unique id	

Python Syntax	CreateImage()
Python Example	<pre>oEditor.CreateImage (["NAME:ImageData", "X1:=", 0.00229684174875881, "Y1:=", _ 0.00100608189091032, "X2:=", 0.00890589813779289, "Y2:=", _ 0.00432967005619977, "File:=", _ "C:/Documents and Settings/user/Desktop/logo.gif", "DisplayBorder:=", _ false, "LineWidth:=", 0, "Color:=", 0, "Id:=", 22], ["NAME:Attributes", "Page:=", 1])</pre>

CreateLine [Schematic]

Create a line

UI Access	Draw>Primitive>Line		
Parameters	<i>Name</i> <LineData>	<i>Type</i> Array	<i>Description</i> Array("NAME:LineData" _ "Points:=", Array of points, _ "LineWidth:=", double, // the width of the line, in meters "Color:=", int, // the RBG value of the line color "Id:=", int), // [Opt=New id] Id for this item Array("NAME:Attributes", _ "Page:=", int) _ // [Opt=1] Page number (one-based) [out,retval] string id)
	<Attributes>	Array	
Return Value	String Unique id		

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]

Create a PagePort

UI Access	Draw>Page Connector		
Parameters	<i>Name</i> <PagePortProps> <i>Type</i> Array <i>Description</i> 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		

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>

CreatePlot [Schematic]

Creates a plot on the schematic.

UI Access	Ribbon > Reports > Select Type		
Parameters	<i>Name</i> <RectData>	<i>Type</i> Array	<i>Description</i> ("NAME:RectData", "X1:=", -0.14986, // x position of upper left "Y1:=", 0.27432, // y position of upper left "X2:=", -0.10668, // x position of lower right "Y2:=", 0.254, // y position of lower right "Id:=", 17 // id)
	<ReportInfo>	Array	//Create new empty report ("NAME:ReportInfo", Array("NAME:Rectangular Plot0", "PlotCreationType:=", "Empty", "ReportType:=", "Standard", "DisplayType:=", 1)) Or //use existing report with id 3, creates on schematic. ("NAME:ReportInfo", "ReportID:=", 3))
	<attributes>	Array	Array("NAME:Attributes", _ "Page:=", int) _ // [Opt=1] Page number (one-based)
Return Value	<i>Name</i> <id>	<i>Type</i> String	<i>Description</i> Unique ID of this object. Use it to perform actions (such as select and move) on object.

Python Syntax	oEditor.CreatePlot([<RectData>],[<ReportInfo>],[<attributes>])
Python Example	<pre> plotID = oEditor.CreatePlot(["NAME:RectData", "X1:=" , -0.1016, "Y1:=" , 0.23114, "X2:=" , -2.08166817117217E-17,</pre>

```
"Y2:=" , 0.30734,  
"Id:=" , 34,  
[  
    "NAME:ReportInfo",  
    [  
        "NAME:Rectangular Plot0",  
        "PlotCreationType:=" , "Empty",  
        "ReportType:=" , "Standard",  
        "DisplayType:=" , 1  
    ]  
]  
],  
[  
    "NAME:Attributes",  
    "Page:=" , 1  
]  
)
```

or

```
plotID = oEditor.CreatePlot(  
    [  
        "NAME:RectData",  
        "X1:=" , 0.04318,  
        "Y1:=" , 0.23114,  
        "X2:=" , 0.14478,  
        "Y2:=" , 0.30734,  
        "Id:=" , 36,  
        [  
            "NAME:ReportInfo",  
            "ReportID:=" , 3  
        ]  
    ]  
)
```

	<pre>], ["NAME:Attributes", "Page:=" , 1]) </pre>

CreateIPort [Schematic]

Create an interface port

UI Access	Draw>Interface Port		
Parameters	<i>Name</i> <IPortProps>	<i>Type</i> Array	<i>Description</i> Array("NAME:IPortProps"), "Name:=", string), // [Opt=default name] Name for this port
	<Attributes>	Array	"Id:=", int), // Port id number 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		

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]

Create a polygon

UI Access	Draw>Primitive>Polygon		
Parameters	<i>Name</i> <PolygonData>	<i>Type</i> Array	<i>Description</i> 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
	<Attributes>	Array	Array ("NAME:Attributes", _ "Page:=", int) _ // [Opt=1] Page number (one-based) [out,retval] string id)
Return Value	String Unique id		

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]

Create a rectangle

UI Access	Draw>Primitive>Rectangle		
Parameters	Name	Type	Description
	<RectData>	Array	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 Array("NAME:Attributes", _ "Page:=", int) _ // [Opt=1] Page number (one-based) [out,retval] string id)
Return Value	String Unique id

Python Syntax	CreateRectangle()
Python Example	<pre>oEditor.CreateRectangle (["NAME:RectData", "X1:=", -0.0755449856733525, "Y1:=", _ 0.0335755491881566, "X2:=", -0.0611831900668577, "Y2:=", _ 0.0254242597898758, "LineWidth:=", _ 0, "BorderColor:=", 0, "Fill:=", 0, "Color:=", 0, "Id:=", 31], _ ["NAME:Attributes", "Page:=", 1])</pre>

CreateText [Schematic]

Create text

UI Access	Draw>Primitive>Text		
Parameters	<i>Name</i> <TextData>	<i>Type</i> Array	<i>Description</i> 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 RGB 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 RGB 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 RGB value of the rectangle fill color Array ("NAME:Attributes", _ "Page:=", int) _ // [Opt=1] Page number (one-based) [out,retval] string id)
Return Value	String Unique id

Python Syntax	CreateText()
Python	oEditor.CreateText (

Example	<pre>["NAME:TextData", "X:=", -0.0764183381088826, "Y:=", _ 0.040174212034384, "Size:=", 12, "Angle:=", _ 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]

Create a wire

UI Access	Draw>Wire		
Parameters	<i>Name</i> <WireData>	<i>Type</i> Array	<i>Description</i> Array("NAME:WireData" _ "Name:=", string), // [Opt=default name] Name for this wire "Id:=", int), // Wire idnum "Points:=", Array/List of x,y points of the wire) // arrays with float/string values or array variables converted to strings are acceptable
	<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		

Python Syntax	CreateWire()
Python Example	<pre>oEditor.CreateWire(["NAME:WireData", "Name:=", "", "Id:=", 41, "Points:=", [_ "(-0.033020, 0.015240)", "(-0.017780, 0.015240)", "(-0.017780, 0.012700)"]], ["NAME:Attributes", "Page:=", 1])</pre>

General Method List [Schematic]

This section presents the general schematic script methods that are available.

Activate [Schematic]

Enable items previously disabled.

UI Access	Edit>Activate		
Parameters	<i>Name</i> Name	<i>Type</i> Type	<i>Description</i> Description
Return Value	Value		

Python Syntax	Activate(["NAME:Selections","Selections:=", [<components>]])
Python Example	oEditor.Activate(["NAME:Selections","Selections:=", ["CompInst@STEP;1;1"]])

AddPinPageConnectors [Schematic]

Adds Page Ports at unconnected pins of all selected components.

UI Access	Draw>Add at unconnected pins>Grounds		
Parameters	Name <PagePortName,NetName> Type Array	Description Array("NAME:Selections", _ // Page Port Name "Selections:=", _ // Net Name	
Return Value	None		

Python Syntax	AddPinPageConnectors ()		
Python Example	<pre>oEditor.AddPinPageconnectors["NAME:Selections", "Selections:=", [_ "CompInst@C;1;1"]]</pre>		

AddPinIPorts [Schematic]

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	<i>Name</i> <IntPortName,NetName>	<i>Type</i> Array	<i>Description</i> Array("NAME:Selections", _ // Interface Port Name "Selections:=", _ // Net Name
Return Value	None		

Python Syntax	AddPinIPorts ()		
Python Example	<pre>oEditor.AddPinIPorts ["NAME:Selections", "Selections:=", ["CompInst@R;2;37"]]</pre>		

AddPinPageConnectors [Schematic]

Adds Page Ports at unconnected pins of all selected components

UI Access	Draw>Add at unconnected pins>Grounds		
Parameters	<i>Name</i> Name	<i>Type</i> Type	<i>Description</i> Description
Return Value	None		

Python Syntax	AddPinPageConnectors ()		
Python Example	<pre>oEditor.AddPinPageconnectors ["NAME:Selections", "Selections:=", [_ "CompInst@C;1;1"]]</pre>		

AlignHorizontal [Schematic]

Align items horizontally

UI Access	Draw>Align Horizontal		
Parameters	<i>Name</i> <Selections>	<i>Type</i> Array	<i>Description</i> 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
Return Value	Value

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]

Align items vertically

UI Access	Draw>Align Vertical		
Parameters	<i>Name</i> <Selections>	<i>Type</i> Array	<i>Description</i> 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		

Python	AlignVertical()

Syntax	
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]

Bring the selected object to the front of the view

UI Access	Draw>Bring to Front		
Parameters	<i>Name</i> <Selections>	<i>Type</i> Array	<i>Description</i> ["NAME: Selections", "Selections:=", [<Selected Components>]] <Selected Components> CompInst@DefName; ID, schematic ID , CompInst@DefName; ID, schematic ID CompInst@R;1,2
Return Value	Value		

Python Syntax	BringToFront()
Python Example	<pre>oEditor.BringToFront (["NAME:Selections", "Selections:=", ["CompInst@R;2;37"]])</pre>

ChangePinVisibility [Schematic]

Changes visibility for named pin of specified component.

UI Access	(Context Menu) > Pin Visibility > <PinName>		
Parameters	<i>Name</i> <compID> <PinName> <visible>	<i>Type</i> String String Boolean	<i>Description</i> Component identifier The name of pin to modify True to make visible, false to hide
Return Value	Boolean True on success		

Python Syntax	ChangePinVisibility(<compID>, <pinName>, <visible>)		
Python Example	<pre>oEditor.ChangePinVisibility ("CompInst@R;2;37", "N1", FALSE)</pre>		

CloseEditor [Schematic]

Close an Editor

UI Access	NA		
Parameters	<i>Name</i> None	<i>Type</i>	<i>Description</i>
Return Value	None		

Python Syntax	CloseEditor()		
Python Example	<pre>oDefinitionEditor.CloseEditor()</pre>		

Copy [Schematic]

Copy items for pasting

UI Access	Edit>Copy		
Parameters	<i>Name</i> <Selections>	<i>Type</i> Array	<i>Description</i> Array("NAME:Selections", _ "Page:=", page number, _ // [Opt=1] Page number "Selections:=", IDs to modify))
Return Value	None		

Python Syntax	Copy()		
Python Example	<pre>oEditor.Copy (["NAME:Selections", "Selections:=", ["CompInst@R;2;37", _ "CompInst@C;1;1"]])</pre>		

CopySubdesign [Schematic]

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	<i>Name</i> <selections>	<i>Type</i> Comma-separated list of strings	<i>Description</i> 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]

Create a page at the end of the existing pages

UI Access	Schematic>New Page		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<PageName>	String	Name of the page that has been created
	<PageNum>	Integer	Page Number
Return Value	None		

Python Syntax	CreatePage(string name, // Page name [out,retval] int num) // Page number
Python Example	oEditor.CreatePage ("TestPage1")

Cut [Schematic]

Cut Page Selection

UI Access	Edit>Cut		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<Selections>	Array	Array("NAME:Selections", _ "Page:=", page number, _ // [Opt=1] Page number "Selections:=", IDs to modify))
Return Value	None		

Python Syntax	Cut()

Python Example	<pre>oEditor.Cut (["NAME:Selections", "Selections:=", ["CompInst@R;10;93"]])</pre>

DeactivateOpen [Schematic]

Disable items and replace with an open circuit

UI Access	Edit>Deactivate (Open)		
Parameters	<i>Name</i> <Selections>	<i>Type</i> Array	<i>Description</i> Array("NAME:Selections", _ "Page:=", page number, _ // [Opt=1] Page number "Selections:=", IDs to modify))
Return Value	None		

Python Syntax	DeactivateOpen()
Python Example	<pre>oEditor.DeactivateOpen (["NAME:Selections", "Selections:=", ["CompInst@R;2;37"]])</pre>

DeactivateShort [Schematic]

Disable items and replace with a closed circuit

UI Access	Edit>Deactivate (Short)		
Parameters	<i>Name</i> <Selections>	<i>Type</i> Array	<i>Description</i> Array("NAME:Selections", _ "Page:=", page number, _ // [Opt=1] Page number "Selections:=", IDs to modify))
Return Value	None		

Python Syntax	DeactivateShort ()
Python Example	<pre>oEditor.DeactivateShort(["NAME:Selections", "Selections:=", [_ "CompInst@R;2;37"]])</pre>

Delete [Schematic]

Delete items

UI Access	Edit>Delete		
Parameters	<i>Name</i> <Selections>	<i>Type</i> Array	<i>Description</i> Array("NAME:Selections", _ "Page:=", page number, _ // [Opt=1] Page number "Selections:=", IDs to modify))
Return Value	None		

Python Syntax	Delete()
Python Example	<pre>oEditor.Delete (["NAME:Selections", "Selections:=", ["CompInst@R;10;93"]])</pre>

DeletePage [Schematic]

To reduce the number of pages in the schematic

UI Access	Schematic>Remove Page		
Parameters	Name	Type	Description
	None		

ExportImage [Schematic]

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	<i>Name</i> <filename>	<i>Type</i> String	<i>Description</i> 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		

Python Syntax	ExportImage (<filename>, <pagenum>, <dx>, <dy>)
Python Example	oEditor.ExportImage ("c:\mysch.png", 1, 800, 400)

FindElements [Schematic]

Select elements based on properties

UI Access	Edit>Find Elements		
Parameters	<i>Name</i> <propname>	<i>Type</i> String	<i>Description</i> 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 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
	<types>	Integer	
	<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
	<only>	Boolean	True = search within current selections False = search schematic(s)
Return Value	Array of strings, the names of the elements selected		

Python Syntax	<pre>FindElements(["NAME:SearchProps", "Prop:=", [<propname>, <propvalue>, <criteria>], [<propname>, <propvalue>, <criteria>]]*, ["NAME:Parameters", "Filter:=", <types>, "MatchAll:=", <match>, "MatchCase:=", <case>, "SearchSubCkt:=", <sub>, "SearchSelectionOnly:=", <only>])</pre>
Python Example	<pre>oEditor.FindElement(["NAME:SearchProps", "Prop:=", ["CompName", "", 0]], ["NAME:Parameters",</pre>

	<pre> "Filter:=" , 30, "MatchAll:=" , False, "MatchCase:=" , False, "SearchSubCkt:=" , True, "SearchSelectionOnly:=" , False]) </pre>

FitToBorder [Schematic]

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	oEditor.FitToBorder()

FlipHorizontal [Schematic]

Flip items about the horizontal (x) axis.

UI Access	NA		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<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

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]

Flip items about the vertical (y) axis

UI Access	NA		
Parameters	<i>Name</i> <Selections>	<i>Type</i> Array	<i>Description</i> Array("NAME:Selections", _ "Page:=", page number, _ // [Opt=1] Page number "Selections:=", IDs to modify)) Array("NAME:FlipParameters", _ "Disconnect:=", bool _ // [Opt=0] Should wires disconnect "Rubberband:=", bool) _ // [Opt=1] Should wires staircase
	<FlipParameters>	Array	
Return Value	None		

Python Syntax	FlipVertical(
Python Example	<pre>oEditor.FlipVertical (["NAME:Selections", "Selections:=", ["CompInst@R;2;4", _ "CompInst@C;1;1"]], ["NAME:FlipParameters", "Disconnect:=", true, "Rubberband:=", false])</pre>

GetAllNets [Schematic]

Retrieve all available nets.

UI Access	NA
Parameters	None
Return Value	Array of object IDs for each net on the schematic.

Python Syntax	GetAllNets()
Python Example	<pre>oProject = oDesktop.GetActiveProject() oDesign = oProject.GetActiveDesign() oEditor = oDesign.SetActiveEditor("SchematicEditor") oEditor.GetAllNets()</pre>

GridSetup [Schematic]

Changes the settings on the schematic grid.

UI Access	Schematic>Grid Setup		
Parameters	<i>Name</i> <Options>	<i>Type</i> Array	<i>Description</i> 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))
Return Value	None

Python Syntax	GridSetup()
Python Example	<pre>oEditor.GridSetup(["NAME:Options", _ "MajorGrid:=", "10mm", _ "Divisions:=", 10, _ "MajorColor:=", 0x00ff0000, _ # Red "MinorColor:=", 0x0000ff00, _ # Green "ShowGrid:=", true, _ "SnapToGrid:=", true, _ "BackgroundColor:=", 0x00ffffff, _ # White "SaveAsDefault:=", false])</pre>

Move [Schematic]

Move items.

UI Access	NA		
Parameters	<i>Name</i> <Selections>	<i>Type</i> Array	<i>Description</i> 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		

Python Syntax	Move()
Python Example	<pre>oEditor.Move (["NAME:elements", "circle_0", "rect_2"], [0.0165613577023499, -0.001])</pre>

MovePins [Schematic]

To move pins around the periphery of a component, and/or to resize the component.

UI Access	(Right-click on single selected component) choose Move Pins... on context menu.		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>

	<compid>	String	The identifier for the schematic component instance
	<llx>	Double	The x dimension change for the lower left corner of the symbol's rectangle
	<lly>	Double	The y dimension change for the lower left corner of the symbol's rectangle
	<urx>	Double	The x dimension change for the upper right corner of the symbol's rectangle
	<ury>	Double	The y dimension change for the upper right corner of the symbol's rectangle
	<pinname>	String	The name of the pin to be moved (note that there may be any number of pins moved in this command)
	<xdelta>	Double	The x dimension change for the named pin
	<ydelta>	Double	The y dimension change for the named pin
Return Value	Boolean True if command succeeds.		

Python Syntax	<code>MovePins(<compid>, <llx>, <lly>, <urx>, <ury>, ["NAME:PinMoveData", "<pin-name>:=", <xdelta>, <ydelta>][, "<pinname>:=", <xdelta>, <ydelta>],...])</code>
Python Example	<pre>oEditor.MovePins ("CompInst@MxTranTranData;1;1", 0, -0.00508, 0.00508, 0, ["NAME:PinMoveData", "T2:=", [0, -0.00254], "T3:=", [0, -0.00254], "T4:=", [0,-0.00508], "T5:=", [0.00508, 0], "T6:=", [0.00508, -0.00254], "T7:=", [0.00508, -0.00254], "T8:=", [0.00508, -0.00508]]])</pre>

NameNets [Schematic]

Change names of nets on the schematic.

UI Access	Schematic>Auto-Name Wires		
Parameters	<i>Name</i> None	<i>Type</i>	<i>Description</i>
Return Value	None		

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]

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*.

Pan [Schematic]

Pan Display

UI Access	View>Pan		
Parameters	<i>Name</i> None	<i>Type</i>	<i>Description</i>
Return Value	None		

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]

Paste copied items

UI Access	Edit>Paste		
Parameters	<i>Name</i> None	<i>Type</i>	<i>Description</i>
Return Value	None		

Python Syntax	Paste(VARIANT attrs)# specified attributes
Python Example	<pre>oEditor.Paste (["NAME:Attributes", "Page:=", 1, "X:=", -0.0591312000000001, "Y:=", _ 0.0307847999999999, "Angle:=", 0, "Flip:=",false])</pre>

RevertToDefaultSymbol [Schematic]

Revert to the default symbol for a component whose pins were moved around the component periphery, and/or if the component symbol was resized. Pin visibility also reverts.

UI Access	(Right-click on single selected component) choose Revert To Default Symbol in the context menu

Parameters	<i>Name</i> <compid>	<i>Type</i> String	<i>Description</i> The identifier for the schematic component instance.
Return Value	Boolean True if command succeeds.		

Python Syntax	RevertToDefaultSymbol (<compid>)
Python Example	<pre>oEditor.RevertToDefaultSymbol ("CompInst@MxTranTranData;1;1")</pre>

Rotate [Schematic]

Rotate items.

UI Access	Draw>Rotate		
Parameters	<i>Name</i> <Selections>	<i>Type</i> Array	<i>Description</i> Array("NAME:Selections", _ "Page:=", page number, _ // [Opt=1] Page number "Selections:=", IDs to modify)) Array("NAME:RotateParameters", _ "Disconnect:=", bool _ // [Opt=0] Should wires disconnect "Rubberband:=", bool _ // [Opt=1] Should wires staircase "Angle:=", double) // [Opt=/2] Angle to rotate Note: Rotation occurs around center of ids
Return Value	None		

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]

Select all elements on the given page.

UI Access	Edit>Select All		
Parameters	<i>Name</i> None	<i>Type</i>	<i>Description</i>
Return Value	None		

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), <i>SelectAll</i> 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>

SelectElement [Schematic]

To select the specified element(s) in the schematic.

UI Access	NA		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>

	<i><page number></i> Integer Page number, optional, /defaulting to 1 <i><IDs></i> String; Element IDs in the schematic comma separated
Return Value	None

Python Syntax	SelectElement(["NAME:Selections", "Page:=", <page number>, "Selections:=", [<IDs>]])
Python Example	<pre>oEditor.SelectElement(["NAME:Selections", "Selections:=", ["CompInst@R;2;37", "CompInst@C;1;1"]])</pre>

SelectPage [Schematic]

Select a page in the UI.

UI Access	NA		
Parameters	<i>Name</i> None	<i>Type</i>	<i>Description</i>
Return Value	None		

Python Syntax	SelectPage(int page) # Page number
Python Example	oEditor.SelectPage(1)

SendToBack [Schematic]

Send the selected object to the back of the view.

UI Access	Draw>Send To Back		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>

	<Object>	String	Object to send to the back.
Return Value	None		

Python Syntax	SendToBack (["NAME:Selections", "Selections:=", [<Object>, <Object>, ...]])		
Python Example	<pre>oDefinitionEditor.SendToBack(["NAME:Selections", "Selections:=", ["SchObj@10"]])</pre>		

SetPageData [Schematic]

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	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<bordertype>	String	One of: "None", "Outline", "ANSI", "ISO", "DIN"
	<sizeX>	Double(with Size of X axis units)	
	<sizeY>	Double(with Size of Y axis units)	
	<marginX>	Double(with Margin of X axis units)	
	<marginY>	Double(with Margin of Y axis units)	
	<zonesH>	Integer	
	<zonesV>	Integer	
	<symbolName>	String	The name of the title block symbol.
	<libName>	String	The name of the library (with path from lib-

		rary directory root) that contains the title block symbol
	<code><Hsnap></code>	String One of: "Left", "Right", "Center", "None"
	<code><Vsnap></code>	String One of: "Top", "Bottom", "Center", "None"
	<code><rotation></code>	String One of "0", "90", "270"
	<code><propName></code>	String The name of a page property followed by ":="
	<code><propVal></code>	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/", </pre>

```
"Design:=", "TwinBuilder1", "Date:=", "9:53:11 PM Jan 27, 2016"]])
```

SortComponents

Sorts all components, or a block of components, using the specified method.

UI Access	Schematic>Sort Components		
Parameters	<i>Name</i> <Type>	<i>Type</i> Array	<i>Description</i> 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"
Return Value	None		

Python Syntax	SortComponents ()
Python Example	oEditor.SortComponents ()

ZoomArea [Schematic]

Zoom Area

UI Access	View>Zoom Area		
Parameters	<i>Name</i> <Size>	<i>Type</i> Array	<i>Description</i> 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

Python Syntax	ZoomArea()
Python Example	<pre>oEditor.ZoomArea ([0.00307569800375911, 0.021101611633739], [-0.050433530091516, -0.0220260723256825])</pre>

ZoomIn [Schematic]

Zoom into the schematic.

UI Access	View>Zoom In		
Parameters	<i>Name</i> None	<i>Type</i>	<i>Description</i>
Return Value	None		

Python Syntax	ZoomIn()
Python Example	<pre>oEditor.ZoomIn()</pre>

ZoomOut [Schematic]

Zoom out from the schematic.

UI Access	View>Zoom Out		
Parameters	<i>Name</i> None	<i>Type</i>	<i>Description</i>
Return Value	None		

Python Syntax	ZoomOut()		
Python Example	<code>oEditor.ZoomOut()</code>		

ZoomPrevious [Schematic]

Restore the zoom to the previous settings.

UI Access	NA		
Parameters	<i>Name</i> None	<i>Type</i>	<i>Description</i>
Return Value	None		

Python Syntax	ZoomPrevious()		
Python Example	<code>oEditor.ZoomPrevious()</code>		

ZoomToFit [Schematic]

Set the Zoom to Fit.

UI Access	View>Fit Drawing		
Parameters	<i>Name</i> None	<i>Type</i>	<i>Description</i>
Return Value	None		

Python Syntax	ZoomToFit()
Python Example	<code>oEditor.ZoomToFit()</code>

Property Script Methods [Schematic]

This section presents the property script methods that are available in the schematic editor.

ChangeProperty [Schematic]

Change a property

UI Access	NA		
Parameters	<i>Name</i> Name	<i>Type</i> Type	<i>Description</i> 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>

GetProperties [Schematic]

To get the names of properties for a specified tab and element.

UI Access	NA		
Parameters	<i>Name</i> <tab>	<i>Type</i> String	<i>Description</i> One of five choices, "PassedParameterTab", "ComponentTab", "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]

To find the value of an attribute of a specified property.

UI Access	NA		
Parameters	<i>Name</i> <tab>	<i>Type</i> String	<i>Description</i> One of five choices, "PassedParameterTab", "ComponentTab", "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]

Get a property

UI Access	NA		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<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]

Set a property.

UI Access	NA		
Parameters	<i>Name</i> <tab> <item> <propname> <value>	<i>Type</i> String String String String	<i>Description</i> Tab with the property Name of the objects Name of the property The new value of the property
Return Value	None		

Python Syntax	SetPropertyValue(<params>)
Python Example	None

Information Method List [Schematic]

This section presents the information methods that are available.

GetCompInstanceFromRefDes [Schematic]

UI Access	NA		
Parameters	<i>Name</i> Component	<i>Type</i> String	<i>Description</i> Name of the Component
Return Value	Returns IDispatch for CompInstance		

Python Syntax	<pre>GetCompInstanceFromRefDes(string) # string is refDes for the component</pre>
Python Example	<pre>GetCompInstanceFromRefDes ("R1")</pre>

GetComponentInfo [Schematic]

Informational

UI Access	NA		
Parameters	<i>Name</i> <compID>	<i>Type</i> String	<i>Description</i> 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	<pre>GetComponentInfo(<compID>)</pre>
Python Example	<pre>For i in selectionArray:</pre>

	<pre> compArray = oEditor.GetComponentInfo(i) For Each k in compArray: AddInfoMessage(str(k) + "Got with GetComponentInfo") </pre>

GetComponentPins [Schematic]

Informational

UI Access	NA		
Parameters	<i>Name</i> <compID>	<i>Type</i> String	<i>Description</i> 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 AddInfoMessage("Pin: " + str(j)) </pre>

GetComponentPinInfo [Schematic]

Informational - gets details about a pin on a schematic component.

UI Access	NA

Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<compID>	String	The Schematic component's ID
	<pinName>	String	The pin name
	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]

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	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<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	Float values, as follows:		
	"X=number" "Y=number"		

Python Syntax	GetComponentPinLocation <compID>, <pinname>, <XorY>
Python Example	<pre>varx = oEditor.GetComponentPinLocation (compid, "N1", TRUE)</pre>

GetNumPages [Schematic]

To determine how many pages there are in the schematic.

UI Access	NA		
Parameters	<i>Name</i> None	<i>Type</i>	<i>Description</i>
Return Value	Integer, the number of pages in the schematic.		

Python Syntax	GetNumPages()
Python Example	<pre>num = oEditor. GetNumPages()</pre>

GetPortInfo [Schematic]

Informational – get details about a port on a schematic component.

UI Access	NA		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<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: info = oEditor.GetPortInfo(i) for j in info: AddInfoMessage(str("GetPortInfo: " + j))</pre>

GetSelections [Schematic]

Returns an array of currently selected objects.

UI Access	N/A
Parameters	None.
Return Value	Array containing object IDs

Python Syntax	GetSelections()
Python Example	oEditor.GetSelections()

GetSignals [Schematic]

Informational

UI Access	NA		
Parameters	<i>Name</i> <wirename>	<i>Type</i> String	<i>Description</i> 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 from GetSignals :" + str(ll)) </pre>

GetWireConnections [Schematic]

Informational

UI Access	NA		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	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=compId pinname x,y" where x,y is the connection point.		

Python Syntax	GetWireConnections(<wireID>)
Python Example	<pre>selectionArray = oEditor.GetSelections() for i in selectionArray: AddInfoMessage("Selected element" + i) netArray = oEditor.GetWireConnections(i) for m in netArray: AddInfoMessage("GetWireConnections" + str(m))</pre>

GetWireInfo [Schematic]

Informational

UI Access	NA

Parameters	<i>Name</i> <wire id> <i>Type</i> String <i>Description</i> The Schematic wire's ID, obtained from GetSelections or through FindElements.
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	GetWireInfo(<wireID>)
Python Example	<pre> selectionArray = oEditor.GetSelections() for i in selectionArray: AddInfoMessage("Selected element" + str(i)) netArray2 = oEditor.GetWireInfo(i) for n in netArray2: AddInfoMessage("GetWireInfo" + str(n)) </pre>

GetWireSegments [Schematic]

Informational

UI Access	NA		
Parameters	<i>Name</i> <wireID>	<i>Type</i> String	<i>Description</i> The Schematic wire's ID, obtained from GetSelections or through FindElements
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: AddInfoMessage("GetWireSegments" + str(n))</pre>

Symbol Editor Script Commands

```
oSymbolEditor.CreateArc Array()
```

Note:

The object IDs in many of the following methods can be obtained from the return value of `Createxxx` methods, or `SelectAll`.

AddLevel [Symbol]

To add a graphics level to the symbol.

UI Access	Add Level in the View>Levels dialog.		
Parameters	<i>Name</i> <level>	<i>Type</i> Integer	<i>Description</i> The new level, an integer greater than 1.
Return Value	Integer The previous active level. The current active level is the newly added level.		

Python Syntax	AddLevel <level>
Python Example	<code>oldLevel = oSymbolEditor.AddLevel(4)</code>

AlignHorizontal [Symbol]

To align the specified objects to the Y axis value of the first object.

UI Access	Draw>Align Horizontal		
Parameters	<i>Name</i> <selections>	<i>Type</i> Comma-separated list of strings	<i>Description</i> Identifiers for each selected item to be aligned, in the form SchObj@<id>, where id is the value of the SchematicID property for each element.
Return Value	None		

Python Syntax	<code>AlignHorizontal(["NAME:Selections", "Selections:=", [<selections>]], 0)</code>

Python Example	<pre>oSymbolEditor = oProject.SetActiveDefinitionEditor("Symbol-Editor", _ "Simplorer Elements\basic Elements\Circuit\PassiveElements:R") oSymbolEditor.AlignHorizontal (["NAME:Selections",_ "Selections:=", ["SchObj@11"]], 0)</pre>

AlignVertical [Symbol]

To align the specified objects to the Y axis value of the first object

UI Access	Draw>Align Vertical		
Parameters	<i>Name</i> <selections>	<i>Type</i> Comma-separated list of strings	<i>Description</i> Identifiers for each selected item to be aligned, in the form SchObj@<id>, where id is the value of the SchematicID property for each element.
Return Value	Value		

Python Syntax	AlignVertical (["NAME:Selections", "Selections:=", [<selections>]], 0)
Python Example	<pre>oSymbolEditor = oProject.SetActiveDefinitionEditor("Symbol-Editor", _ "Simplorer Elements\basic Elements\Circuit\PassiveElements:R") oSymbolEditor.AlignVertical (["NAME:Selections", "Selections:=", ["SchObj@11"]], 0)</pre>

BringToFront [Symbol]

Changes the drawing for the symbol so that the specified objects are drawn on top of other overlapping objects

UI Access	Draw > Bring To Front		
Parameters	<i>Name</i> <ObjectName>	<i>Type</i> String	<i>Description</i> Object to bring to the front
Return Value	None		

Python Syntax	BringToFront(["NAME:Selections", "Selections:=", [<Object>, <Object>, ...]])		
Python Example	<pre>oDefinitionEditor.BringToFront(["NAME:Selections", "Selections:=", ["SchObj@10"]])</pre>		

CloseEditor [Symbol]

Close the Symbol Editor

UI Access	NA		
Parameters	<i>Name</i> None	<i>Type</i>	<i>Description</i>
Return Value	None		

Python Syntax	CloseEditor()		
Python Example	<pre>oDefinitionEditor.CloseEditor()</pre>		

Copy [Symbol]

Copy items for pasting

UI Access	Edit>Copy		
Parameters	<i>Name</i> Name	<i>Type</i> Type	<i>Description</i> Description
Return Value	Value		

Python Syntax	Copy(["NAME:Selections", "Selections:=", [<selections>]])		
Python Example	<pre>oSymbolEditor.Copy(["NAME:Selections", "Selections:=", ["SchObj@11", "SchObj@34"]])</pre>		

CreateArc [Symbol]

Create an arc

UI Access	Draw>Arc		
Parameters	<i>Name</i> <ArcData>	<i>Type</i> Array	<i>Description</i> 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 Array("NAME:Attributes", _ "Page:=", int) _ // [Opt=1] Page number
	<Attributes>	Array	

	(one-based) [out,retval] string id)
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]

Create a circle

UI Access	Draw>Circle		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<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 Array("NAME:Attributes", _ "Page:=", int) _ // [Opt=1] Page number (one-based) [out,retval] string id)
Return Value	Value

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]

Create picture from a BMP or GIF file

UI Access	Draw>Image		
Parameters	<i>Name</i> <ImageData>	<i>Type</i> Array	<i>Description</i> 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)
Return Value	None	

Python Syntax	CreateImage()
Python Example	<pre>oDefinitionEditor.CreateImage (["NAME:ImageData", "X1:=", _ 0.00229684174875881, "Y1:=", 0.00100608189091032, "X2:=", _ 0.00890589813779289, "Y2:=", -0.00432967005619977, "File:=", _ "C:/Documents and Settings/user/Desktop/logo.gif", "DisplayBorder:=", _ false, "LineWidth:=", 0, "Color:=", 0, "Id:=", 22], _ ["NAME:Attributes", "Page:=", 1])</pre>

CreateLine [Symbol]

Create a line

UI Access	Draw>Line		
Parameters	<i>Name</i> <LineData>	<i>Type</i> Array	<i>Description</i> 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
	<Attributes>	Array	"Id:=", int), // [Opt=New id] Id for this item Array("NAME:Attributes", _ "Page:=", int) _ // [Opt=1] Page number (one-based) [out,retval] string id)
Return Value	None		

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]

Draws a pin in the symbol editor

UI Access	NA		
Parameters	<i>Name</i> None	<i>Type</i>	<i>Description</i>
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]

Create a polygon

UI Access	Draw>Polygon		
Parameters	<i>Name</i> <PolygonData>	<i>Type</i> Array	<i>Description</i> Array("NAME:PolygonData" _ "Points:=", Array of points, _ "LineWidth:=", double, // the width of the Polygon 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 Polygon fill color
	<Attributes>	Array	"Id:=", int), // [Opt=New id] Id for this item Array("NAME:Attributes", _ "Page:=", int) _ // [Opt=1] Page number (one-based) [out,retval] string id)
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]

Create a rectangle

UI Access	Draw>Rectangle		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<RectData>	Array	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

	<Attributes> Array <code>Array("NAME:Attributes", _ "Page:=", int) _ // [Opt=1] Page number (one-based) [out,retval] string id)</code>
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]

Create text

UI Access	Draw>Primitive>Text		
Parameters	Name <TextData>	Type Array	Description 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 RGB 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 RGB value of the rectangle fill color Array("NAME:Attributes", _ "Page:=", int) _ // [Opt=1] Page number (one-based) [out,retval] string id)
Return Value	None	

Python Syntax	CreateText()
Python Example	<pre>oDefinitionEditor.CreateText(["NAME:TextData","X=", 0.001524, "Y=", 0.00127, "Text:=", "My text", "Id=", 20], ["NAME:Attributes", "Page=", 1])</pre>

Cut [Symbol]

Cut items - remove from the symbol and copy for pasting

UI Access	Edit>Cut		
Parameters	<i>Name</i> <selections>	<i>Type</i> Comma-separated list of strings	<i>Description</i> Identifiers for each selected item to be cut and copied for pasting, in the form SchObj@<id>, where id is the value of the SchematicID property for each element.
Return Value	None		

Python Syntax	Cut(["NAME:Selections", "Selections:=", [<selections>]])
Python Example	<pre>oSymbolEditor = oProject.SetActiveDefinitionEditor("Symbol-Editor",_ "Simplorer Elements\basic Elements\Circuit\PassiveElements:R") oSymbolEditor.Cut (["NAME:Selections", "Selections:=", ["SchObj@11", "SchObj@3"]])</pre>

DisplayPinNames [Symbol]

To turn on or off the display of pin names

UI Access	NA		
Parameters	<i>Name</i> <onoff>	<i>Type</i> Boolean	<i>Description</i> True or False to turn on or off the display of pin names.
Return Value	None		

Python Syntax	DisplayPinNames(<onoff>)
Python Example	<pre>oSymbolEditor = oProject.SetActiveDefinitionEditor("Symbol-Editor",_ "Simplorer Elements\basic</pre>

	<pre>Elements\Circuit\PassiveElements:R") oSymbolEditor.DisplayPinNames (True)</pre>

Delete [Symbol]

Delete items - remove from the symbol

UI Access	Edit>Delete		
Parameters	<i>Name</i> <selections>	<i>Type</i> Comma-separated list of strings	<i>Description</i> 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.
Return Value	None		

Python Syntax	Delete(["NAME:Selections", "Selections:=", [<selections>]])
Python Example	<pre>oSymbolEditor = oProject.SetActiveDefinitionEditor("Symbol-Editor",_ "Simplorer Elements\basic Elements\Circuit\PassiveElements:R") oSymbolEditor.Delete(["NAME:Selections", "Selections:=", ["SchObj@11", "SchObj@34"]])</pre>

ExportFile [Symbol]

Delete items - remove from the symbol.

UI Access	Symbol>Export File		
Parameters	<i>Name</i> <type>	<i>Type</i> String	<i>Description</i> Type of file to export - only "EMF" is supported currently.

	<filename>	String	Path, name and extension for the file to be saved
Return Value	None		

Python Syntax	ExportFile(<type>, <filename>)
Python Example	<pre>oSymbolEditor = oProject.SetActiveDefinitionEditor("Symbol-Editor",_ "Simplorer Elements\basic Elements\Circuit\PassiveElements:R") oSymbolEditor.ExportFile ("EMF", "c:\sym.emf")</pre>

GetVisibleLevels [Symbol]

To determine the levels in the symbol.

UI Access	Symbol>Export File		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	None		
Return Value	An array of integers, which are the levels defined for the particular symbol.		

Python Syntax	GetVisibleLevels()
Python Example	<pre>oSymbolEditor = oProject.SetActiveDefinitionEditor("Symbol-Editor",_ "Twin Builder Elements\basic Elements\Circuit\PassiveElements:R") levels = oSymbolEditor.GetVisibleLevels()</pre>

GridSetup [Symbol]

Changes the settings on the symbol grid.

UI Access	Symbol>Grid Setup		
Parameters	<i>Name</i> <Options>	<i>Type</i> Array	<i>Description</i> 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
Return Value	None		

Python Syntax	GridSetup()
Python Example	<pre>oEditor.GridSetup(["NAME:Options", _ "MajorGrid:=", "10mm", _ "Divisions:=", 10, _ "MajorColor:=", 0x00ff0000, _ # Red "MinorColor:=", 0x0000ff00, _ # Green "ShowGrid:=", true, _ "SnapToGrid:=", true, _</pre>

	<pre>"BackgroundColor:=", 0x00ffffff, _ # White "SaveAsDefault:=", false])</pre>

FlipHorizontal [Symbol]

To flip the specified objects about the horizontal (x) axis

UI Access	Draw>Flip Horizontal		
Parameters	<i>Name</i> <selections>	<i>Type</i> Comma-separated list of strings	<i>Description</i> 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
Return Value	None		

Python Syntax	FlipHorizontal(["NAME:Selections", "Selections:=", [<selections>]], 0)		
Python Example	<pre>oSymbolEditor.FlipHorizontal(["NAME:Selections", "Selections:=", ["SchObj@11"]], 0)</pre>		

FlipVertical [Symbol]

To flip the specified objects about the vertical (y) axis

UI Access	Draw>Flip Vertical		
Parameters	<i>Name</i> <selections>	<i>Type</i> Comma-separated list of strings	<i>Description</i> 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.
Return Value	None		

Python Syntax	<code>FlipVertical(["NAME:Selections", "Selections:=", [<selections>]], 0)</code>
Python Example	<pre>oSymbolEditor.FlipVertical(["NAME:Selections", "Selections:=", ["SchObj@11"]], 0)</pre>

Move [Symbol]

To move the specified objects in X and Y.

UI Access	NA		
Parameters	<i>Name</i> <selections>	<i>Type</i> Comma-separated list of strings	<i>Description</i> 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>	Real number	The X Delta to move
	<yval>	Real number	The Y Delta to move
Return Value	None		

Python Syntax	<code>Move(["NAME:Selections", "Selections:=", [<selections>]], ["NAME:FlipParameters", "xdelta:=", <xval>, "ydelta:=", <yval>])</code>
Python Example	<pre>oSymbolEditor = oProject.SetActiveDefinitionEditor("Symbol-Editor", _ "Simplorer Elements\basic Elements\Circuit\PassiveElements:R") oSymbolEditor.Move(["NAME:Selections", "Selections:=", ["SchObj@34"]], _ ["NAME:FlipParameters", "xdelta:=", -0.00254, "ydelta:=", 0.00254])</pre>

Pan [Symbol]

Pan Display

UI Access	View>Pan		
Parameters	<i>Name</i> None	<i>Type</i>	<i>Description</i>
Return Value	None		

Python Syntax	Pan(double, _ # Move the visible area by X meters double) # Move the visible area by Y meters		
Python Example	<pre>oDefinitionEditor.Pan ([-0.00922653869663931, 0.00686839904222147])</pre>		

Paste [Symbol]

To paste copied objects to a specified X and Y location.

UI Access	NA		
Parameters	<i>Name</i> <xval> <yval>	<i>Type</i> Real number Real number	<i>Description</i> The X placement value. The Y placement value.
Return Value	None		

Python Syntax	Paste(["NAME:Attributes", "X:=", <xval>, "Y:=", <yval>])		
Python Example	<pre>oSymbolEditor = oProject.SetActiveDefinitionEditor("Symbol-Editor", _</pre>		

```
"Simplorer Elements\basic
Elements\Circuit\PassiveElements:R")

oSymbolEditor.Paste (["NAME:Attributes",
"X:=", -0.00254, "Y:=", 0.00254])
```

Redo [Symbol]

Use: To redo an action that had previously been undone.

Command: Edit>Redo

Syntax: Redo

Return Value: None.

Parameters: None.

Example:

```
oSymbolEditor = oProject.SetActiveDefinitionEditor("SymbolEditor", _
"Simplorer Elements\basic Elements\Circuit\Passive Elements:R")
oSymbolEditor.Redo
```

RemoveLevels [Symbol]

To remove one or more graphics levels from the symbol.

UI Access	Remove Level in the View>Levels dialog.		
Parameters	<i>Name</i> <levels>	<i>Type</i> A comma-separated list of positive integers.	<i>Description</i> The levels to be deleted
Return Value	Integer the current active level.		

Python Syntax	<code>RemoveLevel([<levels>])</code>
Python Example	<pre>oSymbolEditor = oProject.SetActiveDefinitionEditor("Symbol-Editor",_ "Simplorer Elements\basic Elements\Circuit\PassiveElements:R") oldLevel = oSymbolEditor.RemoveLevels([3,4])</pre>

Rotate [Symbol]

To rotate the specified objects about their common center..

UI Access	Draw>Rotate		
Parameters	<i>Name</i> <selections> <angle>	<i>Type</i> Comma-separated list of strings A comma-separated string and real number	<i>Description</i> 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. Either "Angle:=", <rad>, where <rad> is the angle in radians, or "Degrees:=", <deg>, where <deg> is the angle in degrees.
Return Value	None		

Python Syntax	<code>Rotate(["NAME:Selections", "Selections:=", [<selections>]], ["NAME:FlipParameters", <angle>])</code>
Python Example	<pre>oSymbolEditor = oProject.SetActiveDefinitionEditor("Symbol-Editor",_ "Simplorer Elements\basic Elements\Circuit\PassiveElements:R") oSymbolEditor.Rotate(["NAME:Selections", "Selections:=", ["SchObj@11"]],_</pre>

	<code>["NAME:FlipParameters", "Angle:=", 1.5707963267949])</code>

Save [Symbol]

Saves any changes in the current symbol to the project.

UI Access	Symbol>Update Project		
Parameters	<i>Name</i> None	<i>Type</i>	<i>Description</i>
Return Value	None		

Python Syntax	Save()
Python Example	<pre>oSymbolEditor = oProject.SetActiveDefinitionEditor("Symbol-Editor",_ "Simplorer Elements\basic Elements\Circuit\PassiveElements:R") oSymbolEditor.Save()</pre>

SelectAll [Symbol]

Select all elements in the symbol editor.

UI Access	NA		
Parameters	<i>Name</i> None	<i>Type</i>	<i>Description</i>
Return Value	None		

Python Syntax	SelectAll()

Python Example	<code>removedDefs = oSymbolManager.SelectAll()</code>

SendToBack [Symbol]

Changes the drawing for the symbol so that the specified objects are drawn behind other overlapping objects.

UI Access	Draw > Send To Back		
Parameters	<i>Name</i> <Object>	<i>Type</i> String	<i>Description</i> Object to send to the back.
Return Value	None		

Python Syntax	<code>SendToBack(["NAME:Selections", "Selections:=", [<Object>, <Object>, ...]])</code>
Python Example	<code>oDefinitionEditor.SendToBack(["NAME:Selections", "Selections:=", ["SchObj@10"]])</code>

SetActiveLevel [Symbol]

To set the active graphics level in the symbol editor.

UI Access	"Current" radio button for a level in the View>Levels dialog.		
Parameters	<i>Name</i> <level>	<i>Type</i> Integer	<i>Description</i> The new active level.
Return Value	Integer the previous active level. The current active level is the newly added level.		

Python Syntax	<code>SetActiveLevel (<level>)</code>
Python	<code>oSymbolEditor = oProject.SetActiveDefinitionEditor("Symbol-</code>

Example	<pre>Editor",_ "Simplorer Elements\basic Elements\Circuit\PassiveElements:R") oldLevel = oSymbolEditor. SetActiveLevel(4)</pre>

SetInitialLevels [Symbol]

To set one or more graphics levels from the symbol as the default visible levels when used in a schematic. The levels submitted do not need to contain 1 or 0, which are always visible.

UI Access	"Default" checkbox for a level in the View>Levels dialog.		
Parameters	<i>Name</i> <levels>	<i>Type</i> A comma-separated list of positive integers	<i>Description</i> The levels to be marked as default.
Return Value	None		

Python Syntax	SetInitialLevels([<levels>])
Python Example	<pre>oSymbolEditor = oProject.SetActiveDefinitionEditor("Symbol-Editor",_ "Simplorer Elements\basic Elements\Circuit\PassiveElements:R") oldLevel = oSymbolEditor.SetInitialLevels([2, 3])</pre>

SetVisibleLevels [Symbol]

To make one or more graphics levels visible in the symbol editor.

UI Access	NA

Parameters	<i>Name</i> <levels>	<i>Type</i> A comma-separated list of positive integers	<i>Description</i> The levels to be marked as visible.
Return Value	None		

Python Syntax	SetVisibleLevels ([<levels>])
Python Example	<pre>oSymbolEditor = oProject.SetActiveDefinitionEditor("Symbol-Editor",_ "Simplorer Elements\basic Elements\Circuit\PassiveElements:R") oldLevel = oSymbolEditor.SetVisibleLevels([0, 2])</pre>

ToggleLevel [Symbol]

To toggle display of the graphics level in the symbol editor.

UI Access	"Visible" checkbox for a level in the View>Levels dialog.		
Parameters	<i>Name</i> <level>	<i>Type</i> Integer	<i>Description</i> The level to turn on or off.
Return Value	None		

Python Syntax	ToggleLevel (<level>)
Python Example	<pre>oSymbolEditor = oProject.SetActiveDefinitionEditor("Symbol-Editor",_ "Simplorer Elements\basic Elements\Circuit\PassiveElements:R") oSymbolEditor.ToggleLevel (2)</pre>

Undo [Symbol]

To undo an action.

UI Access	Edit>Undo		
Parameters	<i>Name</i> None	<i>Type</i>	<i>Description</i>
Return Value	None		

Python Syntax	Undo()
Python Example	<pre>oSymbolEditor = oProject.SetActiveDefinitionEditor("Symbol-Editor", _ "Simplorer Elements\basic Elements\Circuit\PassiveElements:R") oSymbolEditor.Undo()</pre>

ZoomArea [Symbol]

To change the current graphics view area.

UI Access	View>Zoom Area		
Parameters	<i>Name</i> <xpt>	<i>Type</i> Real number	<i>Description</i> The X value of the zoom area box, in meters.
	<ypt>	Real number	The Y value of the zoom area box, in meters.
	<xdelta>	Real number	The width of the zoom area box, in meters.
	<ydelta>	Real number	The height of the zoom area box, in meters.
Return Value	None		

Python Syntax	<code>ZoomArea([<xpt>, <ypt>], [<xdelta>, <ydelta>])</code>
Python Example	<pre>oSymbolEditor = oProject.SetActiveDefinitionEditor("Symbol-Editor",_ "Simplorer Elements\basic Elements\Circuit\PassiveElements:R") oSymbolEditor.ZoomArea ([0, 0], [0.004,0.004])</pre>

ZoomIn [Symbol]

To zoom in a standard amount to the current graphics view area.

UI Access	View>Zoom In		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	None		
Return Value	None		

Python Syntax	<code>ZoomIn()</code>
Python Example	<pre>oSymbolEditor = oProject.SetActiveDefinitionEditor("Symbol-Editor",_ "Simplorer Elements\basic Elements\Circuit\PassiveElements:R") oSymbolEditor.ZoomIn()</pre>

ZoomOut [Symbol]

To zoom out a standard amount to the current graphics view area.

UI Access	View>Zoom Out		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>

	None
Return Value	None

Python Syntax	ZoomOut()
Python Example	<pre>oSymbolEditor = oProject.SetActiveDefinitionEditor("Symbol-Editor",_ "Simplorer Elements\basic Elements\Circuit\PassiveElements:R") oSymbolEditor.ZoomOut()</pre>

ZoomPrevious [Symbol]

To zoom the current graphics view area to the previous extents.

UI Access	View>Zoom Previous		
Parameters	<i>Name</i> None	<i>Type</i>	<i>Description</i>
Return Value	None		

Python Syntax	ZoomPrevious()
Python Example	<pre>oSymbolEditor = oProject.SetActiveDefinitionEditor("Symbol-Editor",_ "Simplorer Elements\basic Elements\Circuit\PassiveElements:R") oSymbolEditor.ZoomPrevious()</pre>

ZoomToFit [Symbol]

To zoom the current graphics view area to include all the symbol elements in the field of view.

UI Access	View>Zoom to Fit		
Parameters	<i>Name</i> None	<i>Type</i>	<i>Description</i>
Return Value	None		

Python Syntax	ZoomToFit()
Python Example	<pre>oSymbolEditor = oProject.SetActiveDefinitionEditor("Symbol-Editor",_ "Simplorer Elements\basic Elements\Circuit\PassiveElements:R") oSymbolEditor.ZoomToFit()</pre>

20 - Twin Builder Result Database (SdbCtrl) Automation Script Commands

To use the Twin Builder result database (**SdbCtrl**) automation capabilities, every scripting/programming languages or application can be used that supports Microsoft COM. In this documentation, all automation functions are described independent of the language used.

Before an **SdbCtrl** automation function can be called it is necessary to create an instance of an **SdbCtrl** object. The name of the object to create is "Simplorer2021.1.SdbCtrl".

```
Set sdbctrlapp = CreateObject("Simplorer2021.1.SdbCtrl")
```

Note:

IronPython does not support CreateObject for **SdbCtrl**. However, you can still create the **SdbCtrl** instance by directly accessing Microsoft's .NET Framework in IronPython. Use this script snippet to create an **SdbCtrl** object and access the commands:

```
import System

typ=System.Type.GetTypeFromProgID("Simplorer2025.2.SdbCtrl")

oSdbCtrl = System.Activator.CreateInstance(typ)
```

You can now access all regular **SdbCtrl** commands through the **oSdbCtrl** object.

General Functions

The `GetLastError` command returns the description of the last error that occurred for the SDB module.

GetLastError (SDB)

Returns the error description of the last error.

UI Access	NA		
Parameters	<i>Name</i> None	<i>Type</i>	<i>Description</i>
Return Value	String Returns the error description of the last error. If there was no error, this function returns an empty string.		

File Control Functions

Use these commands to access and manipulate Twin Builder files.

OpenSDBFile

Opens an SDB file for read access.

UI Access	NA		
Parameters	<i>Name</i> <lpszFileName>	<i>Type</i> String	<i>Description</i> Name of the SDB file to open.
Return Value	Long Returns a handle to the opened SDB file. This handle can be used as a parameter for other automation functions. On error, this function returns 0.		

CloseSDBFile

Closes an open SDB file. The file must have been previously opened by a call to OpenSDBFile. This function is used when access to a SDB file is no longer needed

UI Access	NA		
Parameters	<i>Name</i> <ISDBHandle>	<i>Type</i> Long	<i>Description</i> Handle to the SDB file model that was previously obtained by a call to the function OpenSDBFile.
Return Value	Boolean Returns 1 if the SDB file was successfully closed; otherwise, it returns 0		

Data Access Functions

Use these commands to access data stored in an SDB file.

ReadData

Reads data that is stored in a SDB file.

UI Access	NA		
Parameters	<i>Name</i> <ISDBHandle>	<i>Type</i> Long	<i>Description</i> Handle to the SDB file model that was previously obtained by a call to the function <code>OpenSDBFile</code> .
	<lpszTaskName>	String	Name of the task from which the data is read.
	<lpszChannelName>	String	Name of the channel (signal) from which the data is read. This name must be fully qualified.
	<bDigital>	Boolean	Specifies the channel type, either digital (1) or analog(0). Ignored, except for "Time". The SDB may contain two "Time" channels, for the digital and analog timesteps in a mixed signal simulation. If there is only one "Time" channel, as for an analog-only circuit, the parameter is ignored.
	<lRunNo>	Long	Number of the run from which the data is read. If <lpszTaskName> specifies the task name of a single run simulation, this parameter is ignored. If <lpszTaskName> specifies the task name of an experiment, and <lRunNo> is -1, then the data generated by the Optimetrics analysis is read; otherwise, the data from the <lRunNo>-th simulation run is read.
	<lAnaType>	Long	Default Value: -1 Analysis type of which the data shall be read. Possible values are: -1 - Default analysis 1 - Transient analysis

			2 - DC-Analysis 3 - AC-Analysis Default Value: -1 Index array of the channel (signal) of which the data shall be read. If the channel is not an array or record, this parameter is ignored.
Return Value	Variant		Returns the read data as a variant array. On error, this functions returns an empty variant.

ReadImagData

Reads data that is stored in a SDB file.

UI Access	NA		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<ISDBHandle>	Long	Handle to the SDB file model that was previously obtained by a call to the function <code>OpenSDBFile</code> .
	<lpszTaskName>	String	Name of the task from which the data is read.
	<lpszChannelName>	String	Name of the channel (signal) from which the data is read. This name must be fully qualified.
	<lRunNo>	Long	Number of the run from which the data is read. If <lpszTaskName> specifies the task name of a single run simulation, this parameter is ignored. If <lpszTaskName> specifies the task name of an experiment, and <lRunNo> is -1, then the data generated by the Optimetrics analysis is read; otherwise, the data from the <lRunNo>-th simulation run is read.

	<IAnaType> Long Default Value: -1 Analysis type of which the data shall be read. Possible values are: -1 - Default analysis 1 - Transient analysis 2 - DC-Analysis 3 - AC-Analysis <arrayIdx> Variant Default Value: -1 Index array of the channel (signal) of which the data shall be read. If the channel is not an array or record, this parameter is ignored.
Return Value	Variant Returns the read data as a variant array. On error, this functions returns an empty variant.

GetAvailableTasks

UI Access	NA		
Parameters	Name <ISDBHandle>	Type Long	Description Handle to the SDB file model that was previously obtained by a call to the function OpenSDBFile.
Return Value	Variant Returns the a variant array which contains the available task names. On error this functions returns an empty variant.		

GetTaskType

Obtains the type of a specified task name.

UI Access	NA		
Parameters	<i>Name</i> <ISDBHandle>	<i>Type</i> Long	<i>Description</i> Handle to the SDB file model that was previously obtained by a call to the function <code>OpenSDBFile</code> .
	<lpszTaskName>	String	Name of the task for which the type is to be obtained.
Return Value	Long Returns the type of the task. On error, this functions returns -1. Possible values are: -1 - Error 0 - Twin Builder 7 and earlier only, Free task type 1 - Nominal Simulation (not Optimetrics) 2 - Parametric Analysis 3 - Simplorer 7 and earlier only, Trend Analysis 4 - Statistical (Monte Carlo) Analysis 5 - Sensitivity Analysis 6 - Simplorer 7 and earlier only, Successive Approximation 7 - Optimization (using any Optimetrics method) 8 - Simplorer 7 and earlier only, Frequency Analysis 9 - Simplorer 7 and earlier only, DC Sweep		

GetNumberOfRuns

Obtains the number of runs that are stored for a specified task.

UI Access	NA		
Parameters	<i>Name</i> <ISDBHandle>	<i>Type</i> Long	<i>Description</i> Handle to the SDB file model that was previously obtained by a call to the function <code>OpenSDBFile</code> .

	<lpszTaskName>	String	Name of the task for which the number of runs is to be obtained.
Return Value	Long Returns the number of runs that are stored for the specified task. Note that the runs in an SDB file may not start at 1.		

GetRunNumbers

Obtains an array of the run numbers available for the specified task.

UI Access	NA		
Parameters	<i>Name</i> <ISDBHandle>	<i>Type</i> Long	<i>Description</i> Handle to the SDB file model that was previously obtained by a call to the function <code>OpenSDBFile</code> .
	<lpszTaskName>	String	Name of the task for which the number of runs is to be obtained.
Return Value	Array of strings, the setup names that have been selected for the specified Optimetrics setup.		

GetAvailableAnalysis

Obtains a list of all analyses stored for the specified task

UI Access	NA		
Parameters	<i>Name</i> <ISDBHandle>	<i>Type</i> Long	<i>Description</i> Handle to the SDB file model that was previously obtained by a call to the function <code>OpenSDBFile</code> .
	<lpszTaskName>	String	Name of the task for which the available analyses are to be obtained.
Return Value	Variant Returns a variant array containing the available analyses. On error, this functions returns an empty variant. Possible values, which can be stored with the variant array, are:		

	1 - Transient analysis
	2 - DC-Analysis
	3 - AC-Analysis

GetAnalysisType

Obtains the default analyses stored for the specified task. The default analysis is the most recent analysis that was simulated

UI Access	NA		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<ISDBHandle>	Long	Handle to the SDB file model that was previously obtained by a call to the function <code>OpenSDBFile</code> .
	<IpszTaskName>	String	Name of the task for which the available analyses are to be obtained
Return Value	Long Returns a number that describes the default analysis. On error, this function returns -1. Possible values, which can be stored with the variant array, are: -1 - Error 1 - Transient analysis 2 - DC-Analysis 3 - AC-Analysis		

GetAvailableChannels

Obtains a list of all channels that are stored for a specified analysis.

UI Access	NA		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<ISDBHandle>	Long	Handle to the SDB file model that was previously obtained by a call to the function <code>OpenSDBFile</code> .

	<lpszTaskName> String Name of the task for which the names of the available channels are to be obtained. <lAnaType> Long Analysis type for which the names of the available channels are to be obtained. Possible values are: -1 - Default analysis 1 - Transient analysis 2 - DC-Analysis 3 - AC-Analysis Default Value:-1
Return Value	Variant Returns a variant array containing the names of the available channels. On error, this functions returns an empty variant.

GetAvailableExpChannels

Obtains a list of all Optimetrics analysis channels stored for a specified analysis.

UI Access	NA		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<ISDBHandle> Long Handle to the SDB file model that was previously obtained by a call to the function <code>OpenSDBFile</code>. <lpszTaskName> String Name of the task for which the names of the available channels are to be obtained.		
Return Value	Variant Returns a variant array containing the names of the available channels. On error, this function returns an empty variant.		

GetDataType

Obtains the data type of a specified channel

UI Access	NA		
Parameters	<i>Name</i> <ISDBHandle>	<i>Type</i> Long	<i>Description</i> Handle to the SDB file model that was previously obtained by a call to the function <code>OpenSDBFile</code> .
	<lpszTaskName>	String	Name of the task for which the data shall be read.
	<lpszChannelName>	String	Name of the channel (signal) for which the data type shall be obtained. This name must be fully qualified.
	<lAnaType>	Long	Analysis type for which the data shall be read. Possible values are: -1 - Default analysis 1 - Transient analysis 2 - DC-Analysis 3 - AC-Analysis
	<arrayIdx>	Variant	Default Value:-1 Index array of the channel (signal) for which the data type shall be obtained. If the channel is not an array or record, this parameter is ignored.
Return Value	Long Returns the data type of the specified channel. On error, this functions returns -1. Possible values are: -1 - Error 0 - Free data type 1 - Real (double) Value 2 - Complex Value 4 - Array 5 - Enumeration 6 - Integer		

	7 - Character
	8 - Record

GetDataTypeExp

Obtains the data type of a parameter being varied for an Optimetrics analysis.

UI Access	NA		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<ISDBHandle>	Long	Handle to the SDB file model that was previously obtained by a call to the function <code>OpenSDBFile</code> .
	<IpszTaskName>	String	Name of the task for which the data shall be read
	<IpszChannelName>	String	Name of the channel (signal) for which the data type shall be obtained. This name must be fully qualified.
Return Value	Long Returns the data type of a parameter being varied for an Optimetrics analysis, not a channel whose data is being calculated and saved. It will return an error if the given name is not one of the varying parameters. The return value is always real currently. Possible values are: -1 - Error 1 - Real (double) Value 12 - Record		

GetNumberOfSamples

Obtains the number of data samples for a specified run.

UI Access	NA		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<ISDBHandle>	Long	Handle to the SDB file model that was pre-

	viously obtained by a call to the function <code>OpenSDBFile</code> <lpszTaskName> String Name of the task for which the number of data samples shall be obtained. <bDigital> Boolean Specifies the channel type, either digital (1) or analog(0). Ignored, except for "Time". The SDB may contain two "Time" channels, for the digital and analog timesteps in a mixed signal simulation. If there is only one "Time" channel, as for an analog-only circuit, the parameter is ignored. <lRunNo> Long Number of the run for which the number of data samples shall be obtained. If <lpszTaskName> specifies the task name of a single run simulation, this parameter is ignored. If <lpszTaskName> specifies the task name of an experiment, and <lRunNo> is -1, then the number of data samples generated by the Optimetrics analysis is obtained. Otherwise, the number of data samples of the <lRunNo>-th simulation run is obtained. <lAnaType> Long Default Value:-1 Analysis type for which the number of data samples shall be obtained. Possible values are: -1 - Default analysis 1 - Transient analysis 2 - DC-Analysis 3 - AC-Analysis Default Value:-1
Return Value	Long Returns the number of data samples for the specified run. On error, this functions returns -1.

GetBaseChannelName

Obtains the name of the base channel for a specified analysis. This function cannot be used for Optimetrics data.

UI Access	NA		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<ISDBHandle>	Long	Handle to the SDB file model that was previously obtained by a call to the function <code>OpenSDBFile</code> .
	<lpszTaskName>	String	Name of the task for which the name of the base channel shall be obtained.
	<lAnaType>	Long	Analysis type for which the name of the base channel shall be obtained. Possible values are: -1 - Default analysis 1 - Transient analysis 2 - DC-Analysis 3 - AC-Analysis Default Value:-1
Return Value	String Returns the name of the base channel (generally "Time" or "F") for the specified analysis. On error, this functions returns an empty string.		

GetEnumString

Provides the corresponding character string of the value.

UI Access	NA		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<ISDBHandle>	Long	Handle to the SDB file model that was previously obtained by a call to the function <code>OpenSDBFile</code>
	<lpszTaskName>	String	Name of the task for which the data shall be read
	<lpszChannelName>	String	Name of the channel (signal) for which the

	<IValue> Long <IAnaType> Long <arrayIdx> Variant	data shall be obtained. This name must be fully qualified. Value that is to be converted. Analysis type for which the data shall be read. Possible values are: -1 - Default analysis 1 - Transient analysis 2 - DC-Analysis 3 - AC-Analysis Default Value:-1 Index array of the channel (signal) for which the data shall be read. If the channel is not an array or record, this parameter is ignored.
Return Value	String	Returns the corresponding character string of the value <IValue>

GetChannelDim

Obtains the channel dimension for a specified channel (signal).

UI Access	NA		
Parameters	<i>Name</i> <ISDBHandle>	<i>Type</i> Long	<i>Description</i> Handle to the SDB file model that was previously obtained by a call to the function <code>OpenSDBFile</code>.
	<lpszTaskName>	String	Name of the task for which the data shall be read.
	<lpszChannelName>	String	Name of the channel (signal) for which the data shall be read. This name must be fully qualified.
	<IAnaType>	Long	Analysis type for which the data shall be read. Possible values are:

			-1 - Default analysis 1 - Transient analysis 2 - DC-Analysis 3 - AC-Analysis Default Value:-1 Index array of the channel (signal) for which the data shall be read. If the channel is not an array or record, this parameter is ignored.
Return Value	Long		Returns the dimension of the channel if it is an ARRAY or RECORD. On error, this functions returns -1

GetArrayDim

Obtains dimensions for all array levels

UI Access	NA		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<ISDBHandle>	Long	Handle to the SDB file model that was previously obtained by a call to the function <code>OpenSDBFile</code> .
	<lpszTaskName>	String	Name of the task for which the data shall be read.
	<lpszChannelName>	String	Name of the channel (signal) for which the data shall be read. This name must be fully qualified.
	<lAnaType>	Long	Analysis type for which the data shall be read. Possible values are: -1 - Default analysis 1 - Transient analysis 2 - DC-Analysis 3 - AC-Analysis

	Default Value:-1
Return Value	Variant Returns a variant array containing the dimension of all array levels. On error, this functions returns an empty variant.

GetChannelName

Obtains the channel name for a specified array channel.

UI Access	NA		
Parameters	<i>Name</i> <ISDBHandle>	<i>Type</i> Long	<i>Description</i> Handle to the SDB file model that was previously obtained by a call to the function <code>OpenSDBFile</code> .
	<lpszTaskName>	String	Name of the task for which the data shall be read.
	<lpszChannelName>	String	Name of the channel (signal) for which the data shall be read. This name must be fully qualified.
	<lAnaType>	Long	Analysis type for which the data shall be read. Possible values are: -1 - Default analysis 1 - Transient analysis 2 - DC-Analysis 3 - AC-Analysis Default Value:-1
	<arrayIdx>	Variant	Index array of the channel (signal) for which the data shall be read. If the channel is not an array or record, this parameter is ignored.
Return Value	String Returns the channel name of the given array channel. If there was no error, this function returns an empty string.		

21 - Twin Builder Mathematical Automation (MathMod) Interface Script Commands

To use the Twin Builder mathematical automation capabilities, every scripting/programming languages or application can be used that supports Microsoft COM. In this documentation, all automation functions are described independent of the language used.

Before a **MathMod** automation function can be called it is necessary to create an instance of an **MathMod** object. The name of the object to create is "Simplorer2021.1.MathMod".

```
Set mathmod = CreateObject("Simplorer2021.1.MathMod")
```

Following is the overall procedure used for mathematical scripting:

1. Initialize the mathematical module -- the "engine".
2. Create some data channels
 - by loading simulation results from an SDB file,
 - by passing an array of data from the script, or
 - by calling math functions in the engine.
3. Define a base channel, typically time or frequency.
4. Call the desired calculation functions.

Note:

IronPython does not support CreateObject for **MathMod**. However, you can still create the **MathMod** instance by directly accessing Microsoft's .NET Framework in IronPython. Use this script snippet to create an **MathMod** object and access the commands:

```
import System
typ=System.Type.GetTypeFromProgID("Simplorer2020.2.MathMod")
oSdbCtrl = System.Activator.CreateInstance(typ)
```

You can now access all regular **MathMod** commands through the **oMathMod** object.

Example Scripts

Example scripts can be found in the installation directory under
 \Examples\Twin Builder\Applications\Scripting\Ex6_MathMod

Error Handling Functions

Any function defined as type Boolean returns TRUE for success, and FALSE for failure. After any function that returns FALSE, the function GetLastError should be called to display an error message.

GetLastError (MathMod)

UI Access	NA		
Parameters	<i>Name</i> None	<i>Type</i>	<i>Description</i>
Return Value	Variant Returns a description of the last error. If there was no error, this function returns an empty string.		

Example: Error Handling

This example shows how to detect errors in the two most common situations, after a function call to the engine, and after attempting to read data from an SDB file. For a more concise presentation, the subsequent examples don't show this error handling, but in actual scripts it is important to detect and handle all errors, before subsequent function calls.

```
bRet = MathMod.MakeLinearChannel ("t", 0.0, 0.1, 500.0e-6)
```

```
if bRet = 0 Then
```

```
  errmsg = MathMod.GetLastError
```

```
  MsgBox( errmsg )
```

```
end if
```

```
x_val = SDBCtrlApp.ReadData(sdbHandle, taskName, "t")
```

```
if IsEmpty(x_val) then
```

```
  MsgBox SDBCtrlApp.GetLastError & " t"
```

```
else
```

```
  MathMod.SetChannel "t", x_val
```

end if

Base and Reference Channel Functions

These functions define base and reference channels that apply to all subsequent function calls, until the base or reference channel is changed.

Many functions require that the base channel be monotonic, and that all input channels have the same number of points as the base channel (or reference channel). When these conditions are violated, function calls will fail and `GetLastError` should be used to display the appropriate message. `GetMonotony` and `GetChannelLength` may be used to pre-test for these conditions.

SetBaseChannel

UI Access	NA		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<ChannelName>	String	The name of an existing channel
	<Start>	<variant>	If provided, calculations do not start until the base channel reaches this value.
Return Value	<Stop>	<variant>	If provided, calculations top after the base channel reaches this value.
	Boolean Returns 1 if the base time or frequency channel was successfully set; otherwise, it returns 0		

Python Syntax	SetBaseChannel (<ChannelName>, <Start>, <Stop>)
Python Example	None

SetFundMethod

UI Access	NA		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<UseLeastSquares>	Integer	If 0, uses the DFT method; if 1, uses the

	<Order> <long> Least Squares method. The highest harmonic order to resolve if using the DFT method, or the minimum number of equally spaced samples in a cycle if using the least squares method. <Stop> Variant If provided, calculations top after the base channel reaches this value.
Return Value	Boolean Returns 1 if successfully selected; otherwise 0.

SetImageRefChannel

UI Access	NA		
Parameters	<i>Name</i> <ChannelName>	<i>Type</i> String	<i>Description</i> The name of an existing channel.
Return Value	Boolean Returns 1 if the base channel is successfully set; otherwise, it returns 0.		

SetRealRefChannel

UI Access	NA		
Parameters	<i>Name</i> <ChannelName>	<i>Type</i> String	<i>Description</i> The name of an existing channel.
Return Value	Boolean Returns 1 if the setpoint channel or real base is successfully set; otherwise, it returns 0.		

UnsetRefChannels

Removes the effect of `SetRealRefChannel` and `SetImagRefChannel`.

UI Access	NA

Parameters	Name	Type	Description
	None		
Return Value	Boolean Returns 1 if the effects are successfully removed; otherwise, it returns 0.		

Example: Base Channel with Start and Stop Times

This example applies windowing from 0 to 1.5 ms, then removes the start and stop window times.

```
MathMod.SetBaseChannel "t", 0.0, 0.0015
```

```
MathMod.SetBaseChannel "t"
```

Example: SetFundMethod for TR Probe and PWR Probe Functions

This example displays the fundamental apparent power first by a 5th-order DFT, and then by least squares with 1000 samples per cycle.

```
MathMod.SetFundMethod 0, 5
```

```
MathMod.SetupPWR Array ("E2.EMF"), Array ("E2.I"), 50.0
```

```
ExcelApp.Range("G17") = "E2 S1 by DFT 5"
```

```
ExcelApp.Range("H17") = MathMod.GetPWRSystemS1
```

```
MathMod.SetFundMethod 1, 1000
```

```
MathMod.SetupPWR Array ("E2.EMF"), Array ("E2.I"), 50.0
```

```
ExcelApp.Range("G18") = "E2 S1 by LSQ 1000"
```

```
ExcelApp.Range("H18") = MathMod.GetPWRSystemS1
```

Example: Reference Channels for CTRL AC and CTRL ERR Probes

This example calculates the integral squared error of a channel using its own value as a setpoint, then the mean and RMS values as setpoints (previously calculated and saved in channels "Mean" and "RMS"), and finally using 1.0 as a setpoint).

```
ExcelApp.Range("L1") = "ISE wrt C3.V"
```

```
MathMod.SetRealRefChannel "C3.V"
```

```
ExcelApp.Range("M1") = MathMod.GetISE ("C3.V", 1.0)
```

```
ExcelApp.Range("L2") = "ISE wrt Mean"
```

```
MathMod.SetRealRefChannel "Mean"
```

```
ExcelApp.Range("M2") = MathMod.GetISE ("C3.V", 1.0)
```

```
ExcelApp.Range("L3") = "ISE wrt RMS"
```

```
MathMod.SetRealRefChannel "RMS"
```

```
ExcelApp.Range("M3") = MathMod.GetISE ("C3.V", 1.0)
```

```
ExcelApp.Range("L4") = "ISE wrt 1"
```

```
MathMod.UnsetRefChannels
```

```
ExcelApp.Range("M4") = MathMod.GetISE ("C3.V", 1.0)
```

Channel Create, Access, and Delete Functions

These functions are used to create and delete channels or to access the channel data.

Calculate

UI Access	NA		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<Expression>	String	Expression to be calculated. This expression must not contain an assignment operator but can contain multiple operators. The operands can be names of channels or constants. If an operand is specified as a channel name, this channel must have been previously set by <code>SetChannel</code> or it calculated by one of the calculation functions. All channels used must be compatible with the base channel, or they must have been created with <code>MakeConstant</code> .

	<OutputName>	String	Name of the channel where the result should be stored. If this is a name of an existing channel, the channel is overwritten. The value of this channel can be obtained by a call to <code>GetChannel</code> .
Return Value	Boolean Returns 1 if the expression is successfully calculated and output; otherwise, it returns 0		

GetChannel

UI Access	NA		
Parameters	<i>Name</i> <ChannelName>	<i>Type</i> String	<i>Description</i> The name of an existing channel.
Return Value	Variant Returns an array of values from the specified channel. On error, this functions returns an empty variant.		

MakeConstant

UI Access	NA		
Parameters	<i>Name</i> <ChannelName>	<i>Type</i> String	<i>Description</i> The name of a channel, of length 1, to create. Any existing channel by this name is overwritten.
	<Value>	Double	The constant value to assign for this channel.
Return Value	Boolean Returns 1 if the channel is successfully created; otherwise, it returns 0.		

MakeLinearChannel

UI Access	NA		
Parameters	<i>Name</i> <ChannelName>	<i>Type</i> String	<i>Description</i> The name of a channel to create. Any existing channel by this name is overwritten.
	<Start>	Double	First channel value.
	<Stop>	Double	Channel stop value. Must be greater than the Start value.
	<Step>	Double	Equal step size between points. Must be positive.
Return Value	Boolean Returns 1 if the channel is successfully created; otherwise, it returns 0.		

MakeLogChannel

UI Access	NA		
Parameters	<i>Name</i> <ChannelName>	<i>Type</i> String	<i>Description</i> The name of a channel to create. Any existing channel by this name is overwritten.
	<Start>	Double	First channel value. Must be positive.
	<Stop>	Double	Channel stop value. Must be greater than the Start value.
	<PointsPerDecade>	Integer	The number of logarithmically spaced points per decade. Must be positive.
Return Value	Boolean Returns 1 if the channel is successfully created; otherwise, it returns 0		

RemoveChannel

UI Access	NA		
Parameters	<i>Name</i> <ChannelName>	<i>Type</i> String	<i>Description</i> The name of the channel to remove.

Return Value	Boolean Returns 1 if the channel is successfully removed; otherwise, it returns 0.

Note:

`RemoveChannel` can be used to delete a channel that was added by [SetChannel](#).

SetChannel

UI Access	NA		
Parameters	<i>Name</i> <ChannelName>	<i>Type</i> String	<i>Description</i> The name of the channel to be set. Any existing channel by this name is overwritten.
	<Values>	Variant	A value array containing the values of the base channel.
Return Value	Boolean Returns 1 if the values are successfully assigned to the channel; otherwise, it returns 0.		

Example: Creating Channels from Data Arrays

This example creates a time base and "C3.V" channel from data arrays, then it sets the time array as the base channel. These channels are used in many of the subsequent examples.

```
bRet = MathMod.SetChannel("t", Array( 0.0, 0.00001, 0.00002, 0.00003, 0.00004, 0.00005,
0.00006, 0.00007, 0.00008, 9.02E-05, 0.000102, 0.00012, 0.000141, 0.000163, 0.000173,
0.000188, 0.000203, 0.000224, 0.000246, 0.000267, 0.00029, 0.000313, 0.000343, 0.000353,
0.000368, 0.000383, 0.000405, 0.000426, 0.000436, 0.000451, 0.000466, 0.000487, 0.000509,
0.00053, 0.000552, 0.000573, 0.000583, 0.000594, 0.000605, 0.000619, 0.000629, 0.00064,
0.00065, 0.000664, 0.000679, 0.000696, 0.000714, 0.000736, 0.000758, 0.000768, 0.000783,
0.000798, 0.00082, 0.000841, 0.000863, 0.000885, 0.000908, 0.000931, 0.000959, 0.000984,
0.000994, 0.001009, 0.001025, 0.001047, 0.001057, 0.001072, 0.001087, 0.001109, 0.001131,
0.001153, 0.001175, 0.001197, 0.00122, 0.001243, 0.001253, 0.001268, 0.001283, 0.001305,
0.001328, 0.001352, 0.001362, 0.001377, 0.001392, 0.001415, 0.001437, 0.001459, 0.001481,
0.001504, 0.001527, 0.001551, 0.001578, 0.001615, 0.001625, 0.00164, 0.001656, 0.00168,
```

```
0.00169, 0.001705, 0.00172, 0.001743, 0.001766, 0.001788, 0.001811, 0.001835, 0.001859,  
0.001885, 0.001914, 0.001943, 0.001953, 0.001968, 0.001984, 0.002 ) )
```

```
bRet = MathMod.SetChannel( "C3.V", Array( 0.0, 0.009803922, 0.029123414, 0.057582491,  
0.094722934, 0.140010398, 0.192841144, 0.252549315, 0.318414685, 0.391082243,  
0.480817709, 0.626779114, 0.808275693, 0.994416339, 1.082038801, 1.206032964,  
1.327993775, 1.482421223, 1.612165649, 1.712956533, 1.780070239, 1.80406446,  
1.780294282, 1.755436968, 1.697744568, 1.634629635, 1.519435863, 1.382369428,  
1.313626369, 1.205418568, 1.096673912, 0.941797166, 0.792435206, 0.655037678,  
0.535253794, 0.438952497, 0.403423043, 0.376220804, 0.352508614, 0.332661993,  
0.327422518, 0.332377902, 0.34056355, 0.363678741, 0.401770034, 0.457834365,  
0.536774808, 0.64660274, 0.771371398, 0.831815838, 0.923850072, 1.016946828,  
1.145126876, 1.264216006, 1.36933295, 1.456009398, 1.520281827, 1.558306148,  
1.563292215, 1.530216563, 1.508053077, 1.461398902, 1.409348309, 1.318571486,  
1.272916741, 1.19918333, 1.122764744, 1.011412985, 0.90167765, 0.798656457,  
0.706726693, 0.629867736, 0.57166179, 0.536374137, 0.528910255, 0.53176795,  
0.539889427, 0.569270195, 0.621232578, 0.693855693, 0.729497898, 0.788721716,  
0.851586577, 0.945441415, 1.040028196, 1.130675748, 1.213459268, 1.284751779,  
1.341248119, 1.37980272, 1.396369125, 1.373409445, 1.358425883, 1.326452938,  
1.288630428, 1.222204689, 1.190179643, 1.138494953, 1.083014042, 1.002323599,  
0.92315516, 0.849321572, 0.78410982, 0.730555257, 0.691526743, 0.670306798,  
0.672836386, 0.7012956, 0.716938838, 0.747683927, 0.784029839, 0.82490091 ) )
```

```
bRet = MathMod.SetBaseChannel ("t")
```

Example: Creating Channels from SDB Files

This example creates several channels from two different SDB files, including real and imaginary channels from an AC analysis. These channels are used later in some of the examples.

```
Set FileSystemObj = CreateObject("Scripting.FileSystemObject")  
PathName = FileSystemObj.GetParentFolderName(WScript.ScriptFullName)  
Path = PathName & "\"  
  
sdbHandle = SDBCtrlApp.OpenSDBFile(Path & "pwr_probe.sdb")  
taskName = "TR"
```

```
x_val = SDBCtrlApp.ReadData(sdbHandle, taskName, "t")
MathMod.SetChannel "t", x_val
y_val = SDBCtrlApp.ReadData(sdbHandle, taskName, "C1.V")
MathMod.SetChannel "C1.V", y_val
y_val = SDBCtrlApp.ReadData(sdbHandle, taskName, "C1.I")
MathMod.SetChannel "C1.I", y_val
y_val = SDBCtrlApp.ReadData(sdbHandle, taskName, "E2.EMF")
MathMod.SetChannel "E2.EMF", y_val
y_val = SDBCtrlApp.ReadData(sdbHandle, taskName, "E2.I")
MathMod.SetChannel "E2.I", y_val
SDBCtrlApp.CloseSDBFile(sdbHandle)

sdbHandle = SDBCtrlApp.OpenSDBFile(Path & "filter.sdb")
taskName = "AC"
x_val = SDBCtrlApp.ReadData(sdbHandle, taskName, "f")
MathMod.SetChannel "filter_f", x_val
y_val = SDBCtrlApp.ReadData(sdbHandle, taskName, "N0011.V")
MathMod.SetChannel "N0011.Re", y_val
y_val = SDBCtrlApp.ReadImagData(sdbHandle, taskName, "N0011.V")
MathMod.SetChannel "N0011.Im", y_val
SDBCtrlApp.CloseSDBFile(sdbHandle)
```

Example: Creating Channels from Calculate Functions

This example creates a time-based channel and a frequency-based channel from expressions.

```
bRet = MathMod.MakeLinearChannel ("t", 0.0, 0.1, 500.0e-6)
bRet = MathMod.MakeLogChannel ("f", 0.001, 100.0, 20)
```

```
bRet = MathMod.MakeConstant ("omega", 60.0 * 3.14159 * 2.0)
bRet = MathMod.SetBaseChannel ("t")
bRet = MathMod.Calculate ("10 * (exp (-t/0.03) + sin (omega * t))", "f(t)")
bRet = MathMod.SetBaseChannel ("f")
bRet = MathMod.Calculate ("10 / sqrt (1 + f * f)", "G(f)")
```

Channel Utility Functions

These functions transform channels or extract basic information about a channel.

The **T Probe** example shows a use of the `GetChannelMax` and `GetChannelMin` functions.
The **F Probe** example shows a use of the `RectToPolar` function.

CalculateCommonBase

UI Access	NA		
Parameters	<i>Name</i> <Base1>	<i>Type</i> String	<i>Description</i> The name of the first existing channel to merge
	<Base2>	String	The name of the second existing channel to merge
	<NewBase>	String	The name of a channel for the merged output. Any existing channel with this name is over-written.
	<SetNewBase>	Long	If 1, call SetBaseChannel for the NewBase output.
Return Value	Boolean Returns 1 if the union is successfully created; otherwise, it returns 0		

GetChannelLength

UI Access	NA		
Parameters	<i>Name</i> <ChannelName>	<i>Type</i> String	<i>Description</i> The name of an existing channel.

Return Value	Long Returns the number of points in a channel.

GetChannelMax

UI Access	NA		
Parameters	<i>Name</i> <ChannelName>	<i>Type</i> String	<i>Description</i> The name of an existing channel.
Return Value	Double Returns the maximum value of a channel.		

GetChannelMin

UI Access	NA		
Parameters	<i>Name</i> <ChannelName>	<i>Type</i> String	<i>Description</i> The name of an existing channel.
Return Value	Double Returns the minimum value of a channel.		

GetMonotony

UI Access	NA		
Parameters	<i>Name</i> <ChannelName>	<i>Type</i> String	<i>Description</i> The name of an existing channel.
Return Value	Monotonic		
	Returns the increasing/decreasing behavior.		
	Enum	Monotonic Type	
	monoError	As Long=-1 "Description: Invalid channel state	
	monoNone	As Long=0 "Description: No monotonic behavior	

	monoRising	As Long=1 "Description: Never decreases
	monoStrictRising	As Long=2 "Description: Always increases
	monoFalling	As Long=3 "Description: Never increases
	monoStrictFalling	As Long=4 "Description: Always decreases
	monoConstant	As Long=5 "Description: Never increases or decreases

Interpolate

UI Access	NA		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<OrgChannel>	String	The name of an existing channel for the Y data values.
	<OrgBaseChannel>	String	The name of an existing channel for the X data values. Normally, this would not be the current base channel.
	<NewChannel>	String	The name of an output channel to create with the interpolated and extrapolated values. Any existing channel by this name is overwritten.
	<NewBaseChannel>	Variant	If provided, it is the name of an existing channel of X values to interpolate on. If not provided, the currently defined base channel is used for the interpolation.
Return Value	Boolean Returns 1 if the channel is successfully re-mapped; otherwise, it returns 0.		

PolarToRect

UI Access	NA		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<MagName>	String	The name of an existing magnitude channel.
	<PhaseName>	String	The name of an existing phase channel, in radians.
	<RealName>	String	The name of an output channel to create for

	<ImagName> String the real part. Any existing channel with the same name is overwritten. The name of an output channel to create for the imaginary part. Any existing channel with the same name is overwritten.
Return Value	Boolean Returns 1 if the channel is the conversion succeeds; otherwise, it returns 0.

RectToPolar

UI Access	NA		
Parameters	<i>Name</i> <RealName> String The name of an existing channel for the real part.		
	<ImagName> String The name of an existing channel for the imaginary part.		
	<MagName> String The name of an output channel to create for the magnitude. Any existing channel with the same name is overwritten.		
	<PhaseName> String The name of an output channel to create for the phase, in radians. Any existing channel with the same name is overwritten.		
Return Value	Boolean Returns 1 if the channel is the conversion succeeds; otherwise, it returns 0.		

Channel Calculation Functions

These functions perform calculations that generally correspond to features available in the post-processor.

CalculateCharacteristic

Returns a number based on `ValType`, corresponding to the "characteristics" provided in the post processor.

UI Access	NA		
Parameters	<i>Name</i> <ChannelName>	<i>Type</i> String	<i>Description</i> The name of an existing channel to calculate.
	<ValType>	Characteristic	Specifies the type of characteristic to evaluate.
		Enum	Characteristic Type
		chMin	As Long=1 "Description: Minimum value"
		chMax	As Long=2 "Description: Maximum value"
		chPeakToPeak	As Long=3 "Description: Peak to Peak value"
		chMean	As Long=4 "Description: Mean (DC)"
		chMeanAbsolute	As Long=5 "Description: Mean of rectified (absolute) value"
		chRMS	As Long=6 "Description: Total RMS"
		chAC-RMS	As Long=7 "Description: AC RMS"
		chIntegral	As Long=8 "Description: Integral"
		chRippleFactor	As Long=9 "Description: AC RMS/Mean, called Ripple Value"
		chPkPkToMean	As Long=10 "Description: Peak-to-Peak/Mean, called Ripple Factor"
		chAC-Content	As Long=11

	"Description: AC RMS/RMS, called Harmonic Content"* chFormFactor As Long=12 "Description: RMS/Mean Abs"* chCrestFactor As Long=13 "Description: Peak/RMS"* * For characteristic types marked with an asterisk (*), a base channel must first be set. For more information, see "Base and Reference Channel Functions" on page 1
Return Value	Double Returns a number based on ValType, corresponding to the "characteristics" provided in the post processor.

Differentiation

Differentiates a channel from the base channel and writes the result to an output channel

UI Access	NA		
Parameters	<i>Name</i> <ChannelName>	<i>Type</i> String	<i>Description</i> The name of an existing channel to differentiate
	<OutputName>	String	The name of a channel in which to store the results. Any existing channel with the same name is overwritten
Return Value	Boolean Returns 1 if the action succeeds; otherwise, it returns 0		

FFT

Performs a power-of-2 Fast Fourier Transform (FFT) on a channel.

UI Access	NA		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<ChannelName>	String	The name of the existing channel on which the FFT should be performed. A channel with this name must have been set by a previous call to SetChannel or calculated by one of the calculation functions.
	<OutputName>	String	Scratch channel, which contains the real part of the transform once completed. Any existing channel with this name is overwritten.
	<MagOut>	String	Name of the channel where the magnitude values should be stored. Any existing channel with this name is overwritten.
	<PhRadOut>	String	Name of the channel where the phase values (in radians) should be stored. Any existing channel with this name is overwritten.
	<PhDegOut>	String	Same as PhRadOut, except the units are degrees.
	<RealOut>	String	Name of the channel where the real part values should be stored. Any existing channel with this name is overwritten.
	<ImagOut>	String	Name of the channel where the imaginary part values should be stored. Any existing channel with this name is overwritten.
	<FreqOut>	String	Name of the channel where the transformed frequency values (in Hertz) should be stored. Any existing channel with this name is overwritten.
	<Window>	FFTWindowType	Specifies the window type to be applied to samples before the FFT.

		<table><tr><th>Enum</th><th>FFT Window Type</th></tr><tr><td>fftNoWindow</td><td>As Long=0 ' Description: No window (rectangle)</td></tr><tr><td>fftBartlett</td><td>As Long=1 ' Description: Bartlett window (triangle)</td></tr><tr><td>fftVonHann</td><td>As Long=2 ' Description: Von Hann window (cosine, zero end)</td></tr><tr><td>fftHamming</td><td>As Long=3 ' Description: Hamming window (cos, no zero end)</td></tr><tr><td>fftBlackmann</td><td>As Long=4 ' Description: Blackmann window (double cos)</td></tr><tr><td>fftLanczos</td><td>As Long=5 ' Description: Lanczos window (abs(sin))</td></tr><tr><td>fftWeber</td><td>As Long=6 ' Description: Weber window (polynomial)</td></tr></table>	Enum	FFT Window Type	fftNoWindow	As Long=0 ' Description: No window (rectangle)	fftBartlett	As Long=1 ' Description: Bartlett window (triangle)	fftVonHann	As Long=2 ' Description: Von Hann window (cosine, zero end)	fftHamming	As Long=3 ' Description: Hamming window (cos, no zero end)	fftBlackmann	As Long=4 ' Description: Blackmann window (double cos)	fftLanczos	As Long=5 ' Description: Lanczos window (abs(sin))	fftWeber	As Long=6 ' Description: Weber window (polynomial)
Enum	FFT Window Type																	
fftNoWindow	As Long=0 ' Description: No window (rectangle)																	
fftBartlett	As Long=1 ' Description: Bartlett window (triangle)																	
fftVonHann	As Long=2 ' Description: Von Hann window (cosine, zero end)																	
fftHamming	As Long=3 ' Description: Hamming window (cos, no zero end)																	
fftBlackmann	As Long=4 ' Description: Blackmann window (double cos)																	
fftLanczos	As Long=5 ' Description: Lanczos window (abs(sin))																	
fftWeber	As Long=6 ' Description: Weber window (polynomial)																	
	<MaxAmpl> Double <FilterVal> Double Window types are specified by an integer: 0 : Rectangular 1 : Bartlett 2 : Von Hann 3 : Hamming 4 : Blackmann 5 : Lanczos 6 : Weber The maximum amplitude for % normalization If non-zero, this specifies the % magnitude threshold for displaying a component. If the magnitude is less than this value, it is reported as zero																	
Return Value	Boolean Returns a 1 if the operation succeeds; otherwise, it returns 0																	

GetChannelBasedISE

Calculates the integral squared error, with a variable target. (The CTRL ERR function GetISE, using of SetRealRefChannel and SetBaseChannel, provides the same functionality.)

UI Access	NA

Parameters	<i>Name</i> <Channel>	<i>Type</i> String	<i>Description</i> The name of an existing channel.
	<RefChannel>	String	The name of a channel for the set point.
	<Start>	Variant	Optional calculation starting time with respect to the base channel.
	<End>	Variant	Optional calculation stopping time with respect to the base channel.
Return Value	Double Returns the integral squared error		

Integration

Integrates a channel vs. the base channel and writes the result to an output channel.

UI Access	NA		
Parameters	<i>Name</i> <ChannelName>	<i>Type</i> String	<i>Description</i> The name of an existing channel to integrate.
	<OutputName>	String	The name of a channel to store the results. Any existing channel with the same name is overwritten.
Return Value	Boolean Returns 1 if the integration succeeds; otherwise, it returns 0.		

Smooth

Smooths a channel vs. the base channel and writes the result to an output channel.

UI Access	NA		
Parameters	<i>Name</i> <ChannelName>	<i>Type</i> String	<i>Description</i> The name of an existing channel to differentiate.
	<FilterSize>	Integer	Defines the sample size, in number of data points, for a moving average window of the smoothed output. The sliding window is centered on the current data point.

	<OutputName>	String	The name of a channel to store the results. Any existing channel with the same name is overwritten.
Return Value	Boolean Returns 1 if the action succeeds; otherwise, it returns 0.		

Example: CalculateCharacteristic Function

This example displays all of the characteristics for the example "C3.V" channel.

```

sub ShowDayLabels(xlApp, col, row)
  xlApp.Range(col & row) = "Minimum"
  xlApp.Range(col & row+1) = "Maximum"
  xlApp.Range(col & row+2) = "Peak-to-Peak"
  xlApp.Range(col & row+3) = "Mean"
  xlApp.Range(col & row+4) = "Mean Abs"
  xlApp.Range(col & row+5) = "RMS"
  xlApp.Range(col & row+6) = "AC RMS"
  xlApp.Range(col & row+7) = "Integral"
  ' this is the commonly understood ripple factor, ACRMS / MEAN
  xlApp.Range(col & row+8) = "Ripple Val (ACRMS/MEAN)"
  ' this is actually PK_to_PK / MEAN, also called PKAVG
  xlApp.Range(col & row+9) = "Ripple Factor (PK_PK/MEAN)"
  ' this is actually the ratio ACRMS / RMS, should really call it the AC content
  xlApp.Range(col & row+10) = "Harmonic Content (ACRMS/RMS)"
  xlApp.Range(col & row+11) = "Form Factor (RMS/MEANABS)"
  xlApp.Range(col & row+12) = "Crest Factor (PK/RMS)"
end sub

sub ShowDayFunctions(xlApp, engine, channel, col, row)
  xlApp.Range(col & row) = engine.CalculateCharacteristic (channel, chMin)

```

```
xlApp.Range(col & row+1) = engine.CalculateCharacteristic (channel, chMax)
xlApp.Range(col & row+2) = engine.CalculateCharacteristic (channel, chPeakToPeak)
xlApp.Range(col & row+3) = engine.CalculateCharacteristic (channel, chMean)
xlApp.Range(col & row+4) = engine.CalculateCharacteristic (channel, chMeanAbsolute)
xlApp.Range(col & row+5) = engine.CalculateCharacteristic (channel, chRMS)
xlApp.Range(col & row+6) = engine.CalculateCharacteristic (channel, chACRMS)
xlApp.Range(col & row+7) = engine.CalculateCharacteristic (channel, chIntegral)
xlApp.Range(col & row+8) = engine.CalculateCharacteristic (channel, chRippleFactor)
xlApp.Range(col & row+9) = engine.CalculateCharacteristic (channel, chPkPkToMean)
xlApp.Range(col & row+10) = engine.CalculateCharacteristic (channel, chACContent)
xlApp.Range(col & row+11) = engine.CalculateCharacteristic (channel, chFormFactor)
xlApp.Range(col & row+12) = engine.CalculateCharacteristic (channel, chCrestFactor)
end sub
```

```
ExcelApp.Range("A1") = "Characteristic"
```

```
ExcelApp.Range("B1") = "Value"
```

```
ShowDayLabels ExcelApp, "A", 2
```

```
ShowDayFunctions ExcelApp, MathMod, "C3.V", "B", 2
```

Example: Power-of-2 FFT Function

This example displays the output channels from a power-of-2 FFT on "C3.V".

```
bRet = MathMod.FFT( "C3.V", "C3.V FFT", "FFT Mag", "FFT Rad", "FFT Deg", "FFT Real", "FFT  
Imag", "FFT Freq", fftNoWindow, 1.8084, 1 )
```

```
if bRet = 0 Then
```

```
errmsg = MathMod.GetLastError
```

```
MsgBox( errmsg )
```

```
else
```

```
ShowChannelInColumn ExcelApp, MathMod, "FFT Freq", "D"
```



```
ShowChannelInColumn ExcelApp, MathMod, "FFT Real", "E"  
ShowChannelInColumn ExcelApp, MathMod, "FFT Imag", "F"  
ShowChannelInColumn ExcelApp, MathMod, "FFT Rad", "G"  
ShowChannelInColumn ExcelApp, MathMod, "FFT Deg", "H"  
ShowChannelInColumn ExcelApp, MathMod, "FFT Mag", "I"
```

```
end if
```

Example: Integration, Differentiation, and Smooth Functions

This example displays the output from integration, differentiation, and smoothing on "C3.V", with a time window applied to differentiation.

```
MathMod.SetBaseChannel "t"
```

```
bRet = MathMod.Integration( "C3.V", "C3.V Int")
```

```
MathMod.SetBaseChannel "t", 0.0, 0.0015
```

```
bRet = MathMod.Differentiation( "C3.V", "C3.V Diff 0-1.5m")
```

```
MathMod.SetBaseChannel "t"
```

```
bRet = MathMod.Smooth( "C3.V", 25, "C3.V Smooth 25(V)" )
```

```
ShowChannelInColumn ExcelApp, MathMod, "C3.V Int", "C"
```

```
ShowChannelInColumn ExcelApp, MathMod, "C3.V Diff 0-1.5m", "D"
```

```
ShowChannelInColumn ExcelApp, MathMod, "C3.V Smooth 25(V)", "E"
```

AC Probe Functions

These functions correspond to the AC Probe component. They report the minimum, maximum, and total RMS values of a complex signal in the frequency domain.

GetACFrqMax

Obtains the frequency at which the maximum AC magnitude occurs.

UI Access	NA		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<RealName>	String	The name of an existing input channel for the real part
	<ImagSize>		The name of an existing input channel for the imaginary part.
	<OutputName>		Optional name of an output channel to create with values vs. the base channel. Any existing channel with the same name is overwritten. If not supplied, only the final value is returned from the function call.
Return Value	Double Returns the frequency at which the maximum AC magnitude occurs		

GetACFrqMin

Obtains the frequency at which the minimum AC magnitude occurs

UI Access	NA		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<RealName>	String	The name of an existing input channel for the real part
	<ImagSize>	String	The name of an existing input channel for the imaginary part
	<OutputName>	Variant	Optional name of an output channel to create with values vs. the base channel. Any existing channel with the same name is overwritten. If not supplied, only the final value is returned from the function call
Return Value	Double Returns the frequency at which the minimum AC Magnitude occurs.		

GetACMax

Description

UI Access	NA		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<RealName>	String	The name of an existing input channel for the real part
	<ImagSize>	String	The name of an existing input channel for the imaginary part
	<OutputName>	Variant	Optional name of an output channel to create with values vs. the base channel. Any existing channel with the same name is overwritten. If not supplied, only the final value is returned from the function call.
Return Value	Double Returns the maximum value of the frequency-domain magnitude.		

GetACMin

Obtains the minimum value of the frequency-domain magnitude

UI Access	NA		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<RealName>	String	The name of an existing input channel for the real part
	<ImagSize>	String	The name of an existing input channel for the imaginary part
	<OutputName>	Variant	Optional name of an output channel to create with values vs. the base channel. Any existing channel with the same name is overwritten. If not supplied, only the final value is returned from the function call.
Return Value	Double Returns the minimum value of the frequency-domain magnitude.		

GetACRMS

Obtains the total RMS magnitude in the frequency domain

UI Access	NA		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<RealName>	String	The name of an existing input channel for the real part
	<ImagSize>	String	The name of an existing input channel for the imaginary part
	<OutputName>	Variant	Optional name of an output channel to create with values vs the base channel. Any existing channel with the same name is overwritten. If not supplied, only the final value is returned from the function call.
Return Value	Double Returns the total RMS magnitude in the frequency domain		

Example: AC Probe Functions

This example displays the AC Probe results for a real and imaginary channel pair, previously loaded from an SDB file.

```
ExcelApp.Range("A4") = "Min Mag"
```

```
ExcelApp.Range("B4") = MathMod.GetACMin ("N0011.Re", "N0011.Im")
```

```
ExcelApp.Range("A5") = "Max Mag"
```

```
ExcelApp.Range("B5") = MathMod.GetACMax ("N0011.Re", "N0011.Im")
```

```
ExcelApp.Range("A6") = "Frq Min"
```

```
ExcelApp.Range("B6") = MathMod.GetACFrqMin ("N0011.Re", "N0011.Im")
```

```
ExcelApp.Range("A7") = "Frq Max"
```

```
ExcelApp.Range("B7") = MathMod.GetACFrqMax ("N0011.Re", "N0011.Im")
```

```
ExcelApp.Range("A8") = "RMS Mag"
```

```
ExcelApp.Range("B8") = MathMod.GetACRMS ("N0011.Re", "N0011.Im")
```

CTRL AC Probe Functions

These functions correspond to the CTRL AC Probe component. They report control system response measures that are traditionally read from a Bode plot in the frequency domain.

GetBandwidth

Obtains the bandwidth in Hz (FC2 - FC1). (See also the SetRealRefChannel, SetImagRefChannel, and UnsetRefChannels functions.)

UI Access	NA		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<RealName>	String	The name of an existing input channel for the real part
	<ImagSize>	String	The name of an existing input channel for the imaginary part
	<BaseMag>	Double	The input source magnitude for the gain calculation, in dB (typically 1m). Ignored if SetRealRefChannel and SetImagRefChannel have been called and are still active.
	<BasePhaseDeg>	Double	The input source angle for the phase calculation, in degrees (typically 0). Ignored if SetRealRefChannel and SetImagRefChannel have been called and are still active.
	<OutputName>	Variant	Optional name of an output channel to create with values vs. the base channel. Any existing channel with the same name is overwritten. If not supplied, only the final value is returned from the function call.
Return Value	Double Returns the bandwidth in Hz		

GetGainCrossover

Obtains the gain crossover frequency, in Hz, where the gain is 0 dB. (See also the SetRealRefChannel, SetImagRefChannel, and UnsetRefChannels functions.)

UI Access	NA

Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<RealName>	String	The name of an existing input channel for the real part.
	<ImagSize>	String	The name of an existing input channel for the imaginary part.
	<BaseMag>	Double	The input source magnitude for the gain calculation, in dB (typically 1m). Ignored if SetRealRefChannel and SetImagRefChannel have been called and are still active.
	<BasePhaseDeg>	Double	The input source angle for the phase calculation, in degrees (typically 0). Ignored if SetRealRefChannel and SetImagRefChannel have been called and are still active.
	<OutputName>	Variant	Optional name of an output channel to create with values vs. the base channel. Any existing channel with the same name is overwritten. If not supplied, only the final value is returned from the function call.
Return Value	Double Returns the gain crossover frequency in Hz		

GetGainMargin

Obtains the gain margin, in dB, at the phase crossover frequency. (See also the SetRealRefChannel, SetImagRefChannel, and UnsetRefChannels functions.)

UI Access	NA		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<RealName>	String	The name of an existing input channel for the real part.
	<ImagSize>	String	The name of an existing input channel for the imaginary part.
	<BaseMag>	Double	The input source magnitude for the gain calculation, in dB (typically 1m). Ignored if SetRealRefChannel and SetImagRefChannel have been called and are still active.
	<BasePhaseDeg>	Double	The input source angle for the phase cal-

	<OutputName> Variant ulation, in degrees (typically 0). Ignored if SetRealRefChannel and SetImagRefChannel have been called and are still active. Optional name of an output channel to create with values vs. the base channel. Any existing channel with the same name is overwritten. If not supplied, only the final value is returned from the function call.
Return Value	Double Returns the gain margin in dB

GetLowerCutoff

Obtains the 3dB lower cutoff frequency, in Hz (FC1). (See also the SetRealRefChannel, SetImagRefChannel, and UnsetRefChannels functions.)

UI Access	NA		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<RealName>	String	The name of an existing input channel for the real part.
	<ImagSize>	String	The name of an existing input channel for the imaginary part.
	<BaseMag>	Double	cre still active.
	<BasePhaseDeg>	Double	The input source angle for the phase calculation, in degrees (typically 0). Ignored if SetRealRefChannel and SetImagRefChannel have been called and are still active.
	<OutputName>	Varajr	Optional name of an output channel to create with values vs. the base channel. Any existing channel with the same name is overwritten. If not supplied, only the final value is returned from the function call.
Return Value	Double Returns the 3dB lower cutoff frequency in Hertz		

GetPeakGain

Obtains the peak gain, in dB. (See also the SetRealRefChannel, SetImagRefChannel, and UnsetRefChannels functions.)

UI Access	NA		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<RealName>	String	The name of an existing input channel for the real part.
	<ImagSize>	String	The name of an existing input channel for the imaginary part.
	<BaseMag>	Double	The input source magnitude for the gain calculation, in dB (typically 1m). Ignored if SetRealRefChannel and SetImagRefChannel have been called and are still active.
	<BasePhaseDeg>	Double	The input source angle for the phase calculation, in degrees (typically 0). Ignored if SetRealRefChannel and SetImagRefChannel have been called and are still active.
	<OutputName>	Variant	Optional name of an output channel to create with values vs. the base channel. Any existing channel with the same name is overwritten. If not supplied, only the final value is returned from the function call.
Return Value	Double Returns the peak gain in dB		

GetPeakGainFreq

Obtains the frequency, in Hz, where the peak gain occurs. (See also the SetRealRefChannel, SetImagRefChannel, and UnsetRefChannels functions.)

UI Access	NA		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<RealName>	String	The name of an existing input channel for the real part.
	<ImagSize>	String	The name of an existing input channel for the imaginary part.

	<BaseMag> Double The input source magnitude for the gain calculation, in dB (typically 1m). Ignored if SetRealRefChannel and SetImagRefChannel have been called and are still active. <BasePhaseDeg> Double The input source angle for the phase calculation, in degrees (typically 0). Ignored if SetRealRefChannel and SetImagRefChannel have been called and are still active. <OutputName> Variant Optional name of an output channel to create with values vs. the base channel. Any existing channel with the same name is overwritten. If not supplied, only the final value is returned from the function call.
Return Value	Double Returns the frequency where the peak gain occurs.

GetPhaseCrossover

Obtains the phase crossover frequency, in Hz, where the phase is -180 deg. (See also the SetRealRefChannel, SetImagRefChannel, and UnsetRefChannels functions.)

UI Access	NA		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<RealName>	String	The name of an existing input channel for the real part.
	<ImagSize>	String	The name of an existing input channel for the imaginary part.
	<BaseMag>	Double	The input source magnitude for the gain calculation, in dB (typically 1m). Ignored if SetRealRefChannel and SetImagRefChannel have been called and are still active.
	<BasePhaseDeg>	Double	The input source angle for the phase calculation, in degrees (typically 0). Ignored if SetRealRefChannel and SetImagRefChannel have been called and are still active.
	<OutputName>	Variant	Optional name of an output channel to create with values vs. the base channel. Any

	existing channel with the same name is overwritten. If not supplied, only the final value is returned from the function call.
Return Value	Double Returns the phase crossover frequency.

GetPhaseMargin

Obtains the phase margin, in deg, at the gain crossover frequency. (See also the SetRealRefChannel, SetImagRefChannel, and UnsetRefChannels functions.)

UI Access	NA		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<RealName>	String	The name of an existing input channel for the real part.
	<ImagSize>	String	The name of an existing input channel for the imaginary part.
	<BaseMag>	Double	The input source magnitude for the gain calculation, in dB (typically 1m). Ignored if SetRealRefChannel and SetImagRefChannel have been called and are still active.
	<BasePhaseDeg>	Double	The input source angle for the phase calculation, in degrees (typically 0). Ignored if SetRealRefChannel and SetImagRefChannel have been called and are still active.
	<OutputName>	Variant	Optional name of an output channel to create with values vs. the base channel. Any existing channel with the same name is overwritten. If not supplied, only the final value is returned from the function call.
Return Value	Double Returns the phase margin in deg.		

GetUpperCutoff

Obtains the 3dB upper cutoff frequency, in Hz (FC2). (See also the SetRealRefChannel, SetImagRefChannel, and UnsetRefChannels functions.)

UI Access	NA		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<RealName>	String	The name of an existing input channel for the real part
	<ImagSize>	String	The name of an existing input channel for the imaginary part.
	<BaseMag>	Double	The input source magnitude for the gain calculation, in dB (typically 1m). Ignored if SetRealRefChannel and SetImagRefChannel have been called and are still active.
	<BasePhaseDeg>	Double	The input source angle for the phase calculation, in degrees (typically 0). Ignored if SetRealRefChannel and SetImagRefChannel have been called and are still active.
	<OutputName>	Variant	Optional name of an output channel to create with values vs. the base channel. Any existing channel with the same name is overwritten. If not supplied, only the final value is returned from the function call.
Return Value	Double Returns the 3dB upper cutoff frequency in Hz.		

Example: CTRL AC Probe Functions

This example displays the CTRL AC probe outputs, using a reference channel, with data previously loaded from an SDB file.

```
MathMod.SetBaseChannel "ctrl_f"
```

```
MathMod.SetRealRefChannel "STEP1.Re"
```

```
MathMod.SetImagRefChannel "STEP1.Im"
```

```
ExcelApp.Range("D1") = "GS3 PM"
```

```
ExcelApp.Range("E1") = MathMod.GetPhaseMargin ("GS3.Re", "GS3.Im", 0.0, 0.0)
```

```
ExcelApp.Range("D2") = "GS3 GM"
ExcelApp.Range("E2") = MathMod.GetGainMargin ("GS3.Re", "GS3.Im", 0.0, 0.0)
ExcelApp.Range("D3") = "GS3 PC"
ExcelApp.Range("E3") = MathMod.GetPhaseCrossover ("GS3.Re", "GS3.Im", 0.0, 0.0)
ExcelApp.Range("D4") = "GS3 GC"
ExcelApp.Range("E4") = MathMod.GetGainCrossover ("GS3.Re", "GS3.Im", 0.0, 0.0)
ExcelApp.Range("D5") = "GS3 BW"
ExcelApp.Range("E5") = MathMod.GetBandwidth ("GS3.Re", "GS3.Im", 0.0, 0.0)
ExcelApp.Range("D6") = "GS3 FC1"
ExcelApp.Range("E6") = MathMod.GetLowerCutoff ("GS3.Re", "GS3.Im", 0.0, 0.0)
ExcelApp.Range("D7") = "GS3 FC2"
ExcelApp.Range("E7") = MathMod.GetUpperCutoff ("GS3.Re", "GS3.Im", 0.0, 0.0)
ExcelApp.Range("D8") = "GS3 MP"
ExcelApp.Range("E8") = MathMod.GetPeakGain ("GS3.Re", "GS3.Im", 0.0, 0.0)
ExcelApp.Range("D9") = "GS3 FP"
ExcelApp.Range("E9") = MathMod.GetPeakGainFreq ("GS3.Re", "GS3.Im", 0.0, 0.0)
```

CTRL ERR Probe Functions

These functions correspond to the CTRL ERR Probe component. They report control system error response with respect to a fixed or variable setpoint.

GetFinalError

Obtains the difference between the channel value and SetPoint. (See also the SetRealRefChannel and UnsetRefChannels functions.)

UI Access	NA		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<ChannelName>	String	The name of an existing input channel.
	<SetPoint>	Double	The target value for the input channel. Ignored if SetRealRefChannel has been called and is still active.
	<OutputName>	Variant	Optional name of an output channel to cre-

	ate with values vs. the base channel. Any existing channel with the same name is over-written. If not supplied, only the final value is returned from the function call.
Return Value	Double Returns the difference between the channel value and SetPoint.

GetIAE

Obtains the integral of the absolute deviation from SetPoint. (See also the `SetRealRefChannel` and `UnsetRefChannels` functions.)

UI Access	NA		
Parameters	<i>Name</i> <ChannelName> <SetPoint> <OutputName>	<i>Type</i> String Double Variant	<i>Description</i> The name of an existing input channel. The target value for the input channel. Ignored if SetRealRefChannel has been called and is still active. Optional name of an output channel to create with values vs. the base channel. Any existing channel with the same name is over-written. If not supplied, only the final value is returned from the function call.
Return Value	Double Returns the integral of the absolute deviation from SetPoint.		

GetISE

Obtains the integral of the squared deviation from SetPoint. (See also the `SetRealRefChannel` and `UnsetRefChannels` functions.)

UI Access	NA		
Parameters	<i>Name</i> <ChannelName> <SetPoint>	<i>Type</i> String Double	<i>Description</i> The name of an existing input channel The target value for the input channel. Ignored if SetRealRefChannel has been

	<OutputName> Variant called and is still active. Optional name of an output channel to create with values vs. the base channel. Any existing channel with the same name is overwritten. If not supplied, only the final value is returned from the function call.
Return Value	Double Returns the integral of the squared deviation from SetPoint

GetITAE

Obtains the time-weighted absolute deviation from SetPoint. (See also the `SetRealRefChannel` and `UnsetRefChannels` functions.)

UI Access	NA		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<ChannelName>	String	The name of an existing input channel.
	<SetPoint>	Double	The target value for the input channel. Ignored if <code>SetRealRefChannel</code> has been called and is still active.
	<OutputName>	Variant	Optional name of an output channel to create with values vs. the base channel. Any existing channel with the same name is overwritten. If not supplied, only the final value is returned from the function call.
Return Value	Double Returns the time-weighted absolute deviation from SetPoint.		

GetITSE

Obtains the time-weighted squared deviation from SetPoint. (See also the `SetRealRefChannel` and `UnsetRefChannels` functions.)

UI Access	NA		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>

	<ChannelName> String The name of an existing input channel. <SetPoint> Double The target value for the input channel. Ignored if SetRealRefChannel has been called and is still active. <OutputName> Variant Optional name of an output channel to create with values vs. the base channel. Any existing channel with the same name is overwritten. If not supplied, only the final value is returned from the function call.
Return Value	Double Returns the time-weighted squared deviation from SetPoint.

Example: CTRL ERR Probe Functions

This example shows the summary results from the CTRL ERR probe functions, on the "C3.V" channel data with a fixed setpoint of 1.0.

```
sub ShowErrLabels(xlApp, col, row)
xlApp.Range(col & row) = "IAE"
xlApp.Range(col & row+1) = "ISE"
xlApp.Range(col & row+2) = "ITAE"
xlApp.Range(col & row+3) = "ITSE"
xlApp.Range(col & row+4) = "Final Err"
end sub
```

```
sub ShowErrFunctions(xlApp, engine, channel, col, row, dTarget)
xlApp.Range(col & row) = engine.GetIAE (channel, dTarget)
xlApp.Range(col & row+1) = engine.GetISE (channel, dTarget)
xlApp.Range(col & row+2) = engine.GetITAE (channel, dTarget)
xlApp.Range(col & row+3) = engine.GetITSE (channel, dTarget)
xlApp.Range(col & row+4) = engine.GetFinalError (channel, dTarget)
end sub
```

dTarget = 1.0

row = 13

ShowErrLabels ExcelApp, "D", row

ShowErrFunctions ExcelApp, MathMod, "C3.V", "E", row, dTarget

CTRL TR Probe Functions

These functions correspond to the CTRL TR Component. They evaluate control system time response with respect to a fixed setpoint.

GetDeadTime

Obtains the latest time when the channel is within a tolerance of zero

UI Access	NA		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<ChannelName>	String	The name of an existing input channel
	<Tolerance>	Double	The +/- bandwidth around zero
	<OutputName>	Variant	Optional name of an output channel to create with values vs. the base channel. Any existing channel with the same name is overwritten. If not supplied, only the final value is returned from the function call.
Return Value	Double Returns the latest time when the channel is within a tolerance of zero		

GetDelayTime

Obtains the time from zero to 50% of SetPoint.

UI Access	NA		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<ChannelName>	String	The name of an existing input channel

	<SetPoint> Double The target value for the input. Remains constant. SetRealRefChannel is ignored for this function. <OutputName> Variant Optional name of an output channel to create with values vs. the base channel. Any existing channel with the same name is overwritten. If not supplied, only the final value is returned from the function call.
Return Value	Double Returns the time from zero to 50% of SetPoint.

GetOvershoot

Obtains the peak overshoot over `SetPoint`.

UI Access	NA		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<ChannelName>	String	The name of an existing input channel.
	<SetPoint>	Double	The target value for the input. Remains constant. <code>SetRealRefChannel</code> is ignored for this function.
	<OutputName>	Variant	Optional name of an output channel to create with values vs. the base channel. Any existing channel with the same name is overwritten. If not supplied, only the final value is returned from the function call.
Return Value	Double Returns the peak overshoot.		

GetOvershootTime

Obtains the time at which the peak overshoot occurs.

UI Access	NA		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>

	<ChannelName> String The name of an existing input channel. <SetPoint> Double The target value for the input. Remains constant. SetRealRefChannel is ignored for this function. <OutputName> Variant Optional name of an output channel to create with values vs. the base channel. Any existing channel with the same name is overwritten. If not supplied, only the final value is returned from the function call.
Return Value	Double Returns the time at which the peak overshoot occurs.

GetRiseTime

Obtains the time the channel takes to go from 10% to 90% of SetPoint.

UI Access	NA		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<ChannelName>	String	The name of an existing input channel.
	<SetPoint>	Double	The target value for the input. Remains constant. SetRealRefChannel is ignored for this function.
	<OutputName>	Variant	Optional name of an output channel to create with values vs. the base channel. Any existing channel with the same name is overwritten. If not supplied, only the final value is returned from the function call.
Return Value	Double Returns the time the channel takes to go from 10% to 90% of SetPoint.		

GetSettlingTime

Obtains the latest time when the value of the channel fell outside the `Tolerance` band.

UI Access	NA

Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<ChannelName>	String	The name of an existing input channel.
	<SetPoint>	Double	The target value for the input. Remains constant. SetRealRefChannel is ignored for this function.
	<Tolerance>	Double	The +/- bandwidth around the SetPoint.
	<OutputName>	Variant	Optional name of an output channel to create with values vs. the base channel. Any existing channel with the same name is overwritten. If not supplied, only the final value is returned from the function call.
Return Value	None		

Example: CTRL TR Probe Functions

This example shows the CTRL TR probe summary results for the "C3.V" channel.

```
sub ShowOvrLabels(xlApp, col, row)
```

```
xlApp.Range(col & row) = "Overshoot"
```

```
xlApp.Range(col & row+1) = "Overshoot Time"
```

```
xlApp.Range(col & row+2) = "Settling Time"
```

```
xlApp.Range(col & row+3) = "Rise Time"
```

```
xlApp.Range(col & row+4) = "Delay Time"
```

```
xlApp.Range(col & row+5) = "Dead Time"
```

```
end sub
```

```
sub ShowOvrFunctions(xlApp, engine, channel, col, row, dTarget, dTol, dZero)
```

```
xlApp.Range(col & row) = engine.GetOvershoot (channel, dTarget)
```

```
xlApp.Range(col & row+1) = engine.GetOvershootTime (channel, dTarget)
```

```
xlApp.Range(col & row+2) = engine.GetSettlingTime (channel, dTarget, dTol)
```

```
xlApp.Range(col & row+3) = engine.GetRiseTime (channel, dTarget)
```

```
xlApp.Range(col & row+4) = engine.GetDelayTime (channel, dTarget)
```

```
xlApp.Range(col & row+5) = engine.GetDeadTime (channel, dZero)
```

```
end sub
```

dTarget = 1.0

dTolerance = 0.2

dZero = 0.02

row = row + 1 + 5

ShowOvrLabels ExcelApp, "D", row

ShowOvrFunctions ExcelApp, MathMod, "C3.V", "E", row, dTarget, dTolerance, dZero

F Probe Functions

This function corresponds to the F Probe component. It reports a single value by logarithmic interpolation, which is typically used for interpolating a magnitude response in the frequency domain.

GetValueLogInterpolation

Obtains the log-interpolated channel value at the base frequency (Base).

UI Access	NA		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<ChannelName>	String	The name of an existing input channel.
	<Base>	Double	The value of the base channel frequency, in Hertz, at which to interpolate.
	<OutputName>	Variant	Optional name of an output channel to create with values vs. the base channel. Any existing channel with the same name is overwritten. If not supplied, only the final value is returned from the function call.
Return Value	Double		
	Returns the log-interpolated channel value at the base frequency.		

Example: F Probe Functions

This example shows the different results of linear and logarithmic interpolation at 200 Hz, for two real and imaginary channels previously loaded from an SDB file.

```
MathMod.SetBaseChannel "filter_f"
```

```
MathMod.RectToPolar "N0011.Re", "N0011.Im", "N0011.Mag", "N0011.Phs"
```

```
ExcelApp.Range("A1") = "Linear N0011 at 200"
```

```
ExcelApp.Range("B1") = MathMod.GetValue ("N0011.Mag", 200.0)
```

```
ExcelApp.Range("A2") = "Log N0011 at 200"
```

```
ExcelApp.Range("B2") = MathMod.GetValueLogInterpolation ("N0011.Mag", 200.0)
```

FFT Probe Functions

These functions correspond to the FFT Probe component, performing a mixed-radix Fast Fourier Transform on an input channel vs. time. To use these functions, SetupFFT must first be called, then any of the other functions may be called to access the results. To perform another FFT, the SetupFFT function must be called again.

SetupFFT

Performs a mixed-radix FFT on a channel. Results available via the `GetFFT*` functions.

UI Access	NA		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<ChannelName>	String	The name of an existing input channel.
	<WindowType>	Integer	Specifies the window type that is to be applied to samples before the FFT.
		Enum	FFT Window Type
		fftNoWindow	As Long=0 "Description: No window (rectangle)"
		fftBartlett	As Long=1 "Description: Bartlett window (triangle)"
		fftvonHann	As Long=2 "Description: VonHann window (cosine, zero end)"
		fftHamming	As Long=3 "Description:

			Hamming window (cosine, no zero end)" fftBlackmann As Long=4 "Description: Blackmann window (double cos)" fftLanczos As Long=5 "Description: Lanczos window (abs(sin))" fftWeber As Long=6 "Description: Weber window (polynomial)"
	<FundFreq>	Double	The fundamental frequency, in Hz.
	<HighestOrder>	Integer	The highest integer harmonic order for which results are needed. Sampling is set so that the Nyquist rate is the HighestOrder + 1 harmonic.
	<Normalize>	Integer	Set to 1 to normalize the magnitude outputs as a percentage of the fundamental. Set to 0 for actual magnitude outputs.
	<Delay>	Double	Set a non-zero time delay to start the FFT computation.
	<MagFilter>	Double	A percentage of the fundamental. If non-zero, then the components of magnitude less than MagFilter are reported as zero.
Return Value	Boolean Returns 1 if the FFT succeeds; otherwise, it returns 0.		

GetFFTFreq

Saves the FFT frequency base in a channel.

UI Access	NA		
Parameters	<i>Name</i> <OutputName>	<i>Type</i>	<i>Description</i> The name of an output channel to create with values vs. the base channel. Any existing channel by the same name is overwritten. For GetFFTFreq, GetFFTReal, GetFFTImag, GetFFTMag, and GetFFT-Phase, the output is vs. frequency.

	For function names beginning with <code>GetFFTResult</code> , the <code>OutputName</code> parameter is optional. If provided, the output is vs. time. If not provided, only the final value is returned from the function call.
Return Value	Boolean Returns 1 if the frequency is successfully saved; otherwise, it returns 0.

GetFFTImag

Saves the imaginary part in a channel

UI Access	NA		
Parameters	<i>Name</i> <OutputName>	<i>Type</i> String	<i>Description</i> The name of an output channel to create with values vs. the base channel. Any existing channel by the same name is over-written. The output is vs. frequency.
Return Value	Boolean Returns 1 if the save succeeds; otherwise, it returns 0.		

GetFFTMag

Saves the FFT magnitude in a channel.

UI Access	NA		
Parameters	<i>Name</i> <InDB>	<i>Type</i> Integer	<i>Description</i> Set to 1 to output the magnitude in dB relative to the fundamental; set 0 to output the actual magnitude.
	<OutputName>	String	The name of an output channel to create with values vs. the base channel. Any existing channel by the same name is over-written.

	The output is vs. frequency.
Return Value	Boolean Returns 1 if the save succeeds, otherwise, it returns 0.

GetFFTPHase

Saves the FFT phase in a channel

UI Access	NA		
Parameters	<i>Name</i> <InDegrees>	<i>Type</i> Integer	<i>Description</i> Set to 1 to output degrees; set to 0 to output radians
	<Wrap>	Integer	Set to 1 to output angles shifted into the range -180 to +180 degrees or - to + radians. Set to 0 for a continuous angle output.
	<OutputName>	String	The name of an output channel to create with values vs. the base channel. Any existing channel by the same name is over-written. The output is vs. frequency.
Return Value	Boolean Returns 1 if the save succeeds, otherwise, it returns 0.		

GetFFTReal

Saves the real part in a channel

UI Access	NA		
Parameters	<i>Name</i> <OutputName>	<i>Type</i> String	<i>Description</i> The name of an output channel to create with values vs. the base channel. Any existing channel by the same name is over-written.
	The output is vs. frequency.		
Return Value	Boolean		

	Returns 1 if the save succeeds; otherwise, it returns 0.

GetFFTResultCIT

Obtains the C-message IT product.

UI Access	NA		
Parameters	<i>Name</i> <OutputName>	<i>Type</i> Variant	<i>Description</i> The name of an output channel to create with values vs. the base channel. Any existing channel by the same name is over-written. For GetFFTFreq, GetFFTReal, GetFFTImag, GetFFTMag, and GetFFTPhase, the output is vs. frequency. For function names beginning with GetFFTResult, the OutputName parameter is optional. If provided, the output is vs. time. If not provided, only the final value is returned from the function call.
Return Value	Double Returns the C-Message IT Product		

GetFFTResultCMWI

Obtains the C-message influence factor.

UI Access	NA		
Parameters	<i>Name</i> <OutputName>	<i>Type</i> Variant	<i>Description</i> The name of an output channel to create with values vs. the base channel. Any existing channel by the same name is over-written. Optional. If provided, the output is vs. time. If not provided, only the final value is returned from the function call.

Return Value	Double Returns the C-mesage influence factor

GetFFTResultDF

Obtains the distortion factor.

UI Access	NA		
Parameters	<i>Name</i> <OutputName>	<i>Type</i> Variant	<i>Description</i> The name of an output channel to create with values vs. the base channel. Any existing channel by the same name is over-written. Optional. If provided, the output is vs. time. If not provided, only the final value is returned from the function call.
Return Value	Double Returns the distortion factor.		

GetFFTResultIECClassA

Performs an *IEC 1000-3-2 Class A* current limit check on the last FFT result.

UI Access	NA		
Parameters	<i>Name</i> None	<i>Type</i>	<i>Description</i>
Return Value	Variant Returns a String		

GetFFTResultIECClassB

Performs an *IEC 1000-3-2 Class B* current limit check on the last FFT result.

UI Access	NA		
Parameters	<i>Name</i> None	<i>Type</i>	<i>Description</i>
Return Value	Variant Returns a String		

GetFFTResultIECClassC

Performs an *IEC 1000-3-2 Class C* current limit check on the last FFT result.

UI Access	NA		
Parameters	<i>Name</i> <PowerFactor>	<i>Type</i> Double	<i>Description</i> For a harmonic check function, this is the circuit power factor
Return Value	Variant Returns a String		

GetFFTResultIECClassD

Performs an *IEC 1000-3-2 Class D* current limit check on the last FFT result.

UI Access	NA		
Parameters	<i>Name</i> <Power>	<i>Type</i> Double	<i>Description</i> For a harmonic check function, this is the load nominal power, in W
Return Value	Variant Returns a String		

GetFFTResultIECInstallation

Performs an *IEC 1000-3-4* current limit check for total installation on the last FFT result.

UI Access	NA		
Parameters	<i>Name</i> None	<i>Type</i>	<i>Description</i>
Return Value	Variant Returns a String		

GetFFTResultIECThreePhase

Performs an *IEC 1000-3-4* current limit check for 3-phase equipment on the last FFT result.

UI Access	NA		
Parameters	<i>Name</i> <ShortCircuitRatio>	<i>Type</i> Double	<i>Description</i> For a harmonic check function, this is the ratio of system short-circuit strength, in MVA or KVA, at the point of common coupling to the load size, in MVA or KVA.
Return Value	Variant Returns a String		

GetFFTResultIEEE519Current

Performs an *IEEE 519-1992* current limit check on the last FFT result.

UI Access	NA		
Parameters	<i>Name</i> <ShortCircuitRatio>	<i>Type</i> Double	<i>Description</i> For a harmonic check function, this is the ratio of system short-circuit strength, in MVA or KVA, at the point of common coupling to the load size, in MVA or KVA.
Return Value	Variant Returns a String		

GetFFTResultIEEE519Voltage

Performs an *IEEE 519-1992* voltage limit check on the last FFT result.

UI Access	NA		
Parameters	<i>Name</i> <BusKV>	<i>Type</i> Double	<i>Description</i> For an IEEE 519 voltage check, this is the system line-to-line bus voltage, in kV, at the point of common coupling.
Return Value	Variant Returns a string		

GetFFTResultIT

Obtains the IT interference product.

UI Access	NA		
Parameters	<i>Name</i> <OutputName>	<i>Type</i> Variant	<i>Description</i> The name of an output channel to create with values vs. the base channel. Any existing channel by the same name is overwritten. For GetFFTFreq, GetFFTReal, GetFFTImag, GetFFTMag, and GetFFTPhase, the output is vs. frequency. For function names beginning with GetFFTResult, the OutputName parameter is optional. If provided, the output is vs. time. If not provided, only the final value is returned from the function call.
Return Value	Double Returns the interference product.		

GetFFTResultKFactor

Obtains the transformer K-factor

UI Access	NA		
Parameters	<i>Name</i> <OutputName>	<i>Type</i> Variant	<i>Description</i> The name of an output channel to create with values vs. the base channel. Any existing channel by the same name is over-written. Optional. If provided, the output is vs. time. If not provided, only the final value is returned from the function call.
Return Value	Double Returns the transformer K-factor.		

GetFFTResultMagnitude

Obtains the magnitude at a given order

UI Access	NA		
Parameters	<i>Name</i> <Order>	<i>Type</i> Integer	<i>Description</i> Specifies the harmonic order (for accessing an individual component in GetFFTResult-Magnitude or GetFFTResultPhaseDeg).
	<OutputName>	Variant	The name of an output channel to create with values vs. the base channel. Any existing channel by the same name is over-written. Optional. If provided, the output is vs. time. If not provided, only the final value is returned from the function call.
Return Value	Double Returns the magnitude at the given order		

GetFFTResultPhaseDeg

Obtains the phase, in deg, at a given order.

UI Access	NA		
Parameters	<i>Name</i> <Order>	<i>Type</i> Integer	<i>Description</i> Specifies the harmonic order (for accessing an individual component in GetFFTResult-Magnitude or GetFFTResultPhaseDeg).
	<OutputName>	Variant	The name of an output channel to create with values vs. the base channel. Any existing channel by the same name is over-written. Optional. If provided, the output is vs. time. If not provided, only the final value is returned from the function call.
Return Value	Double Returns the phase at the given order.		

GetFFTResultRipple

Obtains the ripple factor (ACRMS/DCRMS)

UI Access	NA		
Parameters	<i>Name</i> <OutputName>	<i>Type</i> Variant	<i>Description</i> The name of an output channel to create with values vs. the base channel. Any existing channel by the same name is over-written. Optional. If provided, the output is vs. time. If not provided, only the final value is returned from the function call.
Return Value	Double Returns the ripple factor		

GetFFTResultRMS1

Obtains the RMS of the fundamental

UI Access	NA		
Parameters	<i>Name</i> <OutputName>	<i>Type</i> Variant	<i>Description</i> The name of an output channel to create with values vs. the base channel. Any existing channel by the same name is over-written. Optional. If provided, the output is vs. time. If not provided, only the final value is returned from the function call.
Return Value	Double Returns the RMS of the fundamental		

GetFFTResultRMSAC

Obtains the AC RMS

UI Access	NA		
Parameters	<i>Name</i> <OutputName>	<i>Type</i> Variant	<i>Description</i> The name of an output channel to create with values vs. the base channel. Any existing channel by the same name is over-written. Optional. If provided, the output is vs. time. If not provided, only the final value is returned from the function call.
Return Value	Double Returns the AC RMS		

GetFFTResultRMSDC

Obtains the DC RMS (mean).

UI Access	NA		
Parameters	<i>Name</i> <OutputName>	<i>Type</i> Variant	<i>Description</i> The name of an output channel to create

	with values vs. the base channel. Any existing channel by the same name is over-written. Optional. If provided, the output is vs. time. If not provided, only the final value is returned from the function call.
Return Value	Double Returns the DC RMS (mean)

GetFFTResultTHD

Obtains the total harmonic distortion, in %.

UI Access	NA		
Parameters	<i>Name</i> <OutputName>	<i>Type</i> Variant	<i>Description</i> The name of an output channel to create with values vs. the base channel. Any existing channel by the same name is over-written.
			Optional. If provided, the output is vs. time. If not provided, only the final value is returned from the function call.
Return Value	Double Returns the total harmonic distortion		

GetFFTResultTIF

Obtains the telephone influence factor

UI Access	NA		
Parameters	<i>Name</i> <OutputName> <i>Type</i> Variant <i>Description</i> The name of an output channel to create with values vs. the base channel. Any existing channel by the same name is over-written.		

Return Value	Double Returns the telephone influence factor.

Example: FFT Probe Functions

This example performs an FFT of the example "C3.V" channel up to the 37th order, and displays summary results.

```
sub ShowFFTLLabels(xlApp, col, row)
xlApp.Range(col & row) = "RMS DC"
xlApp.Range(col & row+1) = "RMS 1"
xlApp.Range(col & row+2) = "RMS AC"
xlApp.Range(col & row+3) = "THD"
xlApp.Range(col & row+4) = "Ripple"
xlApp.Range(col & row+5) = "K Factor"
xlApp.Range(col & row+6) = "TIF"
xlApp.Range(col & row+7) = "IT"
xlApp.Range(col & row+8) = "CMWI"
xlApp.Range(col & row+9) = "CIT"
xlApp.Range(col & row+10) = "DF"
xlApp.Range(col & row+11) = "Mag(3)"
xlApp.Range(col & row+12) = "Deg(3)"
end sub
```

```
sub ShowFFTFunctions(xlApp, engine, channel, col, row, iWindow, dFund, iOrder, bNormal,
dDelay, dFilter)
```

```
engine.SetupFFT channel, iWindow, dFund, iOrder, bNormal, dDelay, dFilter
```

```
xlApp.Range(col & row) = engine.GetFFTResultRMSDC
```

```
xlApp.Range(col & row+1) = engine.GetFFTResultRMS1
```

```
xlApp.Range(col & row+2) = engine.GetFFTResultRMSAC
```

```
xlApp.Range(col & row+3) = engine.GetFFTResultTHD
xlApp.Range(col & row+4) = engine.GetFFTResultRipple
xlApp.Range(col & row+5) = engine.GetFFTResultKFactor
xlApp.Range(col & row+6) = engine.GetFFTResultTIF
xlApp.Range(col & row+7) = engine.GetFFTResultIT
xlApp.Range(col & row+8) = engine.GetFFTResultCMWI
xlApp.Range(col & row+9) = engine.GetFFTResultCIT
xlApp.Range(col & row+10) = engine.GetFFTResultDF
xlApp.Range(col & row+11) = engine.GetFFTResultMagnitude (3)
xlApp.Range(col & row+12) = engine.GetFFTResultPhaseDeg (3)
end sub
```

```
iWindow = fftNoWindow
```

```
dFund = 1.0 / 0.0006
```

```
iOrder = 37
```

```
bNormal = 0
```

```
dDelay = 0.0
```

```
dFilter = 0.0
```

```
row = 1
```

```
ShowFFTLabels ExcelApp, "K", row
```

```
ShowFFTFunctions ExcelApp, MathMod, "C3.V", "L", row, iWindow, dFund, iOrder, bNormal,
dDelay, dFilter
```

Example: FFT Probe Output Channel Functions

This example displays the magnitude and phase (in degrees) vs. frequency, and the fundamental RMS vs. time, after calculating the FFT by a call to SetupFFT.

```
inDB = 0
```

```
inDegrees = 1
```

bWrap = 1

MathMod.GetFFTFreq "FFTW(f)"

MathMod.GetFFTMag inDB, "FFTW(mag)"

MathMod.GetFFTPhase inDegrees, bWrap, "FFTW(deg)"

ShowChannelInColumn ExcelApp, MathMod, "FFTW(f)", "N"

ShowChannelInColumn ExcelApp, MathMod, "FFTW(mag)", "O"

ShowChannelInColumn ExcelApp, MathMod, "FFTW(deg)", "P"

MathMod.GetFFTResultRMS1 "RMS1"

ShowChannelInColumn ExcelApp, MathMod, "t", "R"

ShowChannelInColumn ExcelApp, MathMod, "RMS1", "S"

Example: Harmonic Checking Functions

These functions may be called after SetupFFT has been called. The return value is a string "PASSED" if the magnitude spectrum meets all criteria. If the magnitude spectrum violates the standard, the return value will contain the word "FAILED" and one or more of the following indicators:

- "Overall" - the total harmonic distortion is too high
- "2" ... "63" - the listed individual harmonic component is too high
- "Misc" - an individual component higher than the 63rd order is too high, or the partially weighted harmonic distortion is too high

ExcelApp.Range("T1") = "519 I"

ExcelApp.Range("U1") = MathMod.GetFFTResultIEEE519Current (50.0)

ExcelApp.Range("T2") = "519 V"

ExcelApp.Range("U2") = MathMod.GetFFTResultIEEE519Voltage (138.0)

```
ExcelApp.Range("T3") = "Class A"
```

```
ExcelApp.Range("U3") = MathMod.GetFFTResultIECClassA
```

```
ExcelApp.Range("T4") = "Class B"
```

```
ExcelApp.Range("U4") = MathMod.GetFFTResultIECClassB
```

```
ExcelApp.Range("T5") = "Class C"
```

```
ExcelApp.Range("U5") = MathMod.GetFFTResultIECClassC (0.8)
```

```
ExcelApp.Range("T6") = "Class D"
```

```
ExcelApp.Range("U6") = MathMod.GetFFTResultIECClassD (250.0)
```

```
ExcelApp.Range("T7") = "Three Phase"
```

```
ExcelApp.Range("U7") = MathMod.GetFFTResultIECThreePhase (50.0)
```

```
ExcelApp.Range("T8") = "Installation"
```

```
ExcelApp.Range("U8") = MathMod.GetFFTResultIECInstallation
```

Moving Average Functions

These functions correspond to the Moving_Average math block.

GetSlidingMean

Obtains the moving average calculation of the mean, integrating over Period.

UI Access	NA		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<ChannelName>	String	The name of an existing input channel.
	<Period>	Double	The integration period, in s, for the moving average. Must be greater than zero.
	<OutputName>	Variant	Optional name of an output channel to cre-

	ate with values vs. the base channel. Any existing channel with the same name is overwritten. If not supplied, only the final value is returned from the function call.
Return Value	Double Returns the moving average calculation of the mean.

GetSlidingRMS

Obtains the moving average calculation of the RMS, integrating over Period

UI Access	NA		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<ChannelName>	String	The name of an existing input channel.
	<Period>	Double	The integration period, in s, for the moving average. Must be greater than zero.
	<OutputName>	Variant	Optional name of an output channel to create with values vs. the base channel. Any existing channel with the same name is overwritten. If not supplied, only the final value is returned from the function call.
Return Value	Double Returns the moving average calculation of the RMS.		

Example: Moving Average Functions

This example displays the final sliding mean and RMS value of the example "C3.V" channel over a 0.5 ms period, and then re-calculates these two quantities to provide output channels vs. time.

```
sub ShowMaLabels(xlApp, col, row)
xlApp.Range(col & row) = "Sliding Mean"
xlApp.Range(col & row+1) = "Sliding RMS"
end sub
```

```

sub ShowMaFunctions(xlApp, engine, channel, col, row, dPeriod)
xlApp.Range(col & row) = engine.GetSlidingMean (channel, dPeriod)
xlApp.Range(col & row+1) = engine.GetSlidingRMS (channel, dPeriod)
end sub

row = row + 1 + 6

```

```
dPeriod = 0.0005
```

```

ShowMaLabels ExcelApp, "D", row
ShowMaFunctions ExcelApp, MathMod, "C3.V", "E", row, dPeriod

```

```

MathMod.GetSlidingRMS "C3.V", dPeriod, "RMS"
MathMod.GetSlidingMean "C3.V", dPeriod, "Mean"

```

```

ShowChannelInColumn ExcelApp, MathMod, "t", "G"
ShowChannelInColumn ExcelApp, MathMod, "C3.V", "H"
ShowChannelInColumn ExcelApp, MathMod, "RMS", "I"
ShowChannelInColumn ExcelApp, MathMod, "Mean", "J"

```

PULSE Probe Functions

These functions correspond to the PULSE Probe component. They report front time and tail time values with respect to the actual peak value of a channel, not with respect to a known set point.

GetPulseFall9010

Obtains the pulse fall time, by the 90%-10% estimate, within the base channel Start/Stop values.

UI Access	NA		
Parameters	<i>Name</i> <ChannelName>	<i>Type</i> String	<i>Description</i> The name of an existing input channel.

	<OutputName>	Variant	Optional name of an output channel to create with values vs. the base channel. Any existing channel with the same name is overwritten. If not supplied, only the final value is returned from the function call.
Return Value	Double Returns the pulse fall time.		

GetPulseFront1090

Obtains the pulse front time, by the 10%-90% estimate, within the base channel Start/Stop values.

UI Access	NA		
Parameters	<i>Name</i> <ChannelName> <OutputName>	<i>Type</i> String Variant	<i>Description</i> The name of an existing input channel. Optional name of an output channel to create with values vs. the base channel. Any existing channel with the same name is overwritten. If not supplied, only the final value is returned from the function call.
Return Value	Double Returns the pulse fall time.		

GetPulseFront3090

Obtains the pulse front time, by the 30%-90% estimate, within the base channel Start/Stop values.

UI Access	NA		
Parameters	<i>Name</i> <ChannelName> <OutputName>	<i>Type</i> String Variant	<i>Description</i> The name of an existing input channel. Optional name of an output channel to create with values vs. the base channel. Any existing channel with the same name is overwritten. If not supplied, only the final value is

	returned from the function call.
Return Value	Double Returns the pulse fall time.

GetPulseMax

Obtains the pulse maximum for the front and tail estimates, within the base channel Start/Stop values.

UI Access	NA		
Parameters	<i>Name</i> <ChannelName> <OutputName>	<i>Type</i> String Variant	<i>Description</i> The name of an existing input channel. Optional name of an output channel to create with values vs. the base channel. Any existing channel with the same name is overwritten. If not supplied, only the final value is returned from the function call.
Return Value	Double Returns the pulse max.		

GetPulseMaxTime

Obtains the time at which the pulse max occurs.

UI Access	NA		
Parameters	<i>Name</i> <ChannelName> <OutputName>	<i>Type</i> String Variant	<i>Description</i> The name of an existing input channel. Optional name of an output channel to create with values vs. the base channel. Any existing channel with the same name is overwritten. If not supplied, only the final value is returned from the function call.
Return Value	Double Returns the time when the pulse max occurs.		

GetPulseMin

Obtains the pulse minimum for the front and tail estimates, within the base channel Start/Stop values.

UI Access	NA		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<ChannelName> <OutputName>	String Variant	The name of an existing input channel. Optional name of an output channel to create with values vs. the base channel. Any existing channel with the same name is overwritten. If not supplied, only the final value is returned from the function call.
Return Value	Double Returns the pulse min.		

GetPulseMinTime

Obtains the time at which the pulse min occurs.

UI Access	NA		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<ChannelName> <OutputName>	String Variant	The name of an existing input channel. Optional name of an output channel to create with values vs. the base channel. Any existing channel with the same name is overwritten. If not supplied, only the final value is returned from the function call.
Return Value	Double Returns the time when the pulse min occurs.		

GetPulseTail150

Obtains the pulse tail time from virtual peak to 50% on the tail, within the base channel Start/Stop values.

UI Access	NA		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<ChannelName> <OutputName>	String Variant	The name of an existing input channel. Optional name of an output channel to create with values vs. the base channel. Any existing channel with the same name is overwritten. If not supplied, only the final value is returned from the function call.
Return Value	Double Returns the pulse tail time.		

GetPulseWidth5050

Obtains the pulse width between 50% points on the front and tail, within the base channel Start/Stop values.

UI Access	NA		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<ChannelName> <OutputName>	String Variant	The name of an existing input channel. Optional name of an output channel to create with values vs. the base channel. Any existing channel with the same name is overwritten. If not supplied, only the final value is returned from the function call.
Return Value	Double Returns the pulse width.		

Example: PULSE Probe Functions

This example displays a summary of the pulse time measurements for the "C3.V" channel.

```
sub ShowPlsLabels(xlApp, col, row)
xlApp.Range(col & row) = "Pulse Min"
xlApp.Range(col & row+1) = "Pulse Max"
xlApp.Range(col & row+2) = "Pulse Tmin"
xlApp.Range(col & row+3) = "Pulse Tmax"
xlApp.Range(col & row+4) = "Pulse T10_90"
xlApp.Range(col & row+5) = "Pulse T30_90"
xlApp.Range(col & row+6) = "Pulse T90_10"
xlApp.Range(col & row+7) = "Pulse T50_50"
xlApp.Range(col & row+8) = "Pulse T100_50"
end sub
```

```
sub ShowPlsFunctions(xlApp, engine, channel, col, row)
xlApp.Range(col & row) = engine.GetPulseMin (channel)
xlApp.Range(col & row+1) = engine.GetPulseMax (channel)
xlApp.Range(col & row+2) = engine.GetPulseMinTime (channel)
xlApp.Range(col & row+3) = engine.GetPulseMaxTime (channel)
xlApp.Range(col & row+4) = engine.GetPulseFront1090 (channel)
xlApp.Range(col & row+5) = engine.GetPulseFront3090 (channel)
xlApp.Range(col & row+6) = engine.GetPulseFall9010 (channel)
xlApp.Range(col & row+7) = engine.GetPulseWidth5050 (channel)
xlApp.Range(col & row+8) = engine.GetPulseTail50 (channel)
end sub
```

```
row = row + 1 + 2
```

```
ShowPlsLabels ExcelApp, "D", row
```

```
ShowPlsFunctions ExcelApp, MathMod, "C3.V", "E", row
```

PWR Probe Functions

These functions correspond to the PWR Probe component, reporting the power and energy for a flexible number of voltage and current input channels. Optionally, it can report real, reactive, apparent, harmonic, and distortion power with respect to a specified fundamental frequency. First, the **SetupPWR** function must be called with a matching number of voltage and current input channels, each of which must be compatible with the base channel. Then any of the other functions may be called to access the results. To perform a new power and energy calculation, SetupPWR must be called again.

SetupPWR

Performs power calculations on matched arrays of Voltage and Current channels. Results are available via the GetPWR* functions. (See also the SetFundMethod function.)

UI Access	NA		
Parameters	<i>Name</i> <VoltageNames>	<i>Type</i> Variant	<i>Description</i> A variant array of existing channel names for the input voltages. Must be the same length as CurrentNames, and each voltage channel must be compatible with the base channel.
	<CurrentNames>	Variant	A variant array of existing channel names for the input currents. Must be the same length as VoltageNames, and each current channel must be compatible with the base channel.
	<FundFreq>	Double	The frequency, in Hz, for calculating the fundamental apparent power, real power, and reactive power according to the method specified by calling SetFundMethod. Set to 0 to skip these calculations.
Return Value	Boolean Returns 1 if the calculation is successful; otherwise, it returns 0.		

GetPWRPhaseDP

Obtains the distortion power $[(V1 \cdot I_h) \text{ and } (V_h \cdot I1)]$. (See also the SetFundMethod function.)

UI Access	NA		
Parameters	<i>Name</i> <Phase>	<i>Type</i> Integer	<i>Description</i> Specifies the individual phase (voltage and current pair) for accessing a power or energy output. Must range from 1 to the length of the VoltageNames and CurrentNames input arrays.
	<OutputName>	Variant	Optional name of an output channel to create with values vs. the base channel. Any existing channel with the same name is overwritten. If not supplied, only the final value is returned from the function call.
Return Value	Double Returns the distortion power value.		

GetPWRPhaseEnergy

Obtains the instantaneous phase energy. (See also the SetFundMethod function.)

UI Access	NA		
Parameters	<i>Name</i> <Phase>	<i>Type</i> Integer	<i>Description</i> Specifies the individual phase (voltage and current pair) for accessing a power or energy output. Must range from 1 to the length of the VoltageNames and CurrentNames input arrays.
	<OutputName>	Variant	Optional name of an output channel to create with values vs. the base channel. Any existing channel with the same name is overwritten. If not supplied, only the final value is returned from the function call.
Return Value	Double Returns the instantaneous phase energy.		

GetPWRPhaseHP

Obtains the harmonic power ($V_h \cdot I_h$). (See also the SetFundMethod function.)

UI Access	NA		
Parameters	<i>Name</i> <Phase>	<i>Type</i> Integer	<i>Description</i> Specifies the individual phase (voltage and current pair) for accessing a power or energy output. Must range from 1 to the length of the VoltageNames and CurrentNames input arrays.
	<OutputName>	Variant	Optional name of an output channel to create with values vs. the base channel. Any existing channel with the same name is overwritten. If not supplied, only the final value is returned from the function call.
Return Value	Double Returns the harmonic power.		

GetPWRPhaseP

Obtains the average power. (See also the SetFundMethod function.)

UI Access	NA		
Parameters	<i>Name</i> <Phase>	<i>Type</i> Integer	<i>Description</i> Specifies the individual phase (voltage and current pair) for accessing a power or energy output. Must range from 1 to the length of the VoltageNames and CurrentNames input arrays.
	<OutputName>	Variant	Optional name of an output channel to create with values vs. the base channel. Any existing channel with the same name is overwritten. If not supplied, only the final value is returned from the function call.
Return Value	Double Returns the average power.		

GetPWRPhaseP1

Obtains the fundamental average power. (See also the SetFundMethod function.)

UI Access	NA		
Parameters	<i>Name</i> <Phase>	<i>Type</i> Integer	<i>Description</i> Specifies the individual phase (voltage and current pair) for accessing a power or energy output. Must range from 1 to the length of the VoltageNames and CurrentNames input arrays.
	<OutputName>	Variant	Optional name of an output channel to create with values vs. the base channel. Any existing channel with the same name is overwritten. If not supplied, only the final value is returned from the function call.
Return Value	Double Returns the fundamental average power.		

GetPWRPhasePF

Obtains the power factor. (See also the SetFundMethod function.)

UI Access	NA		
Parameters	<i>Name</i> <Phase>	<i>Type</i> Integer	<i>Description</i> Specifies the individual phase (voltage and current pair) for accessing a power or energy output. Must range from 1 to the length of the VoltageNames and CurrentNames input arrays.
	<OutputName>	Variant	Optional name of an output channel to create with values vs. the base channel. Any existing channel with the same name is overwritten. If not supplied, only the final value is returned from the function call.
Return Value	Double		

	Returns the power factor.

GetPWRPhasePF1

Obtains the fundamental power factor. (See also the SetFundMethod function.)

UI Access	NA		
Parameters	<i>Name</i> <Phase>	<i>Type</i> Integer	<i>Description</i> Specifies the individual phase (voltage and current pair) for accessing a power or energy output. Must range from 1 to the length of the VoltageNames and CurrentNames input arrays.
	<OutputName>	Variant	Optional name of an output channel to create with values vs. the base channel. Any existing channel with the same name is overwritten. If not supplied, only the final value is returned from the function call.
Return Value	Double Returns the fundamental power factor.		

GetPWRPhasePower

Obtains the instantaneous phase power. (See also the SetFundMethod function.)

UI Access	NA		
Parameters	<i>Name</i> <Phase>	<i>Type</i> Integer	<i>Description</i> Specifies the individual phase (voltage and current pair) for accessing a power or energy output. Must range from 1 to the length of the VoltageNames and CurrentNames input arrays.
	<OutputName>	Variant	Optional name of an output channel to create with values vs. the base channel. Any existing channel with the same name is overwritten. If not supplied, only the final value is returned from the function call.

Return Value	Double Returns the instantaneous phase power.

GetPWRPhaseQ1

Obtains the fundamental reactive power. (See also the SetFundMethod function.)

UI Access	NA		
Parameters	<i>Name</i> <Phase>	<i>Type</i> Integer	<i>Description</i> Specifies the individual phase (voltage and current pair) for accessing a power or energy output. Must range from 1 to the length of the VoltageNames and CurrentNames input arrays.
	<OutputName>	Variant	Optional name of an output channel to create with values vs. the base channel. Any existing channel with the same name is overwritten. If not supplied, only the final value is returned from the function call.
Return Value	Double Returns the fundamental reactive power.		

GetPWRPhaseS

Obtains the apparent power. (See also the SetFundMethod function.)

UI Access	NA		
Parameters	<i>Name</i> <Phase>	<i>Type</i> Integer	<i>Description</i> Specifies the individual phase (voltage and current pair) for accessing a power or energy output. Must range from 1 to the length of the VoltageNames and CurrentNames input arrays.
	<OutputName>	Variant	Optional name of an output channel to create with values vs. the base channel. Any existing channel with the same name is over-

	written. If not supplied, only the final value is returned from the function call.
Return Value	Double Returns the apparent power.

GetPWRPhaseS1

Obtains the fundamental apparent power. (See also the SetFundMethod function.)

UI Access	NA		
Parameters	<i>Name</i> <Phase>	<i>Type</i> Integer	<i>Description</i> Specifies the individual phase (voltage and current pair) for accessing a power or energy output. Must range from 1 to the length of the VoltageNames and CurrentNames input arrays.
	<OutputName>	Variant	Optional name of an output channel to create with values vs. the base channel. Any existing channel with the same name is overwritten. If not supplied, only the final value is returned from the function call.
Return Value	Double Returns the fundamental apparent power.		

GetPWRSysDPe

Obtains the system equivalent distortion power $[(V1e \cdot I1e) \text{ and } (Vhe \cdot I1e)]$. (See also the SetFundMethod function.)

UI Access	NA		
Parameters	<i>Name</i> <OutputName>	<i>Type</i> Variant	<i>Description</i> Optional name of an output channel to create with values vs. the base channel. Any existing channel with the same name is overwritten. If not supplied, only the final value is returned from the function call.

Return Value	Double Returns the system equivalent distortion power.

GetPWRSystemEnergy

Obtains the total instantaneous energy. (See also the SetFundMethod function.)

UI Access	NA		
Parameters	<i>Name</i> <OutputName>	<i>Type</i> Variant	<i>Description</i> Optional name of an output channel to create with values vs. the base channel. Any existing channel with the same name is overwritten. If not supplied, only the final value is returned from the function call.
Return Value	Double Returns the total instantaneous energy.		

GetPWRSystemHPe

Obtains the system equivalent harmonic power ($V_{he} \cdot I_{he}$). (See also the SetFundMethod function.)

UI Access	NA		
Parameters	<i>Name</i> <OutputName>	<i>Type</i> Variant	<i>Description</i> Optional name of an output channel to create with values vs. the base channel. Any existing channel with the same name is overwritten. If not supplied, only the final value is returned from the function call.
Return Value	Double Returns the system equivalent harmonic power.		

GetPWRSystemle

Obtains the system equivalent RMS current. (See also the SetFundMethod function.)

UI Access	NA		
Parameters	<i>Name</i> <OutputName>	<i>Type</i> Variant	<i>Description</i> Optional name of an output channel to create with values vs. the base channel. Any existing channel with the same name is overwritten. If not supplied, only the final value is returned from the function call.
Return Value	Double Returns the system equivalent RMS current.		

GetPWRSystemle1

Obtains the system equivalent fundamental RMS current. (See also the SetFundMethod function.)

UI Access	NA		
Parameters	<i>Name</i> <OutputName>	<i>Type</i> Variant	<i>Description</i> Optional name of an output channel to create with values vs. the base channel. Any existing channel with the same name is overwritten. If not supplied, only the final value is returned from the function call.
Return Value	Double Returns the system equivalent fundamental RMS current.		

GetPWRSystemP

Obtains the total average power. (See also the SetFundMethod function.)

UI Access	NA		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>

	<OutputName> Variant Optional name of an output channel to create with values vs. the base channel. Any existing channel with the same name is overwritten. If not supplied, only the final value is returned from the function call.
Return Value	Double Returns the total average power.

GetPWRSystemP1

Obtains the system equivalent fundamental average power. (See also the SetFundMethod function.)

UI Access	NA		
Parameters	<i>Name</i> <OutputName>	<i>Type</i> Variant	<i>Description</i> Optional name of an output channel to create with values vs. the base channel. Any existing channel with the same name is overwritten. If not supplied, only the final value is returned from the function call.
Return Value	Double Returns the system equivalent fundamental average power.		

GetPWRSystemPF1

Obtains the system equivalent fundamental power factor. (See also the SetFundMethod function.)

UI Access	NA		
Parameters	<i>Name</i> <OutputName>	<i>Type</i> Variant	<i>Description</i> Optional name of an output channel to create with values vs. the base channel. Any existing channel with the same name is overwritten. If not supplied, only the final value is returned from the function call.

Return Value	Double Returns the system equivalent fundamental power factor.

GetPWRSystemPFe

Obtains the system equivalent power factor (P/Se). (See also the SetFundMethod function.)

UI Access	NA		
Parameters	<i>Name</i> <OutputName>	<i>Type</i> Variant	<i>Description</i> Optional name of an output channel to create with values vs. the base channel. Any existing channel with the same name is overwritten. If not supplied, only the final value is returned from the function call.
Return Value	Double Returns the system equivalent power factor.		

GetPWRSystemPower

Obtains the total instantaneous power. (See also the SetFundMethod function.)

UI Access	NA		
Parameters	<i>Name</i> <OutputName>	<i>Type</i> Variant	<i>Description</i> Optional name of an output channel to create with values vs. the base channel. Any existing channel with the same name is overwritten. If not supplied, only the final value is returned from the function call.
Return Value	Double Returns the total instantaneous power.		

GetPWRSystemQ1

Obtains the system equivalent fundamental reactive power. (See also the SetFundMethod function.)

UI Access	NA		
Parameters	<i>Name</i> <OutputName>	<i>Type</i> Variant	<i>Description</i> Optional name of an output channel to create with values vs. the base channel. Any existing channel with the same name is overwritten. If not supplied, only the final value is returned from the function call.
Return Value	Double Returns the system equivalent fundamental reactive power.		

GetPWRSystemS1

Obtains the system equivalent fundamental apparent power. (See also the SetFundMethod function.)

UI Access	NA		
Parameters	<i>Name</i> <OutputName>	<i>Type</i> Variant	<i>Description</i> Optional name of an output channel to create with values vs. the base channel. Any existing channel with the same name is overwritten. If not supplied, only the final value is returned from the function call.
Return Value	Double Returns the system equivalent fundamental apparent power.		

GetPWRSystemSe

Obtains the system equivalent apparent power. (See also the SetFundMethod function.)

UI Access	NA

Parameters	<i>Name</i> <OutputName>	<i>Type</i> Variant	<i>Description</i> Optional name of an output channel to create with values vs. the base channel. Any existing channel with the same name is overwritten. If not supplied, only the final value is returned from the function call.
Return Value	Double Returns the system equivalent apparent power.		

GetPWRSystemVe

Obtains the system equivalent RMS voltage to reference. (See also the SetFundMethod function.)

UI Access	NA		
Parameters	<i>Name</i> <OutputName>	<i>Type</i> Variant	<i>Description</i> Optional name of an output channel to create with values vs. the base channel. Any existing channel with the same name is overwritten. If not supplied, only the final value is returned from the function call.
Return Value	Double Returns the system equivalent RMS voltage to reference.		

GetPWRSystemVe1

Obtains the system equivalent fundamental RMS voltage to reference. (See also the SetFundMethod function.)

UI Access	NA		
Parameters	<i>Name</i> <OutputName>	<i>Type</i> Variant	<i>Description</i> Optional name of an output channel to create with values vs. the base channel. Any existing channel with the same name is overwritten. If not supplied, only the final value is returned from the function call.

Return Value	Double Returns the system equivalent fundamental RMS voltage to reference.

Example: PWR Probe Functions

This example calculates the instantaneous power and energy for a capacitor transient. Then it calculates the sinusoidal power quantities for a 50 Hz voltage source.

```
MathMod.SetBaseChannel "t"
```

```
MathMod.SetupPWR Array ("C1.V"), Array ("C1.I"), 0.0
```

```
ExcelApp.Range("G1") = "C1 Power"
```

```
ExcelApp.Range("H1") = MathMod.GetPWRSystemPower
```

```
ExcelApp.Range("G2") = "C1 Energy"
```

```
ExcelApp.Range("H2") = MathMod.GetPWRSystemEnergy
```

```
ExcelApp.Range("G3") = "C1 Power (0)"
```

```
ExcelApp.Range("H3") = MathMod.GetPWRPhasePower (0, "C3.P")
```

```
ExcelApp.Range("G4") = "C1 Energy (0)"
```

```
ExcelApp.Range("H4") = MathMod.GetPWRPhaseEnergy (0)
```

```
MathMod.SetupPWR Array ("E2.EMF"), Array ("E2.I"), 50.0
```

```
ExcelApp.Range("G6") = "E2 S1"
```

```
ExcelApp.Range("H6") = MathMod.GetPWRSystemS1
```

```
ExcelApp.Range("G7") = "E2 P1"
```

```
ExcelApp.Range("H7") = MathMod.GetPWRSystemP1
```

```
ExcelApp.Range("G8") = "E2 Q1"
```

```
ExcelApp.Range("H8") = MathMod.GetPWRSystemQ1
```

```
ExcelApp.Range("G9") = "E2 PF1"
```

```
ExcelApp.Range("H9") = MathMod.GetPWRSystemPF1
```

```
ExcelApp.Range("G10") = "E2 Se"
```

```
ExcelApp.Range("H10") = MathMod.GetPWRSystemSe
```

```
ExcelApp.Range("G11") = "E2 P"
```

```
ExcelApp.Range("H11") = MathMod.GetPWRSystemP
```

```
ExcelApp.Range("G12") = "E2 DPe"
```

```
ExcelApp.Range("H12") = MathMod.GetPWRSystemDPe
```

```
ExcelApp.Range("G13") = "E2 HPe"
```

```
ExcelApp.Range("H13") = MathMod.GetPWRSystemHPe
```

```
ExcelApp.Range("G14") = "E2 PFe"
```

```
ExcelApp.Range("H14") = MathMod.GetPWRSystemPFe
```

```
ExcelApp.Range("G15") = "E2 Energy"
```

```
ExcelApp.Range("H15") = MathMod.GetPWRSystemEnergy
```

T Probe Functions

This function corresponds to the T Probe component. It performs linear interpolation with respect to a base channel value, which is typically used in the time domain.

GetValue

Obtains the interpolated channel value at the base time (Base)

UI Access	NA		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<ChannelName>	String	The name of an existing input channel.
	<Base>	Double	The value of the base channel at which to interpolate.
	<OutputName>	Variant	Optional name of an output channel to create with values vs. the base channel. Any existing channel with the same name is overwritten. If not supplied, only the final value is returned from the function call.
Return Value	Double Returns the interpolated channel value.		

Example: T Probe Functions

This example calculates the value of "C3.V" at the half-way time.

```
dMin = MathMod.GetChannelMin ("t")
```

```
dMax = MathMod.GetChannelMax ("t")
```

```
dHalf = 0.5 * (dMin + dMax)
```

```
ExcelApp.Range("E5") = MathMod.GetValue ("C3.V", dHalf)
```

TR Probe Functions

These functions correspond to the TR Probe component. It reports the minimum, maximum, integral, mean, total RMS, total AC RMS, and related values. Optionally, it can report the fundamental RMS and total distortion with respect to a specified fundamental frequency.

GetCrestFactor

Obtains the crest factor (peak/RMS), within the base channel Start/Stop values. (See also the SetFundMethod function.)

UI Access	NA		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<ChannelName> <OutputName>	String Variant	The name of an existing input channel. Optional name of an output channel to create with values vs. the base channel. Any existing channel with the same name is overwritten. If not supplied, only the final value is returned from the function call.
Return Value	Double Returns the crest factor		

GetDistortion

Obtains the total distortion (not just harmonics), within the base channel Start/Stop values. (See also the SetFundMethod function.)

UI Access	NA

Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<ChannelName>	String	The name of an existing input channel
	<Freq>	Double	The frequency, in Hz, at which to calculate the fundamental RMS, according to the method specified by calling SetFundMethod. Set to 0 to skip this calculation, in which case the functions GetFundamentalMag and GetDistortion will return 0.
	<OutputName>	Variant	Optional name of an output channel to create with values vs. the base channel. Any existing channel with the same name is overwritten. If not supplied, only the final value is returned from the function call.
Return Value	Double Returns the total distortion.		

GetFormFactor

Obtains the form factor (RMS/Mean Abs), within the base channel Start/Stop values. (See also the SetFundMethod function.)

UI Access	NA		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<ChannelName>	String	The name of an existing input channel
	<OutputName>	Variant	Optional name of an output channel to create with values vs. the base channel. Any existing channel with the same name is overwritten. If not supplied, only the final value is returned from the function call.
Return Value	Double Returns the form factor		

GetFundamentalMag

Obtains the RMS1 (the magnitude at the fundamental frequency), within the base channel Start/Stop values. (See also the SetFundMethod function.)

UI Access	NA		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<ChannelName>	String	The name of an existing input channel
	<Freq>	Double	The frequency, in Hz, at which to calculate the fundamental RMS, according to the method specified by calling SetFundMethod. Set to 0 to skip this calculation, in which case the functions GetFundamentalMag and GetDistortion will return 0.
	<OutputName>	Variant	Optional name of an output channel to create with values vs. the base channel. Any existing channel with the same name is overwritten. If not supplied, only the final value is returned from the function call.
Return Value	None		

GetIntegral

Obtains the integral, within the base channel Start/Stop values. (See also the SetFundMethod function.)

UI Access	NA		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<ChannelName>	String	The name of an existing input channel
	<OutputName>	Variant	Optional name of an output channel to create with values vs. the base channel. Any existing channel with the same name is overwritten. If not supplied, only the final value is returned from the function call.
Return Value	Double Returns the integral		

GetMax

Obtains the maximum value, within the base channel Start/Stop values. (See also the SetFundMethod function.)

UI Access	NA		
Parameters	<i>Name</i> <ChannelName> <OutputName>	<i>Type</i> String Variant	<i>Description</i> The name of an existing input channel. Optional name of an output channel to create with values vs. the base channel. Any existing channel with the same name is overwritten. If not supplied, only the final value is returned from the function call.
Return Value	Double Returns the maximum value.		

GetMaxTime

Obtains the time of the maximum value, within the base channel Start/Stop values. (See also the SetFundMethod function.)

UI Access	NA		
Parameters	<i>Name</i> <ChannelName> <OutputName>	<i>Type</i> String Variant	<i>Description</i> The name of an existing input channel. Optional name of an output channel to create with values vs. the base channel. Any existing channel with the same name is overwritten. If not supplied, only the final value is returned from the function call.
Return Value	None		

GetMean

Obtains the mean value (DC RMS), within the base channel Start/Stop values. (See also the SetFundMethod function.)

UI Access	NA		
Parameters	<i>Name</i> <ChannelName> <OutputName>	<i>Type</i> String Variant	<i>Description</i> The name of an existing input channel. Optional name of an output channel to create with values vs. the base

	channel. Any existing channel with the same name is overwritten. If not supplied, only the final value is returned from the function call.
Return Value	Double Returns the mean value.

GetMeanAbs

Obtains the mean of the absolute value, within the base channel Start/Stop values. (See also the SetFundMethod function.)

UI Access	NA		
Parameters	<i>Name</i> <ChannelName> <OutputName>	<i>Type</i> String Variant	<i>Description</i> The name of an existing input channel Optional name of an output channel to create with values vs. the base channel. Any existing channel with the same name is overwritten. If not supplied, only the final value is returned from the function call.
Return Value	Double Returns the mean of the absolute value.		

GetMin

Obtains the minimum value, within the base channel Start/Stop values. (See also the SetFundMethod function.)

UI Access	NA		
Parameters	<i>Name</i> <ChannelName> <OutputName>	<i>Type</i> String Variant	<i>Description</i> The name of an existing input channel. Optional name of an output channel to create with values vs. the base channel. Any existing channel with the same name is overwritten. If not supplied, only the final value is returned from the function call.

Return Value	Double Returns the minimum value.

GetMinTime

Obtains the time of the minimum value, within the base channel Start/Stop values. (See also the SetFundMethod function.)

UI Access	NA		
Parameters	<i>Name</i> <ChannelName> <i>Type</i> <OutputName> <i>Description</i>	String Variant	The name of an existing input channel. Optional name of an output channel to create with values vs. the base channel. Any existing channel with the same name is overwritten. If not supplied, only the final value is returned from the function call.
Return Value	Double Returns the time at which the minimum value occurs.		

GetPeakAverage

Obtains the ratio of the peak-peak to average, within the base channel Start/Stop values. (See also the SetFundMethod function.)

UI Access	NA		
Parameters	<i>Name</i> <ChannelName> <i>Type</i> <OutputName> <i>Description</i>	String Variant	The name of an existing input channel. Optional name of an output channel to create with values vs. the base channel. Any existing channel with the same name is overwritten. If not supplied, only the final value is returned from the function call.
Return Value	Double Returns the ratio of the peak-peak to average.		

GetPeakPeak

Obtains the peak-to-peak value, within the base channel Start/Stop values. (See also the SetFundMethod function.)

UI Access	NA		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<ChannelName> <OutputName>	String Variant	The name of an existing input channel Optional name of an output channel to create with values vs. the base channel. Any existing channel with the same name is overwritten. If not supplied, only the final value is returned from the function call.
Return Value	Double Returns the peak-to-peak value		

GetRipple

Obtains the ripple factor (AC RMS/Mean), within the base channel Start/Stop values. (See also the SetFundMethod function.)

UI Access	NA		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<ChannelName> <OutputName>	String Variant	The name of an existing input channel. Optional name of an output channel to create with values vs. the base channel. Any existing channel with the same name is overwritten. If not supplied, only the final value is returned from the function call.
Return Value	Double Returns the ripple factor		

GetRMS

Obtains the total RMS, within the base channel Start/Stop values. (See also the SetFundMethod function.)

UI Access	NA		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<ChannelName> <OutputName>	String Variant	The name of an existing input channel. Optional name of an output channel to create with values vs. the base channel. Any existing channel with the same name is overwritten. If not supplied, only the final value is returned from the function call.
Return Value	Double Returns the total RMS		

GetRMSAC

Obtains the AC RMS (not just harmonics), within the base channel Start/Stop values. (See also the SetFundMethod function.)

UI Access	NA		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<ChannelName> <OutputName>	String Variant	The name of an existing input channel. Optional name of an output channel to create with values vs. the base channel. Any existing channel with the same name is overwritten. If not supplied, only the final value is returned from the function call.
Return Value	Double Returns the AC RMS		

Example: TR Probe Functions

```
sub ShowTrLabels(xlApp, col, row)
```

```
xlApp.Range(col & row) = "Min"
xlApp.Range(col & row+1) = "Max"
xlApp.Range(col & row+2) = "Tmin"
xlApp.Range(col & row+3) = "Tmax"
xlApp.Range(col & row+4) = "Mean"
xlApp.Range(col & row+5) = "Mean Abs"
xlApp.Range(col & row+6) = "Total RMS"
xlApp.Range(col & row+7) = "Fund RMS"
xlApp.Range(col & row+8) = "AC RMS"
xlApp.Range(col & row+9) = "Total Distortion"
xlApp.Range(col & row+10) = "Peak-to-Peak"
xlApp.Range(col & row+11) = "Peak/Average"
xlApp.Range(col & row+12) = "Integral"
xlApp.Range(col & row+13) = "Ripple Factor"
xlApp.Range(col & row+14) = "Form Factor"
xlApp.Range(col & row+15) = "Crest Factor"
end sub
```

```
sub ShowTrFunctions(xlApp, engine, channel, col, row, dFreq)
xlApp.Range(col & row) = engine.GetMin (channel)
xlApp.Range(col & row+1) = engine.GetMax (channel)
xlApp.Range(col & row+2) = engine.GetMinTime (channel)
xlApp.Range(col & row+3) = engine.GetMaxTime (channel)
xlApp.Range(col & row+4) = engine.GetMean (channel)
xlApp.Range(col & row+5) = engine.GetMeanAbs (channel)
xlApp.Range(col & row+6) = engine.GetRMS (channel)
xlApp.Range(col & row+7) = engine.GetFundamentalMag (channel, dFreq)
xlApp.Range(col & row+8) = engine.GetRMSAC (channel)
xlApp.Range(col & row+9) = engine.GetDistortion (channel, dFreq)
```

```

xlApp.Range(col & row+10) = engine.GetPeakPeak (channel)
xlApp.Range(col & row+11) = engine.GetPeakAverage (channel)
xlApp.Range(col & row+12) = engine.GetIntegral (channel)
xlApp.Range(col & row+13) = engine.GetRipple (channel)
xlApp.Range(col & row+14) = engine.GetFormFactor (channel)
xlApp.Range(col & row+15) = engine.GetCrestFactor (channel)
end sub

```

```
dFreq = 1.0 / 0.0006
```

```
row = row + 1 + 9
```

```
ShowTrLabels ExcelApp, "D", row
```

```
ShowTrFunctions ExcelApp, MathMod, "C3.V", "E", row, dFreq
```

Random Number Functions

These functions can be used to generate random numbers.

GetUnitRandMinMax

Obtains a uniform distributed number between two values

UI Access	NA		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<MinVal>	Double	Minimum value.
	<MaxVal>	Double	Maximum value.
Return Value	Double		
	Returns a uniform distributed number between MinVal and MaxVal.		

GetUnitRandMeanTol

Obtains a uniform distributed number with a specified mean and tolerance

UI Access	NA		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<MeanVal>	Double	Mean Value
	<Tolerance>	Double	Tolerance in %
Return Value	Double		
	Returns a uniform distributed number with a mean value of <MeanVal> and a tolerance of <Tolerance>		

GetNormRandMinMax

Obtains a normal distributed number between two values

UI Access	NA		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<MinVal>	Double	Minimum value.
	<MaxVal>	Double	Maximum value
Return Value	Double		
	Returns a normal distributed number between MinVal and MaxVal.		

GetNormRandMeanTol

Obtains a normal distributed number with a specified mean and tolerance.

UI Access	NA		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<MeanVal>	Double	Mean value.
	<Tolerance>	Double	Tolerance in %.
Return Value	Double		
	Returns a normal distributed number with a mean value of <MeanVal> and a tolerance of <Tolerance>.		

GetCustomRNDValue

Obtains a random number with a custom distribution that is described in a separate file.

UI Access	NA		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<FileName>	String	This parameter specifies the name of the file that contains the user defined density function. The file can be a Twin Builder Characteristic file (*.mdx), a Microsoft Excel (*.xls) file, an Access file (*.mdb), or a comma-separated value file (*.csv) containing two data sets. The first data set is the value (must be strictly increasing), and the second data set is the probability.
Return Value	Double Returns a random number with a custom distribution that is described in the file <FileName>		

22 - User Defined Solutions Commands

User Defined Solution commands should be executed by the "UserDefinedSolutionModule" module.

```
oDesign = oProject.SetActiveDesign("TestDesign1")  
oModule = oDesign.GetModule("UserDefinedSolutionModule")
```

[CreateUserDefinedSolution](#)

[DeleteUserDefinedSolutions](#)

[EditUserDefinedSolution](#)

CreateUserDefinedSolution

Use: Creates a new user defined solution.

Command: **Create User Defined Solution** popup menu is available in the Result folder context menu when applicable.

Syntax: CreateUserDefinedSolution <SolutionName>, <PluginFileLocation>, <PluginFileRelativePathName>, <PropertyValuesArray>, <ProbeSelectionArray>, <DynamicProbesArray>

Return Value: The name of the user defined solution that was created. **Note:** if the requested user defined solution name is not available because it is already in use, the user defined solution will be created with a different name which will be returned.

Parameters: <SolutionName>

Type: String

Requested name of new user defined solution.

<PluginFileLocation>

Type: String

Indicates the library where the UDS plugin file is located. This parameter must be one of the following values: "SysLib", "UserLib", "PersonalLib".

<PluginFileRelativePathName>

Type: String

The path of the UDS plugin file relative to the "UserDefinedOutputs" subdirectory of the library specified by <PluginFileLocation>.

<PropertyValuesArray>

Type: Array of strings

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")
```

<ProbeSelectionArray>

Type: Array of <ProbeSelection>'s (see below)

The probe specification array specifies how UDS probes are defined and mapped to traces in the design.

<ProbeSelection>

Type: Array of strings representing how a single probe is defined by a trace. The array contains the following items:

<ReportType>

Type: String

See the CreateReport command for more information.

<ProbeName>

Type: String

Name of the probe being specified. Note: this must match a probe name specified in the UDS plugin file.

<SolutionName>

Type: String

See the CreateReport command for more information.

<SimulatedValueContextArray>

Type: Array of strings

See the CreateReport command for more information.

<PointSetDefinitionArray>

Type: Array of values with optional overriding values and optional variable values.

See the CreateReport command for more information.

<TraceExpressionArray>

Type: Array of strings and values.

This is similar to the TracesExpressionsArray used in the CreateReport command, but there will only be a single component expression named "Probe Component." See the CreateReport command for more information.

<ExtendedTraceInformationArray>

Type: Array of strings and values

See the CreateReport command for more information.

<DynamicProbesArray>

Type: Array of <ProbeSelection>'s, representing the probes that are used by dynamic-probes.

DeleteUserDefinedSolutions

Deletes one or more user defined solutions

UI Access	'Delete' button from the "User Defined Solutions" dialog		
Parameters	<i>Name</i> <UserDefinedSolutionNames>	<i>Type</i> Array of strings	<i>Description</i> Name of User Defined Solutions to be deleted.
Return Value	None		

Python Syntax	DeleteUserDefinedSolutions (<UserDefinedSolutionNames>)
Python Example	oModule.DeleteUserDefinedSolutions(["Solution1", "Solution2"])

EditUserDefinedSolution

Updates an existing user defined solution

UI Access	Command: 'Edit' button in the User Defined Solutions dialog		
Parameters	<i>Name</i>	<i>Type</i>	<i>Description</i>
	<ExistingSolutionName>	String	Current name of the User defined solution
	<NewSolutionName>	String	Desired name of the User defined solution
	<PluginFileLocation>	String	Location where the plugin file is located
	<PluginFileRelativePathName>	String	The relative path of the plugin file
	<PropertyValuesArray>	Array	Array of property values
	<ProbeSelectionArray>	Array	Array of selected probes
	<DynamicProbesArray>	Array	Array of Dynamic Probes
Return Value	The name of the user defined solution after being updated. Note: if the requested user defined solution name is not available because it is already in use, the user defined solution will be created with a different name which will be returned.		

Python Syntax	EditUserDefinedSolution (<ExistingSolutionName>, <NewSolutionName>, <PluginFileLocation>, <PluginFileRelativePathName>, <PropertyValuesArray>, <ProbeSelectionArray>, <DynamicProbesArray>)
Python Example	<pre>oModule.EditUserDefinedSolution ("User Defined Solution 1", "User Defined Solution 2", "SysLib", "Example", ["multiplication_factor:=", "1.2"],[["Modal Solution Data", "Probe 1", "Setup1 : LastAdaptive", [], ["Freq:=", ["All"]], ["Probe Component:=", ["dB(S(1,1))"]], [], ["Modal Solution Data", "Probe 2", "Setup1 : LastAdaptive", [], ["Freq:=", ["All"]], ["Probe Component:=", ["mag(S(1,1))"]], []], [["Modal _ Solution Data", "Dynamic Probe 1", "Setup1 : LastAdaptive",</pre>

	<pre>[], ["Freq:=", ["All"]], ["Probe Component:=", ["Freq"]], []])</pre>

GetSimulationOptionFilePath

Retrieves simulation settings for the current design. Select options as described below to extract the current value.

UI Access	NA		
Parameters	<i>Name</i> <option>	<i>Type</i> Integer	<i>Description</i> Choices are: 0 – Save simulation state (KRN). 1 – Load simulation state (KRN). 2 – Save Initial values.
Return Value	value	string	Return value is the user-specified file path of the selected option. It will return a file name if the selected option was supposed to use the user-specified file; otherwise, it will return an empty file path. For auto or ask file name option value, it will return empty. If empty, it failed to retrieve file name for the option.

Python Syntax	<pre>fileName = GetSimulationOptionFilePath(<option>)</pre>
Python Example	<pre>oProject = oDesktop.GetActiveProject() oDesign = oProject.GetActiveDesign() //return value is filename. // empty return value indicates option could be set to use</pre>

```
auto(default) or ask before simulation.  
// it will return value only if option was set to use user specified filepath  
  
saveKRNFileName = oDesign.GetSimulationOptionFilePath(0)  
loadKRNFileName = oDesign.GetSimulationOptionFilePath(1)
```

Note:
To set KRN/initial values file settings from a user-specified location, use the **SetSimulationOptionFilePath** command.

GetSimulationSettings

Retrieves simulation settings for the current design.

UI Access	NA		
Parameters	<i>Name</i> <option>	<i>Type</i> Integer	<i>Description</i> Choices are: 0 – Save simulation state (KRN) status. 1 – Load simulation state (KRN) status. 2 – Save Initial values status. 3 – Automatically sort block diagram before simulation status. 4 – Status of symbol animation and output property display update. 5 – Significant number of digits to display in output property display. 6 – Status of statistical package use. 7 – Is seed specified by user for statistical package. 8 – Get seed value.
Return Value	value	integer	For options 0, 1, or 2, the return value will be: 0 – No 1 – Auto

	2 – Ask 3 – specify file by user For options 3, 4, 6, or 7, the return value will be: 0 – False 1 – True For option 5, the return value will be the number of significant digits. For option 8, the return value will be the seed value.

Python Syntax	<code>returnVal=GetSimulationSettings(<option>)</code>
Python Example	<pre>oProject = oDesktop.GetActiveProject() oDesign = oProject.GetActiveDesign() // return value meanings: 0 (No), 1 (Auto), 2 (Ask), 3 (specify file by user) saveKRNOptVal = oDesign.GetSimulationSettings(0) //save KRN Option status loadKRNOptVal = oDesign.GetSimulationSettings(1) //load KRN Option status //return value will be 0, 1 sortBlock = oDesign.GetSimulationSettings(3) // auto- matically sort blocks animateSym = oDesign.GetSimulationSettings(4) //Symbol Anim- ation status</pre>

Note:

To *get* KRN/initial values file settings from a user-specified location, use the **GetSimulationOptionFilePath** command.

To *set* KRN/initial values file settings from a user-specified location, use the **SetSimulationOptionFilePath** command.

SetSimulationOptionFilePath

Sets simulation settings for the current design. Select options as described below to set the current value.

UI Access	NA		
Parameters	<i>Name</i> <option>	<i>Type</i> Integer	<i>Description</i> Choices are: 0 – Save simulation state (KRN). 1 – Load simulation state (KRN). 2 – Save Initial values.
	value	string	Return value will set selected option to use file specified by this value.
Return Value	ret	integer	0 – Fail 1 – Success

Python Syntax	ret = SetSimulationOptionFilePath(<option>, <FileName>)
Python Example	<pre>oProject = oDesktop.GetActiveProject() oDesign = oProject.GetActiveDesign() //<option> 0: save krn, 1:load krn</pre>


```
//return value is 0: fail, 1:success.

saveKRNFile = "<UserSpecifiedFileNameWithPath>"
// "C:\MyFile.krn"

loadKRNFile = "<UserSpecifiedFileNameWithPath>"
// "C:\MyFile.krn"

ret = oDesign.SetSimulationOptionFilePath(0, saveKRNFile)

ret = oDesign.SetSimulationOptionFilePath(1, loadKRNFile)
```

Note:
To set KRN/initial values to auto/ask location, use the **SetSimulationSettings** command.

SetSimulationSettings

Retrieves simulation settings for the current design.

UI Access	NA		
Parameters	<i>Name</i> <option>	<i>Type</i> Integer	<i>Description</i> Choices are: 0 – Save simulation state (KRN) status. 1 – Load simulation state (KRN) status. 2 – Save Initial values status. 3 – Automatically sort block diagram before simulation status. 4 – Status of symbol animation and output property display update.

			5 – Significant number of digits to display in output property display. 6 – Status of statistical package use. 7 – Is seed specified for statistical package. 8 – Get seed value.
	<value>	Integer	For options 0, 1, or 2, the return value will be: 0 – No 1 – Auto 2 – Ask For options 3, 4, 6, or 7, the return value will be: 0 – False 1 – True For option 5, command will set the number of significant digits. For option 8, command will set the seed value.
Return Value	ret	integer	1 – Success 0 – Fail

Python Syntax	<code>returnVal=SetSimulationSettings(<option>, <value>)</code>
Python Example	<pre> oProject = oDesktop.GetActiveProject() oDesign = oProject.GetActiveDesign() //<value> meanings: 0 (No), 1(Auto), 2(Ask) //<value> = 1(save KRN at default location (auto)) ret = oDesign.SetSimulationSettings(0,1) //<value> = 2(Ask for filename before simulation) ret = oDesign.SetSimulationSettings(1,2) </pre>

<pre>// <value>will be 0, 1 ret = oDesign.SetSimulationSettings(3, 0) // disable auto- matically sort blocks ret = oDesign.SetSimulationSettings(4, 1) // enable Symbol Animation</pre>

Note:

To *get* KRN/initial values file settings from a user-specified location, use the **GetSimulationOptionFilePath** command.

To *set* KRN/initial values file settings from a user-specified location, use the **SetSimulationOptionFilePath** command.

23 - Best Practices and Examples

Best Practices for Recorded Scripts

While scripts recorded by the native script recorder work well on the designs and in the conditions in which they were originally recorded, you can manually edit the scripts to make them more stable for a wider variety of designs and conditions.

- Replace **SetActiveProject()** and **SetActiveDesign()** calls with **GetActiveProject()** and **GetActiveDesign** calls.
- Get and store design name for future use.
- Store **Name:Id** strings for objects created in variables.

Example: Analyze a Transient Solution

This example script demonstrates how to analyze a transient solution, add another transient, create a report and add markers to the report.

**Python
Example**

```
import ScriptEnv

ScriptEnv.Initialize("Ansoft.ElectronicsDesktop")

oDesktop.RestoreWindow()

oProject = oDesktop.SetActiveProject("Example1")

oDesign = oProject.SetActiveDesign("TwinBuilder1")

oDesign.Analyze("TR1")

oModule = oDesign.GetModule("SimSetup")

oModule.AddTransient(
    [
        "NAME:SimSetup",
        "DataBlockID:=", 8,
        "SimSetupID:=", -1,
        "OptionName:=", "(Default Options)",
        "AnalysisEnabled:=", 1,
        "Name:=", "TR",
        "TransientData:=", ["40ms", "10us", "1ms", False, ""],
```

	<pre> "EnableContSolve:=" , False]) oModule = oDesign.GetModule("ReportSetup") oModule.CreateReport("XY Plot 2", "Standard", "Rectangular Plot", "TR1", ["NAME:Context", "SimValueContext:=" , [1,0,2,0,False,False,- 1,1,0,1,1,"",0,0]], ["Time:=" , ["All"]], ["X Component:=" , "Time", "Y Component:=" , ["GS1.VAL"]], []) oDesign.Analyze("TR1") oModule.AddMarker("XY Plot 2", "m1", "GS1.VAL : TR1 : Cartesian", "6.35099999999991s") oModule.AddMarker("XY Plot 2", "m2", "GS1.VAL : TR1 : Cartesian", "38.1610000000001s")</pre>

Example: Export Report Data

The following script demonstrates how to export report data from Twin Builder and save it to a file.

Python Example	<pre>import ScriptEnv ScriptEnv.Initialize("Ansoft.ElectronicsDesktop")</pre>

```
oDesktop.RestoreWindow()

oProject = oDesktop.SetActiveProject("Example1")

oDesign = oProject.SetActiveDesign("TwinBuilder1")

oModule = oDesign.GetModule("ReportSetup")

oModule.ExportToFile("XY Plot 1", "C:/.../Test1.csv")
```

Example: Running in an HPC Environment

The following script demonstrates how to use a settings file in a high performance computing (HPC) environment.

**Python
Example**

```
projectFileName = "<twinbuilder project filepath>"
hpcSettingsFileName = "<.areg filepath>"
jobIds = oDesktop.SubmitJob(hpcSettingsFileName, ProjectFileName)
oDesktop.LaunchJobMonitor(projectFileName)
oDesktop.RefreshJobMonitor()
//user can use same RSM settings file for more other projects as well..
projectFileName2 = "<anotherProject>"
oDesktop.SubmitJob(hpcSettingsFileName, ProjectFileName2)
oDesktop.LaunchJobMonitor(projectFileName2)
oDesktop.RefreshJobMonitor()
```

Note:

Refresh job monitor will return status of job. For more details check command's help page.

Example: Save/Load Custom KRN Files

Run these scripts to use custom save/load **.krn** (simulation state) files.

**Python
Example**

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

saveKRNFile = "<UserSpecifiedFileNameWithPath>"
```

```
// "C:\MyFile.krn"  
loadKRNFile = "<UserSpecifiedFileNameWithPath>"  
// "C:\MyFile.krn"  
ret = oDesign.SetSimulationOptionFilePath(0, saveKRNFile)  
ret = oDesign.SetSimulationOptionFilePath(1, loadKRNFile)  
...  
oDesign.AnalyzeUI("TR", 3)    // "TR" : setup name
```


Index

A

-
- Aborting Scripts 1-10
 - AC Probe Functions 21-23
 - Activate 19-21
 - Add [Symbol Manager] 9-45
 - Add Component 9-7
 - AddACAnalysis 14-1
 - AddAnalysisOptions 14-2
 - AddCartesianXMarker 17-2
 - AddCartesianYMarker 17-2
 - AddDataset 12-1
 - AddDCAnalysis 14-6
 - AddDeltaMarker 17-3
 - AddFMUModel [Component] 9-15
 - AddLevel 19-63
 - AddMarker 17-3
 - AddMaterial 9-1
 - AddMenuProp 11-16
 - AddMessage 13-1
 - AddMessage (Desktop) 5-1
 - AddNote 17-4
 - AddPinIPorts 19-22
 - AddPinPageConnectors 19-21, 19-23
 - AddPortImpedance 13-1
 - AddPortSource 13-2
 - AddProp 11-18
 - AddProp2 11-20
 - AddScript 9-65
 - AddSdbOutput 13-3
 - AddTraceCharacteristics 17-5
 - AddTraces 17-5
 - AddTransient 14-7
 - AlignHorizontal 19-23
 - AlignHorizontal (Symbol) 19-63
 - AlignVertical 19-24
 - AlignVertical (Symbol) 19-64
 - Analyze 13-3
 - AnalyzeAll 7-1, 13-4
 - AnalyzeInBackground 13-4
 - AnalyzeUI 13-5
 - Application Object Script
 - commands 3-1
 - GetAppDesktop 3-1
 - GetProcessResources 4-1
 - arithmetic operators 2-3
-

B

BringToFront 19-25, 19-65

C

CalculateCharacteristic 21-15

ChangeGPort 13-5

ChangeIPort 13-6

ChangePinVisibility 19-25

ChangePortImpedanceProperty
13-7

ChangePortSourceProperty 13-8

ChangeProperty 11-5, 13-8, 17-7,
19-50

ClearAllMarkers 17-8

ClearAllTraceCharacteristics 17-8

ClearMessages 5-2, 7-1

Close 7-2

CloseAllWindows 5-2

CloseEditor 19-26

CloseEditor (Symbol) 19-65

CloseProject 5-3

CloseProjectNoForce 5-3

comparison operators 2-3

CompileAsTwin 13-10

conditional statements 2-1
types of 2-1

converting data types 2-5

ConvertSMLToSMT 4-2

Copy 19-26

Copy (Symbol) 19-65

CopyDesign 7-2

CopyReportData 17-9

CopyReportDefinitions 17-9

CopySetup, Optimetrics module
command 18-4

CopySubdesign 19-27

CopyTraceDefinitions 17-9

CopyTracesData 17-10

Create_Page 19-28

CreateArc 19-66

CreateCircle 19-3, 19-67

CreateComponent 19-4

CreateGlobalPort 19-7

CreateGround 19-8

CreateImage 19-9, 19-68

CreateIPort 19-15

CreateLine 19-11, 19-69

CreateOutputVariable 16-1

CreatePagePort 19-12

CreatePin 19-70

CreatePolygon 19-16, 19-71

CreateRectangle 19-17, 19-72

CreateReport 17-10

CreateReportFromFile 17-15

CreateReportFromTemplate 17-15
CreateReportofAllQuantities 17-16
CreateText 19-18, 19-73
CreateUserDefinedSolution 22-1
CreateWire 19-20
Cut 19-28
Cut (Symbol) 19-74
CutDesign 7-3

D

dataset commands

AddDataset 12-1
DeleteDataset 12-2
EditDataset 12-3
ExportDataset 12-4
ImportDataset 12-3
DeactivateOpen 19-29
DeactivateShort 19-29
Delete 19-30
Delete (Symbol) 19-76
DeleteAllReports 17-17
DeleteDataset 12-2
DeleteDesign 7-3
DeleteDesignInstance 13-11
DeleteMarker 17-17
DeleteOutputVariable 13-11, 13-11,
16-2, 16-2
DeletePage 19-30

DeletePort 13-12
DeletePortImpedance 13-12
DeletePortSource 13-13
DeleteProject 5-4
DeleteReports 17-18
DeleteSetups, Optimetrics module
command 18-5
DeleteSolutionVariation 13-13
DeleteTraceCharacteristics 17-18
DeleteTraces 17-19
Design analysis setup commands
AddACAnalysis 14-1
AddAnalysisOptions 14-2
AddDCAnalysis 14-6
AddTransient 14-7
EditACAnalysis 14-8
EditAnalysisOptions 14-9
EditDCAnalysis 14-13
EditTransient 14-14
GetAllOptiSolutionSetups 14-15
GetAllSolutionSetups 14-15
GetNom-
inalSetupNamesForOptiSetup
14-16
GetSDBNameForSetupName 14-
17
GetSDBNamesForOptiSetup 14-
17

GetSetupInfo 14-18	ClearAllMarkers 17-8
Remove 14-18	ClearAllTraceCharacteristics 17-8
Rename 14-19	CloseAllWindows 5-2
SetSetupInfo 14-19	CompileAsTwin 13-10
Design object commands 17-27, 17-28, 17-29, 17-29, 17-30	CopyReportData 17-9
AddCartesianXMarker 17-2	CopyReportDefinitions 17-9
AddCartesianYMarker 17-2	CopyTraceDefinitions 17-9
AddDeltaMarker 17-3	CopyTracesData 17-10
AddMarker 17-3	CreateOutputVariable 16-1
AddMessage 13-1	CreateReport 17-10
AddNote 17-4	CreateReportFromFile 17-15
AddPortImpedance 13-1	CreateReportFromTemplate 17-15
AddPortSource 13-2	CreateReportOfAllQuantities 17-16
AddSdbOutput 13-3	DeleteAllReports 17-17
AddTraceCharacteristics 17-5	DeleteDesignInstance 13-11
AddTraces 17-5	DeleteOutputVariable 13-11, 13-11, 16- 2, 16-2
Analyze 13-3	DeletePort 13-12
AnalyzeAll 13-4	DeletePortImpedance 13-12
AnalyzeInBackground 13-4	DeletePortSource 13-13
AnalyzeUI 13-5	DeleteReports 17-18
ChangeGPort 13-5	DeleteSolutionVariation 13-13
ChangePort 13-6	DeleteTraceCharacteristics 17-18
ChangePortImpedanceProperty 13-7	DeleteTraces 17-19
ChangePortSourceProperty 13- 8	DoesOutputVariableExist 16-2
ChangeProperty 13-8	EditCartesianXMarker 17-19
	EditDesignSettings 13-14

EditMarker	17-20	GetTempDirectory	5-15
EditOutputVariable	16-3	GetVariables	11-37, 13-27
ExportAsTwin	9-20, 9-72, 13-8	GetVariableValue	11-38, 13-28
ExportAsTwinModel	9-73	GetVersion	5-16
ExportNetlist	13-15	GroupPlotCurvesByGroupingStrategy	17-42
ExportPlotImageToFile	17-22	ImportIntoReport	17-42
ExportReport	17-23	ImportSolution	13-28
ExportToFile	17-24, 17-24	InsertDesign	7-8, 13-29
FFTONReport	17-26	LoadConfiguration	13-30
GetDisplayType	17-33	ManageOutputs	13-31
GetLibraryDirectory	5-12	MovePlotCurvesToGroup	17-44
GetLinkComponents	13-17	MovePlotCurvesToNewGroup	17-45
GetLinkInstances	13-18	OpenWindowForAllReports	17-45
GetModule	13-19	PasteReports	17-46
GetName	13-20	PasteTraces	17-46
GetOutputVariables	16-4	Redo	13-34
GetOutputVariableValue	16-4	RemoveAllSdbOutputs	13-34
GetProjectDirectory	5-13	RemoveReport	17-47, 17-50
GetProperties	13-22	RemoveSdbOutput	13-34
GetPropHost	13-22	RenameDesignInstance	13-35
GetResultsDirectory	13-23	RenamePort	13-35
GetSdbOutputs	13-23	RenameReport	17-47
GetSimulationOptionFilePath	13-24, 22-5	RenameTrace	17-48
GetSimulationSettings	13-25, 22-6	ResetPlotSettings	17-48
GetSubDesigns	13-27	ResetSymbolAnimation	13-36

SaveConfiguration 13-36	17-52
SavePlotSettingsasDefault 17-49	Desktop object commands
SetActiveEditor 13-37	AddMessage 5-1
SetAssignsDuringSimulation 13-37	CloseProject 5-3
SetDCBiasDisplayOptions 13-38	CloseProjectNoForce 5-3
SetEventCallback 13-39	ExportOptionsFiles 5-5
SetLibraryDirectory 5-28	GetActiveProject 5-5
SetLinkStatic 13-40	GetAutoSaveEnabled 5-6
SetProjectDirectoryVBCommand> 5-28	GetBuildDataTimeString 5-6
SetSimulationOptionFilePath 13-40, 22-8	GetCustomMenuSet 5-7
SetSimulationSettings 13-42, 22-9	GetDesktopConfiguration 5-9
SetTempDirectory 5-29	GetDistributedAnalysisMachines 5-9
SetVariableValue 11-39, 13-43	GetDis-trib-utedAna-lysisMachinesForDesignType 5-10
Solve 13-45	GetMessages 5-11
StartAnalysis 13-44	GetProjectList 5-14
ToggleDCBiasDisplay 13-44	GetProjects 5-14
UnGroupPlotCurvesInGroup 17-49	GetSchematicEnvironment 5-14
UpdateAllReports 17-53	LaunchJobMonitor 5-17
UpdateDCBiasDisplay 13-45	NewProject 5-17
UpdateLink 13-46	OpenProject 5-18
UpdateReports 17-54	PauseRecording 5-18
UpdateTracesContextAndSweeps	PauseScript 5-19
	Print 5-19
	QuitApplication 5-20

RestoreJobMonitor 5-20	EditMarker 17-20
RestoreWindow 5-21	EditMaterial 9-2
RunProgram 5-22	EditOutputVariable 16-3
RunScript 5-22	EditScript 9-65
SelectScheduler 5-25	EditSetup
SetActiveProject 5-26	optimization command 18-15
SetActiveProjectByPath 5-26	parametric command 18-12
SetCustomMenuSet 5-27	sensitivity command 18-23
SetDesktopConfiguration 5-27	statistical command 18-29
SetSchematicEnvironment 5-29	EditTransient 14-14
Sleep 5-30	EditWithComps [Symbol Manager] 9-55
SubmitJob 5-31	ElectricRuleCheck 19-31
Differentiation 21-17	EnableAutoSave 5-4
DisplayPinNames 19-75	ExecuteScript 11-22
DistributedAnalyzeSetup, Opti- metrics module command 18-5	Export [Component] 9-32
DoesOutputVariableExist 16-2	Export [Model Manager] 9-70
	Export [Package Manager] 9-77
	Export [Symbol Manager] 9-61
E	ExportAllProjectComps (Component) 9-32
Edit [Symbol Manager] 9-51	ExportAsFMU [Model Manager] 9-70
EditACAnalysis 14-8	ExportAsTwin 9-20, 9-72, 13-8
EditAnalysisOptions 14-9	ExportAsTwinModel 9-73
EditCartesianXMarker 17-19	ExportDataset 12-4
EditComponent 9-21	ExportDefinitionArchive [Component] 9-34
EditDataset 12-3	
EditDCAnalysis 14-13	
EditDesignSettings 13-14	

ExportDefs [Component] 9-17, 9-33

ExportDXConfigFile, Optimetrics
module command 18-6

ExportFile (Symbol) 19-76

ExportFullWaveSpice 9-81

ExportImage 19-32

ExportMarkerTable 17-21

ExportMaterial 9-3

ExportNetlist 13-15

ExportNetworkData 9-83

ExportNMFData 9-85

ExportOptimetricsProfile, Opti-
metrics module command 18-
6

ExportOptimetricsResults, Opti-
metrics module command 18-
7

ExportOptionsFiles 5-5

ExportParametricResults, Opti-
metrics module command 18-
8

ExportPlotImageToFile 17-22

ExportReport 17-23

ExportScript 9-66

ExportToFile 17-24, 17-24

ExportUniformPointsToFile 17-25

F

FFT 21-17

FFTONReport 17-26

FindElements 19-32

FitToBorder 19-34

FlipHorizontal 19-34

FlipHorizontal (Symbol) 19-79

FlipVertical 19-35

FlipVertical (Symbol) 19-79

For Loop 2-2

G

GenerateVariationData Parametric, para-
metric command 18-12

GetACFrequencyMax 21-23

GetACFrequencyMin 21-24

GetACFrqMax 21-23

GetACFrqMin 21-24

GetACMax 21-24

GetACMin 21-25

GetACRMS 21-25

GetActiveDesign 7-4

GetActiveProject 5-5

GetAllCategories 17-27

GetAllOptiSolutionSetups 14-15

GetAllQuantities 17-29

GetAllReportNames 17-28

GetAllSolutionSetups 14-15

GetApplication 11-23

GetAutoSaveEnabled 5-6

GetAvailableDisplayTypes 17-29	GetDelayTime 21-38
GetAvailableReportTypes 17-30	GetDependentFiles 7-5
GetAvailableSolutions 17-30	GetDescription 11-24
GetAvalableTasks 20-5	GetDesign 7-5, 13-16
GetBandwidth 21-27	GetDesigns 7-6
GetBuildDataTimeString 5-6	GetDesignVariableNames 17-32
GetCallBack 11-23	GetDesignVariableUnits 17-32
GetChangedProperty 11-24	GetDesignVariableValue 17-32
GetChannelBasedISE 21-19	GetDesignVariationKey 17-33
GetCompInstanceFromRefDes 19-53	GetDisplayType 17-33
GetComponentInfo 19-54	GetDistortion 21-82
GetComponentName 10-2	GetDistributedAnalysisMachines 5-9
GetComponentPinInfo 19-55	GetDis- trib- utedAna- lysisMachinesForDesignType 5-10
GetComponentPinLocation 19-56	
GetComponentPins 19-55	
GetCrestFactor 21-82	GetEditor 10-2, 11-25
GetCustomRNDValue 21-93	GetEvaluatedText 11-25
GetData [Component] 9-34	GetExeDir 5-10
GetData [Material Manager] 9-4	GetFFTFreq 21-44
GetData [Model Manager] 9-74	GetFFTImag 21-45
GetData [Package Manager] 9-78	GetFFTMag 21-45
GetData [Script Manager] 9-66	GetFFTPhase 21-46
GetDataExpressions 17-31	GetFFTReal 21-46
GetDataUnits 17-31	GetFFTResultCIT 21-47
GetDeadTime 21-38	GetFFTResultCMWI 21-47
GetDefinitionManager 7-4	GetFFTResultDF 21-48

GetFFTResultIECClassA 21-48	GetHidden 11-26
GetFFTResultIECClassB 21-48	GetIAE 21-35
GetFFTResultIECClassC 21-49	GetImagDataValues 17-34
GetFFTResultIECClassD 21-49	GetInstanceID 10-3
GetFFTResultIECInstallation 21-49	GetIntegral 21-84
GetFFTResultIECThreePhase 21-50	GetISE 21-35
GetFFTResultIEEE519Current 21-50	GetITAE 21-36
GetFFTResultIEEE519Voltage 21-51	GetITSE 21-36
GetFFTResultIT 21-51	GetLibContents [Component] 9-35
GetFFTResultKFactor 21-51	GetLibraryDefNames [Model Manager] 9-74
GetFFTResultMagnitude 21-52	GetLibraryDefNames [Package Manager] 9-78
GetFFTResultPhaseDeg 21-52	GetLibraryDefNames [Symbol Manager] 9-62
GetFFTResultRipple 21-53	GetLibraryDirectory 5-12
GetFFTResultRMS1 21-53	GetLinkComponents 13-17
GetFFTResultRMSAC 21-54	GetLinkInstances 13-18
GetFFTResultRMSDC 21-54	GetLowerCutoff 21-29
GetFFTResultTHD 21-55	GetMax 21-84
GetFFTResultTIF 21-55	GetMaxTime 21-85
GetFileName 11-26	GetMean 21-85
GetFinalError 21-34	GetMeanAbs 21-86
GetFormFactor 21-83	GetMenuChoices 11-26
GetFullComponentName 10-3	GetMessages 5-11
GetFundamentalMag 21-83	GetMin 21-86
GetGainCrossover 21-27	GetMinTime 21-87
GetGainMargin 21-28	

GetModule	13-19	GetPhaseCrossover	21-31
GetName	7-6, 13-20	GetPhaseMargin	21-32
GetNames [Component]	9-36	GetPortInfo	19-57
GetNames [Material Manager]	9-4	GetProcessID	5-13
GetNames [Model Manager]	9-75	GetProcessResources	4-1
GetNames [Package Manager]	9-79	GetProjectDirectory	5-13
GetNames [Script Manager]	9-67	GetProjectList	5-14
GetNominalSetupNamesForOptiSetup	14-16	GetProjects	5-14
GetNormRandMeanTol	21-92	GetProperties	7-7, 11-34, 13-22
GetNormRandMinMax	21-92	GetProperties [Material Manager]	9-5
GetNumPages	19-57	GetProperties [Schematic]	19-50
GetOutputVariables	16-4	GetPropertyAttribute [Schematic]	19-51
GetOutputVariableValue	16-4	GetPropertyValue	11-35, 17-35, 17-37
GetOvershoot	21-39	GetPropertyValue [Schematic]	19-52
GetOvershootTime	21-39	GetPropHost	10-4, 13-22
GetParentDesign	10-4	GetPropServerName	10-5
GetPath	7-7	GetPropServers	11-27
GetPeakAverage	21-87	GetPropTabType	11-27
GetPeakGain	21-30	GetPulseFall9010	21-61
GetPeakGainFreq	21-30	GetPulseFront1090	21-62
GetPeakPeak	21-88	GetPulseFront3090	21-62
GetPerQuantityPrimarySweepValues	17-35	GetPulseMax	21-63
GetPersonalLibDirectory	5-12	GetPulseMaxTime	21-63
		GetPulseMin	21-64
		GetPulseMinTime	21-64

GetPulseTail150 21-65	GetPWRSystemVe1 21-79
GetPulseWidth5050 21-65	GetReadOnly 11-28
GetPWRPhaseDP 21-67	GetRealDataValues 17-37
GetPWRPhaseEnergy 21-68	GetReportTraceNames 17-38
GetPWRPhaseHP 21-69	GetResultsDirectory 13-23
GetPWRPhaseP 21-69	GetRipple 21-88
GetPWRPhaseP1 21-70	GetRiseTime 21-40
GetPWRPhasePF 21-70	GetRMS 21-89
GetPWRPhasePF1 21-71	GetRMSAC 21-89
GetPWRPhasePower 21-71	GetScriptName 11-28
GetPWRPhaseQ1 21-72	GetSDBNameForSetupName 14-17
GetPWRPhaseS 21-72	GetSDBNamesForOptiSetup 14-17
GetPWRPhaseS1 21-73	GetSdbOutputs 13-23
GetPWRSystemDPe 21-73	GetSelections 19-58
GetPWRSystemEnergy 21-74	GetSettlingTime 21-40
GetPWRSystemHPe 21-74	GetSetupInfo 14-18
GetPWRSystemle 21-75	GetSetupNames (Optimetrics), Optimetrics module command 18-8
GetPWRSystemle1 21-75	GetSetupNamesByType (Optimetrics), Optimetrics module command 18-9
GetPWRSystemP 21-75	GetSignals 19-59
GetPWRSystemP1 21-76	GetSimulationOptionFilePath 13-24, 22-5
GetPWRSystemPF1 21-76	GetSimulationSettings 13-25, 22-6
GetPWRSystemPFe 21-77	GetSlidingMean 21-59
GetPWRSystemPower 21-77	GetSlidingRMS 21-60
GetPWRSystemQ1 21-78	GetSolutionContexts 17-38
GetPWRSystemS1 21-78	GetSubDesigns 13-27
GetPWRSystemSe 21-78	
GetPWRSystemVe 21-79	

GetSweepNames 17-40

GetSweepUnits 17-41

GetSweepValues 17-41

GetSysLibDirectory 5-15

GetTabTypeName 11-29

GetTempDirectory 5-15

GetText 11-29

GetTopDesignList 7-8

GetUnitRandMeanTol 21-91

GetUnitRandMinMax 21-91

GetUpperCutoff 21-33

GetUserLibDirectory 5-16

GetValue 11-30, 21-81

GetValueLogInterpolation 21-42

GetVariables 11-37, 11-37, 13-27,
13-27GetVariableValue 11-38, 11-38, 13-
28, 13-28

GetVersion 5-16

GetVisibleLevels [Symbol] 19-77

GetWireConnections 19-60

GetWireInfo 19-60

GetWireSegments 19-62

GridSetup 19-36, 19-77

GroupPlotCurvesByGroup-
ingStrategy 17-42

I

ImportDataset 12-3

ImportDefinitionArchive
[Component] 9-36

ImportIntoReport 17-42

ImportModelsFromFile
[Component] 9-37ImportSetup, Optimetrics module com-
mand 18-9ImportSolution, design-level com-
mand 13-28InsertDesign 7-8, 13-29
design-level command 7-8, 13-29InsertSetup
optimization command 18-19
parametric command 18-13
sensitivity command 18-26
statistical command 18-29

Integration 21-20

IsDataComplex 17-43

IsPerQuantityPrimarySweep 17-43

IsUsed [Component] 9-41

IsUsed [Material Manager] 9-5

IsUsed [Model Manager] 9-76

IsUsed [Package Manager] 9-79

IsUsed [Script Manager] 9-67

IsValueConstant 11-30

Iterative Statements 2-2, 2-2

J

JavaScript, script format 1-1

L

LaunchJobMonitor 5-17

Lists 2-4

LoadConfiguration, design-level
command 13-30

LoadExample [Component] 9-42

logical operators 2-4

M

ManageAliases [Component] 9-43

ManageOutputs 13-31

material commands

AddMaterial 9-1

EditMaterial 9-2

ExportMaterial 9-3

RemoveMaterial 9-6

modifying a script 1-8

ModifyLibraries 9-69

modules in HFSS scripting 1-4

Move 19-38

Move (Symbol) 19-80

MovePins 19-38

MovePlotCurvestoGroup 17-44

MovePlotCurvestoNewGroup 17-45

N

NameNets 19-39

NewProject 5-17

O

oAnsoftApp object 1-2

Object Oriented Property Scripting 8-1

oDesign object 1-3

oDesktop object 1-2

oEditor object 1-3

oModule object 1-4

OpenProject 5-18

OpenWindowForAllReports 17-45

operators

arithmetic 2-3

categories in Python 2-2

comparison 2-3

logical 2-4

precedence of 2-2

oProject 1-2

Optimetrics module commands

DeleteSetups 18-5

DistributeAnalyzeSetup 18-5

ExportDXConfigFile 18-6

ExportOptimetricsResults 18-7

ExportParametricResults 18-8	PasteSetup, Optimetrics module command 18-10
GetSetupNames 18-8	PasteTraces 17-46
GetSetupNamesByType 18-9	PauseRecording 5-18
ImportSetup 18-9	PauseScript 5-19
RenameSetup 18-10	pausing a script 1-7
SolveSetup 18-4, 18-10, 18-11, 18-11	Print 5-19
optimization commands	Project object commands
EditSetup 18-15	AddDataset 12-1
InsertSetup 18-19	AddMaterial 9-1
output variable commands	AnalyzeAll 7-1
CreateOutputVariable 16-1	ChangeProperty 11-5
DeleteOutputVariable 13-11, 16-2	ClearMessages 5-2, 7-1
EditOutputVariable 16-3	Close 7-2
GetOutputVariables 16-4	ConvertSMLToSMT 4-2
GetOutputVariableValue 16-4	CopyDesign 7-2
P	CutDesign 7-3
Pan 19-40, 19-81	DeleteDataset 12-2
parametric commands	DeleteDesign 7-3
EditSetup 18-12	EditDataset 12-3
GenerateVariationData Parametric 18-12	EditMaterial 9-2
InsertSetup 18-13	ExportMaterial 9-3
Paste 7-9, 19-41	GetActiveDesign 7-4
Paste [Symbol] 19-81	GetDesigns 7-6
PasteReports 17-46	GetName 7-6
	GetPath 7-7
	GetProperties 7-7, 11-34

-
- GetPropertyValue 11-35, 17-35
 - GetTopDesignList 7-8
 - GetVariables 11-37, 13-27
 - GetVariableValue 11-38, 13-28
 - ImportDataset 12-3
 - InsertDesign 7-8, 13-29
 - Paste 7-9
 - Redo 7-9
 - RemoveMaterial 9-6
 - Rename 7-10
 - Save 7-10
 - SaveAs 7-11
 - SaveProjectArchive 7-11
 - SetActiveDesign 7-12
 - SetPropertyValue 11-38
 - SetVariableValue 11-39, 13-43
 - SimulateAll 7-13
 - Undo 7-13
 - UpdateDefinitions 7-14
 - property commands
 - GetProperties 11-34
 - GetPropertyValue 11-35, 17-35
 - GetVariables 11-37, 13-27
 - GetVariableValue 11-38, 13-28
 - SetPropertyValue 11-38
 - SetVariableValue 11-39, 13-43
 - PropertyExists 11-31
 - PushExcitations 13-31
 - Python
 - operators 2-2
 - references 2-6
- Q**
- QuitApplication 5-20
- R**
- RecompileLibraries 4-1
 - recording a script 1-4
 - Redo
 - design-level command 13-34
 - project-level command 7-9
 - Redo (Symbol) 19-82
 - references, for Python 2-6
 - RefreshJobMonitor 5-20
 - ReleaseData 17-47
 - Remove 14-18
 - Remove [Component] 9-44
 - Remove [Model Manager] 9-76
 - Remove [Package Manager] 9-80
 - Remove [Symbol Manager] 9-63
 - RemoveAllSdbOutputs 13-34
 - RemoveLevels 19-82
 - RemoveMaterial 9-6
 - RemoveProp 11-31
-

RemoveReport 17-47, 17-50	ClearAllTraceCharacteristics 17-8
RemoveScript 9-68	CopyReportDefinition 17-9
RemoveSdbOutput 13-34	CopyReportDefinitions 17-9
RemoveUnused [Component] 9-44	CopyTraceDefinitions 17-9
RemoveUnused [Material Manager] 9-6	CopyTracesData 17-10
RemoveUnused [Model Manager] 9-77	CreateReport 17-10
RemoveUnused [Package Man- ager] 9-80	CreateReportFromFile 17-15
RemoveUnused [Script Manager] 9-68	CreateReportFromTemplate 17-15
Rename 14-19	DeleteAllReports 17-17
project-level command 7-10	DeleteReports 17-18
RenameDesignInstance 13-35	DeleteTraceCharacteristics 17-18
RenamePort 13-35	DeleteTraces 17-19
RenameReport 17-47	EditCartesianXMarker 17-19
RenameSetup, Optimetrics module command 18-10	EditMarker 17-20
RenameTrace 17-48, 17-49	ExportPlotImageToFile 17-22
Reporter editor commands	ExportToFile 17-24, 17-24
AddCartesianXMarker 17-2, 17- 2	ExportUniformPointsToFile 17-25
AddDeltaMarker 17-3	FFTONReport 17-26
AddMarker 17-3	GetAllReportNames 17-28
AddNote 17-4	GetDisplayType 17-33
AddTraceCharacteristics 17-5	GetReportTraceNames 17-38
AddTraces 17-5	ImportIntoReport 17-42
ClearAllMarkers 17-8	OpenWindowForAllReports 17-45
	PasteReports 17-46
	PasteTraces 17-46
	RemoveReport 17-47, 17-50

RenameReport 17-47
RenameTrace 17-48
ResetPlotSettings 17-48
SavePlotSettingsasDefault 17-49
UnGroupPlotCurvesInGroup 17-49
UpdateAllReports 17-53
UpdateReports 17-54
UpdateTracesContextAndSweeps 17-52
ReportSetup commands
 ChangeProperty 17-7
 GetPropertyValue 17-37
ResetPlotSettings 17-48
ResetSymbolAnimation 13-36
RestoreWindow 5-21
ResumeRecording 5-21
resuming a script 1-7
RevertToDefaultSymbol 19-41
Rotate 19-42
Rotate (Symbol) 19-83
running a script 1-5
RunProgram 5-22
RunScript 5-22
RunScriptWithArguments 5-24

S

Save 7-10
Save (Symbol) 19-84
SaveAs 7-11
SaveConfiguration, design-level command 13-36
Schematic 19-50
scripts
 in JavaScript format 1-1
 modifying for easier playback 1-7
 pausing 1-7
 recording 1-4
 resuming 1-7
 running from command prompt 1-1
 stop recording 1-5
 stopping execution of 1-7
SDB control commands
 CloseSDBFile 20-2
 GetAnalysisType 20-8
 GetArrayDim 20-15
 GetAvailableAnalysis 20-7
 GetAvailableChannels 20-8
 GetAvailableExpChannel 20-9
 GetAvalableTasks 20-5
 GetBaseChannelName 20-13
 GetChannelDim 20-14

GetChannelName 20-16	SetAssignsDuringSimulation 13-37
GetDataType 20-9	SetCallback 11-31
GetDataTypeExp 20-11	SetDCBiasDisplayOptions 13-38
GetEnumString 20-13	SetDescription 11-32
GetLastError 20-1	SetEventCallback 13-39
GetNumberOfRuns 20-6	SetHidden 11-32
GetNumberOfSamples 20-11	SetInitialLevels 19-86
GetRunNumbers 20-7	SetLibraryDirectory 5-28
GetTaskType 20-5	SetLinkStatic 13-40
OpenSDBFile 20-2	SetProjectDirectory 5-28
ReadData 20-3	SetPropertyValue 11-38
ReadImagData 20-4	SetPropertyValue [Schematic] 19-53
SelectAll 19-43	SetReadOnly 11-33
SelectAll (Symbol) 19-84	SetSetupInfo 14-19
SelectElement 19-43	SetSimulationOptionFilePath 13-40, 22-8
SelectPage 19-44	SetSimulationSettings 13-42, 22-9
SelectScheduler 5-25	SetTempDirectory 5-29
SendToBack 19-44, 19-85	SetText 11-33
sensitivity commands	SetupFFT 21-43
EditSetup 18-23	SetupPWR 21-67
InsertSetup 18-26	SetValue 11-34
SetActiveDefinitionEditor 7-12	SetVariableValue 11-39, 11-39, 13- 43, 13-43
SetActiveDesign 7-12	SetVisibleLevels 19-86
SetActiveEditor 13-37	SGetAppDesktop 3-1
SetActiveLevel 19-85	Simplorer Application Object com-
SetActiveProject 5-26	
SetActiveProjectByPath 5-26	

mands 4-1
 RecompileLibraries 4-1
 SimulateAll 7-13
 SimulateSMT 4-2
 SimValueContext 16-5
 Sleep 5-30
 Smooth 21-20
 SolveAllSetup, Optimetrics module
 command 18-11
 SolveSetup, Optimetrics module
 command 18-11
 SortComponents 19-47
 StartAnalysis 13-44
 statistical commands
 EditSetup 18-29
 InsertSetup 18-29
 stopping a script 1-7
 stopping script recording 1-5
 SubmitJobr 5-31

T

Template 13-45, 18-6
 ToggleDCBiasDisplay 13-44
 ToggleLevel 19-87

U

Undo
 project-level command 7-13

Undo (Symbol) 19-88
 UnGroupPlotCurvesInGroup 17-49
 UpdateAllReports 17-53
 UpdateDCBiasDisplay 13-45
 UpdateDefinitions, project-level
 command 7-14
 UpdateLink 13-46
 UpdateReports 17-54
 UpdateTracesContextAndSweeps 17-52
 UserDefinedSolutions module commands
 CreateUserDefinedSolution 22-1
 DeleteUserDefinedSolution 22-3
 EditUserDefinedSolution 22-3

V

variables
 assigning information 2-4
 List 2-4
 Value Assignment 2-4

W

While Loop 2-2, 2-2

Z

ZoomArea 19-47
 ZoomArea (Symbol) 19-88
 ZoomIn 19-48
 ZoomIn (Symbol) 19-89

ZoomOut 19-48

ZoomOut (Symbol) 19-89

ZoomPrevious 19-49

ZoomPrevious (Symbol) 19-90

ZoomToFit 19-49

ZoomToFit (Symbol) 19-91