



POWERING INNOVATION THAT DRIVES HUMAN ADVANCEMENT

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

EMIT 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. ICM CFD is a trademark used by ANSYS, Inc. under license. All other brand, product, service and feature names or trademarks are the property of their respective owners. FLEXIm and FLEXnet are trademarks of Flexera Software LLC.

Disclaimer Notice

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

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

U.S. Government Rights

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

Third-Party Software

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

Table of Contents

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

Output Parameters	1-15
Comparison with C function:	1-16
'List Size' Parameters	1-17
Comparison with C function:	1-17
Added Parameters	1-18
Developing a UDM/UDP	1-19
Creation	1-19
Location	1-19
Organize	1-19
Edit/Reload	1-20
UDPExtension	1-20
IUDPExtension Methods	1-20
Mandatory Methods	1-20
GetLengthParameterUnits()	1-20
Optional Methods	1-21
Example UDP	1-22
UDMExtension	1-22
Import	1-22
Main class: UDMExtension	1-22
IUDMExtension Methods	1-23
Mandatory Methods	1-23
Optional Methods	1-24
Example UDM	1-25
UDMFunctionLibrary	1-26
Functions list:	1-26
UDM/UDP Functions	1-27
Return Values for Each UDM and UDP Function	1-27
UDP/UDM Structures and Constants	1-30
UDP/UDM Structures	1-30
List of Structures	1-32

UDP/UDM Constants	1-37
Enum constants:	1-38
Introduction to CPython	1-40
Creating an External Script	1-40
Start ansyedt as GRPC server	1-40
Connect Functions:	1-41
Example:	1-41
Launching Electronics Desktop	1-41
Connecting with a Running Instance of Electronics Desktop	1-42
Closing Electronics Desktop/Ending the Script	1-42
-grpcsrv Flag	1-42
Anslys Electronics Desktop Scripting	1-43
Overview of Electronics Desktop Scripting Objects	1-43
oAnsoftApp	1-44
oDesktop	1-44
oProject	1-45
oDesign	1-45
oEditor	1-45
oModule	1-46
Example Script Opening	1-46
GetActiveProject and GetActiveDesign for Wider Use	1-46
Running a Script	1-46
Within Electronics Desktop	1-47
From the Command Line	1-47
Recording a Script	1-48
Recording a Script to File	1-48
Recording a Script to a Project	1-49
Working with Project Scripts	1-49
Executing a Script from Within a Script	1-50
Electronics Desktop Scripting Conventions	1-51

Named Arguments	1-51
IronPython Example	1-52
Setting Numerical Values	1-53
2 - Application Object Script Commands	2-1
GetAppDesktop	2-2
3 - Desktop Object Script Commands	3-1
AddMessage	3-5
ClearMessages	3-6
CloseAllWindows	3-10
CloseProject	3-10
CloseProjectNoForce	3-11
Count	3-12
DeleteProject	3-12
DownloadJobResults	3-13
EnableAutoSave	3-14
ExportOptionsFiles	3-14
GetActiveProject	3-15
GetAutoSaveEnabled	3-16
GetBuildDateTimeString	3-16
GetDefaultUnit	3-17
GetDesigns	3-19
GetDistributedAnalysisMachines	3-19
GetDistributedAnalysisMachinesForDesignType	3-20
GetExeDir	3-21
GetGDIObjectCount	3-21
GetLibraryDirectory	3-22
GetLocalizationHelper	3-22
GetMessages	3-23
GetName [Desktop]	3-24
GetPersonalLibDirectory	3-24

GetProcessID	3-25
GetProjects	3-26
GetProjectDirectory	3-26
GetProjectList	3-27
GetScriptingToolsHelper	3-27
GetSysLibDirectory	3-28
GetTempDirectory	3-29
GetUserLibDirectory	3-29
GetVersion	3-30
ImportANF	3-30
ImportAutoCAD	3-31
ImportGDSII	3-32
ImportODB	3-33
LaunchJobMonitor	3-33
NewProject	3-34
OpenAndConvertProject	3-34
OpenMultipleProjects	3-35
OpenProject	3-36
OpenProjectWithConversion	3-37
PauseRecording	3-37
PauseScript	3-38
Print	3-38
QuitApplication	3-39
RefreshJobMonitor	3-39
ResetLogging	3-40
RestoreProjectArchive	3-41
RestoreWindow	3-41
ResumeRecording	3-42
RunACTWizardScript	3-43
RunProgram	3-43

RunScript	3-44
RunScriptWithArguments	3-45
SelectScheduler	3-46
SetActiveProject	3-48
SetActiveProjectByPath	3-48
SetLibraryDirectory	3-49
SetProjectDirectory	3-49
SetTempDirectory	3-50
ShowDockingWindow	3-51
Sleep	3-51
SubmitJob	3-52
TileWindows	3-53
Desktop Commands For Registry Values	3-53
DoesRegistryValueExist	3-54
GetRegistryInt	3-55
GetRegistryString	3-56
SetRegistryFromFile	3-56
SetRegistryInt	3-57
SetRegistryString	3-58
4 - Running Instances Manager Script Commands	4-1
GetAllRunningInstances	4-1
GetRunningInstanceByProcessID	4-1
GetRunningInstancesMgr	4-2
5 - Project Object Script Commands	5-1
AddDataset	5-4
AnalyzeAll [project]	5-6
ClearMessages	5-7
Close	5-8
CopyDesign	5-8
CutDesign	5-9

DeleteDataset	5-10
DeleteDesign	5-10
DeleteToolObject	5-11
EditDataset	5-11
GetActiveDesign	5-13
GetChildNames [Project]	5-14
GetChildObject [Project]	5-15
GetChildTypes [Project]	5-15
GetDefinitionManager	5-16
GetDependentFiles	5-17
GetEDBHandle	5-17
GetLegacyName	5-18
GetName [Project]	5-19
GetPath	5-19
GetPropNames [Project]	5-20
GetPropValue [Project]	5-20
GetTopDesignList	5-21
ImportDataset	5-22
Note About File Types:	5-23
InsertDesignWithWorkflow	5-24
InsertToolObject	5-25
Paste (Project Object)	5-26
Redo [Project Level]	5-26
Rename	5-27
RestoreProjectArchive	5-27
Save	5-28
SaveAs	5-29
SaveAsStandAloneProject	5-31
SaveProjectArchive	5-31
SetActiveDefinitionEditor	5-32

SetActiveDesign	5-33
SetPropValue [Project]	5-33
SimulateAll	5-34
Undo [Project]	5-35
UpdateDefinitions	5-35
ValidateDesign	5-36
6 - Property Script Commands	6-1
Object Script Property Function Summary	6-3
Object Path	6-3
Property Object	6-3
Project Object	6-5
Design Object	6-6
3D Modeler Object	6-6
Variable Object	6-7
Optimetrics Module Object:	6-8
Optimetrics Setup Object	6-9
ReportSetup(Results) Module Object:	6-9
ReportSetup(Results) Module Child Objects:	6-10
Radiation Module Object:	6-11
Radiation Module Child Objects:	6-11
Conventions Used in this Chapter	6-12
GetArrayVariables	6-15
GetProperties	6-15
GetPropertyValue	6-16
GetVariables	6-18
GetVariableValue	6-18
SetPropertyValue	6-19
SetVariableValue	6-20
7 - Design Object Script Commands	7-1
AddLink [EMIT]	7-3

AddResult [EMIT]	7-3
CloseResultWindow	7-4
DeleteAllResults	7-4
DeleteDesign	7-5
DeleteLink [EMIT]	7-5
DeleteResult [EMIT]	7-6
EditComponentNodes [EMIT]	7-6
EditNotes	7-8
GetAvailableLinkNames [EMIT]	7-8
GetComponentNodeNames [EMIT]	7-9
GetComponentNodeProperties [EMIT]	7-9
GetComponentWarnings [EMIT]	7-10
GetCouplingWarnings [EMIT]	7-10
GetCurrentResult	7-11
GetDesignType	7-11
GetLinkNames [EMIT]	7-12
GetManagedFilesPath	7-12
GetName	7-13
GetRadioNames [EMIT]	7-13
GetResultList	7-14
GetResultNotes [EMIT]	7-15
GetResultProperties	7-15
GetRevision	7-16
Redo [EMIT]	7-16
RenameDesign [EMIT]	7-17
RenameResult	7-17
RunToolkit [EMIT]	7-18
SetActiveEditor [EMIT]	7-18
SetResultNotes [EMIT]	7-19
ShowAnalysisAndResults [EMIT]	7-19

ShowCouplingDialog [EMIT]	7-20
ShowEditDialog [EMIT]	7-20
ShowResultWindow	7-21
TransferToEmit	7-22
Undo [Design]	7-22
UpdateLink [EMIT]	7-22
8 - Core Global Script Context Commands	8-1
AddErrorMessage	8-1
AddFatalMessage	8-1
AddInfoMessage	8-2
AddWarningMessage	8-2
LogDebug	8-3
LogError	8-3
9 - Library Management Script Commands	9-1
DeleteLibraryComponent	9-1
RenameLibraryComponent	9-1
10 - Schematic Scripting for EMIT	10-1
Editor Scripting IDs (EMIT)	10-3
BringToFront (Schematic Editor)	10-3
ChangeProperty [EMIT]	10-4
ClearSelections	10-5
Copy [EMIT]	10-5
CreateArc (Schematic Editor)	10-6
CreateCircle (Schematic Editor)	10-8
CreateComponent [EMIT]	10-10
CreateComponentAt [EMIT]	10-11
CreateCurve [Schematic]	10-12
CreateEllipse	10-13
CreateLine (Schematic Editor)	10-15
CreatePolygon (Schematic Editor)	10-17

CreateRectangle (Schematic Editor)	10-19
CreateText (Schematic Editor)	10-21
Cut [EMIT]	10-24
Delete [EMIT]	10-24
DisableEmitComponents [EMIT]	10-25
ExportImage (EMIT)	10-25
ExportToLibrary [EMIT]	10-26
GetAllComponents [EMIT]	10-27
GetAllElements [EMIT]	10-27
GetComponentBoundingBox [EMIT]	10-28
GetComponentLocation [EMIT]	10-28
GetComponentOrientation [EMIT]	10-29
GetComponentPortLocation [EMIT]	10-29
GetComponentPorts [EMIT]	10-30
GetEditorName (Schematic Editor)	10-30
GetNumComponentOrientations [EMIT]	10-31
GetPropertyValue [EMIT]	10-31
GetSelections [EMIT]	10-32
GetWireAtPort [EMIT]	10-32
GetWireConnections [EMIT]	10-33
GetZoomArea [EMIT]	10-33
IsEmitComponentDisabled [EMIT]	10-34
Move [EMIT]	10-34
Pan [EMIT]	10-35
Paste [EMIT]	10-36
PlaceComponent [EMIT]	10-36
RenameComponent [EMIT]	10-37
ReorientComponent [EMIT]	10-37
Select [EMIT]	10-38
SelectAll [EMIT]	10-38

SendToBack(Schematic Editor)	10-39
SetPropertyValue [EMIT]	10-40
Wire [EMIT]	10-40
ZoomArea [EMIT]	10-41
ZoomIn [EMIT]	10-42
ZoomOut [EMIT]	10-42
ZoomPrevious [EMIT]	10-43
ZoomToFit [EMIT]	10-43
Index	Index-1

1 - Introduction to Scripting

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

Electronics Desktop can record scripts in Python, and can run external scripts written in Python. Additionally, it contains a Python command shell for executing scripts.

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

The following sections contain more information about scripting:

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

Scripting Help Conventions

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

[ScriptName]

[Description of script use.]

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

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

Variable Types

The following data types are used throughout the help:

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

Introduction to IronPython

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

Scope

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

This document is also not a tutorial on the scripting of Ansys EM products. It complements the existing scripting guide (available from a product's Help menu) and provides a pythonic interpretation of that information.

Python Compatibility

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

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

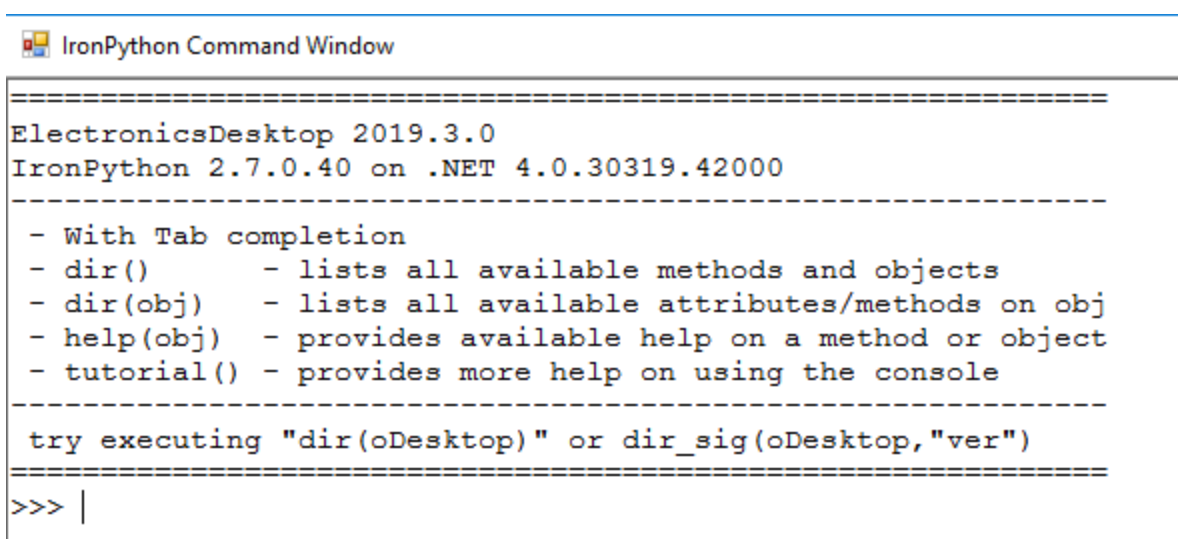
Advantages of IronPython

The advantages that IronPython use provides are significant:

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

- IronPython, in addition, has access to the entire .NET eco system. This allows us, for instance, to create a modern GUI using the **System.Windows.Forms** assembly from IronPython code and call any other .NET assembly for that matter.
- The use of IronPython's technologies enables the ability to interactively script Desktop (feature in development).
- The Python syntax of dictionaries is somewhat easier to read and write when supplying arguments to the scripting methods.

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



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

[Scripting Using Iron Python](#)

[Standalone IronPython and Desktop IronPython](#)

[IronPython Examples](#)

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

Scripting Using Iron Python

The following topics detail scripting using Iron Python:

[IronPython Script Execution Environment](#)

[Scripting with IronPython](#)

[Standalone IronPython and Desktop IronPython](#)

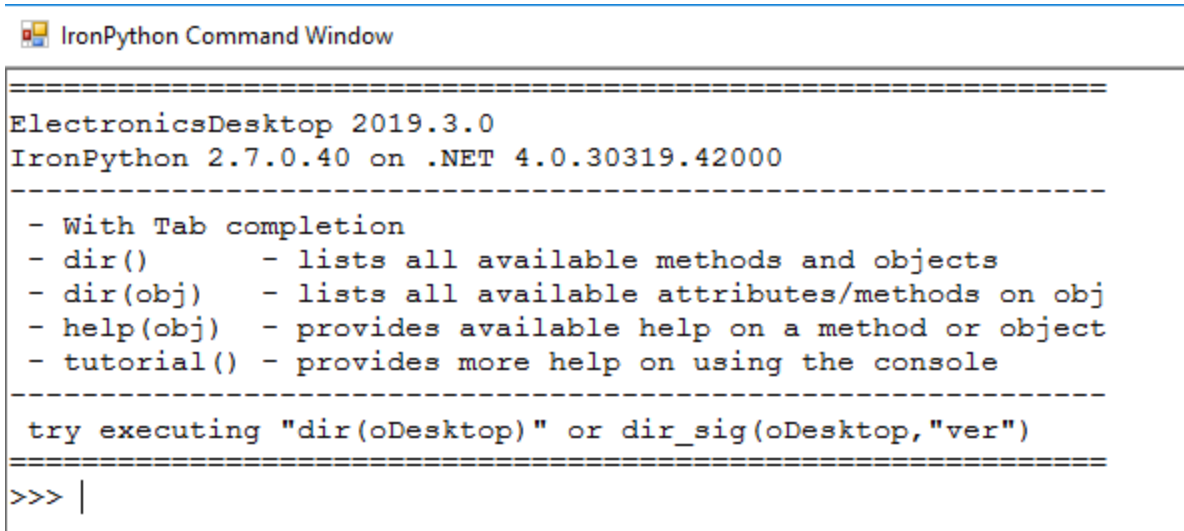
[Introduction to IronPython](#)

[Appendix: IronPython Samples](#)

IronPython Script Execution Environment

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

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



```
IronPython Command Window

=====
ElectronicsDesktop 2019.3.0
IronPython 2.7.0.40 on .NET 4.0.30319.42000
=====

- With Tab completion
- dir()      - lists all available methods and objects
- dir(obj)   - lists all available attributes/methods on obj
- help(obj)  - provides available help on a method or object
- tutorial() - provides more help on using the console
=====

try executing "dir(oDesktop)" or dir_sig(oDesktop,"ver")
=====

>>> |
```

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

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

[Script Argument for IronPython](#)

Script Argument for IronPython

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

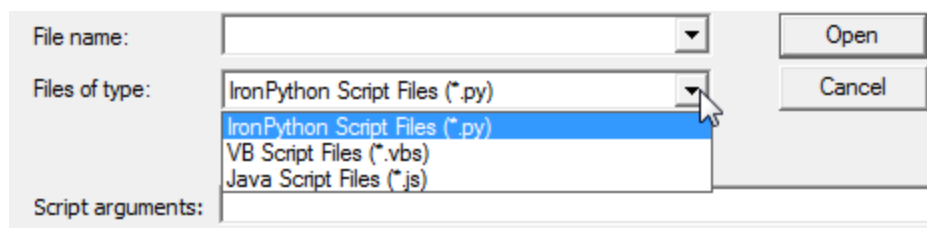


Figure 1: Run Script dialog and script arguments

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

Related Topics

[IronPython Script Execution Environment](#)

Scripting using Embedded VBScript or JavaScript

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

For example, a user can create a parameterized cone in HFSS by executing the following Python script from the **Tools > Run Script** menu:

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

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

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

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

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

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

```
SetScriptingLanguageToVBScript()  
RunScriptCommand(coneScript, "")
```

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

Related Topics

[IronPython Script Execution Environment](#)

Scripting with Python

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

Note the following:

- Any argument is supplied via the built in **ScriptArgument** variable.
- The **oDesktop** object is always available.
- Method calls have to adhere to the rule of ensuring trailing parentheses irrespective of whether the function returns anything or has any arguments.
- Any compound/block arguments should be translated to the appropriate Python array or dictionary syntax.

Related Topics

[IronPython Script Execution Environment](#)

[Standalone IronPython and Desktop IronPython](#)

Standalone IronPython

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

Running Standalone IronPython

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

It is located in:

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

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

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

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

Using a Recorded Script

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

Creating an External Script

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

- `import sys`
Imports the sys module containing system-specific functions native to IronPython.
- `sys.path.append("<InstallationPath>")`
Adds the Electronics Desktop installation path to the list of directories Python searches for modules and files.
- `sys.path.append("<InstallationPath>/PythonFiles/DesktopPlugin")`
Adds the PythonFiles/DesktopPlugin subfolder to the list of directories Python searches for modules and files.
- `import ScriptEnv`
This imports `ScriptEnv.py` from the installation path specified above. `ScriptEnv.py` performs an operating system check and defines functions used in Electronics Desktop scripts. See the annotations in the `ScriptEnv.py` file for more information.
- `ScriptEnv.Initialize("Ansoft.ElectronicsDesktop")`
or `ScriptEnv.InitializeNew(NonGraphical=True)`

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

You must end the script with:

- `ScriptEnv.Shutdown()`

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

Example Script

```
import sys

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

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

import ScriptEnv

ScriptEnv.Initialize("Ansoft.ElectronicsDesktop")

oDesktop.RestoreWindow()

oProject = oDesktop.NewProject()

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

oDesign = oProject.SetActiveDesign("HFSSDesign1")

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

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

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

oDesign.SetDesignSettings(['NAME:Design Settings Data', 'Allow Material Override:=', True, 'Calculate Lossy Dielectrics:=', True])

oEditor.SetModelUnits(['NAME:Units Parameter', 'Units:=', 'mil', 'Rescale:=' , False ])

ScriptEnv.Shutdown()
```

IronPython Samples

Important:

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

Change property

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

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

Sample Script: ChangeProperty command to change color of a cone

```
oEditor.ChangeProperty(
```

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

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

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

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

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

Create a Cone using IronPython

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

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

```

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

```

Sample Script: IronPython script to create a cone

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

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

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

```
oDesign = oProject.InsertDesign("HFSS", "Hersheys Kisses", "DrivenModal", "")
oEditor = oDesign.SetActiveEditor("3D Modeler")

# create the first cone
AddInfoMessage("Creating first cone")
firstConeName = "firstCone"
coneBotRad = "1.5mm"
oEditor.CreateCone(
    {
        "NAME" : "ConeParameters",
        "XCenter" : "0mm",
        "YCenter" : "0mm",
        "ZCenter" : "0mm",
        "WhichAxis" : "Z",
        "Height" : "2mm",
        "BottomRadius": coneBotRad,
        "TopRadius" : "0mm"
    },
    {
        "NAME" : "Attributes",
        "Name" : firstConeName,
        "Flags" : "",
        "Color" : "(132 132 193)",
        "Transparency" : 0,
        "PartCoordinateSystem": "Global",
        "UDMId" : "",
        "MaterialValue" : "\"vacuum\"",
        "SolveInside" : True
    }
}
```

```
)

# Now replicate this a few times and create an array out of it
AddInfoMessage("Replicating it 24 times")
for x in range(5):
    for y in range(5):
        # leave the first one alone in it's created
        # position
        if x == 0 and y == 0:
            continue

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

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

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

# now move the pasted item to it's final position
oEditor.Move(
    {
```

```
"NAME" : "Selections",
"Selections" : pasteName
},
{
"NAME" : "TransalateParameters",
"CoordinateSystemID" : -1,
"TranslateVectorX" : "%d * 3 * %s" % (x, coneBotRad),
"TranslateVectorY" : "%d * 3 * %s" % (y, coneBotRad),
"TranslateVectorZ" : "0mm"
}
)
```

```
# Now fit the display to the created grid
```

```
oEditor.FitAll()
```

Related Topics

[Introduction to IronPython](#)

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

Creating User Defined Primitives and User Defined Models in Python Scripts

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

Advantages Compared to C++

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

Changes compared to C

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

Structures

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

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

Return Values for UDM and UDP Functions

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

Constants

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

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

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

Methods

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

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

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

Output Parameters

The rule for the output parameters is as follows:

- If the function has one output parameter variable and no return value, the variable will become function's return value. The same will happen if the return value is a

'success/failure' boolean ('None' will be returned on failure and parameter variable - on success).

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

one output parameter; return value is ignored

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

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

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

Two output parameter; return value must be preserved

the return tuple is (returnVal, output1, output2)

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

Comparison with C function:

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

'List Size' Parameters

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

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

Example:

```
# input list, input list size
```

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

```
# output list, output list size
```

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

```
# output list, input list size
```

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

Comparison with C function:

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

C	Python
where numObjects and objectIds are input parameters and numObjects is the size of objectIds.	(ignore numObjects as redundant: folded into objectIds callbackData is omitted)
long detachFaces(long nSize, long* facelds, long* partIds, void* callbackData); where partIds is and output list and nSize is an input parameters and nSize is the size of partIds.	(lret, partIdList) = udpFunctionLibrary.DetachFaces(nSize, facelds) (nSize is not ignored, callbackData is omitted)

Added Parameters

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

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

Example:

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

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

Here cloneIDsSize is a new integer parameter.

Comparison with C function:

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

C	Python
<pre>UDPCoordinateSystemAxis axis, double angle, int nClones, long* nClones, void* callbackData);</pre>	<pre>(callbackData is omitted cloneldsSize is a new parameter)</pre>

Developing a UDM/UDP

Creation

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

Location

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

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

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

Organize

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

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

For UDM:extension:

```
from UDM import *
```

For UDP:extension:

```
from UDP import *
```

Edit/Reload

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

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

UDPExtension

Import

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

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

```
from UDP import *
```

Main class: UDPExtension

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

```
class UDPExtension(IUDExtension):
```

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

IUDExtension Methods

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

Mandatory Methods

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

GetLengthParameterUnits()

- returns string.

GetPrimitiveTypeInfo()

- returns UDPPrimitiveTypeInfo.

GetPrimitiveParametersDefinition2()

- returns a list of UDPPrimitiveParameterDefinition2 or None on failure

AreParameterValuesValid2(errorMsg, udpParams)

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

CreatePrimitive2(funcLib, udpParams)

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

Optional Methods

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

GetPrimitiveParameters()

- returns Python list of strings or NULL

GetRegisteredFaceNames()

- returns Python list of strings or NULL

GetRegisteredEdgeNames()

- returns Python list of strings or NULL

GetRegisteredVertexNames()

- returns Python list of strings or NULL

ProjectParametersOnToValidPlane2(currentUDPParams, projectedUDPParams)

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

MapParametersDefinitionVersions2(oldVersion, oldUDPParams)

- oldVersion is a string
- oldUDPParams is a list of UDPParam

- returns Python list of UDPParam or NULL

GetOldPrimitiveParametersDefinition2(version)

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

Example UDP

```
import sys

class UDPExtension(IUDPExtension):

    def GetLengthParameterUnits(self):
        return "mm"

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

        return typeInfo

    ...

    ...
```

UDMExtension

Import

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

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

```
from UDM import *
```

Main class: UDMExtension

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

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

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

IUDMExtension Methods

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

Mandatory Methods

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

GetInfo()

- returns UDMInfo object populated with appropriate UDM information.

IsAttachedToExternalEditor()

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

CreateInstance(funcLib)

- funcLib is UDMFunctionLibrary
- returns UDMParameters.

GetUnits(instanceId)

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

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

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

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

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

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

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

returns True on success, False on failure.

ReleaseInstance(instanceId)

- `instanceId` is an integer.
- This should release any resources assigned to this particular instance of UDM.
- returns True on success, False on failure.

GetAttribNameForEntityId()

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

GetAttribNameForPartId()

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

Optional Methods

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

DialogForDefinitionOptionsAndParams(self, defData, optData, params):

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

- UDM can open a dialog box for UDM definition, options, parameters in this method. Definition, options, and parameters are set/modified by user and returned to application. DII can

also just give default definition, options and parameters.

- Returns two Booleans and a string
 - First Boolean returns whether the method was successful or not.
 - Second Boolean returns whether the application should open a dialog box. If it is True, application will populate a dialog box with definition, options, parameters that are returned.
 - String returned contains length units for parameters.

DialogForDefinitionAndOptions(self, defData, optData) [Deprecated]

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

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

GetInstanceSourceInfo(instanceId)

- instanceId is an integer.
- returns string containing source information of UDM instance. It is used to create initial name for UDM instance.

ShouldAttachDefinitionFilesToProject()

- returns True if any of definition files needs to be attached to project
- returns a Python list of strings containing definition names of files or NULL

Example UDM

```
class UDMExtension(IUDMExtension):

    def IsAttachedToExternalEditor(self):
        return False

    def GetInfo(self)
        udmInfo = UDMInfo(
            name = "SampleUDM",
            purpose = "udm example",
```

```
    company="Ansys",
    date="12.21.12",
    version = "1.0")

    return udmInfo

...

...
```

UDMFunctionLibrary

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

- CreateInstance
- Refresh

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

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

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

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

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

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

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

Functions list:

1. **List_of_UDMDefinition:** udmDefinitionList = **GetDefinition()**
2. **List_of_UDMOption:** udmOptionList = **GetOptions()**
3. **bool:** bret = **SetMaterialName(string: matName, int: partId)**
4. **bool:** bret = **SetMaterialName2(string: matName, string: partName)**

5. **bool:** bret = **SetPartName**(*string:* partName, *int:* partId)
6. **int:** iret = **GetInstanceId**()
7. **string:** str = **GetPartRefId**(*int:* partId)
8. **bool:** bret = **SetPartRefId**(*int:* partId, *string:* refId)
9. **List_of_int:** faceIds = **GetAllFaces**(*int:* partId)
10. **List_of_int:** edgeIds = **GetAllEdges**(*int:* partId)
11. **List_of_int:** vertexIds = **GetAllVertices**(*int:* partId)
12. **bool:** bret = **SetFaceAttribs**(*List_of_int:* faceIds, *List_of_string:* attribs)
13. **bool:** bret = **SetEdgeAttribs**(*List_of_int:* edgeIds, *List_of_string:* attribs)
14. **bool:** bret = **SetVertexAttribs**(*List_of_int:* vertexIds, *List_of_string:* attribs)
15. **string:** str = **GetModelerUnit**()
16. **string:** str = **GetCacheFileForUDMResume**()
17. **bool:** bret = **SetPartColor**(*int:* partId, *int:* nColor)
18. **bool:** bret = **SetPartFlags**(*int:* partId, *int:* nFlags)
19. (**bool:** bret, **UDPPosition:** low, **UDPPosition:** high) = **GetBoundingBox**(*int:* partId, **bool:** exact)
20. **bool:** bret = **IsParametricUpdate**()
21. **bool:** bret = **SetMaterialNameByRefId**(*string:* partRefID, *string:* matName)
22. **bool:** bret = **SetPartNameByRefId**(*string:* partRefId, *string:* partName)
23. **bool:** bret = **SetPartColorByRefId**(*string:* partRefId, *int:* nColor)
24. **bool:** bret = **SetPartFlagsByRefId**(*string:* partRefId, *int:* nFlags)

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

Example:

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

UDM/UDP Functions

Return Values for Each UDM and UDP Function

ID – ID of created Object

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

Functions list:

1. **bool:** SI = **AddMessage**(**MessageSeverity:** messageSeverity, **string:** message)
2. **bool:** SI = **NameAFace**(**UDPPosition:** pointOnFace, **string:** faceName)
3. **bool:** SI = **NameAEdge**(**UDPPosition:** pointOnEdge, **string:** edgeName)
4. **bool:** SI = **NameAVertex**(**UDPPosition:** pointOnVertex, **string:** vertexName)

5. *int*: ID = **GetFaceIDFromPosition**(*UDPPosition*: pointOnFace)
6. *int*: ID = **GetEdgeIDFromPosition**(*UDPPosition*: pointOnEdge)
7. *int*: ID = **CreatePolyline**(*UDPPolylineDefinition*: polylineDefinition)
8. *int*: ID = **CreateRectangle**(*CoordinateSystemPlane*: whichPlane, *UDPPosition*: center-Point, *List_of_double*: widthAndHeight, *int*: isCovered)
9. *int*: ID = **CreateArc**(*CoordinateSystemPlane*: whichPlane, *UDPPosition*: centerPoint, *UDPPosition*: startPoint, *double*: fAngle)
10. *int*: ID = **CreateCircle**(*CoordinateSystemPlane*: whichPlane, *UDPPosition*: center-Point, *double*: fRadius, *int*: isCovered)
11. *int*: ID = **CreateEllipse**(*CoordinateSystemPlane*: whichPlane, *UDPPosition*: center-Point, *double*: fMajorRadius, *double*: fRadiusRatio, *int*: isCovered)
12. *int*: ID = **CreateRegularPolygon**(*CoordinateSystemPlane*: whichPlane, *UDPPosition*: centerPoint, *UDPPosition*: startPoint, *int*: numOfSides, *int*: isCovered)
13. *int*: ID = **CreateEquationBasedCurve**(*UDPEquationBasedCurveDefinition*: curveDefinition)
14. *int*: ID = **CreateEquationBasedSurface**(*UDPEquationBasedSurfaceDefinition*: surfaceDefinition)
15. *int*: ID = **CreateSpiral**(*UDPSpiralDefinition*: spiralDefinition)
16. *int*: ID = **CreateBox**(*UDPPosition*: startPoint, *List_of_double*: boxXYZsize)
17. *int*: ID = **CreateSphere**(*UDPPosition*: centerPoint, *double*: fRadius)
18. *int*: ID = **CreateCylinder**(*CoordinateSystemAxis*: whichAxis, *UDPPosition*: center-Point, *double*: fRadius, *double*: fHeight)
19. *int*: ID = **CreateCone**(*CoordinateSystemAxis*: whichAxis, *UDPPosition*: centerPoint, *double*: fBottomRadius, *double*: fTopRadius, *double*: fHeight)
20. *int*: ID = **CreateTorus**(*CoordinateSystemAxis*: whichAxis, *UDPPosition*: centerPoint, *double*: fMajorRadius, *double*: fMinorRadius)
21. *int*: ID = **CreatePolyhedron**(*CoordinateSystemAxis*: whichAxis, *UDPPosition*: center-Point, *UDPPosition*: startPosition, *int*: numOfSides, *double*: fHeight)
22. *int*: ID = **CreateHelix**(*UDPHelixDefinition*: helixDefinition)
23. *bool*: SI = **Unite**(*List_of_int*: pObjectIDArray)
24. *bool*: SI = **Subtract**(*List_of_int*: pBlankObjectIDArray, *List_of_int*: pToolObjectIDArray)
25. *bool*: SI = **Intersect**(*List_of_int*: pObjectIDArray)
26. *bool*: SI = **Imprint**(*List_of_int*: pBlankObjectIDArray, *List_of_int*: pToolObjectIDArray)
27. *bool*: SI = **SweepAlongVector**(*int*: profileID, *UDPVector*: sweepVector, *UDPSweepOptions*: sweepOptions)
28. *bool*: SI = **SweepAroundAxis**(*int*: profileID, *CoordinateSystemAxis*: whichAxis, *double*: sweepAngle, *UDPSweepOptions*: sweepOptions)
29. *bool*: SI = **SweepAlongPath**(*int*: profileID, *int*: pathID, *UDPSweepOptions*: sweepOptions)

-
30. **bool**: SI = **Translate**(*int*: partID, **UDPVector**: translateVector)
 31. **bool**: SI = **Rotate**(*int*: partID, **CoordinateSystemAxis**: whichAxis, **double**: rotateAngle)
 32. **bool**: SI = **Mirror**(*int*: partID, **UDPPosition**: mirrorPlaneBasePosition, **UDPVector**: mirrorPlaneNormalVector)
 33. **bool**: SI = **Transform**(*int*: partID, **List_of_double**: rotationMatrix, **UDPVector**: translateVector)
 34. **bool**: SI = **Scale**(*int*: partID, **double**: xScale, **double**: yScale, **double**: zScale)
 35. (**bool**: SI, **List_of_int**: cloneIDs) = **DuplicateAlongLine**(*int*: partID, **UDPVector**: translateVector, *int*: numTotalObjs, *int*: cloneIDsListSize)
 36. (**bool**: SI, **List_of_int**: cloneIDs) = **DuplicateAroundAxis**(*int*: partID, **CoordinateSystemAxis**: whichAxis, **double**: rotateAngle, *int*: numTotalObjs, *int*: cloneIDsListSize)
 37. *int*: ID = **DuplicateAndMirror**(*int*: partID, **UDPPosition**: mirrorPlaneBasePosition, **UDPVector**: mirrorPlaneNormalVector)
 38. **bool**: SI = **Connect**(**List_of_int**: objectIDArray)
 39. **bool**: SI = **Offset**(*int*: partID, **double**: offsetDistance)
 40. *int*: ID? = **Section**(*int*: partID, **CoordinateSystemPlane**: sectionPlane)
 41. (**bool**: SI, *int*: ID) = **Split**(*int*: partID, **CoordinateSystemPlane**: splitPlane, **SplitWhichSideToKeep**: whichSideToKeep, **bool**: bSplitCrossingObjectsOnly)
 42. (**bool**: SI, **List_of_int**: importedObjectIDs) = **ImportNativeBody2**(**string**: fileNameWithFullPath)
 43. (**bool**: SI, **List_of_int**: importedObjectIDs) = **ImportAnsoftGeometry**(**string**: fileNameWithFullPath, **List_of_string**: overridingParamsNameArray, **List_of_UDPParam**: overridingParamsArray)
 44. *int*: ID = **Clone**(*int*: partID)
 45. **bool**: SI = **DeletePart**(*int*: partID)
 46. *int*: ID = **CreateObjectFromFace**(*int*: faceID)
 47. **bool**: SI = **Fillet**(**UDPBLNDElements**: entitiesToFillet, **UDPBLNDFilletOptions**: filletOptions)
 48. **bool**: SI = **Chamfer**(**UDPBLNDElements**: entitiesToChamfer, **UDPBLNDChamferOptions**: chamferOptions)
 49. (**bool**: SI, **List_of_int**: newPartIDs) = **DetachFaces**(*int*: newPartIDArraySize, **List_of_int**: faceIDs)
 50. (**bool**: SI, **List_of_int**: newPartIDs) = **DetachEdges**(*int*: newPartIDArraySize, **List_of_int**: edgeIDs)
 51. *int*: ID = **CreateObjectFromEdge**(*int*: edgeID)
 52. **bool**: SI = **SheetThicken**(*int*: partID, **double**: fThickness, **bool**: bThickenBothSides)
 53. (**bool**: SI, **List_of_int**: newPartIDArray) = **SweepFaceAlongNormal**(*int*: newPartIDArraySize, **List_of_int**: faceIDArray, **double**: sweepLength)
-

54. **bool**: SI = **CoverLine**(*int*: partID)
55. **bool**: SI = **CoverSurface**(*int*: partID)
56. **bool**:SI= **UncoverFaces**(*List_of_int*: faceIDArray)
57. (**bool**: SI , *int*: numPartsCreated, *List_of_int*: faceIDArray) = **SeparateBodies**(*int*: partID, *int*: numPartsCreated)
58. **bool**: SI = **MoveFaces**(*List_of_int*: faceIDArray, **bool**: bMoveAlongNormal, **double**: fOffsetDistance, **UDPVector**: moveVector)
59. **bool**: SI = **WrapSheet**(*int*: sheetBodyID, *int*: targetBodyID)
60. **bool**: SI = **ImprintProjection**(*int*: blankBodyID, *List_of_int*: toolBodyIDArray, **bool**: bNormalProjection, **UDPVector**: projectDirection, **double**: projectDistance)
61. **string**: path = **GetTempDirPath**()
62. **string**: path = **GetSysLibDirPath**()
63. **string**: path = **GetUserLibDirPath**()
64. **string**: path = **GetPersonalLibDirPath**()
65. **string**: path = **GetInstallDirPath**()
66. **string**: path = **GetProjectPath**()
67. (**bool**: SI, **bool**: abort) = **SetProgress**(**UDPPProgress**: progress)

UDP/UDM Structures and Constants

The following sections describe:

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

UDP/UDM Structures

Differences compared to C API:

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

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

Example (creating UDMPParameter list):

```
udmParamList = [  
    UDMPParameter(  
        "cubeSizeName", UnitType.LengthUnit,
```

```

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

```

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

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

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

Example:

```

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

```

```

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

```

- **UDP3x3Matrix**

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

Example:

```

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

```

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

List of Structures

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

Example:

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

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

Structure	Construction	Members
	methods: setInt(int val) setBool(bool val) setString(string val) setDouble(double val) setPosition(UDPPosition val)	
UDPPrim- itiveParameterDefinition2	UDPPrim- itiveParameterDefinition2(string name, string description, UnitType unitType, ParamPropType propType, ParamPropFlag propFlag, UDPParam defaultValue)	string Name string Description UnitType UnitType ParamPropType PropType ParamPropFlag PropFlag UDPParam DefaultValue
UDPPosition	UDPPosition(double x, double y, double z)	double X double Y double Z
UDPVector	UDPVector(double x, double y, double z)	double X double Y double Z
UDPSweepOptions	UDPSweepOptions(SweepDraftType draftType, double draftAngle, double twistAngle)	SweepDraftType DraftType double DraftAngle double TwistAngle
UDPPolylineSeg- mentDefinition	UDPPolylineSeg- mentDefinition(PolylineSegmentType Seg- mentType	PolylineSegmentType Seg- mentType

Structure	Construction	Members
	PolylineSegmentType segmentType, int segmentStartIndex, int numberOfPoints, double angle, UDPPosition centerPoint, CoordinateSystemPlane arcPlane)	int segmentStartIndex, int numberOfPoints, double angle, UDPPosition centerPoint, CoordinateSystemPlane arcPlane)
UDPPolylineDefinition	UDPPolylineDefinition() UDPPolylineDefinition(List_of_UDPPosition positions, List_of_UDPPolylineSegmentDefinition segDefs, int closed, int covered)	int IsClosed int IsCovered List_of_UDPPosition ArrayOfPosition List_of_UDPPolylineSegmentDefinition ArrayOfSegmentDefinition
UDPEquationBasedCurveDefinition	UDPEquationBasedCurveDefinition(string functionXt, string functionYt, string functionZt, double tStart, double tEnd, int numOfPointsOnCurve)	string FunctionXt string FunctionYt string FunctionZt double TStart double TEnd int NumOfPointsOnCurve
UDPEquationBasedSurfaceDefinition	UDPEquationBasedSurfaceDefinition(string functionXuv, string functionYuv, string functionZuv, double uStart, double uEnd, double vStart, double vEnd)	string FunctionXuv string FunctionYuv string FunctionZuv double UStart double UEnd double VStart double VEnd

Structure	Construction	Members
	<pre>double vEnd int reserved1 int reserved2)</pre>	<p>two integer arguments that are reserved for future use. They need to be provided, for example as 0. For example:</p> <pre>theSurfaceDefinition = UDPEqua- tionBasedSur- faceDefinition ("u", "v", "1", 0, 1, 0, 1, 0, 0)</pre>
UDPHelixDefinition	<pre>UDPHelixDefinition(int profileID, UDPPosition ptOnAxis, UDPPosition axisDir, double noOfTurns, bool isRightHanded, double radi- usChangePerTurn, double pitch)</pre>	<pre>int ProfileID UDPPosition PtOnAxis UDPPosition AxisDir double NoOfTurns bool IsRightHanded double RadiusChangePerTurn double Pitch</pre>
UDPSpiralDefinition	<pre>UDPSpiralDefinition(int profileID, UDPPosition ptOnAxis, UDPPosition axisDir, double noOfTurns, bool isRightHanded, double radi- usChangePerTurn)</pre>	<pre>int ProfileID UDPPosition PtOnAxis UDPPosition AxisDir double NoOfTurns bool IsRightHanded double RadiusChangePerTurn</pre>
UDPBLNDElements	<pre>UDPBLNDElements(int partID, int noOfEdges; int* listOfEdges;) UDPBLNDElements(int partID,</pre>	<p>UDPBLNDElements can hold either edges or vertices, but not both at the same time. Edges should be applied to solids, and vertices should be applied to sheets.</p> <pre>int PartID /* part to be blended i.e. filleted/chamfered */ int noOfEdges;</pre>

Structure	Construction	Members
	<pre>int noOfVertices; int* listOfVertices;)</pre>	<pre>int* listOfEdges; /* edges to be blended */ int noOfVertices; int* listOfVertices; /* vertices to be blended */</pre>
UDPBLNDFilletOptions	<pre>UDPBLNDFilletOptions(bool supressFillet, BLNDFilletRadiusLaw fil- letRadiusLaw, double filletStartRadius, double filletEndRadius, bool fol- lowSmoothEdgeSequence, BLNDFilletType filletType, double setbackDistance, double bulgeFactor)</pre>	<pre>bool SupressFillet /* Reserved for future */ BLNDFilletRadiusLaw Fil- letRadiusLaw double FilletStartRadius double FilletEndRadius bool Fol- lowSmoothEdgeSequence /* Reserved for future */ BLNDFilletType FilletType double SetbackDistance double BulgeFactor /* Reserved for future */</pre>
UDPBLNDChamferOptions	<pre>UDPBLNDChamferOptions(bool supressChamfer, BLNDChamferRangeLaw chamferRangeLaw, double chamferLeftRange, double chamferRightRange)</pre>	<pre>bool SupressChamfer BLNDChamferRangeLaw Cham- ferRangeLaw double ChamferLeftRange double ChamferRightRange</pre>
UDPPProgress	<pre>UDPPProgress(int prog, int subProg, string mesg, string subMesg)</pre>	<pre>int Prog int SubProg string Mesg string SubMesg</pre>
UDMInfo	<pre>UDMInfo(string name, string purpose,</pre>	<pre>string Name string Purpose string Company</pre>

Structure	Construction	Members
	string company, string date, string version)	string Date string Version
UDMDefinition	UDMDefinition() UDMDefinition(string name, UDParam value, ParamPropType propType, ParamPropFlag propFlag)	string DefName UDPParam DefValue ParamPropType PropType ParamPropFlag PropFlag
UDMOption	UDMOption() UDMOption(string name, UDParam value, ParamPropType propType, ParamPropFlag propFlag)	string OptName UDPParam OptValue ParamPropType PropType ParamPropFlag PropFlag
UDMParameter	UDMParameter() UDMParameter(string name, UDParam value, UnitType unitType, ParamPropType propType, ParamPropFlag propFlag)	string ParamName UDPParam ParamValue UnitType UnitType ParamPropType PropType ParamPropFlag PropFlag

UDP/UDM Constants

Full names of enum constants must be used in scripts.

Example:

```
unitType = UnitType.LengthUnit
```

```
dataType = ParamDataType.Int
```

Enum constants:

enum Constant	Parameters
UnitType	NoUnit LengthUnit AngleUnit
ParamDataType	Int Double String Bool Position Unknown
ParamPropType	Text Menu Number Value FileName Checkbox Position Unknown
ParamPropFlag	NoFlag ReadOnly MustBeInt MustBeReal Hidden Unknown
CoordinateSystemAxis	XAxis YAxis ZAxis
CoordinateSystemPlane	XYPlane YZPlane

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

Introduction to CPython

CPython can be used to:

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

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

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

Creating an External Script

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

```
import sys

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

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

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

Those lines are followed by:

```
import ScriptEnv

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

Follow that with:

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

Start ansyedt as GRPC server

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

-grpcsrv Flag

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

`ansyedt -grpcsrv`

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

`ansyedt -grpcsrv portNumber`

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

`ansyedt -grpcsrv FirstPortNumber:LastPortNumber`

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

`ansyedt -grpcsrv FirstPortNumber:NumberOfPorts`

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

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

Connect Functions:

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

Example:

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

Launching Electronics Desktop

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

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

Where:

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

Note:

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

Connecting with a Running Instance of Electronics Desktop

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

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

Where:

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

Closing Electronics Desktop/Ending the Script

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

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

-grpcsrv Flag

This flag will launch the application in a mode where the executable serves as a scripting server that can be used for CPython scripting in conjunction with the CPython stand alone scripting

instructions that were mentioned earlier. The -ng flag can be combined with -grpcsrv.

On Windows:

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

On Linux:

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

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

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

Related Topics:

[Standalone Scripting in Iron Python](#)

Ansys Electronics Desktop Scripting

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

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

[Running a Script](#)

[Recording a Script](#)

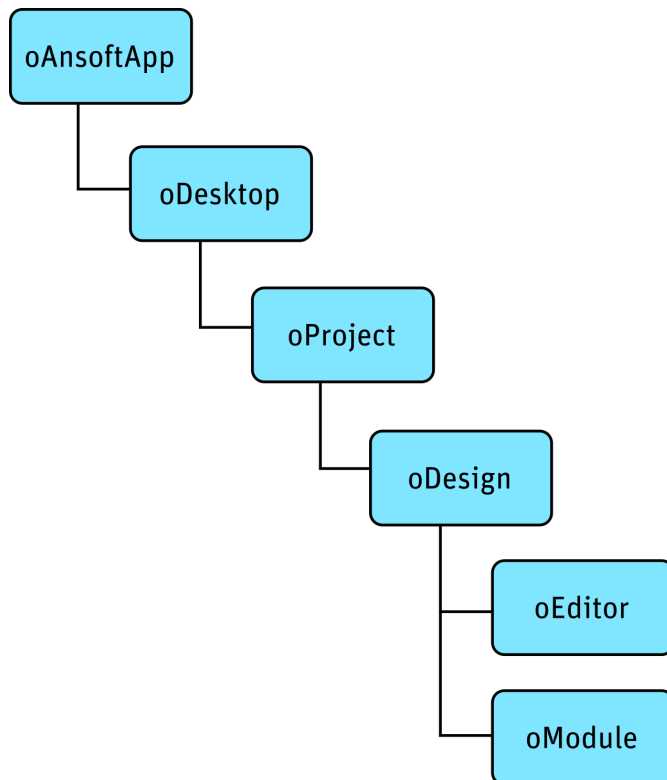
[Working with Project Scripts](#)

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

[Ansys Electronics Desktop Scripting Conventions](#)

Overview of Electronics Desktop Scripting Objects

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



The commands are described below, followed by examples.

oAnsoftApp

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

In Iron Python, for example:

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

oDesktop

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

In Iron Python, for example:

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

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

- [Desktop Object Script Commands](#).

oProject

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

In Iron Python, for example:

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

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

- [Project Object Script Commands](#)

oDesign

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

In Iron Python, for example:

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

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

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

oEditor

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

In Iron Python, for example:

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

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

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

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

Important:

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

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

oModule

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

In IronPython, for example:

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

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

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

Example Script Opening

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

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

GetActiveProject and GetActiveDesign for Wider Use

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

In IronPython:

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

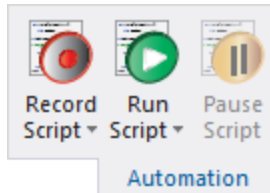
Running a Script

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

Within Electronics Desktop

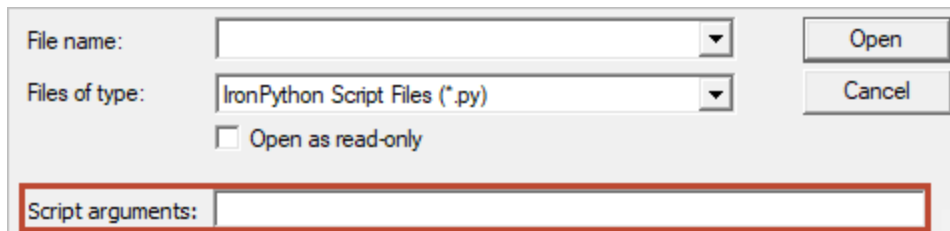
To run scripts in Electronics Desktop:

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



The **Run Script** file browser appears.

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



4. Click **Open**.

Electronics Desktop executes the script.

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

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

From the Command Line

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

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

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

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

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

Recording a Script

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

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

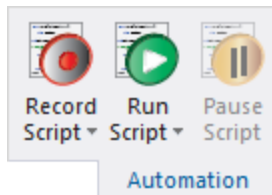
Important:

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

Recording a Script to File

To record a script to file:

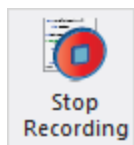
1. Click **Tools > Record Script to File**, or select the **Automation** tab and click the **Record Script** icon:



A **Save As** file browser appears.

2. Navigate to the location where you want to save the script.
3. In the **File Name** field, type a name for the script file.
4. Click **Save**.

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



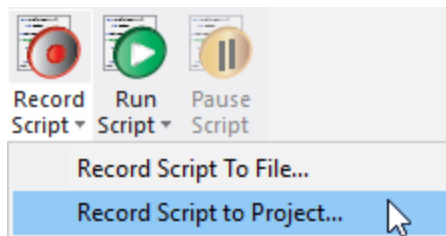
5. Perform the steps you want to record.
6. When you have finished recording the script, click **Stop Recording**, or select **Tools > Stop Script Recording**.

The recorded script is saved to the folder you specified.

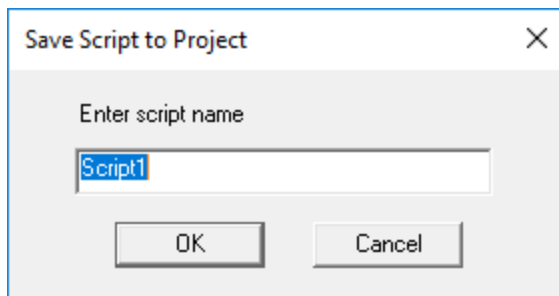
Recording a Script to a Project

To record a script to a project:

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



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



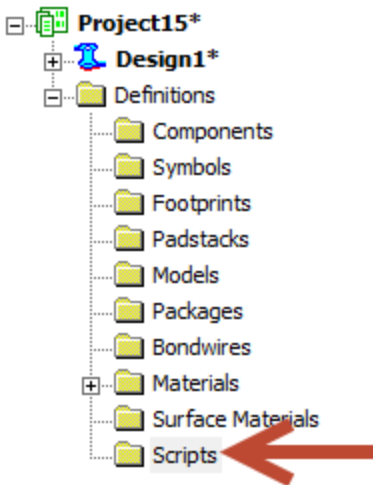
2. Enter a name for the script in the text box, then click **OK**.
3. Perform the steps you want to record.
4. When you have finished, click **Stop Recording**, or select **Tools > Stop Script Recording**.

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

Working with Project Scripts

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

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



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



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

Note:

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

Executing a Script from Within a Script

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

In CPython:

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

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

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

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

Electronics Desktop Scripting Conventions

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

Consult the following topics:

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

Named Arguments

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

1. Named data, where name precedes data.

For example:

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

2. Named Array, where name precedes array.

For example:

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

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

For example:

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

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

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

not want to change. Electronics Desktop will use the names to determine which data you provided.

IronPython Example

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

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

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

```
oModule.Edit("Port1",
    ["NAME:Port1",
```

```
    ["Magnitude:=", "1mA"]  
  ]  
)
```

Setting Numerical Values

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

- Pass in the number directly.

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

- Pass in a string containing the number with units.

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

- Pass in a variable name.

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

This page intentionally
left blank.

2 - Application Object Script Commands

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

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

General Application Script Commands

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

- [GetAppDesktop](#)

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

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

GetAppDesktop

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

UI Access	NA		
Parameters	Name	Type	Description
	None		
Return Value	Object		

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

3 - Desktop Object Script Commands

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

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

[AddMessage](#)

[AreThereSimulationsRunning](#)

[ClearMessages](#)

[CloseAllWindows](#)

[CloseProject](#)

[CloseProjectNoForce](#)

[Count](#)

[DeleteProject](#)

[DeleteRegistryEntry](#)

[DoesRegistryValueExist](#)

[DownloadJobResults](#)

[EnableAutoSave](#)

[ExportOptionsFiles](#)

[GetActiveProject](#)

[GetActiveScheduler](#)

[GetActiveSchedulerInfo](#)

[GetAutoSaveEnabled](#)

[GetBuildDateTimeString](#)

[GetCustomMenuSet](#)

[GetDefaultUnit](#)

[GetDesktopConfiguration](#)

[GetDistributedAnalysisMachines](#)

[GetDistributedAnalysisMachinesForDesignType](#)

[GetExeDir](#)

[GetGDIObjectCount](#)

[GetLibraryDirectory](#)

[GetLocalizationHelper](#)

[GetMessages](#)

[GetMonitorData](#)

[GetPersonalLibDirectory](#)

[GetProcessID](#)

[GetProjectDirectory](#)

[GetProjectList](#)

[GetProjects](#)

[GetRegistryInt](#)

[GetRegistryString](#)

[GetRunningInstancesMgr](#)

[GetSchematicEnvironment](#)

[GetScriptingToolsHelper](#)

[GetSysLibDirectory](#)

[GetTempDirectory](#)

[GetUserLibDirectory](#)

[GetVersion](#)

[IsFeatureEnabled](#)

[KeepDesktopResponsive](#)

[LaunchJobMonitor](#)

[NewProject](#)

[OpenAndConvertProject](#)

[OpenMultipleProjects](#)

[OpenProject](#)

[OpenProjectWithConversion](#)

[PageSetup](#)

[PauseRecording](#)

[PauseScript](#)

[Print](#)

[QuitApplication](#)

[RefreshJobMonitor](#)

[ResetLogging](#)

[RestoreProjectArchive](#)

[RestoreWindow](#)

[ResumeRecording](#)

[RunACTWizardScript](#)

[RunProgram](#)

[RunScript](#)

[RunScriptWithArguments](#)

[SelectScheduler](#)

[SetActiveProject](#)

[SetActiveProjectByPath](#)

[SetCustomMenuSet](#)

[SetDesktopConfiguration](#)

[SetLibraryDirectory](#)

[SetProjectDirectory](#)

[SetRegistryFromFile](#)

[SetRegistryInt](#)

[SetRegistryString](#)

[SetSchematicEnvironment](#)

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

AddMessage

Add a message with severity and context to message window.

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

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

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

ClearMessages

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

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

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

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

	<startSeverity>	Integer	User-specified severity level at which messages start getting cleared: <ul style="list-style-type: none"> • 0 = informational • 1 = warning • 2 = error • 3 = fatal error
	<stopSeverity>	Integer	<i>Optional.</i> User-specified stop severity level until which messages will be cleared. <ul style="list-style-type: none"> • 0 = informational • 1 = warning • 2 = error • 3 = fatal error If not specified, <stopSeverity> has a default value of 0.
Return Value	None.		

Python Syntax	<code>ClearMessages(<projectName>, <designName>, <startSeverity>, <stopSeverity>)</code>
Python Example	<p>Preserve Current Functionality; stopSeverity = 0 (DEFAULT VALUE)</p> <pre># startSeverity = 0; clears just the Info messages oDesktop.ClearMessages("Ex_EC_1_Asymmetrical_Conductor", "Maxwell3DDesign1", 0) # startSeverity = 1; clears all the Info and Warning messages oDesktop.ClearMessages("Ex_EC_1_Asymmetrical_Conductor", "Maxwell3DDesign1", 1) # startSeverity = 2; clears all the Info, Warning, Error oDesktop.ClearMessages("Ex_EC_1_Asymmetrical_Conductor", "Maxwell3DDesign1", 2)</pre>

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

Extend the current functionality by enabling range-based deletion

# startSeverity = 0; clears the Info messages;
oDesktop.ClearMessages("Ex_EC_1_Asymmetrical_Conductor", "Maxwell3DDesign1", 0)
# startSeverity = 0 and stopSeverity = 0; clears all the Info messages
oDesktop.ClearMessages("Ex_EC_1_Asymmetrical_Conductor", "Maxwell3DDesign1", 0,
0)
# startSeverity = 0 and stopSeverity = 1; clears all the Info and Warning mes-
sages
oDesktop.ClearMessages("Ex_EC_1_Asymmetrical_Conductor", "Maxwell3DDesign1", 0,
1)
# startSeverity = 0 and stopSeverity = 2; clears all the Info, Warning, and Error
messages
oDesktop.ClearMessages("Ex_EC_1_Asymmetrical_Conductor", "Maxwell3DDesign1", 0,
2)
# startSeverity = 0 and stopSeverity = 3; clears all the Info, Warning, Error,
and Fatal messages
oDesktop.ClearMessages("Ex_EC_1_Asymmetrical_Conductor", "Maxwell3DDesign1", 0,
3)

# startSeverity = 1; clears all the Info and Warning messages;
oDesktop.ClearMessages("Ex_EC_1_Asymmetrical_Conductor", "Maxwell3DDesign1", 1)
```

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

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

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

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

CloseAllWindows

Closes all MDI child windows on the desktop.

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

Python Syntax	CloseAllWindows()
Python Example	oDesktop.CloseAllWindows()

CloseProject

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

UI Access	File > Close
------------------	------------------------

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

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

CloseProjectNoForce

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

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

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

Count

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

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

DeleteProject

Deletes a project from disk.

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

Python Syntax	DeleteProject (<ProjectName>)
Python Example	

```
oDesktop.DeleteProject("MyProject")
```

DownloadJobResults

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

UI Access	Select Scheduler		
Parameters	Name	Type	Description
	<jobID>	String	Provide the job ID of the target job. The job ID must be able to be found in current .q (or .q.completed) file, or inside the “batchinfo” folder in projectPath. If the job ID is empty, the job ID in current .q (or .q.completed) file will be used.
	<projectPath>	String	A string of path to locate the project folder. The project file may be not necessary, but .q (or .q.completed) file or “batchinfo” folder with valid .jobid files are required.
	<resultPath>	String	A string giving the folder path for the download to save to.
	<Filters> (optional)	String	A string containing filters to download. The delimiter of file types is “;”. If no filter specified, the default filter “*” will be applied, which requests all files for download.
Return Value	A Boolean result about download complete or not		

Python Syntax	<code>DownloadJobResults(jobID, projectPath, resultsPath, filters)</code>
Python Example	<code>boolDownloadCompleted = oDesktop.DownloadJobResults("012345678901234567890",</code>

	"C:\\projects\\basic.aedt", "C:\\projects\\DownloadResults\\", "*")
--	---

EnableAutoSave

Enable or disable autosave feature.

UI Access	N/A		
Parameters	Name	Type	Description
	<enable>	Boolean	True to enable autosave; False disables it.
Return Value	None.		

Python Syntax	EnableAutoSave(<enable>)
Python Example	oDesktop.EnableAutoSave(True)

ExportOptionsFiles

Copies the options config files to the *DestinationDirectory*.

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

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

Note:

GetActiveProject returns normally if there are no active objects.

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

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

GetAutoSaveEnabled

Checks whether the autosave feature is enabled.

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

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

GetBuildDateTimeString

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

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

Python Syntax	GetBuildDateTimeString()
Python Example	oDesktop.GetBuildDateTimeString()

GetDefaultUnit

Returns the default unit for a physical quantity.

UI Access	Tools > Options > General Options > Default Units. Note that this menu only displays units that can be changed, while the script can be used to view additional default units.		
Parameters	Name	Type	Description
	<type>	String	String containing a type of measurement. Valid strings are (case insensitive): <ul style="list-style-type: none"> • "Acceleration" • "Angle" • "AngularAcceleration" • "AngularDamping" • "AngularSpeed" • "Capacitance" • "Conductance" • "Current" • "CurrentChangeRate" • "DataRate" • "DeltaH" (Magnetic Field Strength)

			<ul style="list-style-type: none">• "Density"• "Flux"• "Force"• "Frequency"• "Inductance"• "Length"• "MagneticReluctance"• "Mass"• "MassFlowRate"• "MomentInertia"• "Power"• "Pressure"• "PressureCoefficient"• "Resistance"• "SaturateMagnetization" (Magnetic Inductance)• "Speed"• "Temperature"• "Time"• "Torque"• "Voltage"• "VoltageChangeRate"• "Volume"• "VolumeFlowPerPressureRoot"• "VolumeFlowRate"
--	--	--	---

Return Value	String containing the default unit (for example, "mm").
---------------------	---

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

GetDesigns

Obtains all designs in the current project.

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

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

GetDistributedAnalysisMachines

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

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

Parameters	None.
Return Value	Returns a collection of names of machines used for distributed analysis.

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

GetDistributedAnalysisMachinesForDesignType

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

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

Python Syntax	<code>GetDistributedAnalysisMachinesForDesignType (<designTypeName>)</code>
Python Example	<code>machineNames =oDesktop. GetDistributedAnalysisMachinesForDesignType</code>

	("")
--	------

GetExeDir

Returns the path where the executable is located.

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

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

GetGDIObjectCount

Note:

This command is for internal Ansys use only.

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

GetLibraryDirectory

Get the path to the `SysLib` directory.

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

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

GetLocalizationHelper

Note:

This command is for internal Ansys use only.

Returns the object for the localization helper.

UI Access	NA		
Parameters	Name	Type	Description
	None		

Return Value	Object Localization helper object, such as "IDispatch(ILocalizationHelper)"
---------------------	--

Python Syntax	GetLocalizationHelper()
Python Example	oDesktop.GetLocalizationHelper()

GetMessages

Get the messages from a specified project and design.

UI Access	NA		
Parameters	Name	Type	Description
	<ProjectName>	String	Name of the project for which to collect messages. An incorrect project name results in no messages (design is ignored). An empty project name results in all messages (design is ignored)
	<DesignName>	String	Name of the design in the named project for which to collect messages. An incorrect design name results in no messages for the named project. An empty design name results in all messages for the named project
	<Severity>	Integer	Severity is 0-3, and is tied in to info/warning/error/fatal types as follows: <ul style="list-style-type: none"> • 0 – info and above • 1 – warning and above • 2 – error and fatal • 3 – fatal only (rarely used)
Return Value	Array of string messages.		

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

GetName [Desktop]

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

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

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

GetPersonalLibDirectory

Get the path to the `PersonalLib` directory.

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

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

GetProcessID

Returns the process ID of `ansysedt.exe`.

UI Access	N/A
Parameters	None.
Return Value	Integer process ID of <code>ansysedt.exe</code> . For example, 12716.

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

GetProjects

Returns a list of all the projects that are currently open in Electronics Desktop. Once you have the projects, you can iterate through them using standard scripting methods.

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

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

GetProjectDirectory

Gets the path to the Project directory.

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

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

GetProjectList

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

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

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

GetScriptingToolsHelper

Note:

This command is for internal Ansys use only.

Returns the object for the scripting tools helper.

UI Access	NA		
Parameters	Name	Type	Description
	None		
Return Value	Object ScriptingTools helper object		

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

GetSysLibDirectory

Get the path to the `SysLib` directory.

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

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

GetTempDirectory

Gets the path to the Temp directory.

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

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

GetUserLibDirectory

Gets the path to the UserLib directory.

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

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

--	--

GetVersion

Returns a string representing the version.

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

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

ImportANF

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

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

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

ImportAutoCAD

Imports an AutoCAD file into a new project.

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

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

ImportGDSII

Imports a GDSII file into a new project.

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

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

ImportODB

Imports an ODB++ file into a new project.

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

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

LaunchJobMonitor

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

UI Access	Launch Job Monitor		
Parameters	Name	Type	Description
	<projectPath>	String	Path to the project file to be monitored

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

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

NewProject

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

UI Access	File > New.
Parameters	None.
Return Value	Object, the project that is added.

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

OpenAndConvertProject

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

UI Access	Click File > Open , and choose a legacy project		
Parameters	Name	Type	Description
	<itemPath>	String	full project path of the legacy project
	<legacyChoice>	Integer	0: show conversion dialog box, (same as File > Open of a legacy file) 1: rename (changes extension to .aedt, the original file and results are renamed) 2: copy (creates new file with .aedt extension, and the original file and results remain available)
Return Value	An object reference to the newly opened project which has the .aedt extension		

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

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

OpenMultipleProjects

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

UI Access	N/A		
Parameters	Name	Type	Description
	<directory>	String	Path to the projects.

	<i><fileType></i>	String	Type of projects to open.
Return Value	None.		

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

OpenProject

Opens a specified project.

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

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

OpenProjectWithConversion

Note:

This command is for internal Ansys use only.

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

PauseRecording

Temporarily stop script recording.

UI Access	NA		
Parameters	Name	Type	Description
	None		
Return Value	None		

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

PauseScript

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

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

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

Print

Prints the contents of the active view window.

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

Python Syntax	Print()
----------------------	---------

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

QuitApplication

Exits the desktop.

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

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

RefreshJobMonitor

For use in monitoring a job.

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

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

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

ResetLogging

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

UI Access	N/A		
Parameters	Name	Type	Description
	<logFile>	String	Path to log file.
	<logLevel>	Integer	Specified log level.
Return Value	None.		

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

RestoreProjectArchive

Restores a previously archived project to a specified path.

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

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

RestoreWindow

Restores a minimized Desktop window.

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

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

ResumeRecording

Resume recording a script.

UI Access	N/A
Parameters	None.
Return Value	None

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

RunACTWizardScript

Note:

This command is for internal Ansys use only.

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

RunProgram

Runs an external program.

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

Python Syntax	RunProgram (<ProgName>, <ProgPath>, <WorkPath>, <ArgArray>)
Python Example	<code>oDesktop.RunProgram("winword.exe", _</code>

	<pre>"C:\Program Files\Microsoft Office\Office10",_ "", None)</pre>
--	---

RunScript

Launches another script from within the script currently being executed.

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

			<p>can be specified in the General Options dialog box (click Tools > Options > General Options to open this dialog box) under the Project Options tab.</p> <p>4. HFSS installation directory.</p>
Return Value	<p>Long the return code for the script method.</p>		

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

RunScriptWithArguments

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

UI Access	NA		
Parameters	Name	Type	Description
	<code><ScriptPath></code>	String	<p>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:</p> <ul style="list-style-type: none"> Personal library directory: "PersonalLib" <p>The PersonalLib directory can be specified in Tools > Options > General Options on the 'Project Options' tab.</p> <ul style="list-style-type: none"> User library directory: "userlib"

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

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

SelectScheduler

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

UI Access	Tools > Job Management > Select Scheduler.		
Parameters	Name	Type	Description

	<i><option></i>	String	One of the following options (not case sensitive): <ul style="list-style-type: none"> • Empty string for remote RSM service • "RSM" for local RSM • "Windows HPC" for Windows HPC • "LSF" for Load-Sharing Facility • "SGE" for Grid Engine (GE, OGE, SGE, UGE, etc.) • "PBS" for Portable Batch Scheduler/System (PBSPro, Torque, Maui, etc.) • "Ansys Cloud" for Ansys Cloud
	<i><address (optional)></i>	String	String specifying the IP address or hostname of the head node or for the remote host running the RSM service.
	<i><username (optional)></i>	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.
	<i><forcePasswordEntry (optional)></i>	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></i> , <i><address></i> , <i><username></i> , <i><forcePasswordEntry></i>)
Python Example	<pre>result = oDesktop.SelectScheduler("Windows HPC", "headnode.win.example.com")</pre>

SetActiveProject

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

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

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

SetActiveProjectByPath

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

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

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

SetLibraryDirectory

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

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

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

SetProjectDirectory

Sets the project directory path.

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

Parameters	Name	Type	Description
	<DirectoryPath>	String	The path to the project directory. This should be writeable by the user.
Return Value	None.		

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

SetTempDirectory

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

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

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

ShowDockingWindow

Shows or hides a docking window.

UI Access	Right click docking window > Show/Hide.		
Parameters	Name	Type	Description
	<windowName>	String	The window name (for example, "Message Manager", "Component Libraries", "Properties")
	<show>	Boolean	True to show; False to hide.
Return Value	None.		

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

Sleep

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

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

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

SubmitJob

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

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

Python Syntax	<code>SubmitJob(<<i>settingsPath</i>>, <<i>projectPath</i>>, <<i>design</i>>, <<i>setup</i>>)</code>
Python Example	<code>jobIDs = oDesktop.SubmitJob("C:\\hpc-settings\\Submit_Job_Settings.aredt", "C:\\projects\\basic.aedt")</code>

```
moreIDs = oDesktop.SubmitJob("C:\\hpc-settings\\Submit_
Job_Settings.areg", "C:\\projects\\basic.aedt",
"Design1", "Setup1")
```

TileWindows

Arrange all open windows in a tiled format.

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

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

Desktop Commands For Registry Values

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

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

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

```
#import the setup

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

# Set Active Setup to "test"

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

See the following subtopics:

[DeleteRegistryEntry](#)

[DoesRegistryValueExist](#)

[GetRegistryInt](#)

[GetRegistryString](#)

[SetRegistryFromFile](#)

[SetRegistryInt](#)

[SetRegistryString](#)

DoesRegistryValueExist

Determines whether a registry value exists.

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

Parameters	Name	Type	Description
	<KeyName>	String	Full name of registry key, including path.
Return Value	Boolean: <ul style="list-style-type: none"> • True – Key exists. • False – Key does not exist. 		

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

GetRegistryInt

Obtains registry key integer value.

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

Python Syntax	GetRegistryInt(<KeyName>)
----------------------	---------------------------

Python Example	<pre>num = oDesktop.GetRegistryInt('Desktop/Set- tings/ProjectOptions//UpdateReportsDynamicallyOnEdits')</pre>
-----------------------	--

GetRegistryString

Obtains registry key string value.

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

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

SetRegistryFromFile

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

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

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

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

SetRegistryInt

Sets registry key to an integer value.

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

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

SetRegistryString

Sets registry key to a string value.

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

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

4 - Running Instances Manager Script Commands

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

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

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

GetRunningInstancesMgr

Returns the object of the Running Instances Manager.

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

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

This page intentionally
left blank.

5 - Project Object Script Commands

Project commands should be executed by the oProject object.

One example of accessing this object is:

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

Dataset Script Commands:

[AddDataset](#)

[DeleteDataset](#)

[EditDataset](#)

[ExportDataset](#)

[HasDataset](#)

[ImportDataset](#)

Other Project Object Script Commands:

[AnalyzeAll](#)

ChangeProperty

[ClearMessages](#)

CloneMaterial

[Close](#)

[CopyDesign](#)

[CutDesign](#)

[DeleteDesign](#)

[DeleteToolObject](#)

[ExportMaterial](#)

[GetActiveDesign](#)

[GetArrayVariables](#)

[GetChildNames \[Project\]](#)

[GetChildObject \[Project\]](#)

[GetChildTypes \[Project\]](#)

[GetDefinitionManager](#)

[GetDependentFiles](#)

[GetDesign](#)

[GetDesigns](#)

[GetEDBHandle](#)

[GetLegacyName](#)

[GetName \[Project\]](#)

[GetObjPath \[Project\]](#)

[GetPath](#)

[GetPropEvaluatedValue](#)

[GetPropNames \[Project\]](#)

[GetPropSIValue](#)

[GetPropValue \[Project\]](#)

[GetProperties](#)

[GetPropertyValue](#)

[GetTopDesignList](#)

[GetVariableValue](#)

[GetVariables](#)

InsertDesign

[InsertDesignWithWorkflow](#)

[InsertToolObject](#)

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

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

[RemoveAllUnusedDefinitions](#)

[RemoveMaterial](#)

[RemoveUnusedDefinitions](#)

[Rename](#)

RunToolkit

[Save](#)

[SaveAs](#)

[SaveAsStandAloneProject](#)

[SaveProjectArchive](#)

[SetActiveDefinitionEditor](#)

[SetActiveDesign](#)

[SetPropValue \[Project\]](#)

[SetPropertyValue](#)

[SetVariableValue](#)

[SimulateAll](#)

[Undo \[Project\]](#)

[UpdateDefinitions](#)

AddDataset

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

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

Python Syntax	AddDataset <DatasetDataArray>
Python Example	oProject.AddDataset (

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

```

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

AnalyzeAll [project]

Runs the project-level script command from the script, which simulates all solution setups and Optimetrics setups for all design instances in the project. The UI waits until simulation is finished before continuing with the script.

UI Access	Project > Analyze All.
Parameters	None.

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

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

ClearMessages

Clears information in the **Messages** window.

Note:

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

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

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

Close

Closes the active project.

Warning:

Unsaved changes will be lost.

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

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

CopyDesign

Copies a specified design.

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

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

CutDesign

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

Warning:

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

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

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

DeleteDataset

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

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

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

DeleteDesign

Deletes a specified design in the project.

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

Python Syntax	DeleteDesign (<DesignName>)
Python Example	<code>oProject.DeleteDesign("Design2")</code>

DeleteToolObject

Note:

This command is for internal Ansys use only.

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

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

EditDataset

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

UI Access	Project > Datasets > Edit.
------------------	---

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

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

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

GetActiveDesign

Returns the design in the active project

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

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

Parameters	None.
Return Value	Object of the active design.

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

GetChildNames [Project]

Returns the names of the project's child objects.

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

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

GetChildObject [Project]

Returns the project's child objects.

Note:

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

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

Python Syntax	<code>GetChildObject(<path>)</code>
Python Example	<pre>oProject = oDesktop.GetActiveProject() oDesign = oProject.GetChildObject("TeeModel") oVariable = oProject.GetChildObject("VariableName") oReport = oProject.GetChildObject("TeeModel/Results/S Parameter Plot 1")</pre>

GetChildTypes [Project]

Returns the types of the project's child objects.

UI Access	N/A
Parameters	None.
Return Value	Array of string represents types of the child objects.

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

GetDefinitionManager

Gets the `DefinitionManager` object.

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

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

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

GetDependentFiles

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

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

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

GetEDBHandle

Returns the EDB handle for the project.

Important:

This script is for internal Ansys use only.

UI Access	N/A
Parameters	None.

Return Value	String indicating the EDB handle for the project.
---------------------	---

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

GetLegacyName

Obtains the legacy name of a project.

Note:

This command is for internal Ansys use only.

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

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

GetName [Project]

Obtains the project name

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

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

GetPath

Returns the location of the project on disk.

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

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

--	--

GetPropNames [Project]

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

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

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

GetPropValue [Project]

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

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

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

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

GetTopDesignList

Returns a list of top-level design names.

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

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

ImportDataset

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

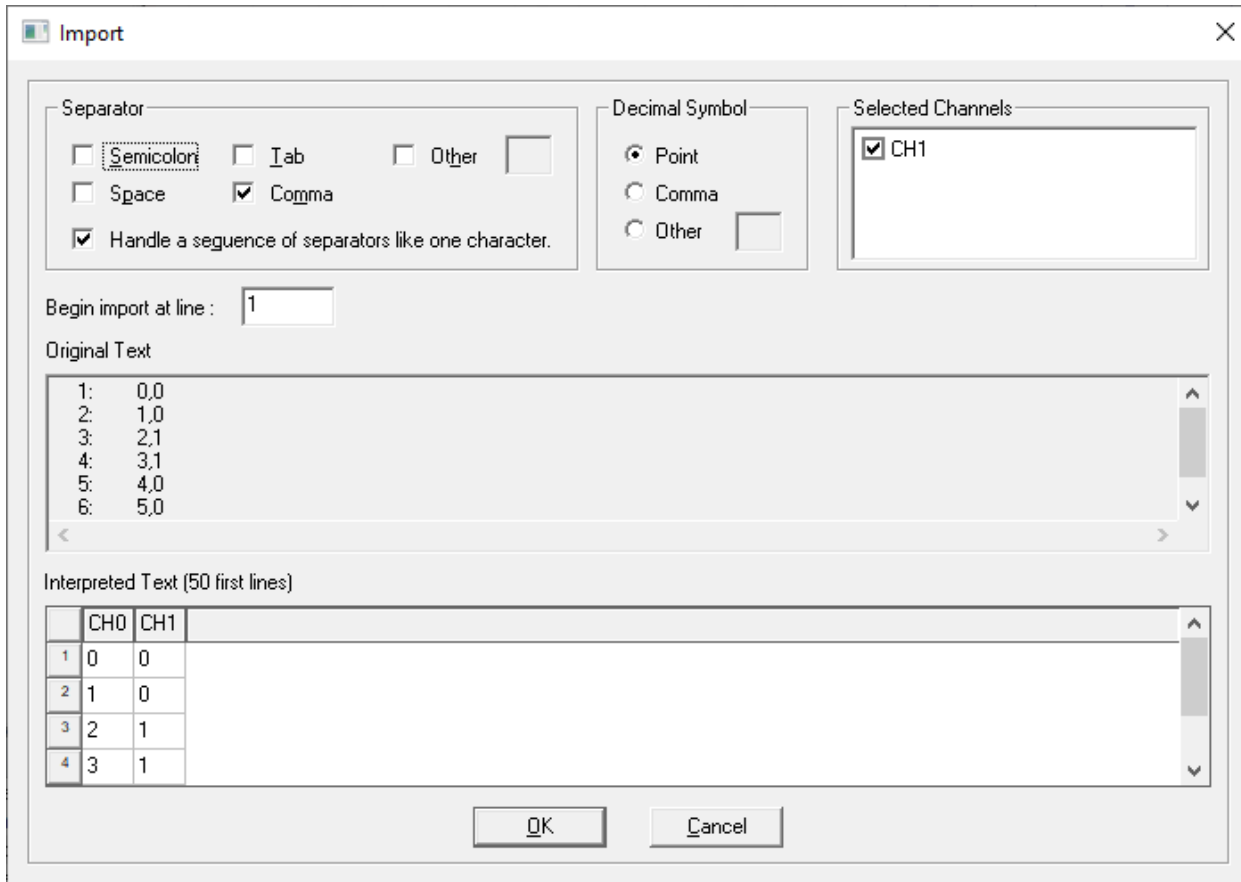
UI Access	Project > Datasets > Import.		
Parameters	Name	Type	Description
	< <i>datasetFullPath</i> >	String	The full path to the file containing the dataset values. *.tab files recommended (see note below).
	< <i>optionalDatasetName</i> >	String	<i>Optional.</i> User-defined dataset name.
Return Value	None.		

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

Note About File Types:

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

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



InsertDesignWithWorkflow

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

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

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

InsertToolObject

Note:

This command is for internal Ansys use only.

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

Paste (Project Object)

UI Access	Edit > Paste.
Parameters	None.
Return Value	

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

Redo [Project Level]

Reapplies the last project-level command.

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

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

Rename

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

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

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

RestoreProjectArchive

Restores a previously archived project to a specified path.

UI Access	File > Restore Archive.
------------------	-----------------------------------

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

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

Save

Saves the active project.

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

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

SaveAs

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

Note:

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

UI Access	File > Save As.		
Parameters	Name	Type	Description
	<NewName>	String	The desired name of the project, with directory and extension.
	<OverWriteOK>	Boolean	True to overwrite the file of the same name, if it exists. False to prevent overwrite.
	<DefaultAction>	String	For Schematic/Layout projects only. Otherwise omit. See note below. Valid actions: ef_overwrite , ef_copy_no_overwrite, ef_make_path_absolute, or empty string.
	<OverwriteActions>	Array	For Schematic/Layout projects only. Otherwise omit. See note below. Structured array: Array("Name: <Action>", <FileName>, <FileName>, ...) Valid actions: ef_overwrite , ef_copy_no_overwrite, ef_make_path_absolute, or empty string.
Return Value	None.		

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

Important:

The DefaultAction and OverrideActions strings correspond to the following actions:

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

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

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

SaveAsStandAloneProject

Saves the project as a standalone copy.

Note:

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

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

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

SaveProjectArchive

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

UI Access	File > Archive.		
Parameters	Name	Type	Description
	<archiveFilePath>	String	Path to archived file.
	<IncludeExternalFiles>	Boolean	True to include external files; False to exclude.
	<IncludeResultsFiles>	Boolean	True to include simulation files associated with the project; False to

			exclude.
	<AdditionalFiles>	Array	Additional specified files to include.
	<ArchiveNotes>	String	String describe the archive.
Return Value	None.		

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

SetActiveDefinitionEditor

Obtains a specified definition editor.

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

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

SetActiveDesign

Sets a design to be the active design.

UI Access	N/A		
Parameters	Name	Type	Description
	<DesignName>	String	Name of the design to set as the active design.
Return Value	None.		

Python Syntax	<code>SetActiveDesign (<DesignName>)</code>
Python Example	<pre>oDesign = oProject.SetActiveDesign("SimpleorerDesign2")</pre>

SetPropValue [Project]

Sets a property value for an active project's child object.

UI Access	Edit Properties on ProjectTree objects.		
Parameters	Name	Type	Description

	<i><propPath></i>	String	A child object's property path. See: Property Function .
	<i><newValue></i>	String	New property value.
Return Value	Boolean: <ul style="list-style-type: none"> • True – property found. • False – property not found. 		

Python Syntax	<code>SetPropValue(<propPath>, <newValue>)</code>		
Python Example	<pre>oProject.SetPropValue("TeeModel/offset", "2mm") oProject.SetPropValue("TeeModel/Results/S Parameter Plot 1/Display Type", "Data Table")</pre>		

SimulateAll

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

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

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

Undo [Project]

Cancels the last project-level command.

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

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

UpdateDefinitions

Updates all definitions. The **Messages** window reports when definitions are updated, or warns when definitions cannot be found.

UI Access	Tools > Project Tools > Update Definitions. Click Select All , then Update .
Parameters	None.
Return Value	None.

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

ValidateDesign

Returns whether a design is valid.

UI Access	> Validation Check.
Parameters	None.
Return Value	Integer: <ul style="list-style-type: none">• 1 – Validation passed.• 0 – Validation failed.

Python Syntax	ValidateDesign()
Python Example	<code>oDesign.ValidateDesign()</code>

6 - Property Script Commands

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

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

`GetName ()`

- Returns the name of the object.

`GetChildTypes ()`

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

`GetChildNames (<type>)`

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

`GetChildObject (<objPath>)`

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

`GetPropNames (<bIncludeReadOnly>)`

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

`GetPropValue (<propertyPath>)`

- <propertyPath> – The property's path; may be a child object's path appended with a property name (for example, TeeModel/Offset/SIValue).

- Returns the property value if found. Otherwise causes script error.

`SetPropValue(<propertyPath>, <data>)`

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

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

Note:

Older property commands should be executed by the oProject object.

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

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

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

[Object Script Property Function Summary](#)

[Conventions Used in this Chapter](#)

[GetArrayVariables](#)

[GetProperties](#)

[GetPropertyValue](#)

[GetVariables](#)

[GetVariableValue](#)

[SetPropertyValue](#)

[SetVariableValue](#)

Example Use of Record Script and Edit Properties

Object Script Property Function Summary

Object Path

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

- An Object path consisted of one or multiple Object-ID-Nodes separated by "/" .
- Object-ID-Node; may exist in the following forms:
 - A simple object name or property name.
 - Type[Name] for object; Tab[name] for property.
 - Name[attr1="v1", attr2 = "v2", ...]. When more than one child object have the same name, use attributes to specify the difference.
 - ArrayName[index]. For example, in an Optimetric setup with multiple calculations, "Calculation[0]" could be used to identify the first calculation.
 - Name beginning with '@' character denoted as a property name, when an object has a child and property with the same name.

Property Object

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

- GetName()
 - Returns the name of the object.

- `GetChildTypes()`
 - An object may have different type of children. For example, a design may have variables, modules, and editors.
 - Returns array of text strings; may be empty if the children are *not* divided to different types.
- `GetChildNames(<type>)`
 - *<type>* – children type name; default returns all children names for all types.
 - Returns an array of immediate children names, belonging to the type if specified.
- `GetChildObject(<objPath>)`
 - *<objPath>* – A child object path. The path may include multiple generations, such as (designObject/moduleObj/SetupObject).
 - Returns a child property object if the object found.
- `GetPropEvaluatedValue(<propName>)`
 - Return the Evaluated-Value for Value-Property and Variable.
 - Return the Property-value as text string for other property types.
- `GetPropSIValue(<propName>)`
 - Return the SI-Value for Value-Property and Variable.
 - Return NAN for other property type if its value is cannot be converted to a double-floating point value.
- `GetPropNames(<blIncludeReadOnly>);`
 - *<blIncludeReadOnly>* – optional, default to true; True will include read-only properties, False will exclude read-only properties.
 - Returns an array of the object's property names.
- `GetPropValue(<propertyPath>)`
 - *<propertyPath>* – the property's full path. A property name or child object's path appended with a property name, like "TeeModel/Offset/SIValue"
 - Returns the property value if the property is found; otherwise causes script error.

- `SetPropValue(<propertyPath>, <data>)`
 - `<propertyPath>` – the property's full path. A property name or child object's path appended with a property name, like "TeeModel/Offset/Value"
 - `<data>` – new data, type is dependent on property type.
 - Returns True if property data is updated successfully; False if the new data is invalid.

Project Object

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

- `GetChildTypes()` always return ["Design", "Variable"].
- `GetChildNames(type)`
`GetChildNames()` & `GetChildName("Design")` will return all Design names of the project.
`GetChildNames("Variable")` return all project variable names.
- `GetChildObject(objPath)`
`oDesign = oProject("TeeModel")`
`oVariable = oProject.GetChildObject("VariableName")`
`oReport = oProject.GetChildObject("TeeModel/Results/S Parameter Plot 1")`
- `GetPropNames(bIncludeReadOnly)` always return empty array since the project has no property.
- `GetPropValue(propertyPath)`
`oProject.GetPropValue("TeeModel/offset")` //get the offset variable value in the TeeModel Design
`oProject.GetPropValue("TeeModel/Results/S Parameter Plot 1/Display Type")` // Get the report display type.
- `SetPropValue(propertyPath, newValue)`
`oProject.SetPropValue("TeeModel/offset", "2mm")` //Set the offset variable value to "2mm" in the TeeModel Design
`oProject.SetPropValue("TeeModel/Results/S Parameter Plot 1/Display Type", "Data Table")` // Set the report display type to data table.

Design Object

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

- GetChildTypes() always return ['Module', 'Editor', 'Variable'].
- GetChildNames(type)
GetChildNames() will return modules & editor child names.
GetChildNames("Variable") will return all variable names
GetChildNames("Module") will return all module names that support property-object-script like ['Optimetrics', 'RadField', 'Results']
GetChildNames("Editor") will return a 3D editor name for all 3D Designs
- GetChildObject()
oVariable = oDesign.GetChildObject("VariableName")
oReport = oDesign.GetChildObject("Results/S Parameter Plot 1")
oRptModule = oDesign.GetChildObject("ReportSetup")
- GetPropNames(bIncludeReadOnly) always return empty array since the design has no property.
- GetPropValue()
oDesign GetPropValue("offset/SIValue") //get the offset variable SI value in the Design
oDesign GetPropValue("offset/SIValue") //get the offset variable SI value in the Design
oDesign.GetPropValue("Results/S Parameter Plot 1/Display Type") // Get the report display type
- SetPropValue()
oDesign.SetPropValue("offset", "2mm") //Set the offset variable value to "2mm" in the Design
oDesign.SetPropValue("Results/S Parameter Plot 1/Display Type", "Data Table") // Set the report display type to data table.

3D Modeler Object

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

```

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

```

Variable Object

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

```

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

```

- GetChildTypes() always return empty array.
- GetChildNames() always return empty array, since variable has no child.
- GetChildObject(objPath) it has no child.
- GetPropNames(bIncludeReadOnly) ['EvaluatedValue', 'SIValue'] are read-only properties
 - Independent variable :['Value', 'EvaluatedValue', 'SIValue', 'Description', 'ReadOnly', 'Hidden', 'Sweep', 'Optimization/Included', 'Optimization/Min', 'Optimization/Max', 'Sensitivity/Included', 'Sensitivity/Min', 'Sensitivity/Max', 'Sensitivity/IDisp', 'Statistical', 'Statistical/Included', 'Tuning/Included', 'Tuning/Step', 'Tuning/Min', 'Tuning/Max'].
 - Dependent variable ['Value', 'EvaluatedValue', 'SIValue', 'Description', 'ReadOnly', 'Hidden', 'Sweep']
- GetPropValue(propName)
 - oVar.GetPropValue() return the variable value as text string.
 - oVar.GetPropValue("Value") return the variable value as text string.
 - oVar.GetPropValue("SIValue") return the SI-value of variable as number.
 - oVar.Get_SIValue() also return the SI value.

- SetPropValue(propName, newValue)
oVar.SetPropValue("Value", 888)
oVar.SetPropValue("Sensitivity/Included", True)
oVar.SetPropValue("Sensitivity/Max", '1.8pF')
oVar.Set_Sensitivity_Max('1.8pF') also works as last call.
oVar.SetPropValue("Sensitivity", ['Min:=' , '0.8pF' , 'Max:=' , '1.8pF'])
//set multiple attributes at one call:
oVar.SetPropValue("@", ['Value:=' , 288 , 'Sensitivity' , ['Included' , True , 'Min' , '0.0']])
oVar.SetPropValue("", ['Value:=' , 288 , 'Sensitivity' , ['Included' , True , 'Min' , '0.0']])

Optimetrics Module Object:

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

- GetChildTypes() there are six type of children, they are ['OptiParametric', 'OptiOptimization', 'OptiSensitivity', 'OptiStatistical', 'OptiDesignExplorer', 'OptiDXDOE']. But the return array only included those that have setup defined, so it may be an empty array if no optimetrics setup is defined. The GetChildNames(type) function also recognized the type name without the prefix "Opti".
- GetChildNames(type)
GetChildNames() will return all setup for all types.
GetChildNames("OptiOptimization") & GetChildNames("Optimization") will return all Optimization setup.
- GetChildObject()
oParamSetup = oOptModule.GetChildObject('ParametricSetup1') get the
oOptSetup = oOptModule.GetChildObject('OptimizationSetup1')
- GetPropNames(bIncludeReadOnly) always return empty array since the it has no property.
- GetPropValue(propPath) may be used to get its child's property value
oOptModule.GetPropValue("OptimizationSetup1\Optimizer") get the optimizer name for OptimizationSetup1
- SetPropValue(propPath, newValue) may be used to set its child's property value
oOptModule.SetPropValue(ParametricSetup1\Enabled", False) //disable ParametricSetup1

Optimetrics Setup Object

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

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

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

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

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

ReportSetup(Results) Module Object:

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

- GetChildTypes() always empty array.
- GetChildNames(type)
 - GetChildNames() return all report names

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

ReportSetup(Results) Module Child Objects:

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

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

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

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

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

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

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

- `SetPropValue(propName, newValue)`
`oTrace.SetPropValue("Primary sweep", "Freq")`

Radiation Module Object:

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

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

Radiation Module Child Objects:

These are Property Objects. It also provides quick function call to get/set its properties by adding functions with property name appended to `Get_` & `Set_` prefix. To find what functions it provides enter `dir(oVar)` the command window.

Those child objects can be access by call all levels of parent object's `GetChildObject(path)` function.

```
oRadSetup = oRadModule.GetChildObject(setupName)
```

```
oRadSetup = oDesign.GetChildObject(RadField/setupName)
```

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

- `GetPropNames(bIncludeReadOnly)` will return the property names listed in the property window when the setup is selected.
- `GetPropValue(propName)`
 - o `oRadSetup.GetPropValue("Num Points")` return the line setup's "Num Points" property value.
 - o `oRadSetup.Get_NumPoints()` will also get the same value.
- `SetPropValue(propName, newValue)`
 - o `oRadSetup.SetPropValue('Num Points', 888)`
 - o `oRadSetup.Set_NumPoints(888)`

Conventions Used in this Chapter

General Definitions:

Property	A single item that can be modified in the Properties window or in the modal Properties pop-up window.
<PropServer>	The item whose properties are being modified. This is usually a compound name, giving all information needed by the editor, design, or project in order to locate the item.
<PropTab>	Corresponds to one tab in the Properties window, the one under which properties are being edited.
<PropName>	The name of a single property.

The following tables list specific <PropServer> and <PropTab> values for different property types.

For Project Variables:

<PropServer>	"ProjectVariables"
<PropTab>	"ProjectVariableTab"

For Local Variables:

<PropServer>	"LocalVariables"
<PropTab>	"LocalVariableTab"

For Passed Parameters:

<PropServer>	"Instance:<Name of Circuit Instance>"
<PropTab>	"PassedParameter Tab"

For Definition Parameters:

<PropServer>	"DefinitionParameters"
<PropTab>	"DefinitionParameters"

For Modules and Editors:

<PropServer>	<ModuleName>:<ItemName> where <ItemName is the boundary name, solution setup name, etc. For example, "BoundarySetup:PerfE1"
<PropTab>	Boundary Module: "HfssTab" Mesh Operations Module: "MeshSetupTab" Analysis Module: "HfssTab" Optimetrics Module: "OptimetricsTab" Solutions Module: <i>Does not support properties.</i> Field Overlays Module: "FieldsPostProcessorTab" Radiation Module: "RadFieldSetupTab" Circuit Module: "CCircuitTab" System Module: "SystemTab" HFSS 3D Layout Module: "HFSS 3D LayoutTab" Nexxim Module: "NexximTab" Layout elements: "BaseElementTab" Schematic elements: "ComponentTab" Optimetrics Module: "OptimetricsTab"

For 3D Model Editor objects:

<PropServer>	Name of the object. For example, "Box1".
<PropTab>	"Geometry3DAttributeTab"

For 3D Model Editor operations:

<PropServer>	<ObjName>:<OperationName>:<int> where <int> is the operation's history index. For example, "Box2:CreateBox:2" refers to the second "CreateBox" operation in Box2's history.
<PropTab>	"Geometry3DCmdTab"

For Reporter operations on Report properties:

<PropServer>	<ReportSetup>
<ChangeProperty>	<p>Array. For example, to set the company name in a plot header to "My Company":</p> <pre>Set oModule = oDesign.GetModule("ReportSetup") oModule.ChangeProperty Array("NAME:AllTabs",_ Array("NAME:Header",_ Array ("NAME:PropServers",_ "XY Plot1:Header"), Array("NAME:ChangedProps",_ Array("NAME:Company Name", "Value:=", "My Company"))))</pre>

Note:

For scripted property changes in the various modules and editors, refer to the chapters on the System, HFSS 3D Layout, and Nexxim tools, as well as the Layout and Schematic editors.

GetArrayVariables

Returns a list of array variables. To get a list of indexed project variables, execute with oProject. To get a list of indexed local variables, use oDesign.

UI Access	N/A
Parameters	None.
Return Value	Array of strings containing names of variables.

Python Syntax	GetArrayVariables()
Python Example	<pre>oProject.GetArrayVariables() oDesign.GetArrayVariables()</pre>

GetProperties

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

UI Access	N/A		
Parameters	Name	Type	Description
	<PropTab>	String	<p>One of the following, where tab titles are shown in parentheses:</p> <ul style="list-style-type: none"> PassedParameterTab ("Parameter Values") DefinitionParameterTab (Parameter Defaults") LocalVariableTab ("Variables" or "Local Variables") ProjectVariableTab ("Project variables")

			<ul style="list-style-type: none"> • ConstantsTab ("Constants") • BaseElementTab ("Symbol" or "Footprint") • ComponentTab ("General") • Component("Component") • CustomTab ("Intrinsic Variables") • Quantities ("Quantities") • Signals ("Signals")
	<code><PropServer></code>	String	An object identifier, generally returned from another script method, such as <code>CompInst@R;2;3</code>
Return Value	Array of strings containing the names of the appropriate properties.		

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

GetPropertyValue

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

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

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

	Name	Type	Description
Parameters	<PropTab>	String	One of the following, where tab titles are shown in parentheses: <ul style="list-style-type: none"> PassedParameterTab ("Parameter Values") DefinitionParameterTab (Parameter Defaults") LocalVariableTab ("Variables" or "Local Variables") ProjectVariableTab ("Project variables") ConstantsTab ("Constants") BaseElementTab ("Symbol" or "Footprint") ComponentTab ("General") Component("Component") CustomTab ("Intrinsic Variables") Quantities ("Quantities") Signals ("Signals")
	<PropServer>	String	An object identifier, generally returned from another script method, such as CompInst@R;2;3
	<PropName>	String	Name of the property.
Return Value	String value of the property.		

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

GetVariables

Returns a list of all defined variables. To get a list of project variables, execute this command using `oProject`. To get a list of local variables, use `oDesign`.

UI Access	N/A
Parameters	None.
Return Value	Array of strings containing the variables.

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

GetVariableValue

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

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

Return Value	String represents the value of the variable.
---------------------	--

Python Syntax	<code>GetVariableValue(<VarName>)</code>
Python Example	<code>oProject.GetVariableValue("var_name")</code>

SetPropertyValue

Sets the value of a single property belonging to a specific PropServer and PropTab. This function is available with the Project, Design or Editor objects, including definition editors. This is not supported for properties of the following types: ButtonProp, PointProp, V3DPointProp, and VPointProp. Only the ChangeProperty command can be used to modify these properties.

Use the script recording feature and edit a property, and then view the resulting script entry or use GetPropertyValue for the desired property to see the expected format.

UI Access	N/A		
Parameters	Name	Type	Description
	<propTab>	String	<p>One of the following, where tab titles are shown in parentheses:</p> <ul style="list-style-type: none"> PassedParameterTab ("Parameter Values") DefinitionParameterTab (Parameter Defaults") LocalVariableTab ("Variables" or "Local Variables") ProjectVariableTab ("Project variables") ConstantsTab ("Constants") BaseElementTab ("Symbol" or "Footprint") ComponentTab ("General") Component("Component")

			<ul style="list-style-type: none"> • CustomTab ("Intrinsic Variables") • Quantities ("Quantities") • Signals ("Signals")
	<code><propServer></code>	String	An object identifier, generally returned from another script method, such as <code>CompInst@R;2;3</code>
	<code><propName></code>	String	Name of the property.
	<code><propValue></code>	String	The value for the property
Return Value	None.		

Python Syntax	<code>SetPropertyValue(<propTab>, <propServer>, <propName>, <propValue>)</code>
Python Example	<code>oEditor.SetPropertyValue("PassedParameterTab", "k", "R", "2200")</code>

SetVariableValue

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

UI Access	N/A		
Parameters	Name	Type	Description
	<code><VarName></code>	String	Variable name.
	<code><VarValue></code>	Value	New value for the variable.
Return Value	None.		

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

This page intentionally
left blank.

7 - Design Object Script Commands

Design object commands should be executed by the oDesign object.

```
oDesign.CommandName <args>
```

For example:

```
Set oDesign = oProject.SetActiveDesign("EMITDesign1")
```

[AddLink](#)

[AddResult](#)

[CloseResultWindow](#)

[DeleteAllResults](#)

[DeleteDesign](#)

[DeleteLink](#)

[DeleteResult](#)

[EditComponentNodes](#)

[EditNotes](#)

[GetAvailableLinkNames](#)

[GetComponentNodeNames](#)

[GetComponentNodeProperties](#)

[GetComponentWarnings](#)

[GetCouplingWarnings](#)

[GetCurrentResult](#)

[GetDesignType](#)

[GetLinkNames](#)

[GetManagedFilePath](#)

[GetName](#)

[GetRadioNames](#)

[GetResultList](#)

[GetResultNotes](#)

[GetResultProperties](#)

[GetRevision](#)

[Redo](#)

[RenameDesign](#)

[RenameResult](#)

[RunToolkit](#)

[SetActiveEditor](#)

[SetResultNotes](#)

[ShowCouplingDialog](#)

[ShowEditDialog](#)

[ShowResultWindow](#)

[TransferToEmit](#)

[Undo](#)

[UpdateLink](#)

AddLink [EMIT]

Create a new link to a HFSS or 3D Layout design.

UI Access	EMIT>Add Link
Parameters	<name> – Name of the HFSS or HFSS 3D Layout design to add as a linked design. The design must be in the current project.
Return Value	The name of the new link.

Python Syntax	AddLink(<name>)
Python Example	<code>oDesign.AddLink("HFSSDesign1")</code>

AddResult [EMIT]

Create a new EMIT result. The new result node is shown under Results in the project manager.

If a current result (result with revision matching the current design revision) already exists, no result node is created and an error is shown in the message manager.

UI Access	EMIT > Analyze when a component is selected in the schematic
Parameters	<desiredName> – Name of the new result node. This is an optional parameter. If omitted, a default name is assigned. If a desiredName is specified and a result with that name already exists, no new result node is created and an error is shown in the message manager.
Return Value	The name of the new result.

Python Syntax	<code>AddResult(<desiredName>)</code>
Python Example	<code>oDesign.AddResult("Mitigated - Updated Antenna Placement")</code>

CloseResultWindow

Close the result window for the specified result. This command has no effect if no window is open for the specified result.

UI Access	Click the 'X' in the upper-right corner of the result window.
Parameters	<code><resultName></code> – Name of the new result node. If no result with the given name exists, an error is shown in the message manager.
Return Value	None

Python Syntax	<code>CloseResultWindow(<resultName>)</code>
Python Example	<code>oDesign.CloseResultWindow("Mitigated - Updated Antenna Placement")</code>

DeleteAllResults

Deletes all results in the Results folder

UI Access	Right-click on the Results folder, and click Delete All Results .
Parameters	None
Return Value	None

Python Syntax	DeleteAllResults()
Python Example	<code>oDesign.DeleteAllResults()</code>

DeleteDesign

Deletes a specified design in the project.

UI Access	Edit > Delete or Delete in the ribbon.
Parameters	None
Return Value	None

Python Syntax	DeleteDesign ()
Python Example	<code>oDesign.DeleteDesign ()</code>

DeleteLink [EMIT]

Delete an existing coupling link.

UI Access	Right-click on the link and click Delete or Edit>Delete with the link selected.
Parameters	<name> – Name of the HFSS or HFSS 3D Layout design link to delete (unlink) from the EMIT design. .
Return Value	The name of the link to be deleted.

Python Syntax	DeleteLink(<name>)
Python Example	<code>oDesign.DeleteLink("HFSSDesign1")</code>

DeleteResult [EMIT]

Delete an EMIT result.

UI Access	Right-click on result node, and click Delete .
Parameters	<resultName> –Name of the result node to delete. If no result with the given name exists, an error is shown in the message manager.
Return Value	None

Python Syntax	DeleteResult(<resultName>)
Python Example	<code>oDesign.DeleteResult("Mitigated - Updated Antenna Placement")</code>

EditComponentNodes [EMIT]

Edit the nodes for an EMIT component. The component properties are stored in a hierarchy (tree) of nodes. These nodes are visible on the left-hand side of the component Edit dialog box. The usage of this method can be observed by editing some properties of a component while recording a script.

UI Access	With a component selected in the schematic, select Edit>Configure .
Parameters	<componentName> – The component to edit.

	<p><i><nodeChanges></i> – The list of component property nodes and the changes to apply to each.</p> <p><i><nodesToDelete></i> – A list of property nodes to remove from the component property tree.</p>
Return Value	None

Python Syntax	<code>EditComponentNodes(<componentName>, <nodeChanges>, <nodesToDelete>)</code>
Python Example	<pre>oDesign.EditComponentNodes("Clock_CTL", [["NAME:node", "fullname:=", "NODE-*--RF Systems-*--RF System-*--Radios- form", ["NAME:properties", "ChannelSpacing:=", "3400000000.0", "StartFrequency:=", "2400000000.0", "StopFrequency:=", "5800000000.0"]]], [])</pre>

EditNotes

Update the notes for the design.

UI Access	EMIT > Edit Notes.		
Parameters	Name	Type	Description
	<DesignNotes>	String	New design notes that are applied, replacing any current notes. .
Return Value	None.		

Python Syntax	EditNotes(<DesignNotes>)
Python Example	<code>oDesign.EditNotes("Initial Cosite analysis, prior to expanded WiFi installation.")</code>

GetAvailableLinkNames [EMIT]

Get the names of linkable designs (excluding those that have been linked to already).

UI Access	N/A
Parameters	None.
Return Value	The list of linkable design names.

Python Syntax	GetAvailableLinkNames ()
----------------------	--------------------------

Python Example	<code>names = oDesign.GetAvailableLinkNames()</code>
-----------------------	--

GetComponentNodeNames [EMIT]

Get the properties nodes for an EMIT component. The component properties are stored in a hierarchy (tree) of nodes. These nodes are visible on the left-hand side of the component Edit Dialog.

UI Access	N/A
Parameters	<i><componentName></i> – The component that will be queried for property nodes.
Return Value	The list of property nodes for the specified component.

Python Syntax	<code>GetComponentNodeNames(<componentName>)</code>
Python Example	<code>oDesign.GetComponentNodeNames("Clock_CTL")</code>

GetComponentNodeProperties [EMIT]

Get the properties nodes for an EMIT component. The component properties are stored in a hierarchy (tree) of nodes. These nodes are visible on the left-hand side of the component Edit Dialog.

UI Access	N/A
Parameters	<i><componentName></i> – The component that will be queried for property nodes. <i><nodeFullName></i> – The node of the property tree.
Return Value	The list of properties and values for the specified component property node.

Python	<code>GetComponentNodeProperties(<componentName>, <nodeFullName>)</code>
---------------	--

Syntax	
Python Example	<code>oDesign.GetComponentNodeProperties("Clock_CTL", 'NODE-*--RF Systems-*--RF System-*--Rados-*--Radio-*--Band')</code>

GetComponentWarnings [EMIT]

Get the warning messages associated with a component configuration.

UI Access	Hover mouse over a schematic component showing a warning indicator.
Parameters	<i><componentName></i> – The component that will be queried for warnings
Return Value	The warnings associated with the component. An empty string ("") is returned if there are no warnings.

Python Syntax	<code>GetComponentWarnings(<componentName>)</code>
Python Example	<code>oDesign.GetComponentWarnings("Radio")</code>

GetCouplingWarnings [EMIT]

Get the warning messages associated with the EMIT coupling configuration.

UI Access	Click the Coupling node in the project manager or click an individual coupling link to observe the warnings in the status bar.
Parameters	<i><optionalLinkName></i> – Warnings for the link are returned if a link is specified. If not specified or an empty string (""), the general coupling warnings are returned.
Return Value	The warnings associated with a specific coupling link or general coupling warnings.

Python Syntax	<code>GetCouplingWarnings(<optionalLinkName>)</code>
Python Example	<pre>link_warnings = oDesign.GetCouplingWarnings("HFSSDesign1") general_warnings = oDesign.GetCouplingWarnings("")</pre>

GetCurrentResult

Get the name of the current EMIT result (result with revision matching the current design revision).

UI Access	N/A
Parameters	None.
Return Value	The name of the current result, if present. An empty string is returned if there is no current result.

Python Syntax	<code>GetCurrentResult()</code>
Python Example	<pre>oDesign.GetCurrentResult() print("Name of current result: {}".format(currentResult))</pre>

GetDesignType

Returns the design type of the active design.

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

Parameters	None.
Return Value	String indicating the design type of the active design ("Circuit Design", "Circuit Netlist", "EMIT", "HFSS 3D Layout Design", "HFSS", "HFSS-IE", "Icepak", "Maxwell 2D", "Maxwell 3D", "Q2D Extractor", "Q3D Extractor", "RMxprt", or "Twin Builder").

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

GetLinkNames [EMIT]

Get the names of all coupling links in the EMIT design.

UI Access	The nodes visible under “Coupling” in the Project Manager.
Parameters	None.
Return Value	The list of link names.

Python Syntax	GetLinkNames()
Python Example	<pre>links = oDesign.GetLinkNames()</pre>

GetManagedFilePath

Get the path to the project's results folder.

UI Access	N/A
Parameters	None.
Return Value	String containing path where project results are located.

Python Syntax	<code>oDesign.GetManagedFilePath()</code>
Python Example	<code>oDesign.GetManagedFilePath()</code>

GetName

Returns the design name of the active design, in that order separated by a semicolon.

UI Access	N/A
Parameters	None.
Return Value	String indicating the name of the active design.

Python Syntax	<code>GetName()</code>
Python Example	<code>design_name = oDesign.GetName()</code>

GetRadioNames [EMIT]

Get a list of all excitations and receivers in the EMIT library.

UI Access	N/A.
Parameters	None.
Return Value	The list of excitations and receivers in the EMIT library.

Python Syntax	GetRadioNames()
Python Example	<pre>radios = oDesign.GetRadioNames()</pre>

GetResultList

Get the list of result names for the EMIT design.

UI Access	N/A
Parameters	None.
Return Value	A list of the names of all results.

Python Syntax	GetResultList()
Python Example	<pre>resultNames = oDesign.GetResultList() for i, name in enumerate(resultNames): print("Result {} is named '{}'.format(i+1, name))</pre>

GetResultNotes [EMIT]

Returns the notes for the specified result.

UI Access	N/A
Parameters	<i><resultName></i> - Name of the result node. If no result with the given name exists, an error is shown in the message manager.
Return Value	String containing the result's notes.

Python Syntax	GetResultNotes(<i><resultName></i>)
Python Example	<pre>notes = oDesign.GetResultNotes("Revision 1")</pre>

GetResultProperties

Get the properties for the specified result.

UI Access	Select the result node in the Project Manager. The properties are shown in the Properties window.
Parameters	<i><resultName></i> – Name of the result node. If no result with the given name exists, an error is shown in the message manager.
Return Value	A list containing 'key=value' entries for each property of the specified result.

Python Syntax	GetResultProperties(" <i><resultName></i> ")
Python Example	<pre>resultProperties = oDesign.GetResultProperties("Revision 2") for property in resultProperties:</pre>

	<code>print (property)</code>
--	-------------------------------

GetRevision

Get the revision of the EMIT design.

UI Access	Select the EMIT design node in the Project Manager. The revision is shown in the Properties window.
Parameters	None.
Return Value	The revision of the EMIT design.

Python Syntax	<code>GetRevision()</code>
Python Example	<code>designRevision = oDesign.GetRevision()</code>

Redo [EMIT]

Redo the last command undo performed on the design.

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

Python Syntax	Redo()
Python Example	<code>oDesign.Redo() # Redo and restore state prior to undo</code>

RenameDesign [EMIT]

Rename the EMIT design represented by the design object.

UI Access	Edit > Rename
Parameters	<code><newName></code> – The new name to assign to the design.
Return Value	None.

Python Syntax	<code>Rename(<newName>)</code>
Python Example	<code>oDesign.Rename("New Design Name")</code>

RenameResult

Rename an EMIT result.

UI Access	Right-click on result node, and click Rename .
Parameters	<p><code><resultName></code> – Name of the result node to rename. If no result with the given name exists, an error is shown in the message manager.</p> <p><code><newResutName></code> – The new name for the result node. If a result with the given name already exists, an error is shown in the message manager.</p>
Return Value	None.

Python Syntax	<code>RenameResult(<resultName>, <newResultName></code>
Python Example	<code>oDesign.RenameResult("Revision 2", "Mitigated - Filter Added")</code>

RunToolkit [EMIT]

Run a toolkit script installed in the Toolkits directory and available from the toolkit menu.

UI Access	EMIT> Toolkit
Parameters	<p><i><libraryType></i> – Library containing the toolkit (for example, 'syslib', 'userlib', 'personallib')</p> <p><i><toolkitName></i> – The name of the toolkit to be run.</p> <p><i><toolkitArgs></i> – Arguments to pass to the toolkit.</p>
Return Value	None.

Python Syntax	<code>RunToolkit(<libraryType>, toolkitName>, <toolkitArgs>)</code>
Python Example	<code>oDesign.RunToolkit("userlib", "CustomImportCSV", "C:\\temp\\datafile.csv")</code>

SetActiveEditor [EMIT]

Activate the EMIT Schematic Editor and get the SchematicEditor object for script-based manipulation.

UI Access	NA.
Parameters	<i><editorName></i> – The name of the editor. The only editor currently supported by EMIT is "SchematicEditor".
Return Value	The SchematicEditor (oEditor) object associated with the EMIT design.

Python Syntax	<code>SetActiveEditor(<editorName>)</code>
Python Example	<pre>oEditor = oDesign.SetActiveEditor("SchematicEditor") component_list = oEditor.GetAllComponents()</pre>

SetResultNotes [EMIT]

Sets the notes for the specified result.

UI Access	N/A
Parameters	<p><i><resultName></i> - Name of the result node. If no result with the given name exists, an error is shown in the message manager.</p> <p><i><newNotes></i> - A string containing the new desired notes for the specified result.</p>
Return Value	N/A

Python Syntax	<code>SetResultNotes(<resultName>, <newNotes>)</code>
Python Example	<code>oDesign.SetResultNotes("Revision 1", "New notes here.")</code>

ShowAnalysisAndResults [EMIT]

Display the EMIT Analysis & Results dialog box.

UI Access	EMIT > Analyze or Analysis & Results on the Simulation ribbon..
Parameters	<i><optionalScriptToRun></i> – A Python script to run within the Analysis & Results dialog box. If this argument is omitted or an empty string ("") is passed, no script is run.

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

Python Syntax	ShowAnalysisAndResults()
Python Example	<code>oDesign.ShowAnalysisAndResults()</code>

ShowCouplingDialog [EMIT]

Display the EMIT Coupling Editor dialog.

UI Access	EMIT>Coupling Editor
Parameters	<p><i><optionalNodeToSelect></i> – Optionally, a property node can be specified and this node will be initially selected when the dialog is shown.</p> <p><i><optionalScriptToRun></i> – A Python script to run within the Coupling Editor dialog box. If this argument is omitted or an empty string (“”) is passed, no script is run.</p>
Return Value	None

Python Syntax	ShowCouplingDialog(<i><optionalNodeToSelect></i> , <i><optionalScriptToRun></i>)
Python Example	<code>oDesign.ShowCouplingDialog('NODE-*--Scene-*--Antenna') # Show the coupling dialog with 'Antenna' selected</code>

ShowEditDialog [EMIT]

Display the EMIT Component Editor dialog for a specified component.

UI Access	Edit>Configure when a component is selected in the schematic
Parameters	<i><componentName></i> – The component to select. May be any component type in the schematic. <i><optionalScriptToRun></i> – A Python script to run within the Edit dialog box. If this argument is omitted or an empty string ("") is passed, no script is run.
Return Value	None

Python Syntax	ShowEditDialog(<i><componentName></i> , <i><optionalScriptToRun></i>)
Python Example	<code>oDesign.ShowEditDialog("Radio")</code>

ShowResultWindow

Show the result window for the specified result.

UI Access	Right-click the result node in the Project Manager, and select Show Result .
Parameters	<i><resultName></i> – Name of the result to show. If no result with the given name exists, an error is shown in the message manager. <i><ScriptToRun></i> - A Python script to run within the result window. If this argument is omitted or an empty string ("") is passed, no script is run. Optional.
Return Value	None.

Python Syntax	ShowResultWindow(<i><resultName></i> , <i><ScriptToRun></i>)
Python Example	<code>oDesign.ShowResultWindow("Mitigated - Filter Added")</code>

--	--

TransferToEmit

This method is no longer supported and should not be used.

Undo [Design]

Cancels the last design-level command.

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

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

UpdateLink [EMIT]

Update the coupling link, retrieving any changes to the port list, setups, sweeps, and associated coupling data from the linked design.

UI Access	Right-click on the link and click Update Link .
Parameters	<name> – Name of the HFSS or HFSS 3D Layout design link to update
Return Value	None.

Python Syntax	UpdateLink(< <i>name</i> >)
Python Example	<code>oDesign.UpdateLink("HFSSDesign1")</code>

This page intentionally
left blank.

8 - Core Global Script Context Commands

To run these commands:

```
import CoreGlobalScriptContextFunctions

CoreGlobalScriptContextFunctions.[CommandName]
```

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

- [AddErrorMessage](#)
- [AddFatalMessage](#)
- [AddInfoMessage](#)
- [AddWarningMessage](#)
- [LogDebug](#)
- [LogError](#)

AddErrorMessage

Adds an error message to the **Message Manager** window. AddErrorMessage is a function of CoreGlobalScriptContextFunctions.

UI Access	N/A		
Parameters	Name	Type	Description
	<message>	String	Error message.
Return Value	None.		

Python Syntax	AddErrorMessage(<message>)
Python Example	<pre>import CoreGlobalScriptContextFunctions CoreGlobalScriptContextFunctions.AddErrorMessage('My error message.')</pre>

AddFatalMessage

Adds a fatal error message to the **Message Manager** window. AddFatalMessage is a function of CoreGlobalScriptContextFunctions.

UI Access	N/A		
Parameters	Name	Type	Description

	<code><message></code>	String	Error message.
Return Value	None.		

Python Syntax	<code>AddFatalMessage(<message>)</code>
Python Example	<pre>import CoreGlobalScriptContextFunctions CoreGlobalScriptContextFunctions.AddFatalMessage('My fatal error message.')</pre>

AddInfoMessage

Adds an informational message to the **Message Manager** window. `AddInfoMessage` is a function of `CoreGlobalScriptContextFunctions`.

UI Access	N/A		
Parameters	Name	Type	Description
	<code><message></code>	String	Informational message.
Return Value	None.		

Python Syntax	<code>AddInfoMessage(<message>)</code>
Python Example	<pre>import CoreGlobalScriptContextFunctions CoreGlobalScriptContextFunctions.AddInfoMessage('My info.')</pre>

AddWarningMessage

Adds a warning message to the **Message Manager** window. `AddWarningMessage` is a function of `CoreGlobalScriptContextFunctions`.

UI Access	N/A		
Parameters	Name	Type	Description
	<code><message></code>	String	Warning message.
Return Value	None.		

Python Syntax	AddWarningMessge(<message>)
Python Example	<pre>import CoreGlobalScriptContextFunctions CoreGlobalScriptContextFunctions.AddWarningMessage('My warning.')</pre>

LogDebug

Adds a debug line to the log specified at **Tools > Debug Logging**. LogDebugis a function of CoreGlobalScriptContextFunctions.

UI Access	N/A		
Parameters	Name	Type	Description
	<message>	String	Debug message.
Return Value	None.		

Python Syntax	LogDebug(<message>)
Python Example	<pre>import CoreGlobalScriptContextFunctions CoreGlobalScriptContextFunctions.LogDebug('My debug message.')</pre>

LogError

Adds an error line to the log specified at **Tools > Debug Logging**. LogErroris a function of CoreGlobalScriptContextFunctions.

UI Access	N/A		
Parameters	Name	Type	Description
	<error>	String	Error to log.
Return Value	None.		

Python Syntax	LogError(<error>)
----------------------	-------------------

Python Example	<pre>import CoreGlobalScriptContextFunctions CoreGlobalScriptContextFunctions.LogError('My error.')</pre>
---------------------------	---

9 - Library Management Script Commands

Library Management script commands provide management functions for libraries in Personal Libraries. These functions are accessed via the Component Manager, which is accessed via the Definition Manager.

```
Set oDefinitionManager = oProject.GetDefinitionManager()
Set oComponentManager = oDefinitionManager.GetManager("Component")
```

The topics for this section include:

[DeleteLibraryComponent](#)

[RenameLibraryComponent](#)

DeleteLibraryComponent

Deletes a component in a specified library.

UI Access	From the Component Libraries window, right-click a component displayed under Personal Libraries and select Delete Library Component .		
Parameters	Name	Type	Description
	<compName>	String	Name of the component to be deleted.
	<libName>	String	Library name of the component to be deleted.
Return Value	None.		

Python Syntax	DeleteLibraryComponent(<compName>,<libName>)
Python Example	oComponentManager.DeleteLibraryComponent("Radio1", "Library1")

RenameLibraryComponent

Renames a component in a specified library.

UI Access	From the Component Libraries window, right-click a component displayed under Personal Libraries and select Rename Library Component .		
Parameters	Name	Type	Description
	<origCompName>	String	Name of the component to be renamed.

	<i><libName></i>	String	Library name of the component to be renamed.
	<i><newCompName></i>	String	New name for the component.
Return Value	None.		

Python Syntax	RenameLibraryComponent(<i><origCompName></i> , <i><libName></i> , <i><newCompName></i>)		
Python Example	oComponentManager.RenameLibraryComponent ("Radio1", "Library1", "NewRadio1")		

10 - Schematic Scripting for EMIT

The Schematic scripting interface is a set of commands that match the data changing methods available in the UI of the Schematic, plus some selection and query methods. Examples of the commands available through the scripting interface are adding items, removing items, and modifying items on the Schematic.

[BringToFront](#)

[ChangeProperty](#)

[ClearSelections](#)

[Copy](#)

[CreateArc](#)

[CreateCircle](#)

[CreateComponent](#)

[CreateComponentAt](#)

[CreateCurve](#)

[CreateEllipse](#)

[CreateImage](#)

[CreateLine](#)

[CreatePolygon](#)

[CreateRectangle](#)

[CreateText](#)

[Cut](#)

[Delete](#)

[DisableEmitComponents](#)

[ExportImage](#)

[ExportToLibrary](#)

[GetAllComponents](#)

[GetAllElements](#)

[GetComponentBoundingBox](#)

[GetComponentLocation](#)

[GetComponentOrientation](#)

[GetComponentPortLocation](#)

[GetComponentPorts](#)

[GetEditorName](#)

[GetNumComponentOrientations](#)

[GetPropertyValue](#)

[GetSelections](#)

[GetWireAtPort](#)

[GetWireConnections](#)

[GetZoomArea](#)

[IsEmitComponentDisabled](#)

[Move](#)

[Pan](#)

[Paste](#)[PlaceComponent](#)[RenameComponent](#)[ReorientComponent](#)[Select](#)[SelectAll](#)[SendToBack](#)[SetPropertyValue](#)[Wire](#)[ZoomArea](#)[ZoomIn](#)[ZoomOut](#)[ZoomPrevious](#)[ZoomToFit](#)

Editor Scripting IDs (EMIT)

Objects in the schematic are identified by text IDs. These IDs are used in scripts to perform actions on objects. These IDs are returned by all methods that add objects, and by the FindElements and GetSelections methods. The IDs are then passed into commands to modify or remove Schematic objects.

BringToFront (Schematic Editor)

Bring the selected object to the front of the view

UI Access	Draw>Bring to Front
------------------	---------------------

Parameters	Name	Type	Description
	<Selections>	Array	["NAME: Selections", "Selections:=", [<Selected Components>]] <Selected Components> CompInst@DefName; ID, schematic ID , CompInst@DefName; ID, schematic ID CompInst@R;1,2
Return Value	None		

Note:

For more information about the component name, schematic IDs, and formats, see the section Editor Scripting IDs.

Python Syntax	BringToFront()
Python Example	<pre>oEditor.BringToFront (["NAME:Selections", "Selections:=", ["SchObj@1"]])</pre>

ChangeProperty [EMIT]

Change the value of a property for one of the primitives (such as. Line, Rectangle) in the Schematic.

UI Access	Properties in the Property Window
Parameters	<args> – The structure specifying the tab, item, and property to be changed and the new value(s) to apply.

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

Python Syntax	ChangeProperty(<args>)
Python Example	<pre>oEditor.ChangeProperty(["NAME:AllTabs", ["NAME:BaseElementTab", ["NAME:PropServers", "SchObj@26"], ["NAME:ChangedProps", ["NAME:Color","R=", 0,"G=", 0,"B=", 255]]]])</pre>

ClearSelections

Clear all selections in the schematic.

UI Access	Click in an empty region in the schematic.
Parameters	None
Return Value	None

Python Syntax	ClearSelections()
Python Example	<pre>oEditor.ClearSelections()</pre>

Copy [EMIT]

Copy items in the schematic for pasting.

UI Access	Edit>Copy or Copy button on Schematic ribbon
Parameters	<elementNames> – A list of component, wire, or primitive names to copy.
Return Value	None.

Python Syntax	Copy(<elementNames>)
Python Example	<pre>oEditor.Copy(["Radio", "Antenna"]) oEditor.Paste(-0.00127, 0.01651)</pre>

CreateArc (Schematic Editor)

Create an arc

UI Access	Draw>Primitive>Arc		
Parameters	Name	Type	Description
	<ArcData>	Array	<pre>Array("NAME:ArcData" _ "x:=", double, _ // X position of the object "y:=", double, _ // Y position of the object "Radius:=", double, _ // Radius of the circle "StartAng:=", double, _ // Start angle of the arc (radians) "EndAng:=", double, _ // End angle of the arc (radians) "LineWidth:=", double, // the width of the line, in meters "Color:=", int, // the RGB value of the arc color</pre>

		"Id=", int), // [Opt=New id] Id for this item
	<Attributes>	Array ("NAME:Attributes", _ "Page:=", int) _ // [Opt=1] Page number (one-based) [out,retval] string id)
Return Value	String Unique id	

Arc is created with the format SchObj@<schematicID>

The Schematic ID can be found on the **Symbols** tab of the **Properties** window when you select the arc.

Properties		
Name	Value	Unit
SchematicID	48	
Color		
Linewidth	0	mil
FillStyle	Hollow	
Center	4100 , 3300	mil
Radius	360.5551275464	mil
StartAngle	123.69006752598	deg
EndAngle	315	deg

< Symbol

Python Syntax	CreateArc()
<p>Python Example</p>	<pre>oEditor.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 (Schematic Editor)

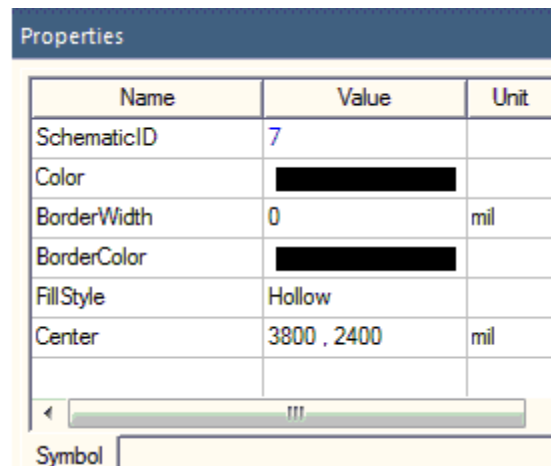
Create a circle

UI Access	Draw>Primitive>Circle								
<p>Parameters</p>	<table border="1"> <thead> <tr> <th data-bbox="453 998 716 1040">Name</th> <th data-bbox="716 998 873 1040">Type</th> <th data-bbox="873 998 1896 1040">Description</th> </tr> </thead> <tbody> <tr> <td data-bbox="453 1040 716 1313"><CircleData></td> <td data-bbox="716 1040 873 1313">Array</td> <td data-bbox="873 1040 1896 1313"> 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 </td> </tr> </tbody> </table>	Name	Type	Description	<CircleData>	Array	Array ("NAME:CircleData" _ "x:=", double, _ // X position of the object "y:=", double, _ // Y position of the object "Radius:=", double, _ // Radius of the circle "LineWidth:=", double, // the width of the circle border, in meters		
Name	Type	Description							
<CircleData>	Array	Array ("NAME:CircleData" _ "x:=", double, _ // X position of the object "y:=", double, _ // Y position of the object "Radius:=", double, _ // Radius of the circle "LineWidth:=", double, // the width of the circle border, in meters							

			<pre>"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 circle fill color "Id:=", int), // [Opt=New id] Id for this item</pre>
	<Attributes>	Array	<pre>Array("NAME:Attributes", _ "Page:=", int) _ // [Opt=1] Page number (one-based) [out,retval] string id)</pre>
Return Value	String Unique id		

Circle is created and the returned value has the format SchObj@<schematicID>

The schematic ID of the circle can be located in the **Properties** window on the **Symbols** tab.



Python Syntax	CreateCircle()
Python Example	<pre>oEditor.CreateCircle(["NAME:CircleData", _ "X:=", -0.004572, "Y:=", -0.000508, _ "Radius:=", 0.001778, "Id:=", 12], _ ["NAME:Attributes", "Page:=", 1])</pre>

CreateComponent [EMIT]

Create a new component of the given type in the schematic

UI Access	Component buttons on Schematic ribbon or dragging a component from the Component Libraries.
Parameters	<p><i><desiredName></i> – The desired name to assign to the new component. If an empty string ("") is given, a name is assigned automatically.</p> <p><i><componentType></i> – The type (for example, Cable) of component to create.</p> <p><i><libraryName [optional]></i> – The library file (*.aclb) containing the specified component. If not specified or an empty string (""), the syslib libraries are used.</p>
Return Value	The name of the new component.

Python Syntax	CreateComponent(<i><desiredName></i> , <i><componentType></i>)
----------------------	--

Python Example	<pre>oEditor.CreateComponent("", "New Radio") oEditor.CreateComponent("Cable C04", "Cable") oEditor.CreateComponent("Horn A01", "Antenna")</pre>
-----------------------	--

CreateComponentAt [EMIT]

Create a new component of the given type in the schematic

UI Access	Component buttons on Schematic ribbon or dragging a component from the Component Libraries.
Parameters	<p><i><desiredName></i> – The desired name to assign to the new component. If an empty string ("") is given, a name is assigned automatically.</p> <p><i><xPosition></i> – The x-location where the new component will be located on the schematic.</p> <p><i><yPosition></i> – The x-location where the new component will be located on the schematic.</p> <p><i><componentType></i> – The type (for example., Cable) of component to create.</p> <p><i><libraryName></i> [optional] – The library file (*.aclb) containing the specified component. If not specified or an empty string (""), the syslib libraries are used.</p>
Return Value	The name of the new component.

Python Syntax	CreateComponentAt(<i><desiredName></i> , <i><xPosition></i> , <i><yPosition></i> , <i><componentType></i>)
Python Example	<pre>oEditor.CreateEmitComponentAt("", -0.02032, -0.0254, "New Radio") oEditor.CreateEmitComponentAt("", -0.00762, -0.0254, "Cable") oEditor.CreateEmitComponentAt("", 0.00508, -0.02032, "Antenna")</pre>

CreateCurve [Schematic]

Creates a Bezier curve.

UI Access	Draw > Primitive > Curve		
Parameters	Name	Type	Description
	<curve data>	Array	("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	Name	Type	Description
	<id>	String	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>

```

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
])

```

CreateEllipse

Creates an ellipse.

UI Access	Draw > Ellipse.		
Parameters	Name	Type	Description
	<Parameters>	Array	Structured array. Array("NAME:EllipseParameters", "IsCovered:=", <string>, "XCenter:=", <string>, "YCenter:=", <string>, "ZCenter:=", <string>,

			"MajRadius:=", <string>, "Ratio:=", <string>, "WhichAxis:=", <string>, "NumSegments:=", <string>)
	<AttributesArray>	Array	Structured array. See: AttributesArray .
Return Value	None.		

Python Syntax	<code>CreateEllipse(<Parameters>, <Attributes>)</code>
Python Example	<pre> oEditor.CreateEllipse(["NAME:EllipseParameters", "IsCovered:=" , True, "XCenter:=" , "0.6mm", "YCenter:=" , "-0.6mm", "ZCenter:=" , "0mm", "MajRadius:=" , "0.2mm", "Ratio:=" , "7", "WhichAxis:=" , "Z", "NumSegments:=" , "0"], ["NAME:Attributes", </pre>

	<pre> >Name:=" , "Ellipse1", Flags:=" , "", Color:=" , "(143 175 143)", Transparency:=" , 0, PartCoordinateSystem:=" , "Global", UDMId:=" , "", MaterialValue:=" , "\"copper\"", SurfaceMaterialValue:=" , "\"\"", SolveInside:=" , False, ShellElement:=" , False, ShellElementThickness:=" , "0mm", IsMaterialEditable:=" , True, UseMaterialAppearance:=" , False, IsLightweight:=" , False] </pre>
--	--

CreateLine (Schematic Editor)

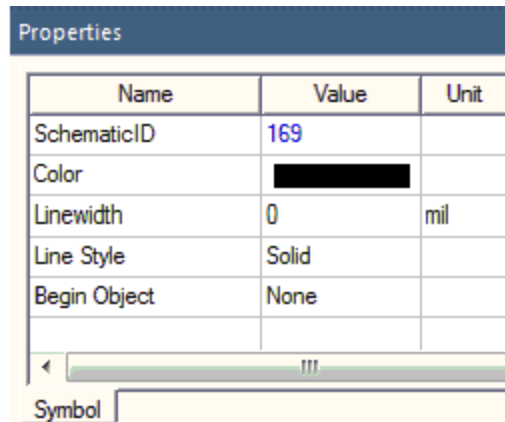
Creates a line.

UI Access	Draw>Primitive>Line		
Parameters	Name	Type	Description
	<LineData>	Array	Array("NAME:LineData" _

			<pre>"Points:=", Array of points, _ "LineWidth:=", double, // the width of the line, in meters "Color:=", int, // the RGB value of the line color "Id:=", int), // [Opt=New id] Id for this item</pre>
	<Attributes>	Array	<pre>Array("NAME:Attributes", _ "Page:=", int) _ // [Opt=1] Page number (one-based) [out,retval] string id)</pre>
Return Value	<p>String Unique id</p>		

Line is created and the returned value has the format SchObj@<schematicID>

The Schematic ID can be found on the **Symbol** tab of the **Properties** window when you select the line.



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>

CreatePolygon (Schematic Editor)

Creates a polygon.

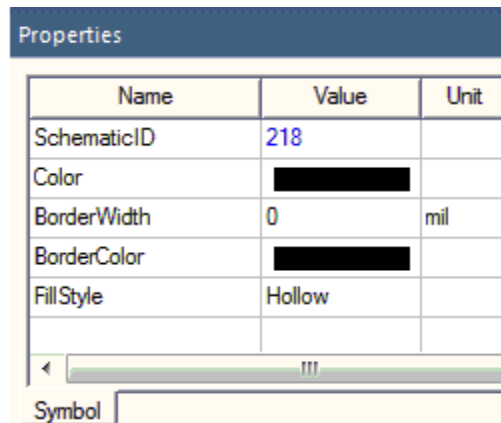
UI Access	Draw > Primitive > Polygon.		
Parameters	Name	Type	Description
	<PolygonData>	Array	Array("NAME:PolygonData" _ "Points:=", Array of points, _ "LineWidth:=", double, // the width of the Polygon border, in meters "Bordercolor:=", int, // the RGB value of the border color "Fill:=", int, // the 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 "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		

Polygon is created and the return value has the format

SchObj@<schematicID>

The Schematic ID can be found in the **Properties** window on the **Symbol** tab when you select the polygon.



Python Syntax	CreatePolygon()
----------------------	-----------------

Python Example	<pre>oEditor.CreatePolygon(["NAME:PolygonData", "Points:=", [_ "(-0.058951, 0.006308)", "(-0.046142, 0.010480)", _ "(-0.046239, 0.001067)"), "LineWidth:=", _ 0, "BorderColor:=", 0, "Fill:=", 0, "Color:=", _ 0, "Id:=", 27], ["NAME:Attributes", "Page:=", 1])</pre>
-----------------------	--

CreateRectangle (Schematic Editor)

Create a rectangle

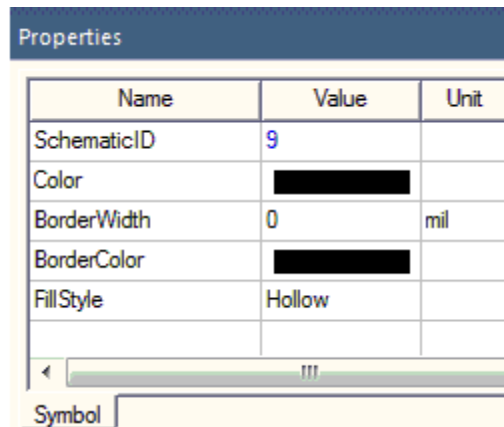
UI Access	Draw>Primitive>Rectangle								
Parameters	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><RectData></td> <td>Array</td> <td> Array("NAME:RectData" _ "x1:=", double, _ // X position of the upper left "y1:=", double, _ // Y position 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 </td> </tr> </tbody> </table>	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		
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 RBG value of the Rectangle 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		

Rectangle is created and the return value has the format

SchObj@<schematicID>

The Schematic ID can be found in the **Properties** window on the **Symbol** tab when you select the rectangle.



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 Editor)

Create text

UI Access	Draw>Primitive>Text								
Parameters	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><TextData></td> <td>Array</td> <td> 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 </td> </tr> </tbody> </table>	Name	Type	Description	<TextData>	Array	Array("NAME:TextData" _ "x:=", double, _ // X position of the object "y:=", double, _ // Y position of the object "Text:=", string, _ // Text to display "Color:=", int, // the 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		
Name	Type	Description							
<TextData>	Array	Array("NAME:TextData" _ "x:=", double, _ // X position of the object "y:=", double, _ // Y position of the object "Text:=", string, _ // Text to display "Color:=", int, // the 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							

			<p>"Y1:=", double, // the text rectangle upper Y value, in meters</p> <p>"X2:=", double, // the text rectangle right X value, in meters</p> <p>"Y2:=", double, // the text rectangle lower Y value, in meters</p> <p>"RectLineWidth:=", double, // the width of the rectangle border, in meters</p> <p>"RectBordercolor:=", int, // the RBG value of the rectangle border color</p> <p>"RectFill:=", int, // the rectangle fill pattern id, 0 = hollow, 1 = solid, 2 = NEDiagonal, 3 = OrthoCross, 4 = DiagCross, 5 = NWDiagonal, 6 = Horizontal, 7 = Vertical</p> <p>"RectColor:=", int, // the RBG value of the rectangle fill color</p>
	<Attributes>	Array	<p>Array("NAME:Attributes", _</p> <p>"Page:=", int) _ // [Opt=1] Page number (one-based)</p> <p>[out,retval] string id)</p>
Return Value	<p>String</p> <p>Unique id</p>		

Text is created and the return value has the format: SchObj@<SchematicID>

The Schematic ID can be found in the **Properties** window on the **Symbols** tab when you select the text.

Properties		
Name	Value	Unit
SchematicID	286	
Color		
Location	4100 , 2600	mil
Angle	0	deg
TextSize	12	
<div style="border: 1px solid black; padding: 2px;"> < </div>		
Symbol		

Python Syntax	CreateText()
Python Example	<pre>oEditor.CreateText (["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:=",</pre>

	0, "RectColor:=", 0], ["NAME:Attributes", "Page:=", 1])
--	---

Cut [EMIT]

Cut items in the schematic for pasting.

UI Access	Edit > Cut or Cut button on Schematic ribbon.
Parameters	< <i>elementNames</i> > – A list of component, wire, or primitive names to cut
Return Value	None.

Python Syntax	<code>Cut(<<i>elementNames</i>>, <<i>elementNames</i>>)</code>
Python Example	<pre>oEditor.Cut(["Radio", "Antenna"]) oEditor.Paste(-0.00127, 0.01651)</pre>

Delete [EMIT]

Delete items from the schematic.

UI Access	Edit>Delete or Delete button on Schematic ribbon.
Parameters	< <i>elementNames</i> > – A list of component, wire, or primitive names to delete.
Return Value	The name of the new component.

Python Syntax	Delete(<elementNames>, <elementNames>)
Python Example	<code>oEditor.Delete(["Radio", "Antenna"])</code>

DisableEmitComponents [EMIT]

Disables/enables specified EMIT component(s). This command only works for emitters and radios.

UI Access	Edit > Disable or Edit > Enable
Parameters	<componentNames> – String component name, or Array of strings containing the components to select. <Disable/Enable> – Boolean True to disable; False to enable.
Return Value	None.

Python Syntax	DisableEmitComponents(<componentNames>)
Python Example	<code>oEditor.DisableEmitComponents("Radio1", True)</code> <code>oEditor.DisableEmitComponents(["Radio1", "Radio2", "Radio3"], False)</code>

ExportImage (EMIT)

To export a picture for a specified page of the current design to a file. The image size can also be specified. The filename extension determines the type of image exported.

UI Access	None.		
Parameters	Name	Type	Description
	<filename>	String	The name of the file, with format-specific extension. Extensions supported are: bmp, gif, jpg, jpeg, png, tif, tiff.
	<dx>	Integer	The width of the image. If dx is less than 160, 160 will be used for the width.

	<dy>	Integer	The height of the image. If dy is less than 160, 160 will be used for the height.
Return Value	None		

Note:

For more information about the component name, schematic IDs, and formats, see the section Editor Scripting IDs.

Python Syntax	ExportImage (<filename>, <dx>, <dy>)
Python Example	<code>oEditor.ExportImage ("c:\mysch.png", 800, 400)</code>

ExportToLibrary [EMIT]

Export a component from the schematic to a new or existing library. If the component already exists in the specified library, update it.

UI Access	Right-click on a component in the schematic and select Export to Library .
Parameters	<p><componentName> – The name of the component to export.</p> <p><libraryType> – The type of library to create or update: “PersonalLib” or “UserLib”.</p> <p><libraryFile> – The relative path to the library files within the PersonalLib or UserLib directory. If a path to a sub-directory is specified, the directory must exist.</p>
Return Value	None.

Python Syntax	<code>ExportToLibrary(<componentName>, <libraryType>, <libraryFile>)</code>
Python Example	<code>oEditor.ExportToLibrary("Radio1", "PersonalLib", "Library1")</code>

GetAllComponents [EMIT]

Get a list of all components in the schematic.

UI Access	N/A
Parameters	None.
Return Value	A list of component names.

Python Syntax	<code>GetAllComponents()</code>
Python Example	<pre>for component_name in oEditor.GetAllComponents(): print(component_name)</pre>

GetAllElements [EMIT]

Get a list of all element (such as component, wire, or primitive) names in the schematic.

UI Access	N/A
Parameters	None.
Return Value	A list of element names.

Python Syntax	<code>GetAllComponents()</code>
----------------------	---------------------------------

Python Example	<pre>for component_name in oEditor.GetAllElements(): print(element_name)</pre>
-----------------------	--

GetComponentBoundingBox [EMIT]

Get the location and extents of a component within the schematic.

UI Access	N/A
Parameters	<componentName> – The name of the component in the schematic.
Return Value	A list containing a pair of points that specify the bottom-left corner of the component and the component width and height: <i>[[xBottom, yBottom], [width, height]]</i>

Python Syntax	GetComponentBoundingBox(<componentName>)
Python Example	<code>oEditor.GetComponentBoundingBox("Radio")</code>

GetComponentLocation [EMIT]

Get the location of a component within the schematic.

UI Access	N/A
Parameters	<componentName> – The name of the component in the schematic.
Return Value	A list containing the x and y location of the component. The location returned is the component origin, typically the center of the component. <i>[xLocation, yLocation]</i>

Python Syntax	<code>GetComponentLocation(<componentName>)</code>
Python Example	<code>oEditor.GetComponentLocation("Radio")</code>

GetComponentOrientation [EMIT]

Get the orientation of a component within the schematic.

UI Access	N/A
Parameters	<componentName> – The name of the component in the schematic.
Return Value	An integer describing the orientation state of the component. The first orientation state is 0 and this corresponds to the default orientation of the component – the orientation when it is first placed in the schematic. The 'Reorient' button in the Schematic ribbon cycles through these states.

Python Syntax	<code>GetComponentOrientation(<componentName>)</code>
Python Example	<code>oEditor.GetComponentOrientation("Multiplexer")</code>

GetComponentPortLocation [EMIT]

Get the location of a component port within the schematic.

UI Access	N/A
Parameters	<componentName> – The name of the component in the schematic. <portName> – The name of the component port.
Return Value	A list containing the x and y location of the component port (that is., the point where wiring connections can be made). [<i>xLocation, yLocation</i>]

Python Syntax	<code>GetComponentPortLocation(<componentName>, <portName>)</code>
Python Example	<code>oEditor.GetComponentPortLocation("Cable", "n2")</code>

GetComponentPorts [EMIT]

Get the names of all ports for a component within the schematic.

UI Access	N/A
Parameters	<code><componentName></code> – The name of the component in the schematic.
Return Value	A list containing the names of all component ports. [<code>"n1"</code> , <code>"n2"</code>]

Python Syntax	<code>GetComponentPorts(<componentName>)</code>
Python Example	<code>oEditor.GetComponentPorts("Cable")</code>

GetEditorName (Schematic Editor)

Get the name of the schematic editor.

UI Access	N/A
Parameters	None
Return Value	String containing the name of the schematic editor. "SchematicEditor"

Python Syntax	GetEditorName()
Python Example	<code>oEditor.GetEditorName()</code>

GetNumComponentOrientations [EMIT]

Get the number of possible orientations for a component within the schematic.

UI Access	N/A
Parameters	<i><componentName></i> – The name of the component in the schematic.
Return Value	An integer describing the number of possible orientation states of the component. The ‘Reorient’ button in the Schematic ribbon cycles through these states. Some components cannot be re-oriented and thus have only 1 possible orientation state.

Python Syntax	GetNumComponentOrientations(<i><componentName></i>)
Python Example	<code>oEditor.GetNumComponentOrientations("Multiplexer")</code>

GetPropertyValue [EMIT]

Get the value of a property for one of the primitives (for example. Line, Rectangle, etc.) in the schematic.

UI Access	Properties in the Property Window.
Parameters	<p><i><tabArg></i> – The structure specifying the tab, item, and property to be changed and the new value(s) to apply.</p> <p><i><serverArg></i> – The name of the element (see GetAllElements and GetSelections). Only primitive-type elements (for example., Rectangle, Line) may be used. Components are not supported.</p> <p><i><propArg></i> – The name of the property.</p>

Return Value	The current value for the property. The return type varies depending on the type of property.
---------------------	---

Python Syntax	<code>GetPropertyValue(<tabArg>, <serverArg>, <propArg>)</code>
Python Example	<pre>line = self.oEditor.GetSelections()[0] oEditor.GetPropertyValue("BaseElementTab", line, "LineWidth")</pre>

GetSelections [EMIT]

Get a list of all selected element (component, wire, or primitive) names in the schematic.

UI Access	NA.
Parameters	None.
Return Value	A list of selected element names.

Python Syntax	<code>GetSelections()</code>
Python Example	<pre>for element_name in oEditor.GetSelections(): print(element_name)</pre>

GetWireAtPort [EMIT]

Get the name of the wire connected to a component port within the schematic.

UI Access	N/A.
Parameters	<i><componentName></i> – The name of the component in the schematic. <i><portName></i> – The name of the component port
Return Value	The name of the wire connected to the component port (that is, the point where wiring connections can be made). An empty string ("") is returned if the port has no wire connected to it. .

Python Syntax	<code>GetWireAtPort<componentName>, <portName></code>)
Python Example	<code>oEditor.GetWireAtPort("Cable", "n2")</code>

GetWireConnections [EMIT]

Get the name of the components and ports connected by the specified wire.

UI Access	N/A.
Parameters	<i><wireElementName></i> – The name of the wire in the schematic. See <code>GetAllElements</code> , <code>GetSelections</code> , and <code>GetWireAtPort</code> for getting wire names.
Return Value	A length-two list of component name, port name pairs. <code>[["Radio", "n1"], ["Cable", "n1"]]</code> .

Python Syntax	<code>GetWireConnections(<wireElementName>)</code>
Python Example	<code>oEditor.GetWireConnections("Wire@net_1;6")</code>

GetZoomArea [EMIT]

Get the location and extents of the current zoom area (visible portion) of the schematic.

UI Access	N/A.
Parameters	None.
Return Value	A list containing a pair of points that specify the bottom-left corner of the zoom area and the zoom area width and height: <code>[[xBottom, yBottom], [width, height]]</code> .

Python Syntax	<code>GetZoomArea()</code>
Python Example	<code>oEditor.GetZoomArea()</code>

IsEmitComponentDisabled [EMIT]

Checks whether a radio or emitter is disabled.

UI Access	N/A
Parameters	<code><componentNames></code> – String name of component to check. Disable/enable only works for emitters and radios.
Return Value	BOOL True if component is disabled; False if component is enabled.

Python Syntax	<code>IsEmitComponentDisabled(<componentNames>)</code>
Python Example	<code>oEditor.IsEmitComponentDisabled("Radio")</code>

Move [EMIT]

Move items in the schematic by a specified amount.

UI Access	Drag selected items within the schematic.
Parameters	<p><elementNames> – A list of elements to move. See GetSelected and GetAllElements for getting names of schematic items.</p> <p><xDelta> – The distance to move the items in the x direction.</p> <p><yDelta> – The distance to move the items in the y direction.</p>
Return Value	None.

Python Syntax	Move([<elementNames>], <xDelta>, <yDelta>)
Python Example	oEditor.Move(["Radio", "Antenna"], -0.1524, 0.0)

Pan [EMIT]

Adjust the current zoom area (visible region) by the specified amount

UI Access	View>Pan or Pan on the View ribbon.
Parameters	<p><xDelta> – The distance to move the view in the x direction.</p> <p><yDelta> – The distance to move the view in the y direction.</p>
Return Value	None.

Python Syntax	Pan(<xDelta>, <yDelta>)
Python Example	oEditor.Pan(-0.1524, 0.0)

Paste [EMIT]

Paste cut or copied items from the schematic at a specified location

UI Access	Edit>Paste or Paste in the Schematic ribbon.
Parameters	< <i>xPosition</i> > – The x location to paste the items. < <i>yPosition</i> > – <i>The y location to paste the items..</i>
Return Value	None.

Python Syntax	<code>Paste(<xPosition>, <yPosition>)</code>
Python Example	<code>oEditor.Paste(-0.00127, 0.01651)</code>

PlaceComponent [EMIT]

Place the specified component in a new location, such that it is connected to the next suitable port or in a new row at the bottom of the schematic.

UI Access	Reposition in the Schematic ribbon
Parameters	< <i>componentName</i> > – The component to place at the next open port or new row at the bottom of the schematic. If an empty string (“”) is passed, the selected component is used. < <i>componentName</i> > – The component to be connected to the first one, if possible; this specifies the component to attach to instead of attaching to the first component it finds. Optional.
Return Value	None.

Python Syntax	<code>PlaceComponent(<componentName>, <componentName>)</code>
Python Example	<code>oEditor.PlaceComponent("Antenna1", "Radio 5")</code>

RenameComponent [EMIT]

Rename the specified component.

UI Access	Properties Window property or name label below the component in the schematic.
Parameters	<p><code><oldName></code> – The name of the component to be renamed.</p> <p><code><newName></code> – The new name for the component.</p>
Return Value	None.

Python Syntax	<code>RenameComponent(<oldName>, <newName>)</code>
Python Example	<code>oEditor.RenameComponent("Antenna", "New Antenna Name")</code>

ReorientComponent [EMIT]

Set the orientation of a component within the schematic.

UI Access	Reorient button in the Schematic ribbon.
Parameters	<p><code><componentName></code> – The name of the component in the schematic.</p> <p><code><orientation></code> – An integer specifying the orientation state to apply to the component. The first orientation state is 0 and this corresponds to the default orientation of the component – the orientation when it is first placed in the schematic. The Reorient button in the Schematic ribbon cycles through these states. See <code>GetNumComponentOrientations</code> for valid values.</p>
Return Value	None.

Python Syntax	<code>ReorientComponent(<componentName>, <orientation>)</code>
Python Example	<code>oEditor.ReorientComponent("Multiplexer", 1)</code>

Select [EMIT]

Select the specified element in the schematic.

UI Access	Click or Shift+Click on an element in the schematic.
Parameters	<code><elementName></code> – The name of the element in the schematic to select. <code><clearExistingSelection></code> – If True, de-select any existing selections before selecting the specified element. If False, preserve the existing selection(s) and additionally select the specified element.
Return Value	None.

Python Syntax	<code>Select(<elementName>, <clearExistingSelection>)</code>
Python Example	<code>oEditor.Select("Multiplexer", True) # Select a multiplexer only</code> <code>oEditor.Select("Radio", False) # Also select a Radio</code>

SelectAll [EMIT]

Select all elements in the schematic.

UI Access	Edit>Select All.
------------------	----------------------------

Parameters	None.
Return Value	None.

Python Syntax	SelectAll()
Python Example	<code>oEditor.SelectAll()</code>

SendToBack(Schematic Editor)

Send the selected object to the back of the view

UI Access	Draw>Send To Back		
Parameters	Name	Type	Description
	<Object>	String	Object to send to the back.
Return Value	None		

Note:

For more information about the component name, schematic IDs, and formats, see the section [Editor Scripting IDs](#)

Python Syntax	<code>SendToBack(["NAME:Selections", "Selections:=", [<Object>, <Object>, ...]])</code>
Python Example	<code>oDefinitionEditor.SendToBack(["NAME:Selections", "Selections:=", ["SchObj@10"]])</code>

SetPropertyValue [EMIT]

Set the value of a property for one of the primitives (such as Line, Rectangle) in the Schematic.

UI Access	Properties in the Property Window.
Parameters	<p><tabArg> – The structure specifying the tab, item, and property to be changed and the new value(s) to apply.</p> <p><serverArg> – The name of the element (see GetAllElements and GetSelections). Only primitive-type elements (such as Rectangle, Line) may be used. Components are not supported.</p> <p><propArg> – The name of the property.</p> <p><valueArg> – The new value for the property.</p>
Return Value	None.

Python Syntax	SetPropertyValue(<tabArg>, <serverArg>, <propArg>, <valueArg>)
Python Example	<pre>line = self.oEditor.GetSelections()[0] oEditor.SetPropertyValue("BaseElementTab", line, "LineWidth", 0.001)</pre>

Wire [EMIT]

Create a wire between the specified ports of two components in the schematic.

UI Access	Click a port and then click a second port to create a wire. Click at additional intermediate points to customize the wire route.
Parameters	<component1> – The name of the first component to connect with a wire.

	<p><i><port1></i> – The port of the first component where one end of the new wire should connect.</p> <p><i><component2></i> – The name of the second component to connect with a wire.</p> <p><i><port2></i> – The port of the second component where the other end of the new wire should connect.</p> <p><i><segmentPointList></i> – An empty list ([]) if the wire should connect directly from one port to the other (ports must be aligned horizontally). Any bends in the wire are specified by including the points (x,y location of the bend) in this list.</p>
Return Value	The name of the new wire.

Python Syntax	<code>Wire(<component1>, <port1>, <component2>, <port2>, [(<segmentPointList>)])</code>
Python Example	<pre>oEditor.Wire("Radio", "n1", "Cable", "n1") oEditor.Wire("Antenna", "in", "Cable", "n2", [[0.01524, -0.01016]])</pre>

ZoomArea [EMIT]

Set the location and extents of the current zoom area (visible portion) of the schematic.

UI Access	View>Zoom Area or the Zoom Area button on the View ribbon.
Parameters	<p><i><pt1></i> – A list specifying the lower left corner of the zoom area ([left, bottom]).</p> <p><i><ptDelta></i> – A list [width, height] specifying the size of the zoom area. If the width and height do not match the aspect ratio of the view area, the view area will be configured to be at least as large as both values.</p>
Return Value	None.

Python Syntax	<code>ZoomArea([<pt1>], [<ptDelta>])</code>
----------------------	---

Python Example	<code>oEditor.ZoomArea([-0.0480, -0.0356], [0.0864, 0.0489])</code>
-----------------------	---

ZoomIn [EMIT]

Zoom in at the specified location in the schematic.

UI Access	View>Zoom In or the Zoom In button on the View ribbon.
Parameters	<i><pt1></i> – The center point of the new zoom area. Optional – if not specified, the center of the view area is used. <i><zoomFactor></i> – The factor by which the view area should be reduced (a value greater than 1).
Return Value	None.

Python Syntax	<code>ZoomIn([<pt1>], <zoomFactor>)</code>
Python Example	<code>oEditor.ZoomIn()</code> <code>oEditor.ZoomIn([-0.0480, -0.0356], 1.1)</code>

ZoomOut [EMIT]

Zoom out around the specified location in the schematic.

UI Access	View>Zoom Out or the Zoom Out button on the View ribbon.
Parameters	<i><pt1></i> – The center point of the new zoom area. Optional – if not specified, the center of the view area is used. <i><zoomFactor></i> – The factor by which the view area should be increased (a value greater than 1).
Return Value	None.

Python Syntax	<code>ZoomOut([<pt1>], <zoomFactor>)</code>
Python Example	<code>oEditor.ZoomArea([-0.0480, -0.0356], [0.0864, 0.0489])</code> <code>oEditor.ZoomOut()</code>

ZoomPrevious [EMIT]

Zoom to the last view area, prior to the most recent zoom event.

UI Access	View>Zoom Previous or the Zoom Previous button on the View ribbon.
Parameters	None.
Return Value	None.

Python Syntax	<code>ZoomPrevious()</code>
Python Example	<code>oEditor.ZoomPrevious()</code>

ZoomToFit [EMIT]

Zoom so that the view area contains all elements in the schematic with a small margin around the edges.

UI Access	View>Fit All or the Fit All button on the View ribbon.
Parameters	None.
Return Value	None.

Python Syntax	ZoomToFit()
Python Example	<code>oEditor.ZoomToFit()</code>

Index

#

3D Modeler editor commands

CreateEllipse 10-13

GetProperties 6-15

GetPropertyValue 6-16

A

AddDataset 5-4

AddLink 7-3

AddMessage 3-5

AddResult 7-3

Ansoft Application Object commands 2-1

GetAppDesktop 2-2

C

ChangeProperty 10-4

ClearMessages 3-6, 5-7

ClearSelections 10-5

Close 5-8

CloseAllWindows 3-10

CloseProject 3-10

CloseProjectNoForce 3-11

CloseResultWindow 7-4

conventions

scripting help 1-1

Copy 10-5

CopyDesign 5-8

Copyright and Trademark Information 2

Core Global Script Context Commands 8-1

Count 3-12

CPython 1-39

CreateComponent 10-10

CreateComponentAt 10-11

CreateEllipse 10-13

Cut 10-24

CutDesign 5-9

D

dataset commands

AddDataset 5-4

DeleteDataset 5-10

EditDataset 5-11

ImportDataset 5-22

Delete 10-24

DeleteAllResults 7-4

DeleteDataset 5-10

DeleteDesign 5-10, 7-5

DeleteLink 7-5

DeleteProject 3-12

- DeleteResult 7-6
- Design object commands
 - CloseAllWindows 3-10
 - GetLibraryDirectory 3-22
 - GetName 7-13
 - GetProjectDirectory 3-26
 - GetProperties 6-15
 - GetPropertyValue 6-16
 - GetRegistryInt 3-55
 - GetRegistryString 3-56
 - GetTempDirectory 3-29
 - GetVariables 6-18
 - GetVariableValue 6-18
 - GetVersion 3-30
 - SetLibraryDirectory 3-49
 - SetProjectDirectoryVBCommand> 3-49
 - SetPropertyValue 6-19
 - SetTempDirectory 3-50
 - SetVariableValue 6-20
 - Solve 7-22
 - ValidateDesign 5-36
- Design Object Script Commands 7-1
- Desktop Commands For Registry Values 3-53
- Desktop object commands
 - AddMessage 3-5
 - ClearMessages 3-6
 - CloseProject 3-10
 - CloseProjectNoForce 3-11
 - Count 3-12
 - DeleteProject 3-12
 - DownloadJobResults 3-13
 - EnableAutosave 3-14
 - ExportOptionsFiles 3-14
 - GetActiveProject 3-15
 - GetAutosaveEnabled 3-16
 - GetBuildTimeDateString 3-16
 - GetChildNames 5-14
 - GetChildTypes 5-15
 - GetDesigns 3-19
 - GetDistributedAnalysisMachines 3-19
 - GetDistributedAnalysisMachinesForDesignType 3-20
 - GetName 3-24
 - GetProjectList 3-27
 - GetProjects 3-23, 3-26
 - GetPropNames 5-20
 - LaunchJobMonitor 3-33
 - NewProject 3-34
 - OpenMultipleProjects 3-35
 - OpenProject 3-36
 - PauseRecording 3-37
 - PauseScript 3-38
 - Print 3-38
 - QuitApplication 3-39
 - RefreshJobMonitor 3-39

ResetLogging 3-40
 RestoreWindow 3-41
 ResumeRecording 3-42
 RunProgram 3-43
 RunScript 3-44
 SelectScheduler 3-46
 SetActiveProject 3-48
 SetActiveProjectByPath 3-48
 Sleep 3-51
 SubmitJob 3-52
 TileWindows 3-53
 Desktop Object Script
 Commands 3-1
 Desktop Scripting Conventions 1-
 51
 DisableEmitComponents 10-25

E

EditComponentNodes 7-6
 EditDataset 5-11
 EditNotes 7-8
 Editor commands
 ClearSelections 10-5
 Editor object commands
 SetPropertyValue 6-19
 Editor Scripting IDs 10-3
 EnableAutoSave 3-14
 ExportOptionsFiles 3-14
 ExportToLibrary 10-26

G

GetActiveDesign 5-13
 GetActiveProject 3-15
 GetAllComponents 10-27
 GetAllElements 10-27
 GetArrayVariables 6-15
 GetAutoSaveEnabled 3-16
 GetAvailableLinkNames 7-8
 GetBuildDateTimeString 3-16
 GetChildNames 5-14
 GetChildObject Design 5-15
 GetChildTypes 5-15
 GetComponentBoundingBox 10-28
 GetComponentLocation 10-28
 GetComponentNodeNames 7-9
 GetComponentNodeProperties 7-9
 GetComponentOrientation 10-29
 GetComponentPortLocation 10-29
 GetComponentPorts 10-30
 GetComponentWarnings 7-10
 GetCouplingWarnings 7-10
 GetCurrentResult 7-11
 GetDefinitionManager 5-16
 GetDependentFiles 5-17
 GetDesigns 3-19
 GetDesignType 7-11
 GetDistributedAnalysisMachines 3-
 19

GetDis-trib-utedAnalysisMachinesForDesignType 3-20

GetEditorName [Schematic] 10-30

GetExeDir 3-21

GetLibraryDirectory 3-22

GetLinkNames 7-12

GetName 3-24, 5-19, 7-13

GetNumComponentOrientations 10-31

GetPath 5-19

GetPersonalLibDirectory 3-24

GetProcessID 3-25

GetProjectDirectory 3-26

GetProjectList 3-27

GetProjects 3-23, 3-26

GetProperties 6-15

GetPropertyValue 6-16, 10-31

GetPropNames 5-20

GetPropValue Project 5-20

GetRadioNames 7-13

GetRegistryInt 3-55

GetRegistryString 3-56

GetResultList 7-14

GetResultNotes 7-15

GetResultProperties 7-15

GetRevision 7-16

GetSelections 10-32

GetSysLibDirectory 3-28

GetTempDirectory 3-29

GetTopDesignList 5-21

GetUserLibDirectory 3-29

GetVariables 6-18

GetVariableValue 6-18

GetVersion 3-30

GetWireAtPort 10-32

GetWireConnections 10-33

GetZoomArea 10-33

H

hierarchy of variables in HFSS 1-43

I

ImportANF 3-30

ImportAutoCAD 3-31

ImportDataset 5-22

ImportGDSII 3-32

ImportODB 3-32

InsertDesignWithWorkflow 5-24

IronPython 1-2

IsEmitComponentDisabled 10-34

L

LaunchJobMonitor 3-33

Library Management Script Commands 9-1

M

modules in HFSS scripting 1-43

Move 10-34

N

Named Arguments 1-51

NewProject 3-34

O

oAnsoftApp object 1-43

oDesign object 1-43

oDesktop object 1-43

oEditor object 1-43

oModule object 1-43

OpenAndConvertProject 3-34

OpenMultipleProjects 3-35

OpenProject 3-36

oProject object 1-43

P

Pan 10-35

Paste 5-26, 10-36

Paste (Project Object) 5-26

PauseRecording 3-37

PauseScript 3-38

PlaceComponent 10-36

Print 3-38

Project object commands

 AddDataset 5-4

 Close 5-8

 CopyDesign 5-8

 CutDesign 5-9

 DeleteDataset 5-10

 DeleteDesign 5-10, 7-5

 EditDataset 5-11, 5-21

 GetActiveDesign 5-13

 GetArrayVariables 6-15

 GetChildObject Design 5-15

 GetDependentFiles 5-17

 GetDesignType 7-11

 GetName 5-19

 GetPath 5-19

 GetProperties 6-15

 GetPropertyValue 6-16

 GetPropvalue Project 5-20

 GetTopDesignList 5-21

 GetVariables 6-18

 GetVariableValue 6-18

 Paste 5-26, 5-26

 Redo 5-26

 Rename 5-27

 RestoreProjectArchive 3-41, 5-27

 Save 5-28

 SaveAs 5-29

 SaveAsStandAloneProject 5-31

 SaveProjectArchive 5-31

 SetActiveDesign 5-33

 SetPropertyValue 6-19

 SetPropValue Project 5-33

 SetVariableValue 6-20

 SimulateAll 5-6, 5-34

 Undo 5-35

- UpdateDefinitions 5-35
- Project Object Script Commands 5-1
 - property commands
 - GetArrayVariables 6-15
 - GetProperties 6-15
 - GetPropertyValue 6-16
 - GetVariables 6-18
 - GetVariableValue 6-18
 - SetPropertyValue 6-19
 - SetVariableValue 6-20
 - Property Script Commands 6-1
 - Conventions uSed in thie Chapter 6-12
 - Object Script Property Function Summary 6-3
- Q**
- QuitApplication 3-39
- R**
- Redo 7-16
 - project-level command 5-26
- RefreshJobMonitor 3-39
- Rename 5-27
- RenameComponent 10-37
- RenameDesign 7-17
- RenameResult 7-17
- ReorientComponent 10-37
- ResetLogging 3-40
- RestoreWindow 3-41
- ResumeRecording 3-42
- Running Instance Manager Script Commands 4-1
- RunProgram 3-43
- RunScript 3-44
- RunScriptWithArguments 3-45
- RunToolkit 7-18
- S**
- Save 5-28
- SaveAs 5-29
- SaveAsStandAloneProject 5-31
- Schematic Scripting 10-1
- scripts
 - CPython 1-39
 - IronPython 1-2
 - overview 1-1
 - pausing 1-47
 - recording 1-48
 - running 1-46
 - stopping 1-47
- Select 10-38
- SelectAll 10-38
- SelectScheduler 3-13, 3-46
- SendToBack 10-39
- SetActiveDefinitionEditor 5-32
- SetActiveDesign 5-33
- SetActiveEditor 7-18
- SetActiveProject 3-48

SetActiveProjectByPath 3-48

SetLibraryDirectory 3-49

SetProjectDirectory 3-49

SetPropertyValue 6-19, 10-40

SetPropValue Project 5-33

SetResultNotes 7-19

SetTempDirectory 3-50

Setting Numerical Values 1-53

SetVariableValue 6-20

SGetAppDesktop 2-1

ShowAnalysisAndResults 7-19

ShowCouplingDialog 7-20

ShowEditDialog 7-20

ShowResultWindow 7-21

SimulateAll 5-6, 5-34

Sleep 3-51

Structures 1-30

SubmitJob 3-52

T

TransferToEmit 7-22

U

UDM Structures 1-30

Undo

 project-level command 5-35

UpdateDefinitions, project-level command 5-35

UpdateLink 7-22

V

ValidateDesign 5-36

variables

 hierarchy in HFSS 1-43

 used in HFSS scripts 1-43

W

Wire 10-40

Z

ZoomArea 10-41

ZoomIn 10-42

ZoomOut 10-42

ZoomPrevious 10-43

ZoomToFit 10-43