Ansys

2025 R2

POWERING INNOVATION THAT DRIVES HUMAN ADVANCEMENT

Mechanical Scripting Guide

Ansys

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 companies.

Copyright and Trademark Information

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-PartySoftware

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

Table of Contents

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/).

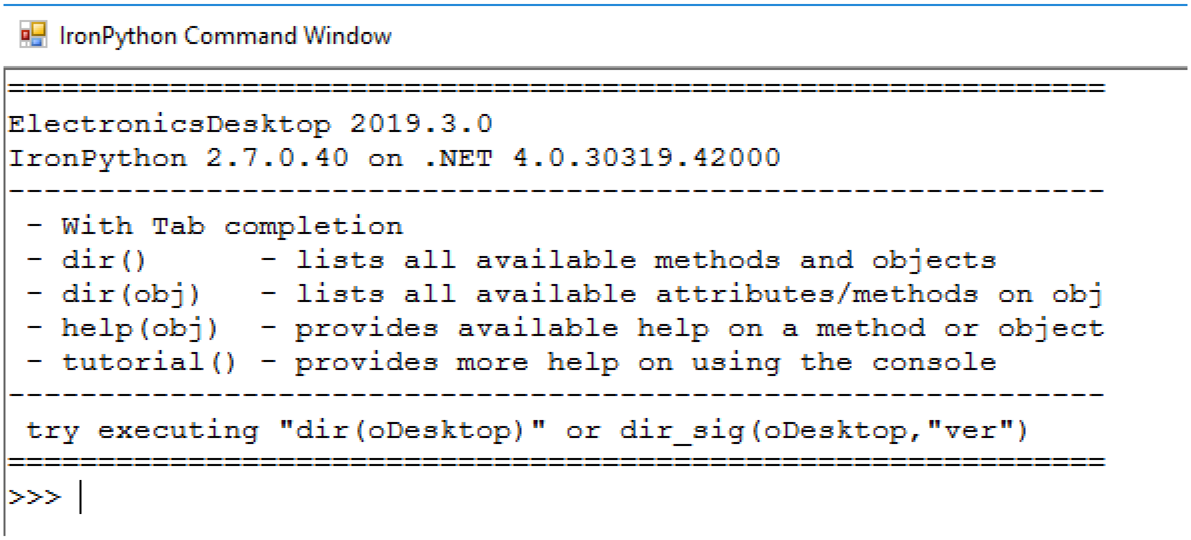
Advantages of IronPython

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**.



```
==============================================================
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]

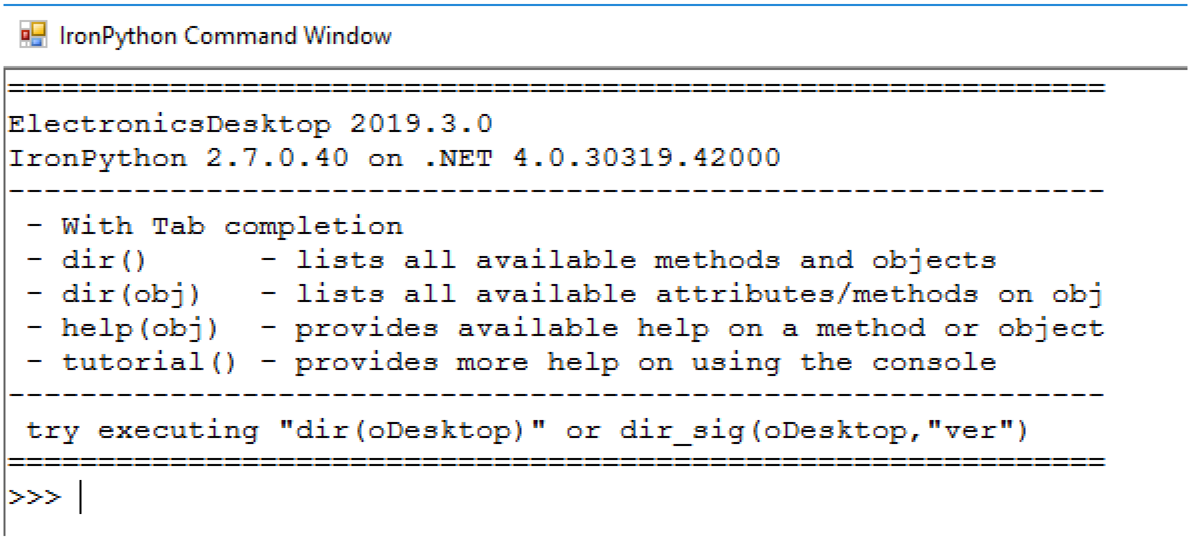
[Introduction to IronPython]

[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:**



```
==============================================================
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")
==============================================================
>>> |
```

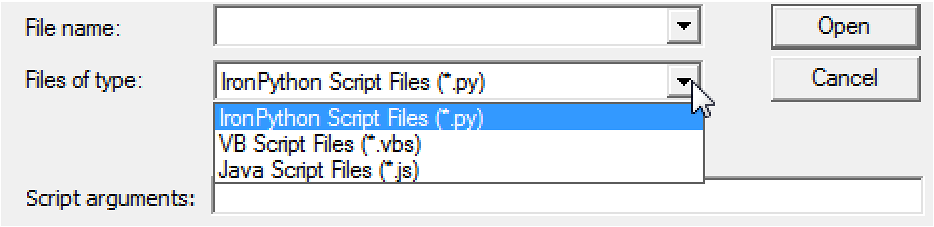
- **Tools** > **Run Script** menu item, select "IronPython" from the file type drop-down list.
- Launch the product with a script argument.
- Register an IronPython script as an external tool using the **Tools** > **External Tools** menu item.

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.



File name:

Files of type: IronPython Script Files (*.py)

Script 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)", "Transparency:=", 0, _
 "PartCoordinateSystem:=", "Global", "UDMId:=", "", "MaterialValue:=", _
 "" & 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\AnsysEM\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 Kis-
ses","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 cmposed of multiple objects
pasteName = oEditor.Paste()[0]

# now move the pasted item to it's final position
oEditor.Move(
  {
    "NAME" : "Selections",
    "Selections" : pasteName
    },
    {
    "NAME" : "TransalateParameters",
    "CoordinateSystemID" : -1,
    "TranslateVectorX" : "%d * 3 * %s" % (x, coneBotRad),
    "TranslateVectorY" : "%d * 3 * %s" % (y, coneBotRad),
    "TranslateVectorZ" : "0mm"
    }
)

# Now fit the display to the created grid
oEditor.FitAll()
```

Related Topics

Introduction to IronPython

Scripting Using Iron Python: Putting it all Together

Creating User Defined Primitives and User Defined Models in Python Scripts

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

Advantages Compared to C++

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

Changes compared to C

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

Structures

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

For a complete list of structures and examples please see UDP/UDM Structures.

Return Values for UDM and UDP Functions

For information on return values for each UDM and UDP function, see the Return Values section.

Constants

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

```
# Example of specifying the LengthUnit enum by qualifying it

# with the type of the enum: UnitType

unitType = UnitType.LengthUnit
```

For a complete list of enum constants please see UDP/UDM Constants.

Methods

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

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

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

Output Parameters

The rule for the output parameters is as follows:

- If the function has one output parameter variable and no return value, the variable will become function's return value.The same will happen if the return value is a 'success/failure' boolean ('None' will be returned on failure and parameter variable - on success).

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

```
# one output parameter; return value is ignored

udmDefinition = udmFunctionLibrary.GetDefinition()


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

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

# Two output parameter; return value must be preserved

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

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

Comparison with C function:

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

'List Size' Parameters

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

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

Example:

```
# input list, input list size

lret = udpFunctionLibrary.Unite(objectIds)


# output list, output list size

faceIdList = udmFunctionLibrary.GetAllFaces(PartId)


# output list, input list size

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

Comparison with C function:

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

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* faceIds, 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, faceIds) (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
UDPCoordinateSystemAxis axis, double angle, int nClones, long* nClones, void* callbackData);	(callbackData is omitted cloneIdsSize is a new parameter)

Developing a UDM/UDP

Creation

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

Location

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

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

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

Organize

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

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

For UDM:extension:

```
from UDM import *
```

For UDP:extension:

```
from UDP import *
```

Edit/Reload

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

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

UDPExtension

Import

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

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

```
from UDP import *
```

Main class: UDPExtension

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

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

The class should implement IUDPextension methods described in the topic that follows.

IUDPExtension Methods

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

Mandatory Methods

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

GetLengthParameterUnits()

- returns string.

GetPrimitiveTypeInfo()

- returns UDPPrimitiveTypeInfo.

GetPrimitiveParametersDefinition2()

- returns a list of UDPPrimitiveParameterDefinition2 or None on failure

AreParameterValuesValid2(errorMsg, udpParams)

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

CreatePrimitive2(funcLib, udpParams)

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

Optional Methods

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

GetPrimitiveParameters()

- returns Python list of strings or NULL

GetRegisteredFaceNames()

- returns Python list of strings or NULL

GetRegisteredEdgeNames()

- returns Python list of strings or NULL

GetRegisteredVertexNames()

- returns Python list of strings or NULL

ProjectParametersOnToValidPlane2(currentUDPParams, projectedUDPParams)

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

MapParametersDefinitionVersions2(oldVersion, oldUDPParams)

- oldVersion is a string
- oldUDPParamsis 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, udmInParams, updatedParams, refreshModifiedPartsOnly, nonEditedPartRefIds)

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

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

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

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

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

returns True on success, False on failure.

ReleaseInstance(instanceId)

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

GetAttribNameForEntityId()

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

GetAttribNameForPartId()

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

Optional Methods

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

DialogForDefinitionOptionsAndParams(self, defData, optData, params):

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

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

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

DialogForDefinitionAndOptions(self, defData, optData) [Deprecated]

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

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

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

30. ***bool***: SI = **Translate(*int***: partID, ***UDPVector***: translateVector**)**
31. ***bool***: SI = **Rotate(*int***: partID, ***CoordinateSystemAxis***: whichAxis, ***double***: rotateAngle**)**
32. ***bool***: SI = **Mirror(*int***: partID, ***UDPPosition***: mirrorPlaneBasePosition, ***UDPVector***: mirrorPlaneNormalVector**)**
33. ***bool***: SI = **Transform(*int***: partID, ***List_of_double***: rotationMatrix, ***UDPVector***: translateVector**)**
34. ***bool***: SI = **Scale(*int***: partID, ***double***: xScale, ***double***v: 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(*UDPProgress***: progress**)**

UDP/UDM Structures and Constants

The following sections describe:

- UDP/UDM Structures
- UDP/UDM Constants

UDP/UDM Structures

Differences compared to C API:

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

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

  Example (creating UDMParameter list):

```
udmParamList = [

  UDMParameter(

    "cubeSizeName", UnitType.LengthUnit,
      UDPParam(ParamDataType.Double, cubeSize),
      ParamPropType.Value,
      ParamPropFlag.MustBeReal),
    UDMParameter(
      "cubeDistanceName", UnitType.LengthUnit,
      UDPParam(ParamDataType.Double, cubeDistance),
      ParamPropType.Value,
      ParamPropFlag.MustBeReal),
    UDMParameter("numCubesName", UnitType.LengthUnit,
      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, trans-
lationVector)
```

List of Structures

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

Example:

```
pos1 = UDPPosition(1,2,3)

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

pos2.Z = pos1.Z

udpParam = UDPParam(ParamDataType.Double,1)

value = udpParam.Data
```

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

UDPPrimitiveParameterDefinition2	UDPPrimitiveParameterDefinition2(string name, string description, UnitType unitType, ParamPropType propType, ParamPropFlag propFlag, UDPParam defaultValue)	string Name string Description UnitType UnitType ParamPropType PropType ParamPropFlag PropFlag UDPParam DefaultValue
UDPPosition	UDPPosition(double x, double y, double z)	double X double Y double Z
UDPVector	UDPVector(double x, double y, double z)	double X double Y double Z
UDPSweepOptions	UDPSweepOptions(SweepDraftType draftType, double draftAngle, double twistAngle)	SweepDraftType DraftType double DraftAngle double TwistAngle
UDPPolylineSegmentDefinition	UDPPolylineSegmentDefinition(	PolylineSegmentType SegmentType

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,	string FunctionXuv string FunctionYuv string FunctionZuv double UStart double UEnd double VStart double VEnd

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

Structure	Construction	Members

	int noOfVertices;
	int* listOfVertices;
	)
		int* listOfEdges; /* edges to be blended */
		int noOfVertices;
		int* listOfVertices; /* vertices to be blended */

UDPBLNDFilletOptions	UDPBLNDFilletOptions(bool supressFillet, BLNDFilletRadiusLaw filletRadiusLaw, double filletStartRadius, double filletEndRadius, bool followSmoothEdgeSequence, BLNDFilletType filletType, double setbackDistance, double bulgeFactor)	bool SupressFillet /* Reserved for future */ BLNDFilletRadiusLaw FilletRadiusLaw double FilletStartRadius double FilletEndRadius bool FollowSmoothEdgeSequence /* Reserved for future */ BLNDFilletType FilletType double SetbackDistance double BulgeFactor /* Reserved for future */
UDPBLNDChamferOptions	UDPBLNDChamferOptions(bool supressChamfer, BLNDChamferRangeLaw chamferRangeLaw, double chamferLeftRange, double chamferRightRange)	bool SupressChamfer BLNDChamferRangeLaw ChamferRangeLaw double ChamferLeftRange double ChamferRightRange
UDPProgress	UDPProgress(int prog, int subProg, string mesg, string subMesg)	int Prog int SubProg string Mesg string SubMesg
UDMInfo	UDMInfo(string name, string purpose,	string Name string Purpose string Company

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

UDP Python Example

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

```
import sys

primitive_info = UDPPrimitiveTypeInfo(

  name="Fillet_Chamfer",

  purpose="Fillet Chamfer Example",

  company="Ansys",

  date="09/11/2020",

  version="1.0")

primitive_param_definitions = [

  UDPPrimitiveParameterDefinition2(

    "x_size",

    "",

    UnitType.LengthUnit,

    ParamPropType.Value,

    ParamPropFlag.MustBeReal,

    UDPParam(ParamDataType.Double, 10)),

  UDPPrimitiveParameterDefinition2(

    "y_size",

    "",

    UnitType.LengthUnit,

    ParamPropType.Value,

    ParamPropFlag.MustBeReal,

    UDPParam(ParamDataType.Double, 5)),

  UDPPrimitiveParameterDefinition2(

    "z_size",

    "",

    UnitType.LengthUnit,

    ParamPropType.Value,
    ParamPropFlag.MustBeReal,
    UDPParam(ParamDataType.Double, 2))
]
length_units = "mm"

################################################################
# Class Implementation
################################################################
class UDPExtension(IUDPExtension):
  def CreatePrimitive2(self, func_lib, param_values):
    """
    Inbuilt function that is called to generate a UDP after successful validation
    :param func_lib: drawing inbuilt class, see in Help: UDMFunctionLibrary
    :param param_values: list of udp parameter values (user input) generated by UDP Core
    :return: None
    """
    param_dict = self.get_param_dict(param_values)

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

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

    [param_dict["x_size"], param_dict["y_size"], param_dict["z_size"]/2],s

    [0, param_dict["y_size"], param_dict["z_size"]/2]

]


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


fillet_rad = 0.1 * param_dict["x_size"] # 10% of X size

fillet_opt = UDPBLNDFilletOptions(True, BLNDFilletRadiusLaw.BLNDConstantRadius, fillet_rad, 0.0, True, BLNDFilletType.BLNDRound, 0.0, 0.0)


chamfer_length = 0.1 * param_dict["x_size"] # 10% of X size

chamfer_opt = UDPBLNDChamferOptions(False, BLNDChamferRangeLaw.BLNDConstantRange, chamfer_length, 0.0)


# select your geometry to which to apply operations

blend_element = UDPBLNDElements(box)


# specify attribute ListOfEdges to which edges to apply fillet operation

blend_element.ListOfEdges = edges[0:2]

func_lib.Fillet(blend_element, fillet_opt)


# redeclare attribute ListOfEdges to which edges to apply chamfer operation

blend_element = UDPBLNDElements(box)

blend_element.ListOfEdges = edges[2:4]

func_lib.Chamfer(blend_element, chamfer_opt)


# Provide to the user Info message indicating success
```

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

    def GetPrimitiveTypeInfo(self):
        return primitive_info

    def GetLengthParameterUnits(self):
        return length_units

    def GetPrimitiveParametersDefinition2(self):
        return primitive_param_definitions

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

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

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

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

    :return: dict of parameter name and values

    """

    udm_param_def = self.GetPrimitiveParametersDefinition2()

    param_dict = {}

    for i, param in enumerate(udm_param_def):

      param_value = param_values[i].Data

      if str(param.PropType) != "Menu":

        param_dict[param.Name] = param_value

      else:

        param_dict[param.Name] = param_value.replace('"', '').split
        (",")[0]

    return param_dict
```

Introduction to CPython

CPython can be used to:

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

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

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

Creating an External Script

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

```
import sys

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

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

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

Those lines are followed by:

```
import ScriptEnv

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

Follow that with:

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

Start ansysedt as GRPC server

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

-grpcsrv Flag

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

ansysedt –grpcsrv

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

ansysedt –grpcsrv portNumber

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

ansysedt –grpcsrv FirstPortNumber:LastPortNumber

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

ansysedt –grpcsrv FirstPortNumber:NumberOfPorts

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

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

Connect Functions:

1. Connect(projectPath)
   a. Connect to the opened project.
   b. Launch ansysedt.exe, open the project, then connect to it.
   c. Error if the project does not exist.

2. Connect(portNumber) Connect to running ansysedt process in the local machine. Error if no ansysedt process listening on this port.
3. Connect(machine, projectPath) connect to an open project in remote Machine. projectPath must be a network path. Error if no ansysedt process opened the project.
4. Connect(machine, portNumber) connect to a running ansysedt process on the remote machine. Error if no ansysedt process listening on this port.

Example:

```
import sys

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

import ScriptEnv

ScriptEnv.Connect(r”c:\myProjects\Project1.aedt”)

oProject = oDesktop.GetActiveProject()

print(oProject.GetName())
```

Launching Electronics Desktop

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

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

Where:

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

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

Connecting with a Running Instance of Electronics Desktop

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

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

Where:

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

Closing Electronics Desktop/Ending the Script

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

```
ScriptEnv.Shutdown()
```

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

-grpcsrv Flag

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

On Windows:

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

On Linux:

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

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

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

Related Topics:

Standalone Scripting in Iron Python

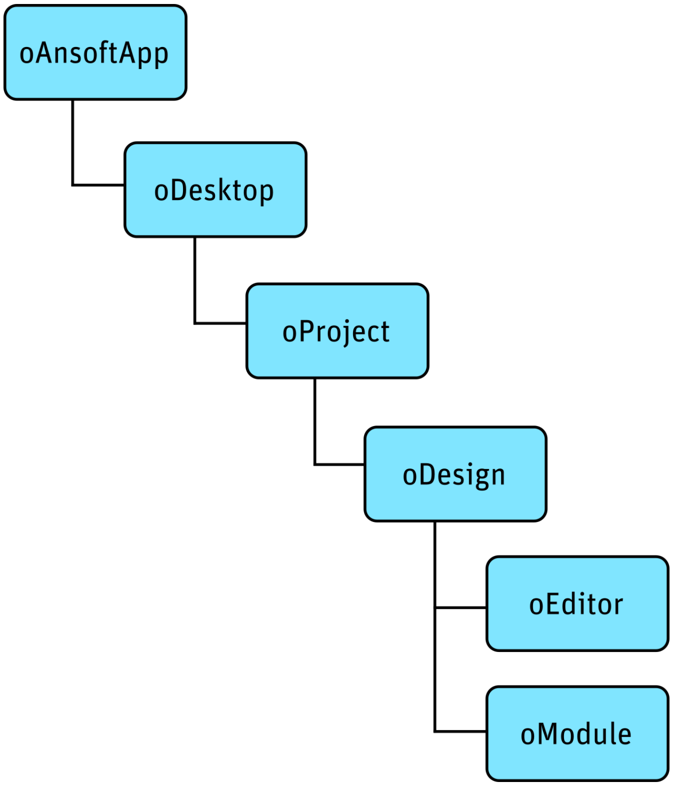
This page intentionally
left blank.

Ansys Electronics Desktop Scripting

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

Overview of Electronics Desktop Scripting Objects

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



The commands are described below, followed by examples.

oAnsoftApp

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

In Iron Python, for example:

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

oDesktop

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

In Iron Python, for example:

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

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

- Desktop Object Script Commands.

oProject

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

In Iron Python, for example:

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

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

- Project Object Script Commands

oDesign

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

In Iron Python, for example:

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

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

- Design Object Script Commands
- Output Variable Script Commands
- Reporter Editor Script Commands

oEditor

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

In Iron Python, for example:

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

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

- 3D Modeler Editor Script Commands

> **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()
```

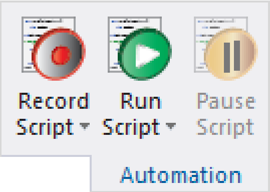
Running a Script

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

Within Electronics Desktop

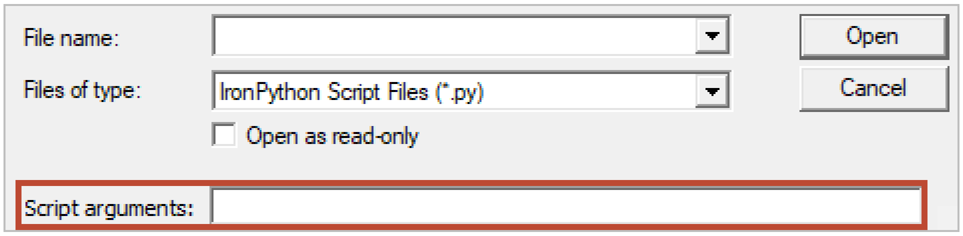
To run scripts in Electronics Desktop:

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:



File name:

Files of type: IronPython Script Files (*.py)

Script arguments:

4. Click **Open**.

Electronics Desktop executes the script.

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

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

From the Command Line

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

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

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

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

For more information about running a script from the command line, consult the Mechanical 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.

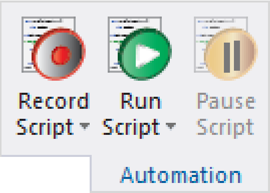
Scripts can be saved to an external file, or to the project.

> **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:

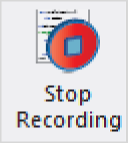
   

   A **Save As** file browser appears.

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.



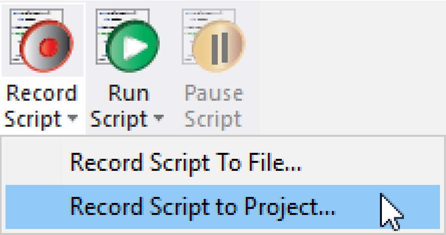
5. Perform the steps you want to record.
6. When you have finished recording the script, click **Stop Recording**, or select **Tools** > **Stop Script Recording**.

   The recorded script is saved to the folder you specified.

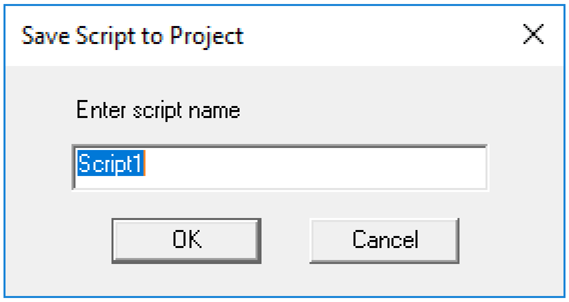
Recording a Script to a Project

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**.

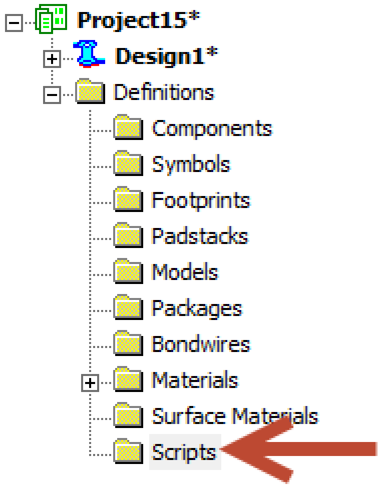
3. Perform the steps you want to record.
4. When you have finished, click **Stop Recording**, or select **Tools** > **Stop Script Recording**.

   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.

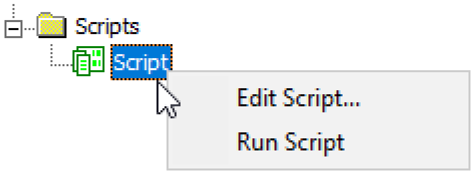
Working with Project Scripts

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]
  ```

Layout Scripts and the Active Layer

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

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

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

Scripts and Locked Layers

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

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

PyAEDT (Beta)

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

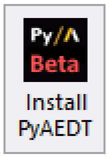
Documentation for PyAEDT can be found online at: https://aedt.docs.pyansys.com/version/stable/

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

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

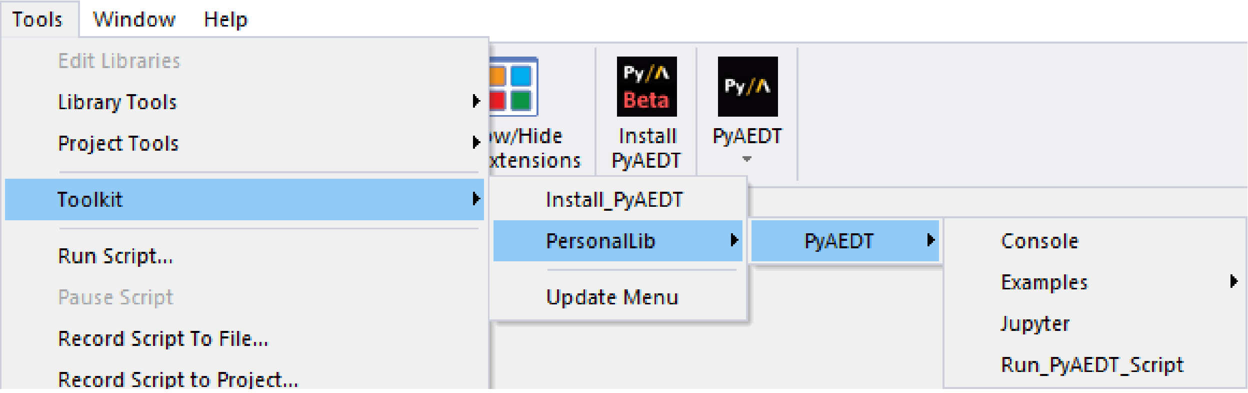
To install PyAEDT:

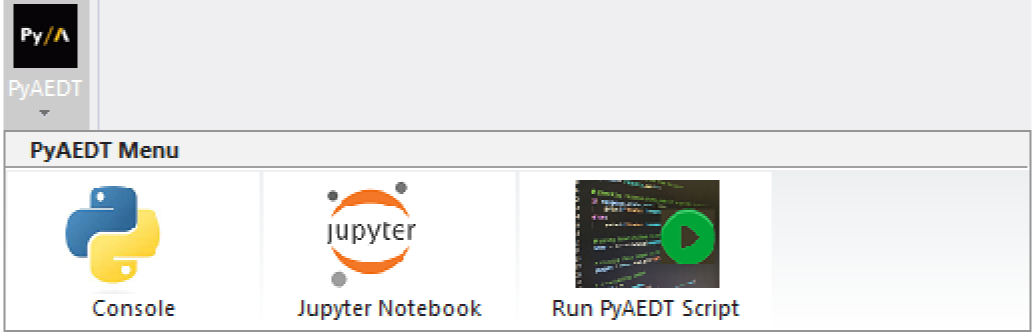
- From the **Automation** tab, click **Install PyAEDT**.



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

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





2 - Object-Oriented Property Scripting

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

This topic covers the following:

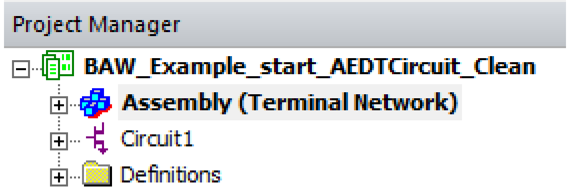
- Logic/syntax
- Basic attributes of project and design properties
- Example Scripts

Object-Oriented Scripting

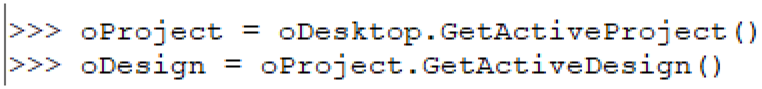
There are five basic functions in object-oriented scripting for retrieving and setting properties:

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

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

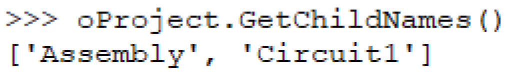

Project Manager
BAW_Example_start_AEDTCircuit_Clean
Assembly (Terminal Network)
Circuit1
Definitions

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



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

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

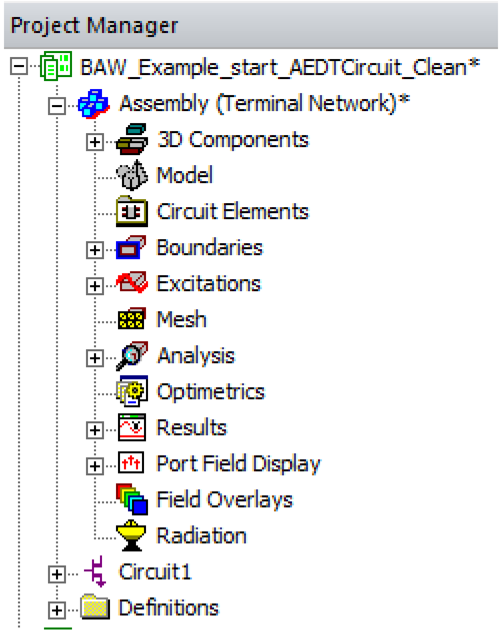


```
>>> oProject.GetChildNames()
['Assembly', 'Circuit1']
```

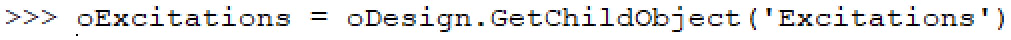
As another example, retrieve the names of the Object Instances available in oDesign, to see the various objects associated with a Design setup:

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

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

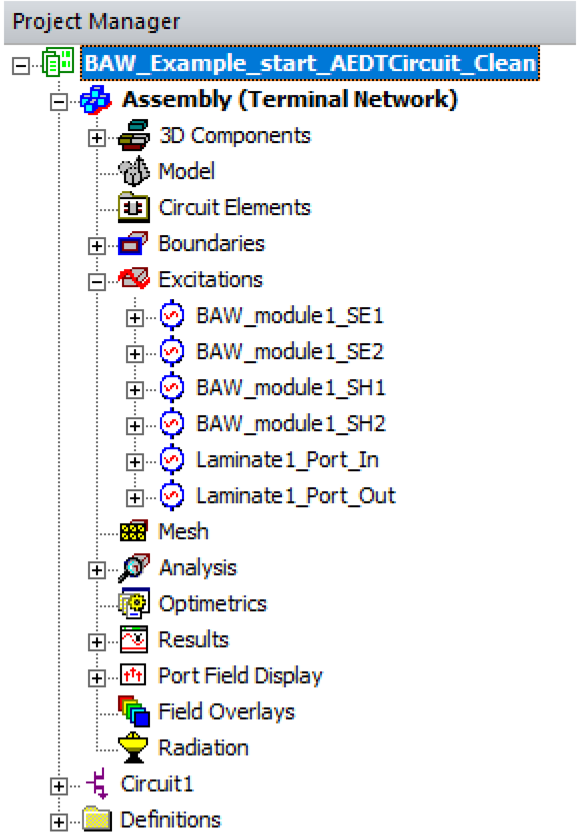


In the **Project Manager** above, any object that has the '+' symbol is populated with children that you can query. Once you know the name of the object you want, you can instance it via the GetChildObject() command. This defines the instance to the desired object. As an example, set an instance for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



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

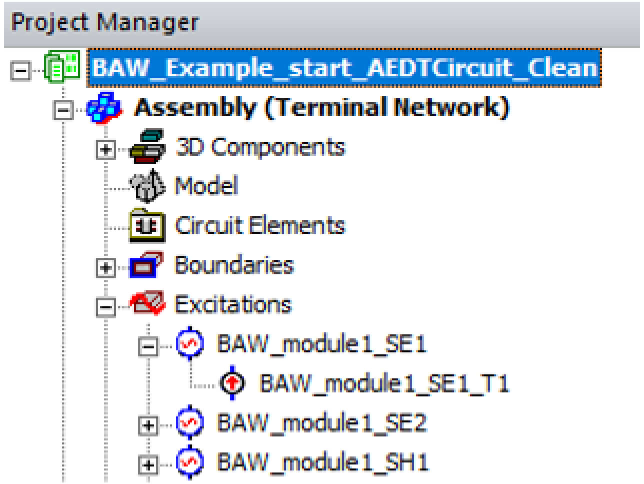
You now have an object, oExcitations defined to be the oDesign Child Object ‘Excitations’. What does this mean? If you expand the Excitations dialogue in the Project Manager, you expect that the Child Names of this object would be the names of the Excitations as defined, in this case six ports:



```
>>> oExcitations.GetChildNames()
['Laminate1_Port_In', 'Laminate1_Port_Out', 'BAW_module1_SE1', 'BAW_module1_SE2', 'BAW_module1_SH1',
'BAW module1 SH2']
```

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

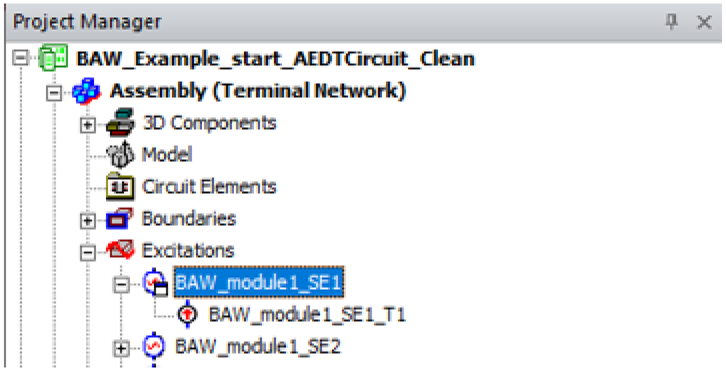
Expand the first port to see its expected Child Object, its Terminal, in the **Project Manager** Window:



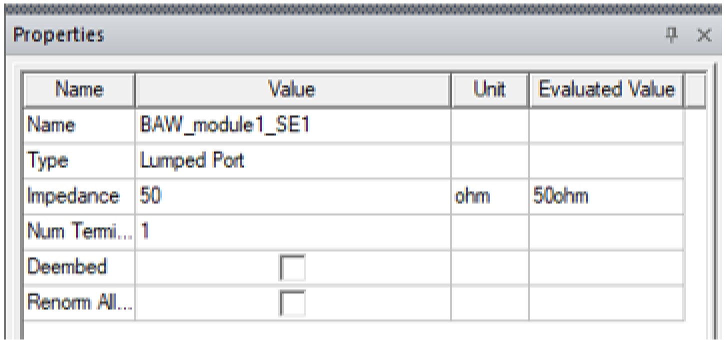
Through scripting, first define the Port Object (in this example, oPort) using the 'GetChildObject()' command for the first port, 'BAW_module1_SE1.' Then determine its Object Child Name, the terminal definition:

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

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



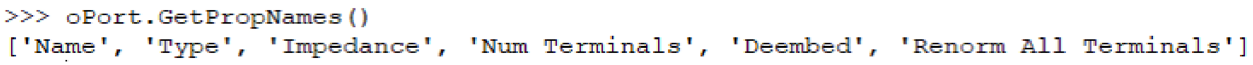
Project Manager
- BAW_Example_start_AEDTCircuit_Clean
  - Assembly (Terminal Network)
    - 3D Components
    - Model
    - Circuit Elements
    - Boundaries
    - Excitations
      - BAW_module1_SE1
        - BAW_module1_SE1_T1
      - BAW_module1_SE2



Properties

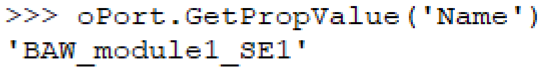
Name	Value	Unit	Evaluated Value
Name	BAW_module1_SE1		
Type	Lumped Port		
Impedance	50	ohm	50ohm
Num Termi...	1		
Deembed	□		
Renorm All...	□		

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



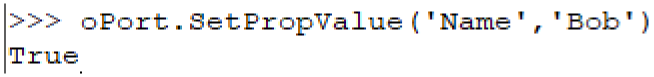
```
>>> oPort.GetPropNames()
['Name', 'Type', 'Impedance', 'Num Terminals', 'Deembed', 'Renorm All Terminals']
```

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



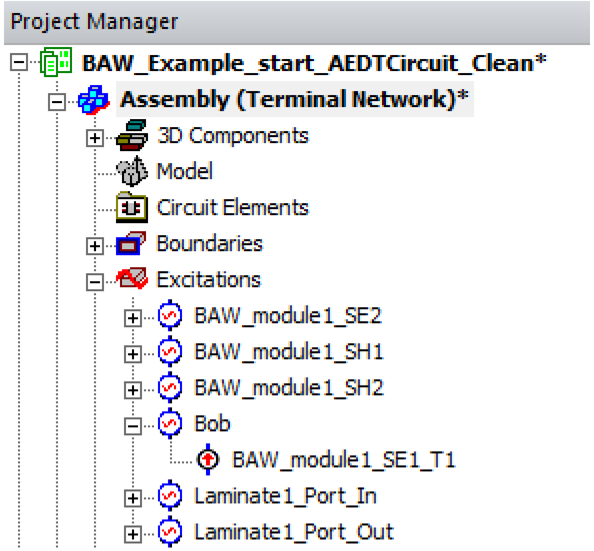
```
>>> oPort.GetPropValue('Name')
'BAW_module1_SE1'
```

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



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

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



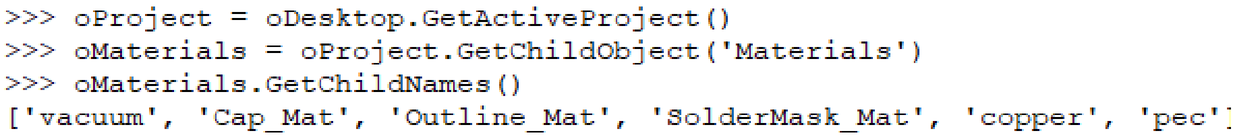
Project Manager

- BAW_Example_start_AEDTCircuit_Clean*
  - Assembly (Terminal Network)*
    - 3D Components
    - Model
    - Circuit Elements
    - Boundaries
    - Excitations
      - BAW_module1_SE2
      - BAW_module1_SH1
      - BAW_module1_SH2
      - Bob
        - BAW_module1_SE1_T1
      - Laminate1_Port_In
      - Laminate1_Port_Out

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

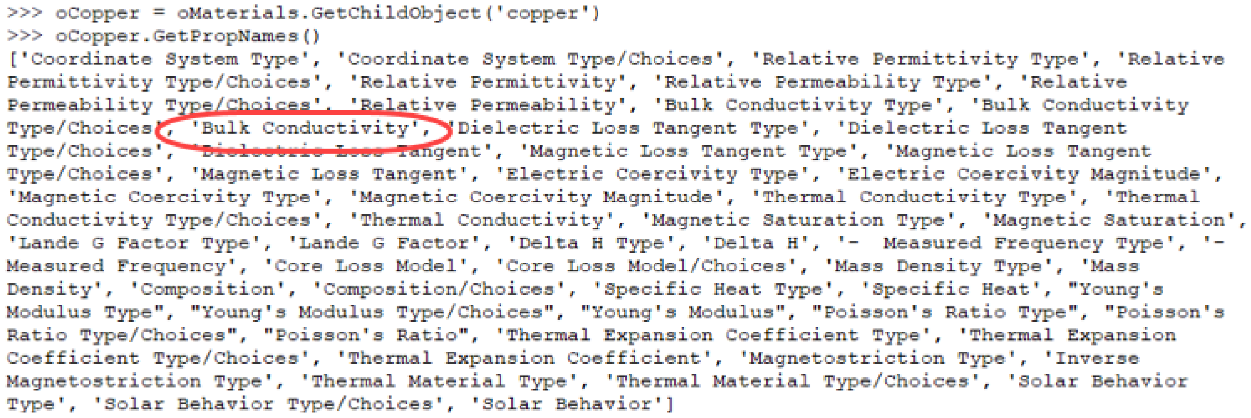
Material Properties and Examples

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



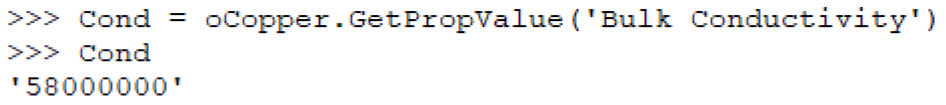
```
>>> oProject = oDesktop.GetActiveProject()
>>> oMaterials = oProject.GetChildObject('Materials')
>>> oMaterials.GetChildNames()
['vacuum', 'Cap_Mat', 'Outline_Mat', 'SolderMask_Mat', 'copper', 'pec']
```

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



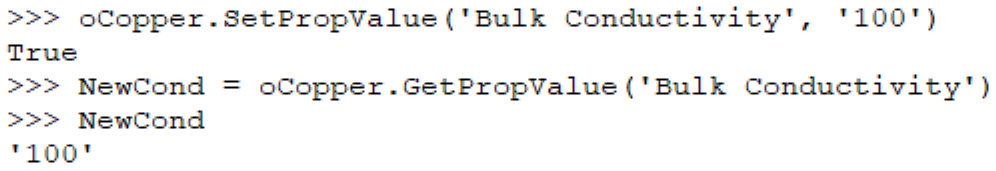
```
>>> oCopper = oMaterials.GetChildObject('copper')
>>> oCopper.GetPropNames()
['Coordinate System Type', 'Coordinate System Type/Choices', 'Relative Permittivity Type', 'Relative
Permittivity Type/Choices', 'Relative Permittivity', 'Relative Permeability Type', 'Relative
Permeability Type/Choices',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

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



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

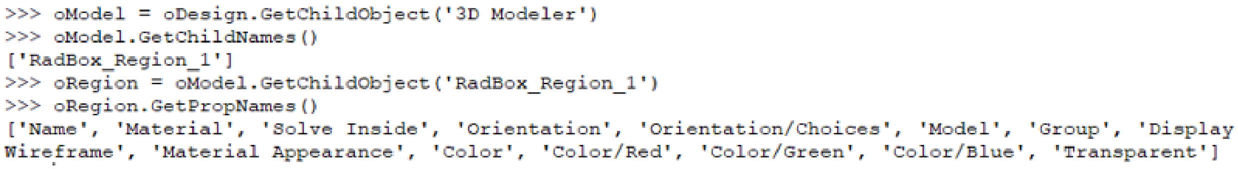
To change the conductivity, use the SetPropValue() function as shown below:



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

Body Properties and Modification

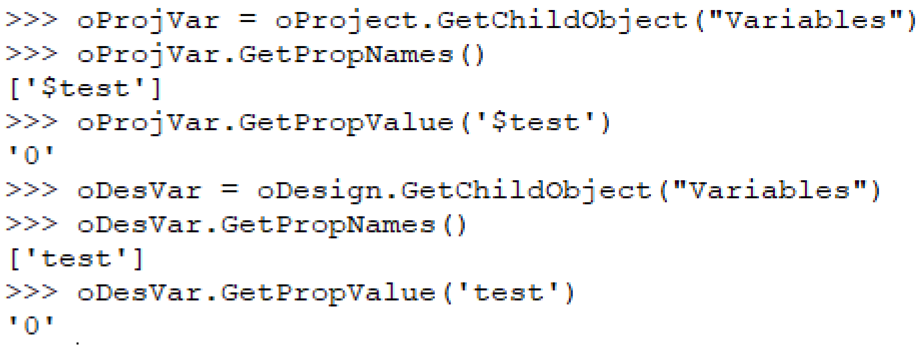
The following example shows how to retrieve the properties of a Body in the model, in this case a Region object. Once you identity the desired property, you can modify it as needed.



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

Retrieving Variables

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



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

Retrieve Datasets and Values

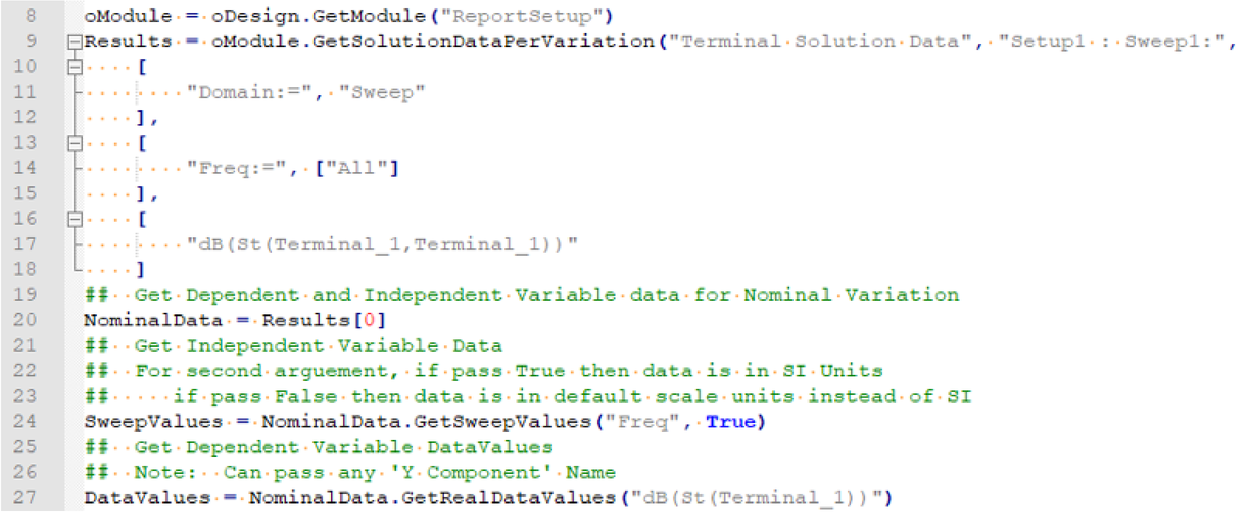
The GetChildObject, GetChildTypes and GetChildNames functions operate on the oDesktop objects. This allows you to retrieve and view datasets and values. The dataset script wrapper store all values internally in SI units, and converts them back to user-supplied units when you request non-SI property values.For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

```
>>>oDesktop.GetChildTypes()

['Projects']

>>>oDesktop.GetChildNames()

['OptimTee']

>>>arrProjectNames = oDesktop.GetChildNames()

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

>>>tp.GetChildTypes()
```

```
['Design', 'Project Data']

>>>tp.GetChildNames('Project Data')

['Variables', 'Materials', 'Surface Materials', 'Datasets']

>>>ds = tp.GetChildObject('datasets')

>>>ds.GetChildNames()

['$ds1']

>>>ds1=ds.GetChildObject('$ds1')

>>>ds1.GetPropValue('[:,:]')

[[1.0, 4.0], [2.0, 5.0], [3.0, 6.0]]

>>>ds1.GetPropSIValue()

[[1.0, 4.0], [2.0, 5.0], [3.0, 6.0]]

>>>ds1.DimUnits

>>>ds1.DimUnits = ['mm','mm']

>>>ds1.DimUnits

['mm', 'mm']

>>>ds1.GetPropSIValue()

[[0.001, 0.0040000000000000001], [0.002, 0.0050000000000000001], [0.0030000000000000001,
0.0060000000000000001]]

>>>ds1.GetPropValue('[:,:]')

[[1.0, 4.0], [2.0, 5.0], [3.0, 6.0]]
```

GetSolutionData API

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



```
oModule = oDesign.GetModule("ReportSetup")
Results = oModule.GetSolutionDataPerVariation("Terminal Solution Data", "Setup1 : Sweep1:",
    [
            "Domain:=", "Sweep"
    ],
    [
            "Freq:=", ["All"]
    ],
    [
            "dB(St(Terminal_1,Terminal_1))"
    ]
##  Get Dependent and Independent Variable data for Nominal Variation
NominalData = Results[0]
##  Get Independent Variable Data
##  For second arguement, if pass True then data is in SI Units
##      if pass False then data is in default scale units instead of SI
SweepValues = NominalData.GetSweepValues("Freq", True)
##  Get Dependent Variable DataValues
##  Note:  Can pass any 'Y Component' Name
DataValues = NominalData.GetRealDataValues("dB(St(Terminal_1))")
```

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

Summary

Scripting has been advancing in Ansys Electronics Desktop to better allow you to customize and automate their repetitive or complex simulations. The ability to easily retrieve and set property values via the Object-Oriented scripting allows for ease or both writing and reading. The ability to extract solution data within a script execution is a new functionality that markedly enables more advanced post processing.

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

Each exposed script object supports the following COM functions:

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

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

```
>>> design.GetChildTypes()

['Module', 'Editor', ‘Design Data’]
```

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

  For example, a Mechanical design object has these children.

```
>>> design.GetChildNames()

['Boundaries', 'Excitations', 'Optimetrics', 'Results', '3D Modeler']
```

Four of the children are of “Module” type

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

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

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

  For example,

```
>>> d = project.GetChildObject("hfss")

>>> d.GetChildObject("3d modeler").GetChildNames()

['Box1', 'Box1_1', 'Box1_1_1']

>>> project.GetChildObject("hfss/3d modeler").GetChildNames()

['Box1', 'Box1_1', 'Box1_1_1']
```

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

  For example,

```
>>> geom = project.GetChildObject("hfss/3d modeler").GetChildObject("Box1")

>>> geom.GetPropNames()

['Name', 'Material', 'Material/SIValue', 'Material/EvaluatedValue', 'Solve Inside', 'Orientation', 'Orientation/Choices', 'Model', 'Group', 'Display Wireframe', 'Material Appearance', 'Color', 'Color/Red', 'Color/Green', 'Color/Blue', 'Transparent']
```

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

For example,

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

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

For example,

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

- GetPropEvaluatedValue (<*PropName*>)

For example,

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

- GetPropSIValue (<*PropName*>

For example,

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

return "length / 2"

```
oCreateBox.GetPropEvaluatedValue("xSize")
```

return '0.4mm'

```
oCreateBox.GetPropSIValue("xSize")
```

return 0.0004

Additional Details Specific to AEDT Solvers

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

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

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

HFSS

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

HFSS 3D Layout

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

Maxwell 3D/2D

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

RMxprt

- Analysis, Field Overlays, Optimetrics, Results

Q3D

- Boundaries, Nets, Analysis, Optimetrics, Q2D

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

Icepak

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

Mechanical

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

Circuit

- Optimetrics, Results

Circuit Netlist

- Results

EMIT

- Coupling

Simplorer/Twin Builder

- Analysis, Optimetrics, Results

Additional details on Boundaries/Excitations

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

These module script objects do not have properties

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

[]
```

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

```
>>> d = p.GetChildObject("q2d")

>>> d.GetChildObject("conductors").GetChildTypes()

['NonIdealGround', 'SignalLine']
```

GetChildNames of these module object returns its immediate children

```
>>> p.GetChildObject("icepak/thermal").GetChildNames()

['Source1', 'Resistance1', 'ConductingPlate1', 'Source2', 'Resistance2', 'ConductingPlate2',
'Source3', 'Resistance3', 'ConductingPlate3']
```

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

```
>>> p.GetChildObject("icepak/thermal").GetChildNames("resistance")

['Resistance1', 'Resistance2', 'Resistance3']
```

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

```
>>> port = p.GetChildObject("hfss/excitations/1")

>>> port.GetPropNames(False)

['Name', 'Deembed', 'Deembed Dist', 'Renorm All Terminals']
```

You can query/edit these properties

```
>>> port.GetPropValue("deembed")

False

>>> port.SetPropValue("deembed", True)

True

>>> port.GetPropValue("deembed")

True
```

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

```
>>> port.GetChildNames()

['Box1_T1']

>>> port.GetChildTypes()

['Terminal']

>>> p.GetChildObject("q3d/nets/s2").GetChildNames()

['Source2', 'Sink2']

>>> p.GetChildObject("q3d/nets/s2").GetChildTypes()

['Sink', 'Source']
```

3D component encapsulation

These script interfaces are compliant with encapsulation. For example,

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

Additional details on Solve setup

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

```
>>> d = oDesktop.GetActiveProject().GetChildObject("hfss")

>>> d.GetChildNames()

['Boundaries', 'Excitations', 'Hybrid Regions', 'Circuit Elements', 'Analysis', 'Optimetrics',
'RadField', 'Results', '3D Modeler']

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

This module script object has no property

```
>>> setups.GetPropNames()

[]
```

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

```
>>> setups.GetChildTypes()

[]
```

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

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

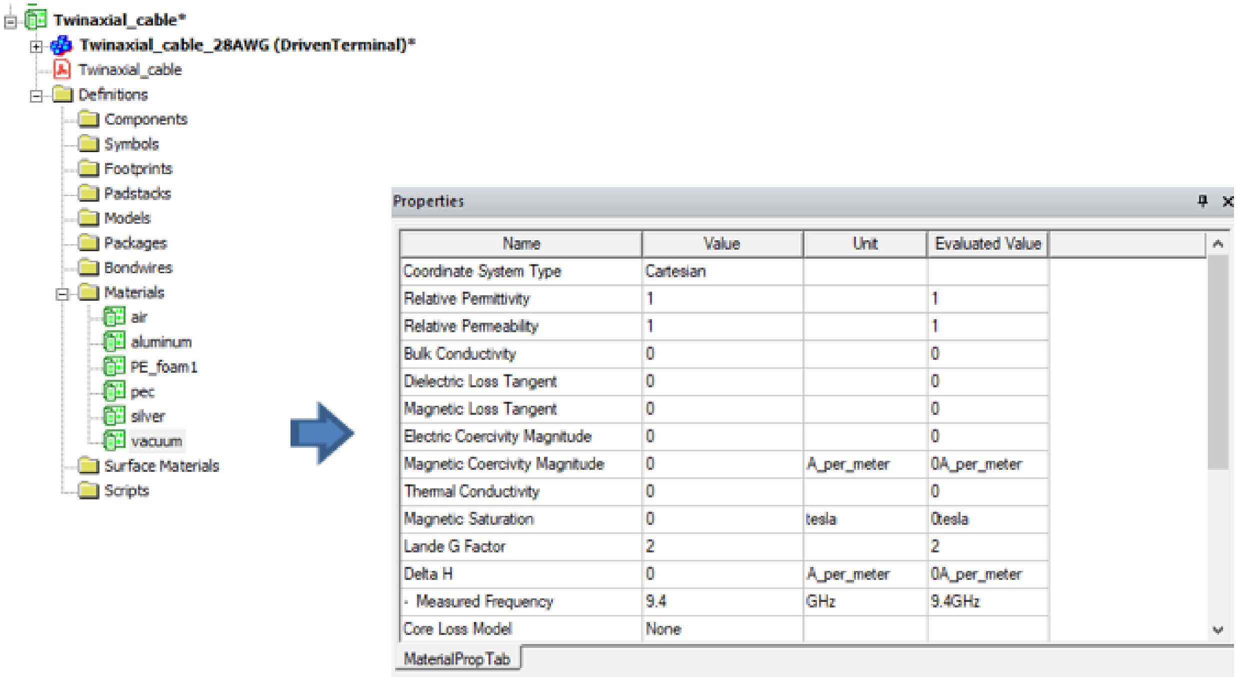
```
>>> setup.GetChildNames()

['Sweep', 'Sweep1', 'Sweep2']

>>> setup.GetChildTypes()

['Discrete', 'Interpolating']

>>> sweep1 = setup.GetChildObject("sweep")
```

Related Topics

Example: GetPropNames and GetPropValues for Layered Impedance Boundary Script

Materials Scripting Support

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



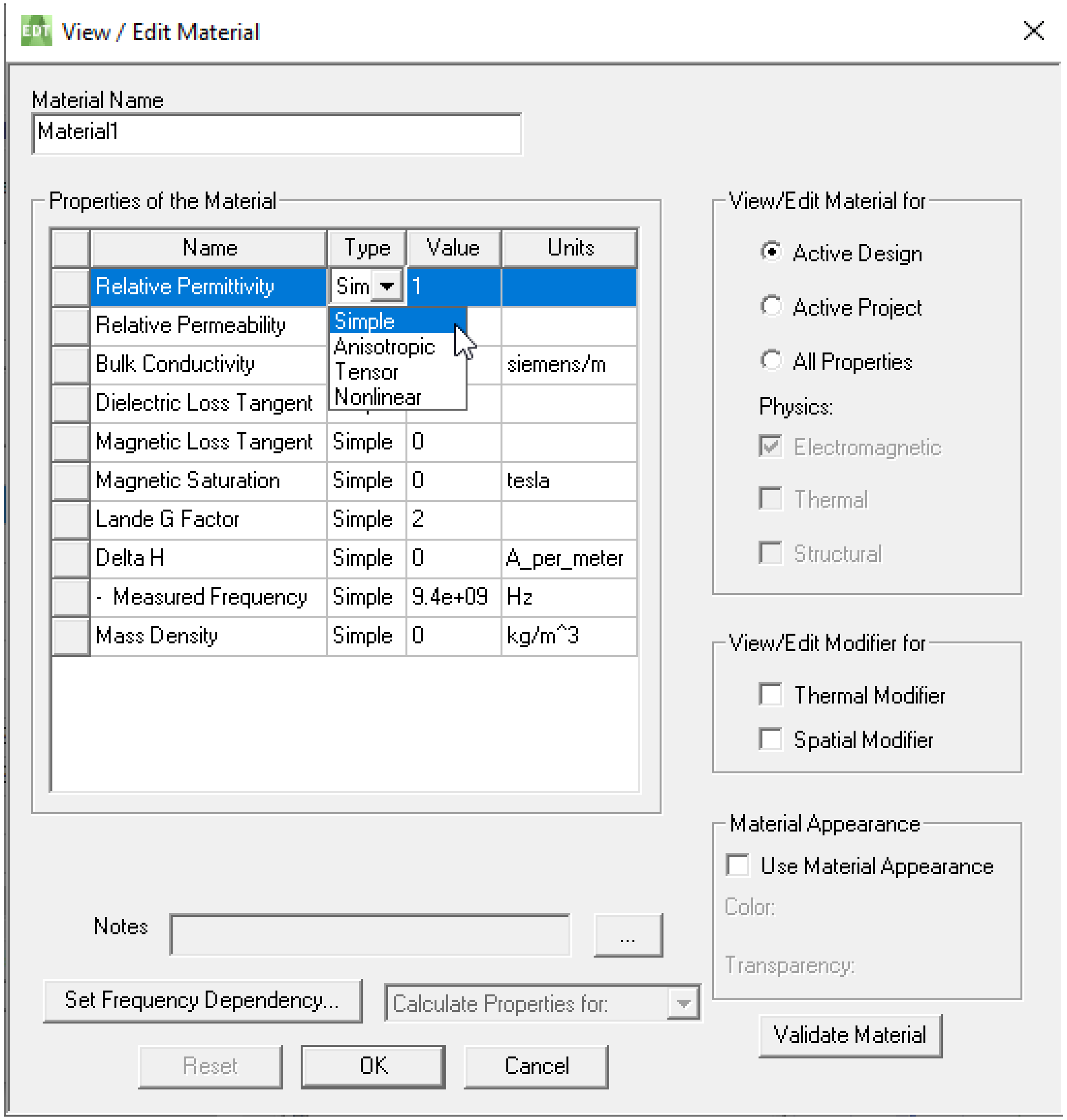
- Twinaxial_cable*
  - Twinaxial_cable_28AWG (DrivenTerminal)*
  - Twinaxial_cable
  - Definitions
    - Components
    - Symbols
    - Footprints
    - Padstacks
    - Models
    - Packages
    - Bondwires
    - Materials
      - air
      - aluminum
      - PE_foam1
      - pec
      - silver
      - vacuum
    - Surface Materials
    - Scripts

Properties

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

MaterialPropTab

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



View / Edit Material

Material Name: Material1

Properties of the Material

	Name	Type	Value	Units
	Relative Permittivity	Sim	1	
	Relative Permeability			
	Bulk Conductivity			siemens/m
	Dielectric Loss Tangent			
	Magnetic Loss Tangent	Simple	0	
	Magnetic Saturation	Simple	0	tesla
	Lande G Factor	Simple	2	
	Delta H	Simple	0	A_per_meter
	- Measured Frequency	Simple	9.4e+09	Hz
	Mass Density	Simple	0	kg/m^3

View/Edit Material for

● Active Design

○ Active Project

○ All Properties

Physics:

☑ Electromagnetic

□ Thermal

□ Structural

View/Edit Modifier for

□ Thermal Modifier

□ Spatial Modifier

Material Appearance

□ Use Material Appearance

Color:

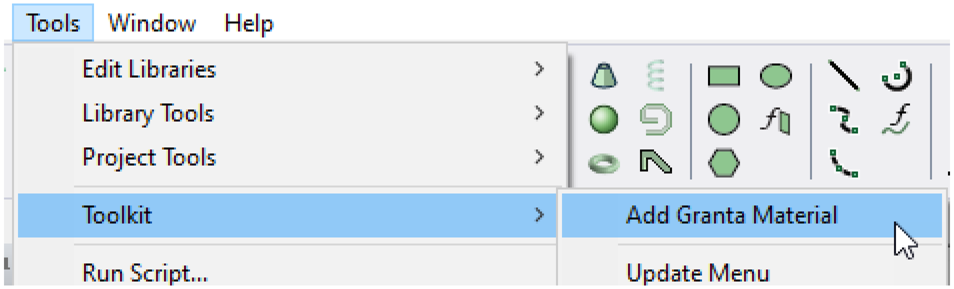
Transparency:

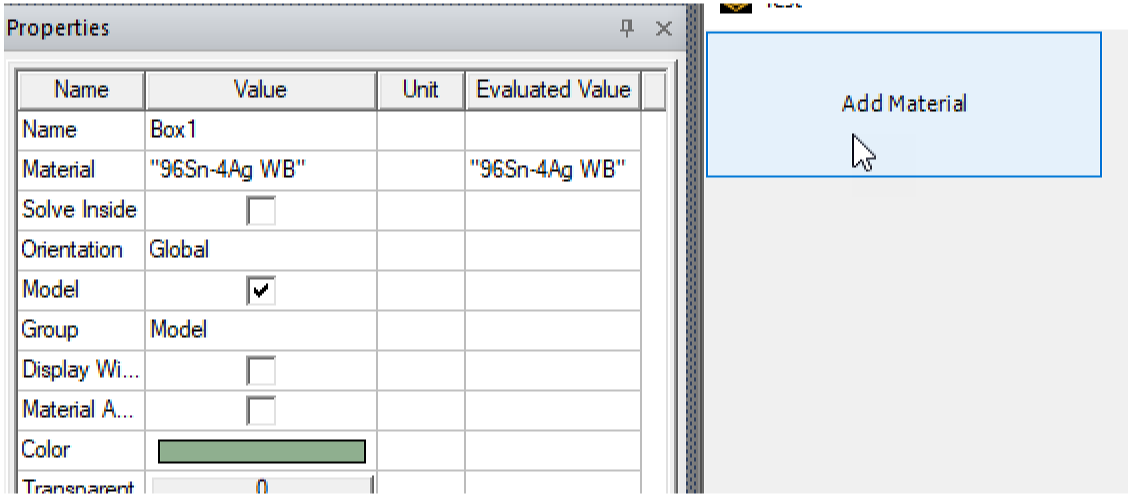
Notes

With this feature enabled you can then:

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

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





Name	Value	Unit	Evaluated Value
Name	Box1		
Material	"96Sn-4Ag WB"		"96Sn-4Ag WB"
Solve Inside	□		
Orientation	Global		
Model	☑		
Group	Model		
Display Wi...	□		
Material A...	□		
Color	🟩		
Transparent	0		

What is not supported:

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

Object Oriented Scripting for Materials

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

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

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

Examples showing change to material property type:

```
>>> omat.GetPropValue("Relative Permeability Type/Choices")

['Simple', 'Anisotropic', 'Tensor', 'Nonlinear']

>>> omat.GetPropValue("Relative Permeability Type")

'Nonlinear'

>>> omat.SetPropValue("Relative Permeability Type", "Simple")

True

>>> omat.GetPropValue("Relative Permeability Type")

'Simple'

>>> omat.SetPropValue("Relative Permeability", 10)

True
```

Examples showing change to a vector component value

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

'0A_per_meter'

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

```
True
```

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

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

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

```
True
```

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

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

Change choice property value

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

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

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

```
True
```

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

```
['Coordinate System Type', 'Coordinate System Type/Choices', 'Relative Permeability Type',
'Relative Permeability Type/Choices', 'Relative Permeability', 'Relative
Permeability/SIValue', 'Relative Permeability/EvaluatedValue', 'Bulk Conductivity Type', 'Bulk
Conductivity Type/Choices', 'Bulk Conductivity', 'Bulk Conductivity/SIValue', 'Bulk
Conductivity/EvaluatedValue', 'Magnetic Coercivity Type', 'Magnetic Coercivity Magnitude',
'Magnetic Coercivity Magnitude/SIValue', 'Magnetic Coercivity Magnitude/EvaluatedValue', 'Magnetic
Coercivity Components', 'Magnetic Coercivity Components/Component1', 'Magnetic Coercivity
Components/Component2', 'Magnetic Coercivity Components/Component3', 'Composition',
'Composition/Choices', '- Stacking Factor Type', '- Stacking Factor', '- Stacking Factor/SIValue',
'- Stacking Factor/EvaluatedValue', '- Stacking Direction', '- Stacking Direction/Choices',
"Young's Modulus Type", "Young's Modulus Type/Choices", "Young's Modulus", "Young's
Modulus/SIValue", "Young's Modulus/EvaluatedValue", "Poisson's Ratio Type", "Poisson's Ratio
Type/Choices", "Poisson's Ratio", "Poisson's Ratio/SIValue", "Poisson's Ratio/EvaluatedValue"]
```

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

True

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

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

Change choice property value

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

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

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

True

>>> omat.GetPropNames()

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

>>>
```

Property Script Commands

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

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

`GetName()`

- Returns the name of the object.

`GetChildTypes()`

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

`GetChildNames(<`*`type`*`>)`

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

`GetChildObject(<`*`objPath`*`>)`

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

`GetPropNames(<bIncludeReadOnly>)`

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

`GetPropValue(<propertyPath>)`

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

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

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

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

> **Note:**
>
> Older property commands should be executed by the oProject object.
>
> ```
> oProject = oDesktop.SetActiveProject("Project1")
> ```
>
> ```
> oProject.CommandName <args>
> ```

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

Object Script Property Function Summary

Object Path

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

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

- Name beginning with '@' character denoted as a property name, when an object has a child and property with the same name.

Property Object

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

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

    - Returns an array of the object’s property names.
- GetPropValue(<*propertyPath*>)
    - <*propertyPath*> – the property's full path. A property name or child object’s path appended with a property name, like “TeeModel/Offset/SIValue”
    - Returns the property value if the property is foun; 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 DesignoDesign GetPropValue("offset/SIValue") //get the offset variable SI value in the Design
  oDesign.GetPropValue("Results/S Parameter Plot 1/Display Type") // Get the report display type

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

3D Modeler Object

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

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

Variable Object

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

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

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

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

Optimetrics Module Object:

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

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

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

Optimetrics Setup Object

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

oOptSetup = oOptModule.GetChildObject('OptimizationSetup1')

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

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

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

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

ReportSetup(Results) Module Object:

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

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

ReportSetup(Results) Module Child Objects:

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

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

oRpt = oRptModule.GetChildObject(reportName)

oRpt = oDesign.GetChildObject(“Results/reportName”)

oTrace = oRpt.GetChildObject(traceName)

oTrace = oRptModule.GetChildObject(ReportName/TraceName)

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

Radiation Module Object:

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

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

Radiation Module Child Objects:

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

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

oRadSetup = oRadModule.GetChildObject(setupName)

oRadSetup = oDesign.GetChildObject(RadField/setupName)

- GetChildTypes() always return empty array.
- GetChildNames() always return empty array , since Radiation setup has no child.
- GetChildObject(objPath) it has no child.
- GetPropNames(bIncludeReadOnly) will return the property names listed in the property window when the setup is selected.
- GetPropValue(propName)
  oRadSetup.GetPropValue(”Num Points") return the line setup’s “Num Points” property value.
  oRadSetup.Get_NumPoints() will also get the same value.
- SetPropValue(propName, newValue)
  oRadSetup.SetPropValue('Num Points', 888)
  oRadSetup.Set_NumPoints(888)

Conventions Used in this Chapter

General Definitions:

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

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

For Project Variables:

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

For Local Variables:

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

For Passed Parameters:

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

For Definition Parameters:

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

For Modules and Editors:

<PropServer>	<ModuleName>:<ItemName> where <ItemName is the boundary name, solution setup name, etc. For example, "BoundarySetup:PerfE1"
<PropTab>	Boundary Module: "HfssTab" Mesh Operations Module: "MeshSetupTab" Analysis Module: "HfssTab" Optimetrics Module: "OptimetricsTab" Solutions Module: *Does not support properties.* Field Overlays Module: "FieldsPostProcessorTab" Radiation Module: "RadFieldSetupTab"

	Circuit Module: "CCircuitTab" System Module: "SystemTab" HFSS 3D Layout Module: "HFSS 3D LayoutTab" Nexxim Module: "NexximTab" Layout elements: "BaseElementTab" Schematic elements: "ComponentTab" Optimetrics Module: "OptimetricsTab"

For 3D Model Editor objects:

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

For 3D Model Editor operations:

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

For Reporter operations on Report properties:

<PropServer>	<ReportSetup>

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

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

GetArrayVariables

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

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

Python Syntax	GetArrayVariables()

Python Example	`oProject.GetArrayVariables()`
	`oDesign.GetArrayVariables()`

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

<table>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td>&lt;<i>PropTab</i>&gt;</td><td>String</td><td>One of the following, where tab titles are shown in parentheses:
• PassedParameterTab ("Parameter Values")
• DefinitionParameterTab (Parameter Defaults")
• LocalVariableTab ("Variables" or "Local Variables")
• ProjectVariableTab ("Project variables")
• ConstantsTab ("Constants")
• BaseElementTab ("Symbol" or "Footprint")
• ComponentTab ("General")
• Component("Component")
• CustomTab ("Intrinsic Variables")
• Quantities ("Quantities")
• Signals ("Signals")</td></tr>
<tr><td>&lt;<i>PropServer</i>&gt;</td><td>String</td><td>An object identifier, generally returned from another script method, such as CompInst@R;2;3</td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>Array of strings containing the names of the appropriate properties.</td></tr>
</table>

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

GetPropertyValue

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

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

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

Python Syntax	GetPropertyValue (<*PropTab*>, <*PropServer*>, <*PropName*>)

Python Example	`selectionArray = oEditor.GetSelections()`
	`for k in selectionArray:`
	`val = oEditor.GetPropertyValue("PassedParameterTab", k, "R")`
	`...`

GetVariables

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

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

Python Syntax	GetVariables ()

Python Example	`oProject.GetVariables()`
	`oDesign.GetVariables()`

GetVariableValue

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

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

Python Syntax	GetVariableValue(*<VarName>*)
Python Example	`oProject.GetVariableValue("var_name")`

SetPropertyValue

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

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

UI Access	N/A
Parameters	Name | Type | Description

	<propTab>	String	One of the following, where tab titles are shown in parentheses: • PassedParameterTab ("Parameter Values") • DefinitionParameterTab (Parameter Defaults") • LocalVariableTab ("Variables" or "Local Variables") • ProjectVariableTab ("Project variables") • ConstantsTab ("Constants") • BaseElementTab ("Symbol" or "Footprint") • ComponentTab ("General") • Component("Component") • CustomTab ("Intrinsic Variables") • Quantities ("Quantities") • Signals ("Signals")
	<propServer>	String	An object identifier, generally returned from another script method, such as CompInst@R;2;3
	<propName>	String	Name of the property.
	<propValue>	String	The value for the property
Return Value	None.		

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

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`.

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td><table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td>&lt;<i>VarName</i>&gt;</td><td>String</td><td>Variable name.</td></tr>
<tr><td>&lt;<i>VarValue</i>&gt;</td><td>Value</td><td>New value for the variable.</td></tr>
</table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	SetVariableValue (<*VarName*>, <*VarValue*>)
Python Example	`oProject.SetVariableValue('$Var1', '3mm')`

3 - Application Object Script Commands

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

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

General Application Script Commands

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

- GetAppDesktop

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

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

GetAppDesktop

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

<table>
<tr><td><b>UI Access</b></td><td>NA</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>None</td><td></td><td></td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>Object</td></tr>
</table>

Python Syntax	GetAppDesktop()
Python Example	`oDesktop = oAnsoftApp.GetAppDesktop()`

4 - Desktop Object Script Commands

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

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

AddMessage

AreThereSimulationsRunning

ClearMessages

CloseAllWindows

CloseProject

CloseProjectNoForce

DeleteProject

DeleteRegistryEntry

DoesRegistryValueExist

DownloadJobResults

EnableAutoSave

ExportOptionsFiles

GetActiveProject

GetActiveScheduler

GetActiveSchedulerInfo

GetAutoSaveEnabled

GetBuildDateTimeString

GetCustomMenuSet

GetDefaultUnit

GetDesktopConfiguration

GetDistributedAnalysisMachines

GetDistributedAnalysisMachinesForDesignType

GetExeDir

GetGDIObjectCount

GetLibraryDirectory

GetLocalizationHelper

GetMessages

GetMonitorData

GetPersonalLibDirectory

GetProcessID

GetProjectDirectory

GetProjectList

GetProjects

GetRegistryInt

GetRegistryString

GetRunningInstancesMgr

GetSchematicEnvironment

GetScriptingToolsHelper

GetSysLibDirectory

GetTempDirectory

GetUserLibDirectory

GetVersion

IsFeatureEnabled

LaunchJobMonitor

NewProject

OpenAndConvertProject

OpenMultipleProjects

OpenProject

OpenProjectWithConversion

PageSetup

PauseRecording

PauseScript

Print

QuitApplication

RefreshJobMonitor

ResetLogging

RestoreProjectArchive

RestoreWindow

ResumeRecording

RunACTWizardScript

RunProgram

RunScript

RunScriptWithArguments

SelectScheduler

SetActiveProject

SetActiveProjectByPath

SetCustomMenuSet

SetDesktopConfiguration

SetLibraryDirectory

SetProjectDirectory

SetRegistryFromFile

SetRegistryInt

SetRegistryString

SetSchematicEnvironment

SetTempDirectory

ShowDockingWindow

Sleep

StopSimulations

SubmitJob

TileWindows

Related Topics:

Desktop Commands For Registry Values

ImportExport Tool Commands

AddMessage

Add a message with severity and context to message window.

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td>&lt;<i>projectName</i>&gt;</td><td>String</td><td>Project name. Passing an empty string adds the message as the desktop global message.</td></tr>
<tr><td>&lt;<i>designName</i>&gt;</td><td>String</td><td>Design name. Ignored if project name is empty. Passing an empty string adds the message to project node in the message tree.</td></tr>
<tr><td>&lt;<i>severity</i>&gt;</td><td>Integer</td><td>One of "Error", "Warning" or "Info". Anything other than the first two is treated as "Info"
0 = Informational, 1 = Warning, 2 = Error, 3 = Fatal</td></tr>
<tr><td>&lt;<i>msg</i>&gt;</td><td>String</td><td>The message for the message window.</td></tr>
<tr><td>&lt;<i>category</i>&gt;</td><td>String</td><td>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.</td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

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

AreThereSimulationsRunning

Returns a bool specifying whether there are simulations running, or either running or pending, depending on the inclusion of alsoPending.

<table>
<tr><td><b>UI Access</b></td><td>NA</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;bool&gt;
<AlsoPending&gt;</td><td>Integer</td><td>Whether to report if simulations are running, or running and pending.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>A human readable status string specifying what happened.</td></tr>
</table>

Python Syntax	AreThereSimulationsRunning (<bool>, clean)
Python Example	`oDesktop.AreThereSimulationsRunning(bool clean)`

ClearMessages

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

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

The ClearMessages function accepts four input arguments. The first two specifies project and design, respectively. This function clears messages in the range specified by the third (interpreted as start severity) and fourth (interpreted as stop severity) arguments. The fourth argument has a default value of 0. The last two arguments need not be specified in any given order, i.e., they can indicate either ascending or descendingtrend. 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	<table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;projectName&gt;</td><td>String</td><td>Name of the project from which to clear messages.</td></tr><tr><td>&lt;designName&gt;</td><td>String</td><td>Name of the design under the <projectName> from which to clear messages.</td></tr><tr><td>&lt;startSeverity&gt;</td><td>Integer</td><td>User-specified severity level at which messages start getting cleared: • 0 = informational • 1 = warning • 2 = error • 3 = fatal error</td></tr><tr><td>&lt;stopSeverity&gt;</td><td>Integer</td><td><i>Optional</i>. User-specified stop severity level until which messages will be cleared. • 0 = informational • 1 = warning • 2 = error • 3 = fatal error If not specified, <stopSeverity> has a default value of 0.</td></tr></table>
Return Value	None.

Python Syntax	ClearMessages(<*projectName*>, <*designName*>, <*startSeverity*>, <*stopSeverity*>)

Python Example	**Preserve Current Functionality; stopSeverity = 0 (DEFAULT VALUE)**
	`# startSeverity = 0; clears just the Info messages`
	`oDesktop.ClearMessages("Ex_EC_1_Asymmetrical_Conductor", "Maxwell3DDesign1", 0)`

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

oDesktop.ClearMessages("Ex_EC_1_Asymmetrical_Conductor", "Maxwell3DDesign1", 1)

# startSeverity = 2; clears all the Info, Warning, Error
```

oDesktop.ClearMessages("Ex_EC_1_Asymmetrical_Conductor", "Maxwell3DDesign1", 2)

```
# startSeverity = 3; clears all messages, i.e., Info, Warning, Error, Fatal

oDesktop.ClearMessages("Ex_EC_1_Asymmetrical_Conductor", "Maxwell3DDesign1", 3)
```

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 messages

oDesktop.ClearMessages("Ex_EC_1_Asymmetrical_Conductor", "Maxwell3DDesign1", 0, 1)

# startSeverity = 0 and stopSeverity = 2; clears all the Info, Warning, and Error messages

oDesktop.ClearMessages("Ex_EC_1_Asymmetrical_Conductor", "Maxwell3DDesign1", 0, 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`
	`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)`

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.

<table>
<tr><td><b>UI Access</b></td><td><b>File</b> > <b>Close</b></td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;ProjectName&gt;</td><td>String</td><td>The name of the project already in the Desktop that is to be closed, without path or extension</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>None</td></tr>
</table>

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

CloseProjectNoForce

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

<table>
<tr><td><b>UI Access</b></td><td><b>File</b> > <b>Close</b></td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;ProjectName&gt;</td><td>String</td><td>The name of the project already on the Desktop that is to be closed, without</td></tr></table></td></tr>
</table>

			path or extension
Return Value	None		

Python Syntax	CloseProjectNoForce (<*ProjectName*>)
Python Example	`oDesktop.CloseProjectNoForce("MyProject")`

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	`projects = oDesktop.GetProjects()`
	`numprojects = projects.Count`

DownloadJobResults

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

UI Access	Select Scheduler
Parameters	<table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;jobID&gt;</td><td>String</td><td>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.</td></tr><tr><td>&lt;projectPath&gt;</td><td>String</td><td>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.</td></tr><tr><td>&lt;resultPath&gt;</td><td>String</td><td>A string giving the folder path for the download to save to.</td></tr><tr><td>&lt;Filters> (optional)</td><td>String</td><td>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.</td></tr></table>
Return Value	A Boolean result about download complete or not

Python Syntax	DownloadJobResults(*jobID*, *projectPath*, *resultsPath*, *filters*)
Python Example	`boolDownloadCompleted = oDesktop.DownloadJobResults("01234567890123456789O", "C:\\projects\\basic.aedt", "C:\\projects\\DownloadResults\\", "*")`

DeleteRegistryEntry

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

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><i>&lt;pathToRegistry&gt;</i></td><td>String</td><td>Path to Registry entry.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>Boolean:
• <b>True</b> – Key has been deleted.
• <b>False</b> – Key does not exist.</td></tr>
</table>

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

DoesRegistryValueExist

Determines whether a registry value exists.

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><i>&lt;KeyName&gt;</i></td><td>String</td><td>Full name of registry key, including path.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>Boolean:
• <b>True</b> – Key exists.
• <b>False</b> – Key does not exist.</td></tr>
</table>

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

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

GetActiveProject

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

> **Note:**
>
> **GetActiveProject** returns normally if there are no active objects.

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

Python Syntax	GetActiveProject()
Python Example	`oProject = oDesktop.GetActiveProject()`

GetActiveScheduler

Obtains the name of the scheduler active in the Desktop.

> **Note:**
>
> **GetActiveScheduler** returns normally if there are no active objects.

UI Access	N/A
Parameters	None.
Return Value	Name of the scheduler that is currently active in the desktop. Gets results such as "RSM", "Remote RSM", "Ansys Cloud", "SLURM", "LSF", "PBS", "SGE", "Windows HPC", etc. (User-friendly scheduler names.)

Python Syntax	GetActiveScheduler()
Python Example	`oSchedulerName = oDesktop.GetActiveScheduler()`

GetActiveSchedulerInfo

Obtains info for the scheduler active in the Desktop.

> **Note:**
>
> **GetActiveSchedulerInfo** returns normally if there are no active objects.

UI Access	N/A
Parameters	None.
Return Value	Info for the scheduler that is currently active in the desktop or a remote scheduler. Gets results such as "RSM", "Remote RSM", "Ansys Cloud", "SLURM", "LSF", "PBS", "SGE", "Windows HPC", as well as server info used to select a server. For a local scheduler, the return format resembles '{"Selected Scheduler":"RSM","Scheduler Name":"RSM","Server":""}'. For a remote scheduler, the return format resembles '{"Selected Scheduler":"Remote RSM","Scheduler Name":"SLURM","Server":"<server>"}'

Python Syntax	GetActiveSchedulerInfo()
Python Example	`oSchedulerInfo = oDesktop.GetActiveSchedulerInfo()`

GetAutoSaveEnabled

Checks whether the autosave feature is enabled.

UI Access	N/A
Parameters	None.
Return Value	Integer: • 1 – Autosave is enabled.

	- 0 – Autosave is not enabled.

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

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()`

GetCustomMenuSet

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

UI Access	N/A
Parameters	None.
Return Value	String containing current menu set. For example, 'Default', 'EM', ‘Twin Builder’.

Python Syntax	GetCustomMenuSet ()
Python Example	`oDesktop.GetCustomMenuSet()`

GetDefaultUnit

Returns the default unit for a physical quantity.

<table>
<tr><td><b>UI Access</b></td><td><b>Tools</b> > <b>Options</b> > <b>General Options</b> > <b>Default Units</b>. Note that this menu only displays units that can be changed, while the script can be used to view additional default units.</td></tr>
<tr><td><b>Parameters</b></td><td><table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td>&lt;type&gt;</td><td>String</td><td>String containing a type of measurement.
Valid strings are (case insensitive):
• "Acceleration"
• "Angle"
• "AngularAcceleration"</td></tr>
</table></td></tr>
</table>

			• "AngularDamping" • "AngularSpeed" • "Capacitance" • "Conductance" • "Current" • "CurrentChangeRate" • "DataRate" • "DeltaH" (Magnetic Field Strength) • "Density" • "Flux" • "Force" • "Frequency" • "Inductance" • "Length" • "MagneticReluctance" • "Mass" • "MassFlowRate" • "MomentInertia" • "Power" • "Pressure" • "PressureCoefficient" • "Resistance" • "SaturateMagnetization" (Magnetic Inductance) • "Speed"

			• "Temperature" • "Time" • "Torque" • "Voltage" • "VoltageChangeRate" • "Volume" • "VolumeFlowPerPressureRoot" • "VolumeFlowRate"
Return Value	String containing the default unit (for example, "mm").		

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

GetDesktopConfiguration

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

UI Access	N/A
Parameters	None.
Return Value	String containing current Desktop configuration. For example, 'All', ‘Twin Builder’.

Python Syntax	GetDesktopConfiguration()
Python Example	

	`oDesktop.GetDesktopConfiguration()`

GetDistributedAnalysisMachines

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

UI Access	N/A
Parameters	None.
Return Value	Returns a collection of names of machines used for distributed analysis.

Python Syntax	GetDistributedAnalysisMachines()
Python Example	`oDesktop.GetDistributedAnalysisMachines()`

GetDistributedAnalysisMachinesForDesignType

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

<table>
<tr><td><b>UI Access</b></td><td>NA</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;designTypeName&gt;</td><td>String</td><td>The name of the type of design, such as "Twin Builder", "HFSS", HFSS-IE", "Maxwell 3D" , "Maxwell 2D", "RMxprt", "EM Design", "Circuit", "System",</td></tr></table></td></tr>
</table>

			"Q3D Extractor", "2D Extractor"
Return Value	Object; returns a collection of machine names.		

Python Syntax	GetDistributedAnalysisMachinesForDesignType (<designTypeName>)

Python Example	`machineNames =oDesktop.`
	`GetDistributedAnalysisMachinesForDesignType`
	`("Mechanical")`

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	`oDesktop.GetExeDir()`

GetGDIObjectCount

> **Note:**
> This command is for internal Ansys use only.

Python Syntax	GetGDIObjectCount()
Python Example	`oDesktop.GetGDIObjectCount()`

GetLibraryDirectory

Get the path to the `SysLib` directory.

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

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

GetLocalizationHelper

> **Note:**
> This command is for internal Ansys use only.

Returns the object for the localization helper.

<table>
<tr><td><b>UI Access</b></td><td>NA</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>None</td><td></td><td></td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>Object
Localization helper object, such as "IDispatch(ILocalizationHelper)"</td></tr>
</table>

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

GetMessages

Get the messages from a specified project and design.

<table>
<tr><td><b>UI Access</b></td><td>NA</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><i>&lt;ProjectName&gt;</i></td><td>String</td><td>Name of the project for which to collect messages. An incorrect project name results in no messages (design is ignored). An empty project name</td></tr></table></td></tr>
</table>

		results in all messages (design is ignored)
<*DesignName*>	String	Name of the design in the named project for which to collect messages. An incorrect design name results in no messages for the named project. An empty design name results in all messages for the named project
<*Severity*>	Integer	Severity is 0-3, and is tied in to info/warning/error/fatal types as follows: • 0 – info and above • 1 – warning and above • 2 – error and fatal • 3 – fatal only (rarely used)

Return Value	Array of string messages.

Python Syntax	GetMessages (<*ProjectName*>, <*DesignName*>, <*Severity*>)
Python Example	`Messages = oDesktop.GetMessages("MyProject", "Mechanical1", 1)`

GetMonitorData

Get monitor data. This command is requires a -monitor flag on the ansysedt.exe command line and is used only with the web client.

UI Access	NA
Parameters	

Name	Type	Description
<Request>	String	a json string describing what monitor data is desired. This is a list of triples separated by spaces of the form: `type startposition maxnumber`.

			The `type` must be one of convergence profile `sweptvar` progress variations, `displaytype`, or a guide that was previously returned in a displaytype block. The `startposition` is an integer that specifies where to start in the sequence of monitor data of that type. Each of these types is available in a list, ordered by time. For example, as the simulation goes on, data points are added to the internal list of convergence items. If the caller needs to incrementally update a display, it can repeatedly call this method with the `startposition` for each of the types given as the size of the list it has previously obtained and displayed, which ensures there will be no duplication and the returned data will be new each time. The `maxnumber`, if greater than 0, is the limit on the number of new items returned for each type. If the caller is able to handle any number of returned values, it can be left at 0.
Return Value	a json string with the results.		

Python Syntax	GetMonitorData (<Request>)
Python Example	`Messages = oDesktop.GetMonitorData(request)`

GetPersonalLibDirectory

Get the path to the `PersonalLib` directory.

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

Python Syntax	GetPersonalLibDirectory()
Python Example	`oDesktop.GetPersonalLibDirectory()`

GetProcessID

Returns the process ID of ansysedt.exe.

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

Python Syntax	GetProcessID ()
Python Example	`oDesktop.GetProcessID()`

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	`oDesktop.GetProjectDirectory()`

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	`list_of_projects = oDesktop.GetProjectList()`

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	`oDesktop.GetProjects()`

GetRegistryInt

Obtains registry key integer value.

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><i>&lt;KeyName&gt;</i></td><td>String</td><td>Full name of registry key, including path.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>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.</td></tr>
</table>

Python Syntax	GetRegistryInt(<*KeyName*>)
Python Exampl-e	`num = oDesktop.GetRegistryInt('Desktop/Settings/ProjectOptions/Mechanical/UpdateReportsDynamicallyOnEdits')`

GetRegistryString

Obtains registry key string value.

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

Python Syntax	GetRegistryString(<*KeyName*>)
Python Example	`activeDSO = oDesktop.GetRegistryString('Desktop/ActiveDSOConfigurations/Mechanical')`

GetRunningInstancesMgr

Returns the object of the Running Instances Manager.

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>None</td><td></td><td></td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>Object
Running instances manager object</td></tr>
</table>

Python Syntax	GetRunningInstancesMgr()
Python Example	`oRunningInstances = oDesktop.GetRunningInstancecMgr()`

GetSchematicEnvironment

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

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

Python Syntax	GetSchematicEnvironment()
Python Example	`oDesktop.GetSchematicEnvironment()`

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	`oDesktop.GetScriptingToolsHelper()`

GetSysLibDirectory

Get the path to the `SysLib` directory.

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

Python Syntax	GetSysLibDirectory()
Python Example	`oDesktop.GetSysLibDirectory()`

GetTempDirectory

Gets the path to the `Temp` directory.

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

Python Syntax	GetTempDirectory()
Python Example	`oDesktop.GetTempDirectory()`

GetUserLibDirectory

Gets the path to the UserLib directory.

UI Access	N/A
Parameters	None.
Return Value	Stringpath to the `UserLib` directory.

Python Syntax	GetUserLibDirectory()
Python Example	`oDesktop.GetUserLibDirectory()`

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	`oDesktop.GetVersion()`

IsFeatureEnabled

Returns a Boolean for whether a queried feature is enabled.

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

Python Syntax	IsFeatureEnabled()

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

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

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

  for i in range(len(feature_strs)):

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

result_txt.close()
```

KeepDesktopResponsive

Specifies the minimum number of milliseconds to keep the desktop from showing hung.

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><i>&lt;MinTimeInMilliseconds&gt;</i></td><td>Integer</td><td>The minimum number of milliseconds to keep the desktop window from showing hung, that is, not responding.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>Boolean True for success and to keep running; False to indicate that the calling script should shut down.</td></tr>
</table>

Python Syntax	KeepDesktopResponsive (<*TimeInMilliseconds*>)
Python Example	`oDesktop.KeepDesktopResponsive(10000)`

LaunchJobMonitor

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

<table>
<tr><td><b>UI Access</b></td><td>Launch Job Monitor</td></tr>
<tr><td><b>Parameters</b></td><td><table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td>&lt;projectPath&gt;</td><td>String</td><td>Path to the project file to be monitored</td></tr>
</table></td></tr>
<tr><td><b>Return Value</b></td><td>None</td></tr>
</table>

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

OpenAndConvertProject

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

<table>
<tr><td><b>UI Access</b></td><td>Click <b>File</b> &gt; <b>Open</b>, and choose a legacy project</td></tr>
<tr><td><b>Parameters</b></td><td><table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td>&lt;itemPath&gt;</td><td>String</td><td>full project path of the legacy project</td></tr>
<tr><td>&lt;legacyChoice&gt;</td><td>Integer</td><td>0: show conversion dialog box, (same as <b>File</b> &gt; <b>Open</b> 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)</td></tr>
</table></td></tr>
<tr><td><b>Return Value</b></td><td>An object reference to the newly opened project which has the .aedt extension</td></tr>
</table>

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(*filePath*, *legacyChoice*)

Python Example	`oProject = oDesktop.OpenAndConvertProject("c:\files\optimtee.hfss", 1)`
	Note: optimtee.hfss is gone after this code executes
	`oProject = oDesktop.OpenAndConvertProject("c:\files\optimtee.hfss", 2)`
	Note: optimtee.hfss remains after this code executes

OpenMultipleProjects

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

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td><table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;directory&gt;</i></td><td>String</td><td>Path to the projects.</td></tr>
<tr><td><i>&lt;fileType&gt;</i></td><td>String</td><td>Type of projects to open.</td></tr>
</table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

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

OpenProject

Opens a specified project.

UI Access	Click **File** > **Open**.

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

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

OpenProjectWithConversion

> **Note:**
> This command is for internal Ansys use only.

Python Syntax	OpenProjectWithConversion()
Python Example	`oDesktop.OpenProjectWithConversion()`

PageSetup

Specifies page settings for printing.

UI Access	**File** > **Page Setup**.

<table>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td>&lt;<i>Parameters</i>&gt;</td><td>Array</td><td>Structured array.
[
"NAME:PageSetupData",
"margins:=", <SetupArray>
]</td></tr>
<tr><td>&lt;<i>SetupArray</i>&gt;</td><td>Array</td><td>Structured Array
[
"left:=", <string>,
"right:=", <string>,
"top:=", <string>,
"bottom:=", <string>)
]</td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	PageSetup(<*Parameters*>)
Python Example	oEditor.PageSetup(["NAME:PageSetupData", "margins:=", ["left:=", "10mm",

	`"right:=", "10mm",`
	`"top:=", "10mm",`
	`"bottom:=", "10mm"]`
	`])`

PauseRecording

Temporarily stop script recording.

UI Access	NA
Parameters	Name | Type | Description None | |
Return Value	None

Python Syntax	PauseRecording()
Python Example	`oDesktop.PauseRecording()`

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**

<table>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;<i>Message</i>&gt;</td><td>String</td><td>Any Text.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>None</td></tr>
</table>

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

Print

Prints the contents of the active view window.

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

Python Syntax	Print()
Python Example	`oDesktop.Print()`

QuitApplication

Exits the desktop.

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

Python Syntax	QuitApplication()
Python Example	`oDesktop.QuitApplication()`

RefreshJobMonitor

For use in monitoring a job.

UI Access	**Tools** > **Job Management** > **Monitor Jobs**.
Parameters	None.
Return Value	A string specifing the job state. The result can be any of the following strings: • "Monitor Not Visible" • "Queued" • "Running" • "Shutting Down" • "Unknown"

	- "Completed" - "Not Monitoring" - "Starting Monitoring"

Python Syntax	RefreshJobMonitor()
Python Example	`oDesktop.RefreshJobMonitor()`

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.

Name	Type	Description
<logFile>	String	Path to log file.
<logLevel>	Integer	Specified log level.

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

RestoreProjectArchive

Restores a previously archived project to a specified path.

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

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

RestoreWindow

Restores a minimized Desktop window.

UI Access	N/A

Parameters	None.
Return Value	None.

Python Syntax	RestoreWindow()
Python Example	`oDesktop.RestoreWindow()`

ResumeRecording

Resume recording a script.

UI Access	N/A
Parameters	None.
Return Value	None

Python Syntax	ResumeRecording()
Python Example	`oDesktop.ResumeRecording()`

RunACTWizardScript

> **Note:**
> This command is for internal Ansys use only.

Python Syntax	RunACTWizardScript()
Python Example	`oDesktop.RunACTWizardScript()`

RunProgram

Runs an external program.

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

Python Syntax	RunProgram (<*ProgName*>, <*ProgPath*>, <*WorkPath*>, <*ArgArray*>)

Python Example	`oDesktop.RunProgram("winword.exe", _`
	`"C:\Program Files\Microsoft Office\Office10",_`
	`"", None)`

RunScript

Launches another script from within the script currently being executed.

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

<table>
<tr><td></td><td><table><tr><td></td><td></td><td>This is the syslib subdirectory in the library directory. The library directory can be specified in the <b>General Options</b> dialog box (click <b>Tools</b> &gt; <b>Options</b> &gt; <b>General Options</b> to open this dialog box) under the <b>Project Options</b> tab.
4. HFSS installation directory.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>Long
the return code for the script method.</td></tr>
</table>

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

RunScriptWithArguments

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

<table>
<tr><td><b>UI Access</b></td><td>NA</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;<i>ScriptPath</i>&gt;</td><td>String</td><td>The name or full path of the script to execute. If the full path to the script is not specified, the software looks for the script in the following locations:
• Personal library directory: "PersonalLib"
The PersonalLib directory can be specified in <b>Tools</b> &gt; <b>Options</b> &gt; <b>General Options</b> on the 'Project Options' tab.</td></tr></table></td></tr>
</table>

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

Python Syntax	RunScriptWithArguments (<*ScriptPath*>, <*Arguments*>)

Python Example	`oDesktop.RunScriptWithArguments`
	`("C:/Project/test2.py", "foo")`

SelectScheduler

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

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

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

Python Syntax	Select Scheduler(*<option>, <address>, <username>, <forcePasswordEntry>*)

Python Example	`result = oDesktop.SelectScheduler("Windows HPC",`
	`"headnode.win.example.com")`

SetActiveProject

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

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><i>&lt;ProjectName&gt;</i></td><td>String</td><td>The name of the project already in the Desktop that is to be activated.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>Object, the project that is activated.</td></tr>
</table>

Python Syntax	SetActiveProject (<*ProjectName*>)
Python Example	`oProject = oDesktop.SetActiveProject("Project1")`

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.

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><i>&lt;ProjectName&gt;</i></td><td>String</td><td>The full path name of the project already in the Desktop that is to be activated.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>Object, the project that is activated.</td></tr>
</table>

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

SetCustomMenuSet

Sets the custom menu set for Electronics Desktop.

<table>
<tr><td><b>UI Access</b></td><td>Navigate to <b>Tools</b> &gt; <b>Options</b> &gt; <b>General Options</b> &gt; <b>General</b> &gt; <b>Desktop Configuration</b>. Select a configuration using the <b>Custom Menu Set</b> drop-down menu.</td></tr>
<tr><td><b>Parameters</b></td><td><table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td>&lt;customMenuSet&gt;</td><td>String</td><td>Name of desired menu set. Can be one of:<ul><li>'Default'</li><li>'EM'</li><li>'RF'</li><li>'RF.0'</li><li>'SI'</li><li>'SI1.0'</li><li>'SI2.0'</li><li>'Twin Builder'</li></ul></td></tr>
</table></td></tr>
<tr><td><b>Return Value</b></td><td>None</td></tr>
</table>

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

SetDesktopConfiguration

Sets the desktop configuration.

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

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

SetLibraryDirectory

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

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

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

SetProjectDirectory

Sets the project directory path.

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

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

SetRegistryFromFile

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

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

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

SetRegistryInt

Sets registry key to an integer value.

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

<table>
<tr><td></td><td><table><tr><td><i>&lt;KeyName&gt;</i></td><td>String</td><td>Full name of registry key, including path.</td></tr><tr><td><i>&lt;int&gt;</i></td><td>Integer</td><td>Integer value to be assigned to registry key.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>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.</td></tr>
</table>

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

SetRegistryString

Sets registry key to a string value.

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

Python Syntax	SetRegistryString(<*KeyName*>, <*value*>)

Python Example	`oDesktop.SetRegistryString('Desktop/ActiveDSOConfigurations/Mechanical', 'Local')`

SetSchematicEnvironment

Sets the schematic environment for Electronics Desktop.

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

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

SetTempDirectory

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

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;<i>DirectoryPath</i>&gt;</td><td>String</td><td>The path to the Temp directory. This should be writeable by the user.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

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

ShowDockingWindow

Shows or hides a docking window.

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

Python Syntax	ShowDockingWindow (<*windowName*>, <show>)

Python Example	`oDesktop.ShowDockingWindow('Message Manager',False)`

Sleep

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

<table>
<tr><td><b>UI Access</b></td><td>NA</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><i>&lt;TimeInMilliseconds&gt;</i></td><td>Integer</td><td>The time that the execution should be suspended in milliseconds</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>None</td></tr>
</table>

Python Syntax	Sleep (<*TimeInMilliseconds*>)
Python Example	`oDesktop.Sleep(1000)`

StopSimulations

Either cleanly stops all running and pending simulations, or aborts them.

<table>
<tr><td><b>UI Access</b></td><td>NA</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;bool&gt;
<Clean&gt;</td><td>Integer</td><td>If true, clean stop for all running and pending simulations. If false, aborts them.</td></tr></table></td></tr>
</table>

Return Value	A human readable status string specifying what happened.

Python Syntax	StopSimulations (<bool>, clean)
Python Example	`oDesktop.StopSimulations(bool clean)`

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.

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

Python Syntax	SubmitJob(<*settingsPath*>, <*projectPath*>, <*design*>, <*setup*>)

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

TileWindows

Arrange all open windows in a tiled format.

<table>
<tr><td><b>UI Access</b></td><td>From main menu, <b>Window</b> &gt;<b>Tile Horizontally</b> or <b>Window</b> &gt;<b>Tile Vertically</b>.</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;<i>TilingFlag</i>&gt;</td><td>Integer</td><td>• 0 – Tile vertically.
• 1 – Tile horizontally.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	TileWindows(<*TilingFlag*>)
Python Example	`oDesktop.TileWindows(0)`

Desktop Commands For Registry Values

The Ansys Registry is stored as XML format file. By default it is located at C:\Users\<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 Mechanical.
2. Go to the DSO and HPC options and create a setup named "test".
3. Export the setup to a file (for example, c:\temp\test.acf).
4. Copy the exported file to a target PC (for example, f:\temp\test.acf).
5. Run the following script:

```
#import the setup

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

# Set Active Setup to "test"

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

See the following subtopics:

DeleteRegistryEntry

DoesRegistryValueExist

GetRegistryInt

GetRegistryString

SetRegistryFromFile

SetRegistryInt

SetRegistryString

DeleteRegistryEntry

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

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><i>&lt;pathToRegistry&gt;</i></td><td>String</td><td>Path to Registry entry.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>Boolean:
• <b>True</b> – Key has been deleted.
• <b>False</b> – Key does not exist.</td></tr>
</table>

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

DoesRegistryValueExist

Determines whether a registry value exists.

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><i>&lt;KeyName&gt;</i></td><td>String</td><td>Full name of registry key, including path.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>Boolean:
• <b>True</b> – Key exists.</td></tr>
</table>

	- **False** – Key does not exist.

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

GetRegistryInt

Obtains registry key integer value.

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

Python Syntax	GetRegistryInt(<*KeyName*>)
Python Example	`num = oDesktop.GetRegistryInt('Desktop/Settings/ProjectOptions/Mechanical/UpdateReportsDynamicallyOnEdits')`

GetRegistryString

Obtains registry key string value.

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

Python Syntax	GetRegistryString(<*KeyName*>)
Python Example	`activeDSO = oDesktop.GetRegistryString('Desktop/ActiveDSOConfigurations/Mechanical')`

SetRegistryFromFile

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

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

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

SetRegistryInt

Sets registry key to an integer value.

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

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

SetRegistryString

Sets registry key to a string value.

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

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

ImportExport Tool Commands

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

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

Scripts run via the ImportExport Tool include:

ImportANF

ImportANFV2

ImportAutoCAD

ImportAWRMicrowaveOffice

ImportEDB

ImportExtracta

ImportGDSII

ImportGerber

ImportIDF

ImportIDFandMerge

ImportIPC

ImportODB

ImportXFL

ImportANF

Imports an ANF file into a new project. For older ANFv2 files, use ImportANFv2.

<table>
<tr><td><b>UI Access</b></td><td><b>File</b> &gt; <b>Import</b> &gt; <b>ANF</b>.</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><i>&lt;ANFfilename&gt;</i></td><td>String</td><td>Full path of ANF file.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	ImportANF(<*ANFfilename*>)

Python Example	`oDesktop.RestoreWindow()`
	`Set oTool = oDesktop.GetTool('ImportExport')`
	`oTool.ImportANF('C:\\AnsTranslator\\results\\package4.anf')`

ImportANFv2

Imports an ANFv2 file into a new project. For newer ANF files, use ImportANF.

<table>
<tr><th>UI Access</th><th><b>File</b> &gt; <b>Import</b> &gt; <b>ANF</b>.</th></tr>
<tr><td><b>Parameters</b></td><td><table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;ANFfilename&gt;</i></td><td>String</td><td>Full path of ANF file.</td></tr>
<tr><td><i>&lt;outputPathName&gt;</i></td><td>String</td><td>Full path of *.aedb output file.</td></tr>
<tr><td><i>&lt;controlFileName&gt;</i></td><td>String</td><td>Full path of XML control file.</td></tr>
<tr><td><i>&lt;cmpFileName&gt;</i></td><td>String</td><td>Full path of CMP file. Pass empty string if none.</td></tr>
</table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	ImportANFv2 (<*ANFfilename*>, <*outputPathName*>, <*controlFileName*>, <*cmpFileName*>)

Python Example	`oDesktop.RestoreWindow()`
	`Set oTool = oDesktop.GetTool('ImportExport')`
	`oTool.ImportANFV2("C:/Users/jdoe/Documents/SAMPLEFILES/my_model.anf",`
	`"C:/Users/jdoe/Documents/Ansoft/my_model.aedb",`
	`"C:/Users/jdoe/Documents/Ansoft/my_model.xml", "")`

ImportAutoCAD

Imports an AutoCAD file into a new project.

<table>
<tr><td><b>UI Access</b></td><td><b>File</b> &gt; <b>Import</b> &gt; <b>AutoCAD</b>.</td></tr>
<tr><td><b>Parameters</b></td><td><table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;dxfFileName&gt;</i></td><td>String</td><td>Full path of DXF file.</td></tr>
<tr><td><i>&lt;outputPathFileName&gt;</i></td><td>String</td><td>Full path of EDB file to create during import.</td></tr>
<tr><td><i>&lt;controlFileName&gt;</i></td><td>String</td><td>Full path of XML control file. Pass empty string if none.</td></tr>
</table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

<table>
<tr><td><b>Python Syntax</b></td><td>ImportAutoCAD (<i>&lt;dxfFileName&gt;</i>, <i>&lt;outputPathFileName&gt;</i>, <i>&lt;controlFileName&gt;</i>)</td></tr>
<tr><td><b>Python Example</b></td><td><code>Set oTool = oDesktop.GetTool('ImportExport')</code>
<code>oTool.ImportAutoCAD('C:/MyPath/a4lines.dxf', 'C:/MyPath/a4lines.aedb.edb', 'C:/MyPath/a4lines.xml')</code></td></tr>
</table>

ImportAWRMicrowaveOffice

Imports an AWRMicrowaveOffice file into a new project.

<table>
<tr><td><b>UI Access</b></td><td><b>File</b> &gt; <b>Import</b> &gt; <b>AWRMicrowaveOffice</b>.</td></tr>
<tr><td><b>Parameters</b></td><td><table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;xmlFileName&gt;</i></td><td>String</td><td>Full path of XML file.</td></tr>
<tr><td><i>&lt;outputPathName&gt;</i></td><td>String</td><td>Full path of output file.</td></tr>
<tr><td><i>&lt;logFileName&gt;</i></td><td>String</td><td>Full path of log file.</td></tr>
</table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

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

Python Example	`oDesktop.RestoreWindow()`
	`Set oTool = oDesktop.GetTool('ImportExport')`
	`oTool.ImportAWRMicrowaveOffice('C:/MyFiles/package4.xml', 'C:/MyFiles/package4.aedb', 'C:/MyFiles/package4.log')`

ImportEDB

Imports an EDB file into a new project.

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

Python Syntax	ImportEDB (<*edbFileName*>)

Python Example	`oDesktop.RestoreWindow()`
	`Set oTool = oDesktop.GetTool('ImportExport')`
	`oTool.ImportEDB('C:/MyFiles/projectforimport.aedb/edb.def')`

ImportExtracta

Imports a Cadence Extracta file into a new project.

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

UI Access	**File** > **Import** > **Cadence APD / Allegro / SiP**.
Parameters	<table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><i>&lt;extractaFileName&gt;</i></td><td>String</td><td>Full path of Extracta file.</td></tr><tr><td><i>&lt;outputPathName&gt;</i></td><td>String</td><td>Full path of output file to be created.</td></tr><tr><td><i>&lt;controlFileName&gt;</i></td><td>String</td><td>Optional. Full path to XML control file.</td></tr></table>
Return Value	None.

Python Syntax	ImportExtracta (<*extractaFileName*>, <*outputPathName*>, <*controlFileName*>)

Python Example	`oDesktop.RestoreWindow()`
	`Set oTool = oDesktop.GetTool('ImportExport')`
	`oTool.ImportExtracta('C:/MyFiles/projectforimport.brd', 'C:/MyFiles/project.edb', '')`

ImportGDSII

Imports a GDSII file into a new project.

<table>
<tr><th>UI Access</th><th><b>File</b> &gt; <b>Import</b> &gt; <b>GDSII</b>.</th></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td>&lt;<i>gdsiiFileName</i>&gt;</td><td>String</td><td>Full path of GDSII file.</td></tr>
<tr><td>&lt;<i>outputPathName</i>&gt;</td><td>String</td><td>Full path of EDB file to create during import.</td></tr>
<tr><td>&lt;<i>controlFileName</i>&gt;</td><td>String</td><td>Optional. Full path of XML control file. Full path of “technology” (corresponding to <code>-t=technologyfile</code> argument in anstranslator). Pass empty string if none.</td></tr>
<tr><td>&lt;<i>mapFileName</i>&gt;</td><td>String</td><td>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 anstranslator) or XML control file. Otherwise, pass empty string.</td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	ImportGDSII(<*gdsiiFileName*>, <*outputPathName*>, <*controlFileName*>, <*mapFileName*>)

Python Example	`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')`

ImportGerber

Imports a GERBER file into a new project.

<table>
<tr><td><b>UI Access</b></td><td><b>File</b> &gt; <b>Import</b> &gt; <b>GERBER</b>.</td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;gerberFileName&gt;</i></td><td>String</td><td>Full path of GERBER file.</td></tr>
<tr><td><i>&lt;outputPathName&gt;</i></td><td>String</td><td>Full path of EDB file to create during import.</td></tr>
<tr><td><i>&lt;controlFileName&gt;</i></td><td>String</td><td>Optional. Full path of XML control file. Pass empty string if none.</td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

<table>
<tr><td><b>Python Syntax</b></td><td>ImportGerber(&lt;<i>gerberFileName</i>&gt;, &lt;<i>outputPathName</i>&gt;, &lt;<i>controlFileName</i>&gt;)</td></tr>
<tr><td><b>Python Example</b></td><td><code>oDesktop.RestoreWindow()</code>
<code>Set oTool = oDesktop.GetTool('ImportExport')</code>
<code>oTool.ImportGERBER('C:/Files/test.gbr', 'C:/Files/test.aedb.edb',</code>
<code>'C:/Files/test.xml')</code></td></tr>
</table>

ImportIDF

Imports an IDF file into a new project. To merge with an existing design, use ImportIDFandMerge.

<table>
<tr><td><b>UI Access</b></td><td>placeholder</td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;NAME&gt;</i></td><td>string</td><td>Settings</td></tr>
<tr><td><i>&lt;Board&gt;</i></td><td>bool</td><td>Full path of board file</td></tr>
</table>
</td></tr>
</table>

<*Library*>	int	Full path of library file
<*Control*>	int	Full path of control file
<*Filters*>	list	["Cap","Height","HeightExclude2D","Ind","Power","Res"]
<*CreateFilteredAsNonModel*>	bool	True or False
<*FootPrint*>	string	Footprint size and unit
<*NAME*>	string	definitionOverridesMap
<*NAME*>	string	instanceOverridesMap
<*HighSurfThickness*>	string	High side surface thickness and unit
<*LowSurfThickness*>	string	Low side surface thickness and unit
<*InternalLayerThickness*>	string	Internal layer thickness and unit
<*NumInternalLayer*>	int	Number of internal layers
<*HighSurfaceCopper*>	int	High side surface coverage percentage
<*LowSurfaceCopper*>	int	Low side surface coverage percentage
<*InternalLayerCopper*>	int	Internal layer coverage percentage
<*TraceMaterial*>	string	Trace material name
<*SubstrateMaterial*>	int	Substrate material name
<*CreateBoard*>	bool	True or False
<*ModelBoardAsRect*>	bool	True or False
<*ModelDeviceAsRect*>	bool	True or False
<*Cutoff*>	bool	True or False
<*IncludeDrilledHoles*>	bool	True or False
<*HoleDiameterCutoff*>	string	Hole diameter size and unit
<*ReplaceDevices*>	bool	True or False
<*CreatePointsOnBoardForInstances*>	bool	True or False

Return Value	None.

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

	<*NAME*>, <*HighSurfThickness*>, <*LowSurfThickness*>, <*InternalLayerThickness*>, <*NumInternalLayer*>, <*HighSurfaceCopper*>, <*LowSurfaceCopper*>, <*InternalLayerCopper*>, <*TraceMaterial*>, <*SubstrateMaterial*>, <*CreateBoard*>, <*ModelBoardAsRect*>, <*ModelDeviceAsRect*>, <*Cutoff*>, <*IncludeDrilledHoles*>, <*ReplaceDevices*>)

Python Example

```
oDesign.ImportIDF(
        [
                "NAME:Settings",
                "Board:="               , "C:\\Users\\Model Files\\brd_board.emn",
                "Library:="             , "C:\\Users\\Model Files\\brd_board.emp",
                "Control:="             , "C:\\Users\\Model Files\\brd_board.xml",
                "Filters:="             , ["HeightExclude2D"],
                "CreateFilteredAsNonModel:=", False,
                "FootPrint:="           , "0.1mm2",
                [
                        "NAME:definitionOverridesMap"
                ],
                [
                        "NAME:instanceOverridesMap"
                ],
                "HighSurfThickness:="   , "0.07mm",
                "LowSurfThickness:="    , "0.07mm",
                "InternalLayerThickness:=", "0.07mm",
                "NumInternalLayer:="    , 2,
                "HighSurfaceCopper:="   , 30,
                "LowSurfaceCopper:="    , 30,
                "InternalLayerCopper:=" , 30,
                "TraceMaterial:="       , "Cu-Pure",
                "SubstrateMaterial:="   , "FR-4",
                "CreateBoard:="         , True,
                "ModelBoardAsRect:="    , True,
                "ModelDeviceAsRect:="   , False,
                "Cutoff:="              , False,
                "IncludeDrilledHoles:=" , True,
                "HoleDiameterCutoff:="  , "0.1mm",
                "ReplaceDevices:="      , False,
                "CreatePointsOnBoardForInstances:=", False
        ])
```

ImportIDFandMerge

Imports an IDF file into a new project. To import without merging, use ImportIDF.

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

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

Python Example	`oDesktop.RestoreWindow()`
	`Set oTool = oDesktop.GetTool('ImportExport')`
	`oTool.ImportIDFandMerge('C:/Files/test.brd', 'C:/Files/test.aedb.lib',`

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

ImportIDX

Imports and IDX model into a new Icepak project.

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

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

Python Syntax	ImportIDX (<*NAME*>, <*Board*>, <*Library*>, <*Control*>, <*Filters*>, <*CreateFilteredAsNonModel*>, <*NAME*>, <*NAME*>, <*HighSurfThickness*>, <*LowSurfThickness*>, <*InternalLayerThickness*>, <*NumInternalLayer*>, <*HighSurfaceCopper*>, <*LowSurfaceCopper*>, <*InternalLayerCopper*>, <*TraceMaterial*>, <*SubstrateMaterial*>, <*CreateBoard*>, <*ModelBoardAsRect*>, <*ModelDeviceAsRect*>, <*Cutoff*>, <*ReplaceDevices*>)
Python Example	

```
oDesign.ImportIDX(
        [
                "NAME:Settings",
                "Board:="               , "C:\\Users\\Model files\\PCB-00278_A.idx",
                "Library:="             , "",
                "Control:="             , "",
                "Filters:="             , ["HeightExclude2D"],
                "CreateFilteredAsNonModel:=", False,
                [
                        "NAME:definitionOverridesMap"
                ],
                [
                        "NAME:instanceOverridesMap"
                ],
                "HighSurfThickness:="   , "0.07mm",
                "LowSurfThickness:="    , "0.07mm",
                "InternalLayerThickness:=", "0.07mm",
                "NumInternalLayer:="    , 2,
                "HighSurfaceCopper:="   , 30,
                "LowSurfaceCopper:="    , 30,
                "InternalLayerCopper:=" , 30,
                    "TraceMaterial:="        , "Cu-Pure",
                    "SubstrateMaterial:="    , "FR-4",
                    "CreateBoard:="          , True,
                    "ModelBoardAsRect:="     , False,
                    "ModelDeviceAsRect:="    , False,
                    "Cutoff:="               , False,
                    "ReplaceDevices:="       , False
            ])
```

ImportIPC

Imports an IPC2581 file into a new project.

<table>
<tr><th>UI Access</th><th><b>File</b> &gt; <b>Import</b> &gt; <b>IPC2581</b>.</th></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;ipcFileName&gt;</i></td><td>String</td><td>Full path of IPC2581 file.</td></tr>
<tr><td><i>&lt;outputPathName&gt;</i></td><td>String</td><td>Full path of EDB file to create during import.</td></tr>
<tr><td><i>&lt;controlFileName&gt;</i></td><td>String</td><td>Optional. Full path of XML control file. Pass empty string if none.</td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	ImportIPC(<*ipcFileName*>, <*outputPathName*>, <*controlFileName*>)

Python Example	`oDesktop.RestoreWindow()`
	`Set oTool = oDesktop.GetTool('ImportExport')`
	`oTool.ImportIPC('C:/Files/test.cvg', 'C:/Files/test.aedb',`

	`'C:/Files/test.xml', 'C:/Files/test.txt')`

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.

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.

Python Syntax	ImportODB(<*odbFileName*>, <*outputPathName*>, <*controlFileName*>)

Python Example	`oDesktop.RestoreWindow()`
	`Set oTool = oDesktop.GetTool('ImportExport')`
	`oTool.ImportODB('C:/Files/test.odb', 'C:/Files/test.aedb',`
	`'C:/Files/test.xml')`

ImportXFL

Imports an XFL file into a new project.

<table>
<tr><th>UI Access</th><th><b>File</b> &gt; <b>Import</b> &gt; <b>XFL</b>.</th></tr>
<tr><td><b>Parameters</b></td><td><table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td>&lt;<i>xflFileName</i>&gt;</td><td>String</td><td>Full path of XFL file.</td></tr>
<tr><td>&lt;<i>outputPathName</i>&gt;</td><td>String</td><td>Full path of EDB file to create during import.</td></tr>
<tr><td>&lt;<i>controlFileName</i>&gt;</td><td>String</td><td>Optional. Full path of XML control file. Pass empty string if none.</td></tr>
</table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

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

Python Example	`oDesktop.RestoreWindow()`
	`Set oTool = oDesktop.GetTool('ImportExport')`
	`oTool.ImportXFL('C:/Files/test.xfl', 'C:/Files/test.aedb',`
	`'C:/Files/test.xml')`

5 - Running Instances Manager Script Commands

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

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

GetAllRunningInstances

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.

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;processID&gt;</td><td>Integer</td><td>Process ID</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>String of the returned instance.</td></tr>
</table>

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

GetRunningInstancesMgr

Returns the object of the Running Instances Manager.

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>None</td><td></td><td></td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>Object
Running instances manager object</td></tr>
</table>

Python Syntax	GetRunningInstancesMgr()
Python Example	`oRunningInstances = oDesktop.GetRunningInstancecMgr()`

This page intentionally
left blank.

6 - Project Object Script Commands

Project commands should be executed by the oProject object.

One example of accessing this object is:

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

Dataset Script Commands:

AddDataset

DeleteDataset

EditDataset

ExportDataset

HasDataset

ImportDataset

Other Project Object Script Commands:

AnalyzeAll

ChangeProperty

ClearMessages

CloneMaterial

Close

CopyDesign

CutDesign

DeleteDesign

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

AddDataset

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

<table>
<tr><td><b>UI Access</b></td><td><b>Project</b> &gt; <b>Datasets</b> &gt; <b>Add</b>.</td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;DatasetDataArray&gt;</i></td><td>Array</td><td>Array("NAME:<DatasetName>",
Array("NAME:Coordinates", <CoordinateArray>,
<CoordinateArray>, ...)</td></tr>
<tr><td><i>&lt;DatasetName&gt;</i></td><td>String</td><td>Name of the dataset.</td></tr>
<tr><td><i>&lt;CoordinateArray&gt;</i></td><td>Array</td><td>Array("NAME:Coordinate", "X:=", <double>, "Y:=",<double&gt;)</td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

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
    ]
  ]
]
)
```

AddMaterial

Adds a local material.

<table>
<tr><td><b>UI Access</b></td><td><b>Add Material</b> in the material editor.</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr></table></td></tr>
</table>

<MaterialParams>	Array	["NAME: <name of the material to be added>", <MatProperty>, <MatProperty>, ...]
<MatProperty>	Array	For simple material: "<PropertyName>:=", <value> For anisotropic material: ["NAME:<PropertyName>", "property_type:=", "AnisoProperty", "unit:=", <Unit>", "component1:=", <value>, "component2:=", <value>, "component3:=", <value>))]
<PropertyName>	String	Should be one of the following (depending on the material, design, and solution types): Electromagnetic (Maxwell-exclusive material properties omitted, see Maxwell Scripting help): "permittivity", "permeability", "conductivity", "dielectric_loss_tangent",

			"magnetic_loss_tangent", "electric_coercivity", "magnetic_coercivity", "saturation_mag", "lande_g_factor", "delta_H", "delta_h_freq", "mass_density" Thermal (including solids, Icepak fluid flow, and Mechanical rotating fluid modeling): "thermal_conductivity", "mass_density", "specific_heat", "thermal_expansion_coefficient", "thermal_material_type", "viscosity", "diffusivity", "molecular_mass", "clarity_type" Structural: "mass_density", "youngs_modulus", "poissons_ratio", "thermal_expansion_coefficient"
	<*Unit*>	String	Possible values (Maxwell-exclusive properties omitted, see Maxwell Scripting Help; other missing entries are unitless): conductivity: "siemens/m" saturation_mag: "uTesla", "mTesla", "tesla", "kTesla", "uGauss", "mGauss", "gauss", "kGauss" delta_H: "A_per_meter", "kA_per_meter", "Oe", "kOe" delta_h_frequency: "Hz", "kHz", "MHz", "GHz", "THz", "rps", "per_sec" mass_density: "kg/m^3" thermal_conductivity: "W/m-C" specific_heat: "J/kg-C"

			youngs_modulus: "N/m^2" thermal_expansion_coefficient: "1/C"
Return Value	None		

<table>
<tr><td><b>Python Syntax</b></td><td>AddMaterial (["NAME:<MaterialName>",
<MatProperty>, <MatProperty>, ...])</td></tr>
<tr><td><b>Python Example</b></td><td><pre>
oDefinitionManager.AddMaterial(
["permittivity:=", "2.2", "0.002"])

oDefinitionManager.AddMaterial [("NAME:Material2",_
  "dielectric_loss_tangent:=", "44",
  Array("NAME:saturation_mag",_
    "property_type:=", "AnisoProperty",_
    "unit:=", "Gauss",_
    "component1:=", "11", _
    "component2:=", "22", _
    "component3:=", "33"), _
    "delta_H:=", "44Oe")]
</pre></td></tr>
</table>

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	`oProject.AnalyzeAll()`

ChangeProperty

Changes the properties of an object in the history tree.

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

Python Syntax

ChangeProperty(<*propertyArgs*>)

Python Example

Example: Changing the Position of a Box and the Reference Temperature (for Structural Solutions only)

```
oEditor.ChangeProperty(
[
    "NAME:AllTabs",
    [
        "NAME:Geometry3DCmdTab",
        [
            "NAME:PropServers" ,
            "Box1:CreateBox:1"
        ],
        [
            "NAME:ChangedProps",
            [
                "NAME:Position",
                "X:="          , "0.35in",
                "Y:="          , "0.55in",
                "Z:="          , "0in"
            ],
            [
                "NAME:Reference Temperature",  # Applicable only to Mechanical-
                "Value:="      , "27cel"       # Structural solutions
            ]
        ]
    ]
])
```

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

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

Example: Changing a Box's Material and Wireframe Display

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

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

```
oEditor.ChangeProperty(

  [
     "NAME:AllTabs",

     [
       "NAME:General",
       [
         "NAME:PropServers",
         "LC1_1"
       ],
       [
         "NAME:ChangedProps",
         [
           "NAME:Handle",
           "Value:="              , " NotchOutY"
         ]
       ]
     ]
   ])


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

Example: Change Material Import for a Discovery Model

```
oEditor = oDesign.SetActiveEditor("3D Modeler")
oEditor.ChangeProperty(
  [
    "NAME:AllTabs",
    [
      "NAME:Options",
      [
        "NAME:PropServers",
        "Discovery1"
      ],
      [
        "NAME:ChangedProps",
        [
           "NAME:Materials",
           "Value:="    , "Assignments and Properties"
        ]
      ]
    ]
  ])
```

ClearMessages

Clears information in the **Messages** window.

> **Note:**
>
> Additional options are available when using the Desktop-level ClearMessages command.

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

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

CloneMaterial

Clones a local material.

UI Access	N/A
Parameters	Name | Type | Description *<matName>* | String | Name of existing material. *<newName>* | String | Name for newly cloned material.
Return Value	Boolean: • 1 - Material is cloned.

	• 0 - Existing material not found or a conflict with the new material name.

Python Syntax	CloneMaterial (<*matName*>, <*newName*>)
Python Example	`oDefinitionManager.CloneMaterial("copper1", "copper3")`

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	`oProject.Close()`

CopyDesign

Copies a specified design.

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

Python Syntax	CopyDesign (<*DesignName*>)
Python Example	`oProject.CopyDesign ("HFSSDesign1")`

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**.

<table>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><i>&lt;DesignName&gt;</i></td><td>String</td><td>Name of the design.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	CutDesign (<*DesignName*>)
Python Example	`oProject.CutDesign("SimplorerDesign1")`

DeleteDataset

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

<table>
<tr><td><b>UI Access</b></td><td><b>Project</b> &gt; <b>Datasets</b> &gt; <b>Remove</b>.</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><i>&lt;DatasetName&gt;</i></td><td>String</td><td>Name of the dataset found in the project.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	DeleteDataset (<*DatasetName*>)

Python Example	`oProject.DeleteDataset('$ds1')`
	`oDesign.DeleteDataset('$ds1')`

DeleteDesign

Deletes a specified design in the project.

<table>
<tr><td><b>UI Access</b></td><td><b>Edit</b> &gt; <b>Delete</b>, or <b>Delete</b> in the ribbon.</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><i>&lt;DesignName&gt;</i></td><td>String</td><td>Name of the design.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	DeleteDesign (<*DesignName*>)
Python Example	`oProject.DeleteDesign("MechanicalDesign2")`

DeleteToolObject

> **Note:**
> This command is for internal Ansys use only.

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><i>&lt;ObjectName&gt;</i></td><td>String</td><td>Name of the tool object.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	DeleteToolObject(<*ObjectName*>)
Python Example	`oProject.DeleteToolObject("object1")`

EditDataset

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

<table>
<tr><td><b>UI Access</b></td><td><b>Project</b> &gt; <b>Datasets</b> &gt; <b>Edit</b>.</td></tr>
<tr><td><b>Parameters</b></td><td><table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td>&lt;<i>OriginalName</i>&gt;</td><td>String</td><td>Name of the original dataset.</td></tr>
<tr><td>&lt;<i>DatasetDataArray</i>&gt;</td><td>Array</td><td>Data for the modified dataset.</td></tr>
</table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	EditDataset (<*OriginalName*> <*DatasetDataArray*>)

Python Example	`oProject.EditDataset ("ds1"`
	`["NAME:ds2",`
	`  ["NAME:Coordinates",`
	`    [`
	`      "NAME:Coordinate",`
	`      "X:=", 1, "Y:=", 2`
	`    ],`

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

	)

EditMaterial

Modifies an existing material.

<table>
<tr><th>UI Access</th><th><b>View/Edit Materials</b> command in the material editor</th></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td>&lt;<i>OriginalName</i>&gt;</td><td>String</td><td>Name of the material before editing.</td></tr>
<tr><td>&lt;<i>MatProperties</i>&gt;</td><td>Array</td><td>Structured array containing material properties:
[
  "NAME:<New material name>",
  "CoordinateSystemType:=", <string>,
  "BulkOrSurfaceType:=" , <integer>,
  [
    "NAME:PhysicsTypes",
    "set:=" , <array containing string physics types>
  ],
  <Optional ModifierDataArray>,
  "permeability:=" , <string containing value>,
  "conductivity:=" , <string containing value>,
  "thermal_conductivity:=", <string containing value>,
  "mass_density:=" , <string containing value&gt;,</td></tr>
</table>
</td></tr>
</table>

		"specific_heat:=" , <string containing value>, "youngs_modulus:=" , <string containing value>, "poissons_ratio:=" , <string containing value>, "thermal_expansion_coefficient:=", <string containing value>]
<ModifierDataArray>	Array	Optional structured array containing thermal or spatial modifiers: ["NAME:ModifierData", ["NAME:<ThermalModifierData or SpatialModifierData>", "modifier_data:=" , <"thermal_modifier_data" or "spatial_modifier_data">, ["NAME:<all_thermal_modifiers or all_spatial_modifiers>", ["NAME:<modifierName>", "Property::=" , <string property being modified>, "Index::=" , <integer>, "prop_modifier:=" , <"thermal_modifier" or "spa-

<table>
<tr><td></td><td><table><tr><td></td><td></td><td>   tial_modifier">,
        "use_free_form:=" , <Boolean>,
        "free_form_value:=" , <string modifier value>,
      ]
    ]
  ]
]</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	EditMaterial (<*OriginalName*>, <*MatProperties*>)

Python Example	**Without Modifiers:**
	oDefinitionManager.EditMaterial("alumina_92pct",
	[
	  "NAME:alumina_92pct",
	  "CoordinateSystemType:=" , "Cartesian",
	  "BulkOrSurfaceType:=" , 1,
	  [
	    "NAME:PhysicsTypes",
	    "set:=" , ["Electromagnetic","Thermal","Structural"]
	  ],

```
  "permittivity:=" , "9.3",
  "dielectric_loss_tangent:=" , "0.008",
  "thermal_conductivity:=" , "26",
  "mass_density:=" , "3720",
  "specific_heat:=" , "790",
  "youngs_modulus:=" , "267000000000",
  "poissons_ratio:=" , "0.26",
  "thermal_expansion_coeffcient:=", "7.2e-006"
]
)
```

With Thermal Modifier:

```
oDefinitionManager.EditMaterial("copper",
  [
  "NAME:copper",
  "CoordinateSystemType:=", "Cartesian",
  "BulkOrSurfaceType:=" , 1,
    [
    "NAME:PhysicsTypes",
    "set:=" , ["Electromagnetic","Thermal","Structural"]
    ],
[
"NAME:ModifierData",
  [
  "NAME:ThermalModifierData",
  "modifier_data:=" , "thermal_modifier_data",
    [
    "NAME:all_thermal_modifiers",
      [
      "NAME:one_thermal_modifier",
      "Property::=" , "permittivity",
      "Index::=" , 0,
      "prop_modifier:=" , "thermal_modifier",
      "use_free_form:=" , True,
      "free_form_value:=" , "if(Temp > 1000cel, 1, if(Temp < -273.15cel, 1, 1))"
      ]
    ]
  ]
],
"permeability:=" , "0.999991",
"conductivity:=" , "58000000",
"thermal_conductivity:=" , "400",
  "mass_density:=" , "8933",
  "specific_heat:=" , "385",
  "youngs_modulus:=" , "120000000000",
  "poissons_ratio:=" , "0.38",
  "thermal_expansion_coefficient:=" , "1.77e-05"
])
```

Transient Solve, Non-linear Drude Data Plasma

```
import ScriptEnv
ScriptEnv.Initialize("Ansoft.ElectronicsDesktop")
oDesktop.RestoreWindow()
oProject = oDesktop.SetActiveProject("Drude_plasma_parameters_r231")
oDefinitionManager = oProject.GetDefinitionManager()
oDefinitionManager.EditMaterial("Drude",
  [
    "NAME:Drude",
    "CoordinateSystemType:=", "Cartesian",
    "BulkOrSurfaceType:="   , 1,
    [
      "NAME:PhysicsTypes",
      "set:="                  , ["Electromagnetic"]
    ],
    [
        "NAME:AttachedData",
        [
            "NAME:MatNonLinearDrudeFreqDepData",
            "property_data:="      , "nonlinear_drude_data",
            "EpsilonInfinity:="    , "1",
            "PlasmaFrequency:="    , "4.62348462366278GHz",
            "CollisionFrequency:="  , "0.00054491190162662GHz",
            "FieldBreakdown:="     , "10000V_per_meter",
            "PlasmaMaintainFrequency:=", "2.31174231183139GHz",
            "NeutralDensity:="     , 2.65164580488373E+20,
            "ElectronDensity:="    , 2.65164580488373E+17,
            "CollisionRateConstant:=", 2.05499505485618E-15
        ]
    ]
])
```

ExportDataset

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

<table>
<tr><td>**UI Access**</td><td>**Project** > **Datasets** > **Export**.</td></tr>
<tr><td>**Parameters**</td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><*datasetFileFullPath*></td><td>String</td><td>The full path to the file.</td></tr></table></td></tr>
<tr><td>**Return Value**</td><td>None.</td></tr>
</table>

<table>
<tr><td>**Python Syntax**</td><td>ExportDataset (<*datasetFileFullPath*>)</td></tr>
<tr><td>**Python Example**</td><td>`oProject.ExportDataset('e:/tmp/dsdata.txt')`
`oDesign.ExportDataset('e:/tmp/dsdata.txt')`</td></tr>
</table>

ExportMaterial

Exports a local material to a library.

<table>
<tr><td>**UI Access**</td><td>**Export to Library** command in the material editor.</td></tr>
<tr><td>**Parameters**</td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><*ExportData*></td><td>Array</td><td>["NAME:<LibraryName>",
<MaterialName>, <MaterialName>, ...]</td></tr><tr><td><*LibraryName*></td><td>String</td><td>Name of the exported library.</td></tr><tr><td><*MaterialName*></td><td>String</td><td>Name of the material to be exported.</td></tr><tr><td><*LibraryLocation*></td><td>String</td><td>Location to save the library. Only "PersonalLib" and "UserLib" are allowed.</td></tr></table></td></tr>
<tr><td>**Return Value**</td><td>None.</td></tr>
</table>

Python Syntax	ExportMaterial (<*ExportData*>, <*LibraryLocation*>)

Python Example	`oDefinitionManager.ExportMaterial (["NAME:mylib",_`
	`"Material1", "Material2", "Material3"], "PersonalLib")`

GetActiveDesign

Returns the design in the active project

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

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

Python Syntax	GetActiveDesign()
Python Example	`oDesign = oProject.GetActiveDesign()`

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	`oProject.GetArrayVariables()`
	`oDesign.GetArrayVariables()`

GetChildNames [Project]

Returns the names of the project's child objects.

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

Python Syntax	GetChildNames (<*type*>)
Python Example	`arrDesignNames = oProject.GetChildNames()`

	`arrVarbleNames = oProject.GetChildNames("Variable")`

GetChildObject [Project]

Returns the project's child objects.

> **Note:**
>
> `GetChildObject` will return normally if there are no active objects.

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

Python Syntax	GetChildObject(*<path>*)

Python Example	`oProject = oDeskop.GetActiveProject()`
	`oDesign = oProject.GetChildObject(“TeeModel”)`
	`oVariable = oProject.GetChildObject("VariableName")`
	`oReport = oProject.GetChildObject("TeeModel/Results/S Parameter Plot 1")`

GetChildTypes [Project]

Returns the types of the project's child objects.

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

Python Syntax	GetChildTypes()
Python Example	`oProject.GetChildTypes()`

GetDefinitionManager

Gets the `DefinitionManager` object.

UI Access	N/A
Parameters	None.
Return Value	`DefinitionManager` object.

Python Syntax	GetDefinitionManager()

Python Example	`oDefinitionManager = oProject.GetDefinitionManager()`

> **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	`files = oProject.GetDependentFiles()`

GetDesign

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

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><i>&lt;designName&gt;</i></td><td>String</td><td>Name of the design.</td></tr></table></td></tr>
</table>

Return Value	Object of the specified design.

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

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	`oProject.GetDesigns()`

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	`oProject.GetEDBHandle()`

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	`oProject.GetLegacyName()`

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	`oProject.GetName()`

GetObjPath [Project]

Obtains the project name from the full path.

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

Python Syntax	GetObjPath()
Python Example	`oProject.GetObjPath()`

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	`oProject.GetPath()`

GetPropEvaluatedValue

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

> **Note:**
>
> This command is not supported by the EMIT and Circuit design types.

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><i>&lt;PropName&gt;</i></td><td>String</td><td>Name of the property.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>String value of the evaluated value.</td></tr>
</table>

Python Syntax	GetPropEvaluatedValue (<*PropName*>)

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

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.

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><i>&lt;includeReadOnly&gt;</i></td><td>Boolean</td><td>(Optional)</td></tr></table></td></tr>
</table>

	<table><tr><td></td><td></td><td><ul><li>True – Include read only props.</li><li>False – Do not include read only props.</li></ul></td></tr></table>
Return Value	Empty array.

Python Syntax	GetPropNames ()
Python Example	`oProject.GetPropNames()`

GetPropSIValue

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

> **Note:**
>
> This command is not supported by the EMIT and Circuit design types.

UI Access	N/A
Parameters	<table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><i>&lt;PropName&gt;</i></td><td>String</td><td>Name of the property.</td></tr></table>
Return Value	Property value as a double floating value, or NAN if the property value cannot be converted to double floating point.

Python Syntax	GetPropSIValue (<*PropName*>)

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

GetPropValue [Project]

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

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td><table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;propPath&gt;</i></td><td>String</td><td>A child object's property path. See: Property Function Summary.</td></tr>
</table></td></tr>
<tr><td><b>Return Value</b></td><td>String of property value.</td></tr>
</table>

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

GetProperties

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

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;PropTab&gt;</i></td><td>String</td><td>One of the following, where tab titles are shown in parentheses:<ul><li>PassedParameterTab ("Parameter Values")</li><li>DefinitionParameterTab (Parameter Defaults")</li><li>LocalVariableTab ("Variables" or "Local Variables")</li><li>ProjectVariableTab ("Project variables")</li><li>ConstantsTab ("Constants")</li><li>BaseElementTab ("Symbol" or "Footprint")</li><li>ComponentTab ("General")</li><li>Component("Component")</li><li>CustomTab ("Intrinsic Variables")</li><li>Quantities ("Quantities")</li><li>Signals ("Signals")</li></ul></td></tr>
<tr><td><i>&lt;PropServer&gt;</i></td><td>String</td><td>An object identifier, generally returned from another script method, such as CompInst@R;2;3</td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>Array of strings containing the names of the appropriate properties.</td></tr>
</table>

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

GetPropertyValue

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

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

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

Return Value	String value of the property.

Python Syntax	GetPropertyValue (<*PropTab*>, <*PropServer*>, <*PropName*>)

Python Example	`selectionArray = oEditor.GetSelections()`
	`for k in selectionArray:`
	`val = oEditor.GetPropertyValue("PassedParameterTab", k, "R")`
	`...`

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	`oProject.GetTopDesignList()`

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`.

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><i>&lt;VarName&gt;</i></td><td>String</td><td>Name of the variable to access.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>String represents the value of the variable.</td></tr>
</table>

Python Syntax	GetVariableValue(<*VarName*>)
Python Example	`oProject.GetVariableValue("var_name")`

GetVariables

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

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

Python Syntax	GetVariables ()

Python Example	`oProject.GetVariables()`
	`oDesign.GetVariables()`

ImportDataset

Imports a dataset from a named file. This can be executed by the oProject, or oDesign variables. The name of the dataset is filename+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.

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

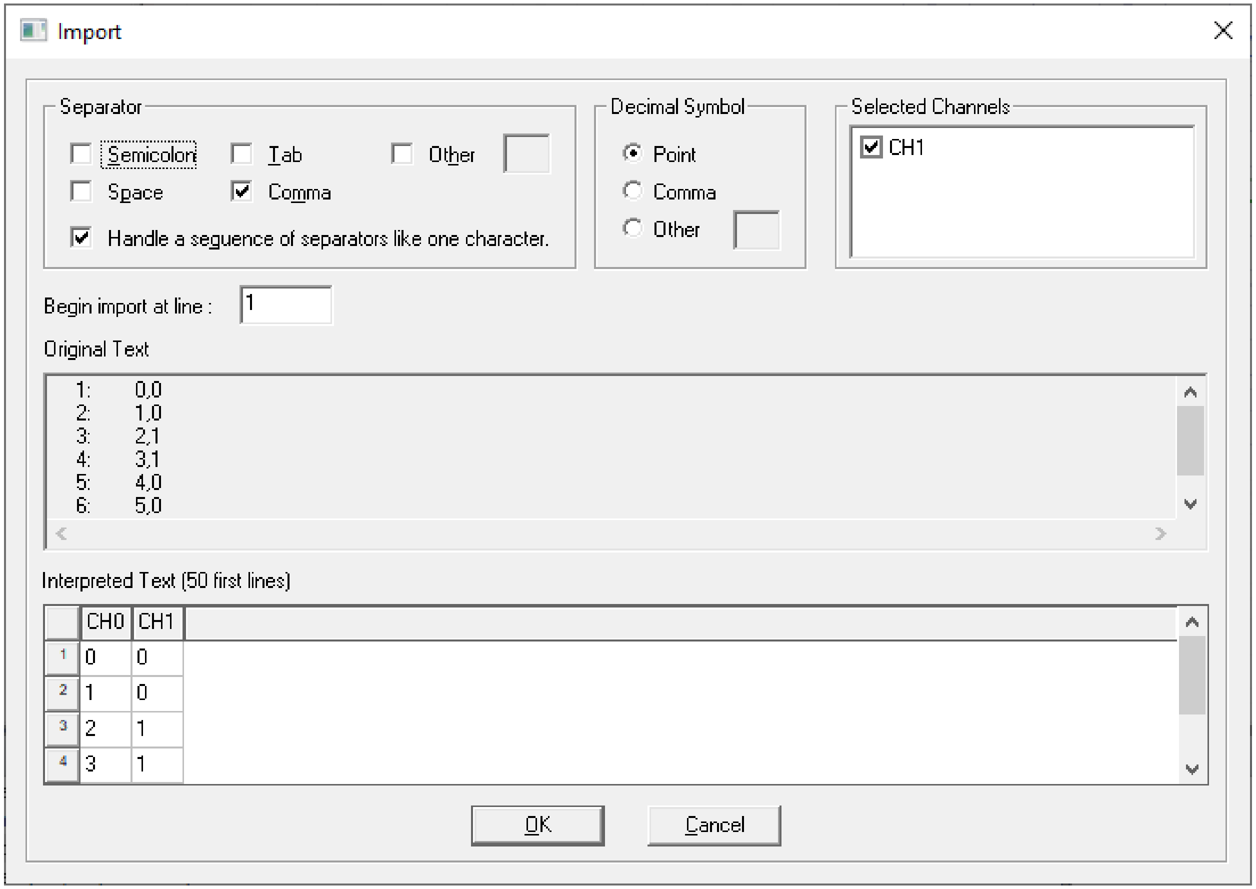
Python Syntax	ImportDataset (<*datasetFileFullPath*>,<*optionalDatasetName*>)

Python Example	`oProject.ImportDataset('e:\tmp\dsdata.tab')`
	`oDesign.ImportDataset('e:\tmp\dsdata.tab')`
	`oProject.ImportDataset('e:\tmp\dsdata.tab', 'MyDatasetName')`
	`oDesign.ImportDataset('e:\tmp\dsdata.tab', 'MyDatasetName')`

Note About File Types:

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

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



Separator

□ Semicolon □ Tab □ Other

□ Space ☑ Comma

☑ Handle a sequence of separators like one character.

Decimal Symbol

● Point

○ Comma

○ Other

Selected Channels

☑ CH1

Begin import at line : 1

Original Text

```
1:      0,0
2:      1,0
3:      2,1
4:      3,1
5:      4,0
6:      5,0
```

Interpreted Text (50 first lines)

	CH0	CH1
1	0	0
2	1	0
3	2	1
4	3	1

InsertDesignWithWorkflow

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

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

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

Python Example	`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")`

InsertToolObject

> **Note:**
> This command is for internal Ansys use only.

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

Paste (Project Object)

Pastes a design in the active project.

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

Python Syntax	Paste()
Python Example	`oProject.Paste()`

Redo [Project Level]

Reapplies the last project-level command.

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

Python Syntax	Redo()
Python Example	`oProject.Redo()`

RemoveAllUnusedDefinitions

Removes all unused project definitions.

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

Python Syntax	RemoveAllUnusedDefinitions()
Python Example	`oProject.RemoveAllUnusedDefinitions()`

RemoveMaterial

Removes a material from a library.

UI Access	**Remove Material(s)** command in the material editor
Parameters	<table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><i><MaterialName></i></td><td>String</td><td>Name of the material to be removed.</td></tr><tr><td><i><IsProjectMaterial></i></td><td>Boolean</td><td>If True, assumes the material is a project material. The last two para-</td></tr></table>

		meters will be ignored. If False, the material is not a project material.
<*LibraryName*>	String	Name of the user or personal library where the material resides.
<*LibraryLocation*>	String	Location of library. Valid options:"UserLib" or "PersonalLib".

Return Value	None.

Python Syntax	RemoveMaterial (<*MaterialName*>, <*IsProjectMaterial*>, <*LibraryName*>, <*LibraryLocation*>)

Python Example	`oDefinitionManager.RemoveMaterial ([`
	`"Material1", false, "mo0907","UserLib"])`

RemoveUnusedDefinitions

Removes any unused project definitions.

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

Name	Type	Description
<*Definitions*>	Array	Definitions to be removed, such as materials and surface materials.

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

```
[
  [
    "NAME:Materials",
    "Al-Extruded"
  ],
  [
    "NAME:SurfaceMaterials",
    "Steel-oxidised-surface"
  ]
])
```

Rename

Renames the project and saves it. Similar to SaveAs().

<table>
<tr><td><b>UI Access</b></td><td><b>Edit</b> &gt; <b>Rename</b>.</td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;NewName&gt;</i></td><td>String</td><td>Desired name of the project. The path is optional.</td></tr>
<tr><td><i>&lt;OverWriteOk&gt;</i></td><td>Boolean</td><td><ul><li>True - overwrite the file on disk if it exists.</li><li>False - prevent overwrite.</li></ul></td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

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

Save

Saves the active project.

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

Python Syntax	Save()
Python Example	`oProject.Save()`

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.

<table>
<tr><th>UI Access</th><th><b>File</b> &gt; <b>Save As</b>.</th></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td>&lt;<i>NewName</i>&gt;</td><td>String</td><td>The desired name of the project, with directory and extension.</td></tr>
<tr><td>&lt;<i>OverWriteOK</i>&gt;</td><td>Boolean</td><td>True to overwrite the file of the same name, if it exists. False to prevent overwrite.</td></tr>
<tr><td>&lt;<i>DefaultAction</i>&gt;</td><td>String</td><td>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.</td></tr>
<tr><td>&lt;<i>OverwriteActions</i>&gt;</td><td>Array</td><td>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.</td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	For non-Schematic/Layout project: SaveAs (<*NewName*> <*OverWriteOK*>) For Schematic/Layout project: SaveAs (<*NewName*> <*OverWriteOK*> <*DefaultAction*> <*OverrideActions*>)

Python Example	`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']]]])`

> **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	<table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>*&lt;projectName&gt;*</td><td>String</td><td>The desired name of the project, with directory and extension.</td></tr></table>
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.

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

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

SetActiveDefinitionEditor

Obtains a specified definition editor.

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td><table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td>&lt;<i>EditorName</i>&gt;</td><td>String</td><td>Name of the definition editor to set active, one of "SymbolEditor", "FootprintEditor".</td></tr>
<tr><td>&lt;<i>DefinitionName</i>&gt;</td><td>String</td><td>The combination name for the symbol or footprint, &lt;<i>libname</i>&gt;:&lt;<i>defname</i>&gt;</td></tr>
</table></td></tr>
<tr><td><b>Return Value</b></td><td>Object for the definition to be edited.</td></tr>
</table>

<table>
<tr><td><b>Python Syntax</b></td><td>SetActiveDefinitionEditor(&lt;<i>EditorName</i>&gt;, &lt;<i>DefinitionName</i>&gt;)</td></tr>
<tr><td><b>Python Example</b></td><td><code>oProject.SetActiveDefinitionEditor("SymbolEditor",</code>
<code>"Simplorer Elements\Basic Elements\Circuit\Passive Elements:R")</code></td></tr>
</table>

SetActiveDesign

Sets a design to be the active design.

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td><table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td>&lt;<i>DesignName</i>&gt;</td><td>String</td><td>Name of the design to set as the active design.</td></tr>
</table></td></tr>
</table>

Return Value	None.

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

SetPropValue [Project]

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

<table>
<tr><td><b>UI Access</b></td><td>Edit Properties on ProjectTree objects.</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><i>&lt;propPath&gt;</i></td><td>String</td><td>A child object's property path. See: Property Function.</td></tr><tr><td><i>&lt;newValue&gt;</i></td><td>String</td><td>New property value.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>Boolean:
• <b>True</b> – property found.
• <b>False</b> – property not found.</td></tr>
</table>

Python Syntax	SetPropValue(<*propPath*>, <*newValue*>)

Python Example	`oProject.SetPropValue("TeeModel/offset", "2mm")`
	`oProject.SetPropValue("TeeModel/Results/S Parameter Plot 1/Display Type", "Data`

	`Table")`

SetPropertyValue

Sets the value of a single propertybelonging 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.

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

	<propName>	String	Name of the property.
	<propValue>	String	The value for the property
Return Value	None.		

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

SetVariableValue

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

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

Python Syntax	SetVariableValue (<*VarName*>, <*VarValue*>)
Python Example	`oProject.SetVariableValue('$Var1', '3mm')`

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	`oProject.SimulateAll()`

Undo [Project]

Cancels the last project-level command.

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

Python Syntax	Undo()
Python Example	`oProject.Undo()`

UpdateDefinitions

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

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

Python Syntax	UpdateDefinitions()
Python Example	`oProject.UpdateDefinitions()`

7 - Design Object Script Commands

Design object commands should be executed by the oDesign object.

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

For example:

Conventions Used in this Chapter

`<ModuleName>` is a placeholder for any one of the following modules:

- Analysis Module – "AnalysisSetup"
- Boundary Module – "BoundarySetup"
- Field Overlays Module – "FieldsReporter"
- Mesh Module – "MeshSetup"
- Optimetrics Module – "Optimetrics"
- Radiation Module – "RadField"
- Reduce Matrix Module – "ReduceMatrix"
- Reporter Module – "ReportSetup"
- Simulation Setup Module – "SimSetup"
- Solutions Module – "Solutions"

ApplyMeshOps

ConstructVariationString

CopyArray

DeleteFieldVariation

DeleteFullVariation

DeleteLinkedDataVariation

EditDesignSettings

EditNotes

ExportConvergence

ExportLayoutViaCurrentDensity

ExportMatrixData

ExportMeshStats

ExportProfile

ExportReport

GenerateMesh

GetAllPorts

GetChildNames [Design]

GetChildObject [Design]

GetChildTypes [Design]

GetDesignType

GetManagedFilesPath

GetModule

GetName

GetNominalVariation

GetNoteText

GetProject

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
```

	]]])

AddModelingProperties

Use: Add a modeling property to a design

Command: None

Syntax: AddModelingProperties <design>

Return Value: None

Parameters: <design>

Type: string

AnalyzeAll [design]

Runs all solution setups and Optimetrics setups for the current design instance.

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>isBlocking</td><td>Boolean</td><td>An optional arg that defaults to <code>true</code>. If it's <code>false</code>, the command immediately returns while the analysis or analyses run in the background. The status of the analyses can be checked with the <code>AreThereSimulationsRunning</code> command.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>None</td></tr>
</table>

Python Syntax	AnalyzeAll()
Python Example	`oDesign.AnalyzeAll(isBlocking)`

AnalyzeAllNominal

Runs all defined setups.

<table>
<tr><td><b>UI Access</b></td><td>Right-click <b>Analysis</b> in the project tree, and select <b>Analyze All</b>.</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>isBlocking</td><td>Boolean</td><td>An optional arg that defaults to <code>true</code>. If it's <code>false</code>, the command immediately returns while the analysis or analyses run in the background. The status of the analyses can be checked with the <code>AreThereSimulationsRunning</code> command.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>None</td></tr>
</table>

Python Syntax	AnalyzeAllNominal()
Python Example	`oDesign.AnalyzeAllNominal()`

ConstructVariationString

Lists and orders the variables and values associated with a design variation.

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td><table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;ArrayOfVariableNames&gt;</i></td><td>Array of Strings</td><td>List of variable names.</td></tr>
<tr><td><i>&lt;ArrayOfVariableValuesIncludingUnits&gt;</i></td><td>Array of Strings</td><td>List of variable values, including units, in the same order as the list of names.</td></tr>
</table></td></tr>
<tr><td><b>Return Value</b></td><td>Returns variation string with the variables ordered to correspond to the order of variables in design variations. The values for the variables are inserted into the variation string. For an example of how ConstructionVariationString can be used, see the Python example.</td></tr>
</table>

Python Syntax	ConstructVariationString(<*ArrayOfVariableNames*>, <*ArrayOfVariableValuesIncludingUnits*>)

Python Example	`varStr = oDesign.ConstructVariationString(["xx", "yy"], ["2mm", "1mm"])`
	`oDesign.ExportProfile("Setup1",  varStr, "C:\profile.prof")`

CopyArray

Copies a specified array.

<table>
<tr><td><b>UI Access</b></td><td><b>Edit</b> &gt; <b>Copy</b>.</td></tr>
<tr><td><b>Parameters</b></td><td><table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;ArrayName&gt;</i></td><td>String</td><td>Name of the array to copy.</td></tr>
</table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	Copy (<*ArrayName*>)

Python Example	`oModule = oDesign.GetModule("ModelSetup")`
	`oModule.CopyArray("Array")`
	`oModule.PasteArray() # returns the new array name`

CopyItemCommand

Use: Copy tree items, such as Altrasim Solution Setups in Nexxim, or Solve Setups and Frequency Sweeps in Ensemble.

Command: None

Syntax: CopyItemCommand <ItemPathList>

Return Value: None

Parameters: <ItemPathList>

Type: Array of strings

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')`

DeleteFieldVariation

Deletes field variations, fields and meshes, or fields and meshes for specified variations.

<table>
<tr><td><b>UI Access</b></td><td>[<b>Solver</b>] &gt; <b>Results</b> &gt; <b>Clean Up Solutions</b> &gt; [<b>Fields Only</b> / <b>Fields and Meshes</b>].</td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td>&lt;<i>variationKeys</i>&gt;</td><td>Array</td><td>Array containing either "All" string to select all variations, or strings of variations to include.</td></tr>
<tr><td>&lt;<i>deleteMesh</i>&gt;</td><td>Boolean</td><td>When true, deletes mesh data.</td></tr>
<tr><td>&lt;<i>deleteLinkedData</i>&gt;</td><td>Boolean</td><td>When true, deletes linked data.</td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	DeleteFieldVariation(<*variationKeys*>, <*deleteMesh*>, <*deleteLinkedData*>

Python Example	`oProject = oDesktop.GetActiveProject()`
	`oDesign = oProject.GetActiveDesign()`
	`Design.DeleteFieldVariation(['All'], True, False)`

DeleteFullVariation

Deletes either all solution data, or selected variation data.

<table>
<tr><td><b>UI Access</b></td><td>[<b>Mechanical</b>] &gt; <b>Results</b> &gt; <b>Clean Up Solutions</b>.</td></tr>
<tr><td><b>Parameters</b></td><td><table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;variationKeys&gt;</i></td><td>Array</td><td>Array containing either "All" string to select all variations, or strings of variations to include.</td></tr>
<tr><td><i>&lt;deleteLinkedData&gt;</i></td><td>Boolean</td><td>When true, deletes linked data as well.</td></tr>
</table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	DeleteFullVariation(<*variationKeys*>, <*deleteLinkedData*>)
Python Example	`oDesign.DeleteFullVariation(['All']), false)`

DeleteLinkedDataVariation

Deletes the linked data of specified variations.

<table>
<tr><td><b>UI Access</b></td><td>[<b>Mechanical</b>] &gt; <b>Results</b> &gt; <b>Clean Up Solutions</b>.</td></tr>
<tr><td><b>Parameters</b></td><td><table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;DesignVariationKeys&gt;</i></td><td>Array</td><td>Array containing strings of the variations keys whose linked data are going to be deleted.</td></tr>
</table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	DeleteLinkedDataVariation(<*DesignVariationKeys*>)
Python Example	`oDesign.DeleteLinkedDataVariation(["current=\'0.9mA\'", "current=\'1.0mA\'"])`

DeleteOutputVariable

Deletes an existing output variable. The variable can only be deleted if it is not in use by any traces.

<table>
<tr><td><b>UI Access</b></td><td><b>Mechanical</b> &gt; <b>Results</b> &gt; <b>Output Variables</b>. In the <b>Output Variables</b> window, click <b>Delete</b>.</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;<i>OutputVarName</i>&gt;</td><td>String</td><td>Name of the output variable.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	DeleteOutputVariable (<*OutputVarName*>)

Python Example	`oModule = oDesign.GetModule("OutputVariable")`
	`oModule.DeleteOutputVariable ("testNew")`

DeletePlotEntities

Delete plot entities including limit line, marker, X-marker, Y-marker and note.

UI Access	[**Edit**] > **Delete**.

<table>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;plotname&gt;</i></td><td>string</td><td>Name of the plot containing entities to delete.</td></tr>
<tr><td>&lt;<i>"NAME:PlotEntityNames"</i>
,"<PlotEntityName&gt;"</td><td>string</td><td>limit line, marker, X-marker, Y-marker and not</td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	DeletePlotEntities(<*plotName*>, ["NAME:PlotEntityNames", "<*PlotEntityName*>"]])

Python Example

```
oProject = oDesktop.SetActiveProject("Tee-23R2-01")
oDesign = oProject.SetActiveDesign("TeeModel")
oModule = oDesign.GetModule("ReportSetup")
oModule.DeletePlotEntities("Z Parameter Plot 1-01",
  [
    [
    "NAME:PlotEntityNames",
    "LimitLine1"
    ]
  ])
```

DeleteSolutionVariation

Deletes all solution data for specific solutions and design variations. This is obsolete and is supported only for backward compatibility. You should use DeleteFullVariation.

<table>
<tr><td><b>UI Access</b></td><td>Right-click on <b>Results</b>, select <b>Browse Solutions...</b>, click <b>Delete</b> button in the dialog.</td></tr>
<tr><td><b>Parameters</b></td><td><table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i><SoluParams></i></td><td>Array</td><td>Structured array.
Array(<DataSpecifierArray>, ...)</td></tr>
<tr><td><i><DataSpecifierArray></i></td><td>Array</td><td>Structured array.
Array(<DesignVariationKey>, <SetupName>, <SolnName>)</td></tr>
</table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

<table>
<tr><td><b>Python Syntax</b></td><td>DeleteSolutionVariation(<<i>SoluParams</i>>)</td></tr>
<tr><td><b>Python Example</b></td><td>oModule.DeleteSolutionVariation([
  ["width=’2in’", "Setup1", "Adaptive_1"],
  ["width=’2in’", "Setup1", "Sweep1"]
])</td></tr>
</table>

DeleteOutputVariable

Deletes an existing output variable. The variable can only be deleted if it is not in use by any traces.

<table>
<tr><td><b>UI Access</b></td><td><b>Mechanical</b> > <b>Results</b> > <b>Output Variables</b>. In the <b>Output Variables</b> window, click <b>Delete</b>.</td></tr>
<tr><td><b>Parameters</b></td><td><table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
</table></td></tr>
</table>

	<*OutputVarName*>	String	Name of the output variable.
Return Value	None.		

Python Syntax	DeleteOutputVariable (<*OutputVarName*>)

Python Example	`oModule = oDesign.GetModule("OutputVariable")`
	`oModule.DeleteOutputVariable ("testNew")`

EditCoSimulationOptions

Sets options for cosimulation.

Command: None

Syntax: EditCoSimulationOptions <array_name>

Return Value: None

Parameters: <array_name>

Type: string

EditInfiniteArray

Edits the properties of an infinite array

Command: None

Syntax: EditInfiniteArray <array_name>

Return Value: None

Parameters: <array_name>

Type: string

EditLayoutForLayoutComponent

Opens the definition component for editing in HFSS 3D Layout for a Layout Component in an HFSS 3D design.

<table>
<tr><td><b>UI Access</b></td><td><b>Edit Layout...</b></td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;<i>ComponentDefinitionName</i>&gt;</td><td>String</td><td>Name of the Layout Component Definition.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	EditLayoutForLayoutComponent ([Name:LayoutComponent EDB", Definitions:=", [<*ComponentDefinitionName*>])

Python Example	`oDesign.EditLayoutForLayoutComponent (`
	`  [`
	`    "NAME:Layout Component EDB",`
	`    "Definitions:="         , ["Diff_Via_diffViaNominal"]`
	`  ])`

EMDesignOptions

Use: Set options for an EM Design.

Command: Right click on design and select **EM Design Options**

Syntax: DesignOptions <Options Array>

Return Value: None

Parameters:

<Options Array>

```
Array("NAME:options",
"SaveSolFilesAsBinary:=", <boolean>,
"LowPriorityForSimulations:=", <boolean>,
"SaveNearFieldSolutions:=", <boolean>,
"SchematicEnabled:=", <boolean>,
"UseGlobalNumProc:=", <boolean>,
"ComputeBothEvenAndOddCPWModes:=", <boolean>,
"NumProcessors:=", <int>,
"NumProcessorsDistrib:=", <int>,
"CausalMaterials:=", <boolean>,
"UseHPCForMP:=", <boolean>,
"HPCLicenseType:=", <int>)
```

SaveSolFilesAsBinary - if true, solutions files are saved using a binary format.

LowPriorityForSimulations - if true, run simulations at a lower CPU priority.

SaveNearFieldSolutions - if true, save near field solutions

SchematicEnabled - if true, enable schematics

UseGlobalNumProc - if true, use global number of processors and ignore NumProcessors

ComputeBothEvenAndOddCPWModes - if true, compute both even and odd cpw modes

NumProcessors - number of processors

NumProcessorsDistrib- number of distributed processors

CausalMaterials - if true, use causal materials

UseHPCForMP - if true, use hpc for mp

HPCLicenseType - number indicating hpc license type

ExportConvergence

For a given variation, exports convergence data (max mag delta S, E, freq) to *.conv file.

<table>
<tr><td><b>UI Access</b></td><td><b>Results</b> > <b>Solution Data</b>. Select <b>Convergence</b> tab and click <b>Export</b>.</td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;Setup&gt;</i></td><td>String</td><td>Setup name.</td></tr>
<tr><td><i>&lt;VariationKeys&gt;</i></td><td>String</td><td>The variation. Pass empty string for the current nominal variation.</td></tr>
<tr><td><i>&lt;FilePath&gt;</i></td><td>String</td><td>Full path to desired *.conv file location.</td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	
Python Example	

ExportArray

Exports an array definition to a csv file.

UI Access	Export Array to CSV File
Parameters	<filepath>.
Return Value	None.

Python Syntax	ExportArray()

Python Example	`oModule = oDesign.GetModule("ModelSetup")`
	`oModule.ExportArray("Array", "C:\\Users\\MyFolder\Export Array3.csv")`

ExportDataset

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

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

Python Syntax	ExportDataset (<*datasetFileFullPath*>)

Python Example	`oProject.ExportDataset('e:/tmp/dsdata.txt')`
	`oDesign.ExportDataset('e:/tmp/dsdata.txt')`

ExportProfile

Exports a solution profile to file.

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td>&lt;<i>SetupName</i>&gt;</td><td>String</td><td>Text of the design's notes.</td></tr>
<tr><td>&lt;<i>VariationString</i>&gt;</td><td>String</td><td>Variation name. Pass empty string for the current nominal variation.</td></tr>
<tr><td>&lt;<i>filePath</i>&gt;</td><td>String</td><td>Full file path, including extension *.prof.</td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	
Python Example	

GetChildNames [Design]

Returns the names of the design's child objects.

UI Access	N/A
Parameters	Name | Type | Description *<type>* | String | Optional. Valid options are "Module", "Editor", "Variable". Default returns Module names.
Return Value	Names of children for the queried object).

#160;

Python Syntax	GetChildNames (<*type*>)
Python Example	`oDesign.GetChildNames()`

GetChildObject [Design]

Returns the design's child objects.

UI Access	N/A
Parameters	Name | Type | Description *<path>* | String | The path may include multiple generations (for example, "designObject/moduleObj/SetupObject"). See: Object Path.
Return Value	A child object if one is found. Otherwise script error.

Python Syntax	GetChildObject(<*path*>)

Python Example	`oDesign = oProject.GetActiveDesign()`
	`oOptimModule = oDesign.GetChildObject("Optimetrics")`
	`oEditor = oDesign.GetChildObject("3D Model")`

	`oBox =  oDesign.GetChildObject("3D Model/Box1")`
	`oRpt = oDesign.GetChildObject("Results/S Parameter Plot 1")`
	`oVariable= oDesign.GetChildObject("Offset")`

GetChildTypes [Design]

Returns the design's child object types.

UI Access	N/A
Parameters	None.
Return Value	String: "Module", "Editor", or "Variable"

Python Syntax	GetChildTypes()
Python Example	`oDesign.GetChildTypes()`

GetDesignID

Returns the unique identification number of the active design.

UI Access	N/A
Parameters	None.

Return Value	String indicating the unique identification number of the active design.

Python Syntax	GetDesignID()

Python Example	`oDesign = oProject.GetActiveDesign()`
	`oDesign.GetDesignID()`

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

GetEdgePositionAtNormalizedParameter

Gets the position on an edge with normalized parameter.

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td><table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;EdgeID&gt;</i></td><td>Integer</td><td>Edge ID.</td></tr>
<tr><td><i>&lt;NormParam&gt;</i></td><td>Double</td><td>Normalized parameter.(Proportional to the length of the specified edge)
For example, 0 leads to the start vertex position of the edge, 1 leads to the end vertex position of the edge, 0.5 leads to the mid position on the edge.</td></tr>
</table></td></tr>
<tr><td><b>Return Value</b></td><td>Array of string containing the x, y and z coordinate values.</td></tr>
</table>

Python Syntax	GetEdgePositionAtNormalizedParameter(<*EdgeID*>, <*NormParam*>)

Python Example	`oEditor.GetEdgePositionAtNormalizedParameter(5, 0)`
	`oEditor.GetEdgePositionAtNormalizedParameter(5, 1)`
	`oEditor.GetEdgePositionAtNormalizedParameter(5, 0.5)`

GetGeometryIdsForNetLayerCombinations

Returns ID numbers of all faces and edges of the active design related to the current target combination of nets and layers.

> **Note:** Intended to support CreateFieldPlot.

UI Access	N/A

<table>
<tr><td><b>Parameters</b></td><td><table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;netName&gt;</i></td><td>String</td><td>Net's name.</td></tr>
<tr><td><i>&lt;layerName&gt;</i></td><td>String</td><td>Layer's name.</td></tr>
<tr><td><i>&lt;setupName&gt;</i></td><td>String</td><td>Name of the setup.</td></tr>
</table></td></tr>
<tr><td><b>Return Value</b></td><td>Array of strings indicating face and edge ID numbers and net/layer combination they are assigned to.</td></tr>
</table>

Python Syntax	GetGeometryIdsForNetLayerCombinations ()

Python Example	`oDesign = oProject.GetActiveDesign()`
	`oDesign.GetGeometryIdsForAllNetLayerCombinations ("<no-net>","Trace","HFSS Setup : Last Adaptive")`

Example with Variables	`oProject = oDesktop.SetActiveProject("Spiral_Inductor_Microstrip")`
	`oDesign = oProject.SetActiveDesign("Spiral")`
	`res = oDesign.GetGeometryIdsForNetLayerCombination("<no-net>","Trace","HFSS Setup : Last Adaptive")`
	`res = ['Surface', 'FacesList', '21', '22']`

GetGeometryIdsForAllNetLayerCombinations

Returns ID numbers of all faces and edges of the active design for all possible combinations of nets and layers the user can choose.

> **Note:** Intended to support CreateFieldPlot.

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><i>&lt;setupName&gt;</i></td><td>String</td><td>Name of the setup.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>Array of strings indicating face and edge ID numbers.</td></tr>
</table>

Python Syntax	GetGeometryIdsForAllNetLayerCombinations ()

Python Example	`oDesign = oProject.GetActiveDesign()`
	`oDesign.GetGeometryIdsForAllNetLayerCombinations("HFSS Setup : Last Adaptive")`

Example with Variables	`oProject = oDesktop.SetActiveProject("Spiral_Inductor_Microstrip")`
	`oDesign = oProject.SetActiveDesign("Spiral")`
	`res = oDesign.GetGeometryIdsForAllNetLayerCombinations("HFSS Setup : Last Adaptive")`
	`res = ['PlotGeomInfo for <no-net>/<no-layer> (net/layer combination):', 'Surface', 'FacesList', '30', '31', '32', '33', '34', '35', '36', '37', '38', '39', '40', '41', '42', '43', '44', '45', '46', '47', '48', '49', '50', '51', '52', '53', '54', '55', '56', '57', 'PlotGeomInfo for <no-net>/AirbridgeMetal (net/layer combination):', 'Surface', 'FacesList', '14', 'PlotGeomInfo for <no-net>/BottomGround (net/layer combination):', 'Surface', 'FacesList', '29', 'PlotGeomInfo for <no-net>/Trace (net/layer combination):', 'Surface', 'FacesList', '21', '22']`

GetManagedFilesPath

Get the path to the project's results folder.

UI Access	N/A
Parameters	None.
Return Value	String containing path where project results are located.

Python Syntax	oDesign.GetManagedFilesPath()
Python Example	`oDesign.GetManagedFilesPath()`

GetModule

Returns the IDispatch for the specified module.

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;modulename&gt;</i></td><td>String</td><td>One of the following:
• Analysis Module – "AnalysisSetup"
• Boundary Module – "BoundarySetup"
• Field Overlays Module – "FieldsReporter"
• Mesh Module – "MeshSetup"
• Optimetrics Module – "Optimetrics"
• Radiation Module – "RadField"</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><td></td><td><table><tr><td></td><td></td><td>• Reduce Matrix Module – "ReduceMatrix"
• Reporter Module – "ReportSetup"
• Simulation Setup Module – "SimSetup"
• Solutions Module – "Solutions"</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>Module IDispatch</td></tr>
</table>

Python Syntax	GetModule (<*modulename*>)
Python Example	`oModule = oDesign.GetModule("SimSetup")`

GetModule (RadiationSetupMgr)

Returns the IDispatch for the specified module.

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><i>&lt;modulename&gt;</i></td><td>String</td><td>One of the following:
• Analysis Module – "AnalysisSetup"
• Boundary Module – "BoundarySetup"
• Field Overlays Module – "FieldsReporter"
• Mesh Module – "MeshSetup"
• Optimetrics Module – "Optimetrics"
• Radiation Module – "RadField"</td></tr></table></td></tr>
</table>

			• Reduce Matrix Module – "ReduceMatrix" • Reporter Module – "ReportSetup" • Simulation Setup Module – "SimSetup" • Solutions Module – "Solutions"
Return Value	Module IDispatch		

Python Syntax	GetModule (<*modulename*>)
Python Example	`oModule = oDesign.GetModule("SimSetup")`

GetName

Returns the design name of the active design, in that order separated by a semicolon.

UI Access	N/A
Parameters	None.
Return Value	String indicating the name of the active design.

Python Syntax	GetName()
Python Example	design_name = oDesign.GetName()

GetNominalVariation

Returns the current nominal variation.

UI Access	N/A
Parameters	None.
Return Value	String containing current nominal variation.

Python Syntax	GetNominalVariation()
Python Example	`oDesign.GetNominalVariation()`

GetNoteText

Returns the text of the note attached to a design.

UI Access	N/A
Parameters	None.
Return Value	String: text of the design note.

Python Syntax	GetNoteText()
Python Example	`oDesign.GetNoteText()`

GetObjPath [Design]

Obtains the path to the design.

UI Access	N/A
Parameters	None.
Return Value	String containing the path to the design.

Python Syntax	GetObjPath()
Python Example	`oDesign.GetObjPath()`

GetOutputVariableValue

Returns the double value of an output variable. Only expressions that return a double value are supported. The expression is evaluated only for a single point.

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;OutputVarName&gt;</i></td><td>String</td><td>Name of the output variable.</td></tr>
<tr><td><i>&lt;IntrinsicVariation&gt;</i></td><td>String</td><td>A set of intrinsic variable value pairs to use when evaluating the output expression.</td></tr>
<tr><td><i>&lt;SolutionName&gt;</i></td><td>String</td><td>Name of the solution as listed in the output variable UI. For example, <code>"Setup1 : Last Adaptive"</code>.</td></tr>
<tr><td><i>&lt;ReportType&gt;</i></td><td>String</td><td>The name of the report type as seen in the output variable UI.</td></tr>
</table>
</td></tr>
</table>

	<ContextArray>	Array	Structured array containing context for which the output variable expression is being evaluated. Can be empty. `Array("Context:=", <string>)`
Return Value	Double value of the output variable.		

Python Syntax	GetOutputVariableValue(*<OutputVarName>*, *<IntrinsicVariation>*, *<SolutionName>*, *<ReportTypeName>*, *<ContextArray>*)
Python Example	`Val = oDesign.GetOutputVariableValue("test","Freq = '20Ghz' Theta='20deg' Phi='30deg'", "TR", "Standard", [])`

GetOutputVariables

Returns the list of output variables.

UI Access	N/A
Parameters	None.
Return Value	Array containing all output variables.

Python Syntax	GetOutputVariables()
Python Example	`oDesign.GetOutputVariables()`

GetPostProcessingVariables

Returns the list of post-processing variables.

UI Access	N/A
Parameters	None.
Return Value	Returns array containing variables.

Python Syntax	GetPostProcessingVariables()
Python Example	`oDesign.GetPostProcessingVariables()`

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.

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;PropTab&gt;</i></td><td>String</td><td>One of the following, where tab titles are shown in parentheses:
<ul>
<li>PassedParameterTab ("Parameter Values")</li>
<li>DefinitionParameterTab (Parameter Defaults")</li>
<li>LocalVariableTab ("Variables" or "Local Variables")</li>
<li>ProjectVariableTab ("Project variables")</li>
<li>ConstantsTab ("Constants")</li>
<li>BaseElementTab ("Symbol" or "Footprint")</li>
</ul></td></tr>
</table>
</td></tr>
</table>

		• ComponentTab ("General") • Component("Component") • CustomTab ("Intrinsic Variables") • Quantities ("Quantities") • Signals ("Signals")
<PropServer>	String	An object identifier, generally returned from another script method, such as CompInst@R;2;3

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

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

GetPropertyValue

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

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

UI Access	N/A

Parameters

Name	Type	Description
<PropTab>	String	One of the following, where tab titles are shown in parentheses:

			• PassedParameterTab ("Parameter Values") • DefinitionParameterTab (Parameter Defaults") • LocalVariableTab ("Variables" or "Local Variables") • ProjectVariableTab ("Project variables") • ConstantsTab ("Constants") • BaseElementTab ("Symbol" or "Footprint") • ComponentTab ("General") • Component("Component") • CustomTab ("Intrinsic Variables") • Quantities ("Quantities") • Signals ("Signals")
	<PropServer>	String	An object identifier, generally returned from another script method, such as `CompInst@R;2;3`
	<PropName>	String	Name of the property.
Return Value	String value of the property.		

Python Syntax	GetPropertyValue (<*PropTab*>, <*PropServer*>, <*PropName*>)

Python Example	`selectionArray = oEditor.GetSelections()`
	`for k in selectionArray:`
	`val = oEditor.GetPropertyValue("PassedParameterTab", k, "R")`
	`...`

GetPropNames [Design]

Returns array containing names of property. For designs, always returns an empty array because the design has no property.

UI Access	N/A
Parameters	<table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>*&lt;includeReadOnly&gt;*</td><td>Boolean</td><td>(Optional). • True - include read only property. • False - do not include read only property.</td></tr></table>
Return Value	Empty array.

Python Syntax	GetPropNames(*<includeReadOnly>*)
Python Example	`oDesign.GetPropNames()`

GetPropValue [Design]

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

UI Access	N/A
Parameters	<table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>*&lt;propPath&gt;*</td><td>String</td><td>A child object's property path. See: Object Property Script Function Summary.</td></tr></table>
Return Value	The property value (integer, string, or array of strings) of the specified child object.

Python Syntax	GetPropValue(<*propPath*>)

Python Example	`oDesign.GetPropValue('offset/SIValue')`
	`oDesign.GetPropValue('Results/S Parameter Plot 1/Display Type')`
	`oDesign.GetPropValue('Results/S Parameter Plot 1/Display Type/Choices')`

GetSelections [Design]

This script serves no function at the Design level. See: GetSelections (Layout Editor), GetSelections (Model Editor), or Get Selections (Schematic Editor).

GetSolutionType

Returns solution type of the design.

UI Access	N/A
Parameters	None.
Return Value	String containing the solution type. Possible values are: "SBR+", "HFSS [Modal | Terminal] [Network | Composite]","Transient [Network | Composite]","Eigenmode", or "Characteristic".

Python Syntax	GetSolutionType()
Python Example	`oDesign.GetSolutionType()`

GetSolveInsideThreshold

Returns the solve inside threshold. This command does not apply to HFSS-IE.

UI Access	N/A
Parameters	None.
Return Value	Double representing the solve inside threshold.

Python Syntax	GetSolveInsideThreshold()
Python Example	`oDesign.GetSolveInsideThreshold()`

GetVariables

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

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

Python Syntax	GetVariables ()

Python Example	`oProject.GetVariables()`
	`oDesign.GetVariables()`

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`.

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;<i>VarName</i>&gt;</td><td>String</td><td>Name of the variable to access.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>String represents the value of the variable.</td></tr>
</table>

Python Syntax	GetVariableValue(<*VarName*>)
Python Example	`oProject.GetVariableValue("var_name")`

GetVariationVariableValue

Returns the value for a specified variation's variable.

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;<i>VariationString</i>&gt;</td><td>String</td><td>The name of the design variation.</td></tr><tr><td>&lt;<i>VariableName</i>&gt;</td><td>String</td><td>The name of the variable.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>Returns a double precision value in SI units, interpreted to mean the value of the variable contained in the variation string.</td></tr>
</table>

Python Syntax	GetVariationVariableValue(<*VariationString*>, <*VariableName*>)
Python Example	`oDesign.GetVariationVariableValue('x_size = 2mm y_size = 1mm', 'y_size')`

ImportArray

Imports an array definition using a csv file.

UI Access	Create Array Through CSV
Parameters	<filepath>.
Return Value	None.

Python Syntax	ImportArray()

Python Example	`oModule = oDesign.GetModule("ModelSetup")`
	`oModule.ImportArray("Array", "C:\\Users\\MyFolder\Export Array3.csv")`

ImportDataset

Imports a dataset from a named file. This can be executed by the oProject, or oDesign variables. The name of the dataset is filename+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.

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

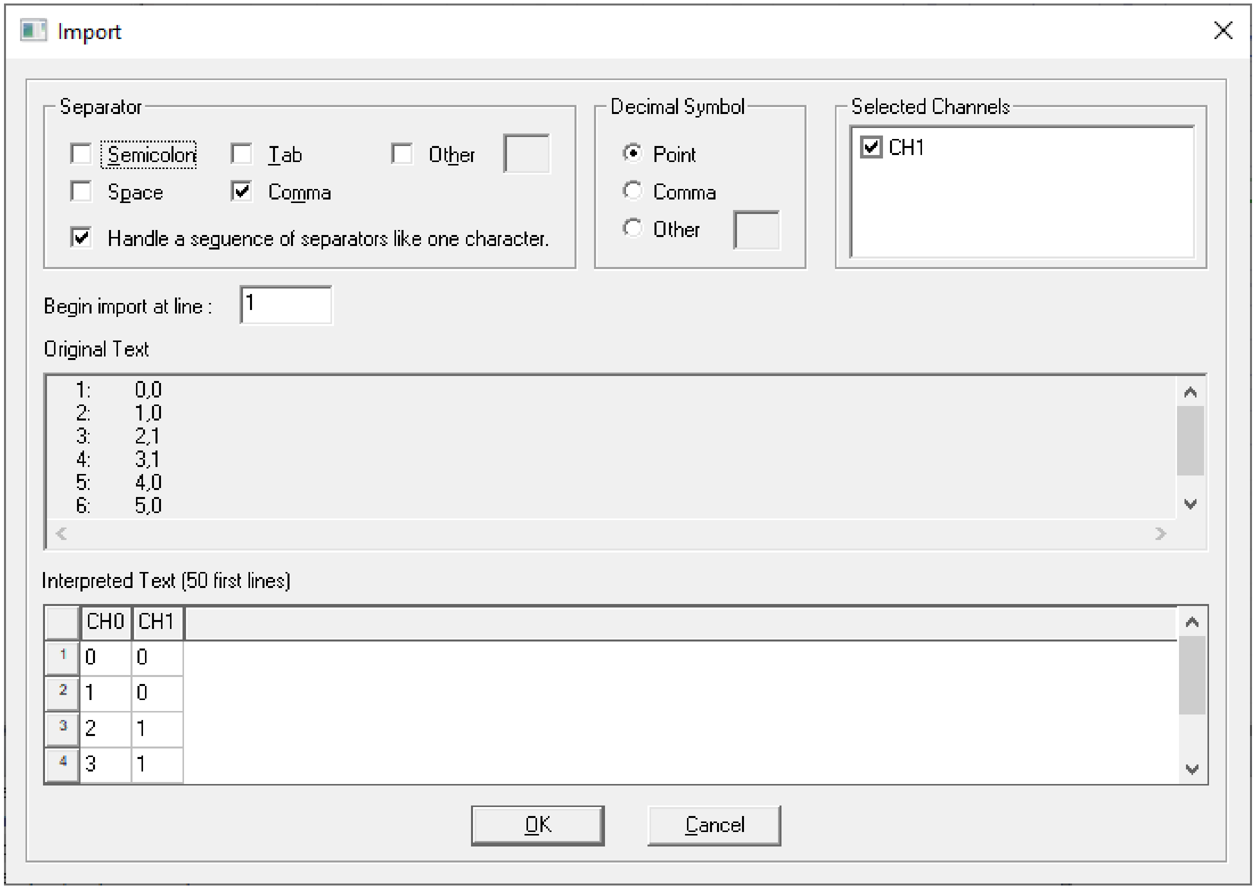
Python Syntax	ImportDataset (<*datasetFileFullPath*>,<*optionalDatasetName*>)

Python Example	`oProject.ImportDataset('e:\tmp\dsdata.tab')`
	`oDesign.ImportDataset('e:\tmp\dsdata.tab')`
	`oProject.ImportDataset('e:\tmp\dsdata.tab', 'MyDatasetName')`
	`oDesign.ImportDataset('e:\tmp\dsdata.tab', 'MyDatasetName')`

Note About File Types:

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

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



Import

Separator

☐ Semicolon ☐ Tab ☐ Other

☐ Space ☑ Comma

☑ Handle a sequence of separators like one character.

Decimal Symbol

● Point

○ Comma

○ Other

Selected Channels

☑ CH1

Begin import at line : 1

Original Text

```
1:      0,0
2:      1,0
3:      2,1
4:      3,1
5:      4,0
6:      5,0
```

Interpreted Text (50 first lines)

	CH0	CH1
1	0	0
2	1	0
3	2	1
4	3	1

PasteArray

Copies a specified array.

UI Access	**Edit** > **Paste**.
Parameters	None. You must previously use CopyArray.
Return Value	New array name.

Python Syntax	Paste ()

Python Example	`oModule = oDesign.GetModule("ModelSetup")`
	`oModule.CopyArray("Array")`
	`oModule.PasteArray() # returns the new array name`

PasteDesign (Design Object)

Pastes a design that has already been copied to the clipboard into another design.

<table>
<tr><th>UI Access</th><th><b>Edit</b> > <b>Paste</b>.</th></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;pasteOption&gt;</i></td><td>Integer</td><td>One of the following:<ul><li><b>0</b> – Link to the existing design that was copied</li><li><b>1</b> – Create a new copy of the design and keep the original layers of the design being copied</li><li><b>2</b> – Create a new copy of the design and merge the layers of the design being copied.</li></ul></td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	PasteDesign(<*pasteOption*>)

Python Example	`oProject.CopyDesign('DesignB')`
	`oDesign = oProject.SetActiveDesign('DesignA')`
	`oDesign.PasteDesign(1)`

Redo [Design]

Reapplies the last design-level command.

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

Python Syntax	Redo()
Python Example	`oDesign.Redo()`

RenameDesignInstance

Renames a design instance.

UI Access	Right-click a design instance in the project tree, and then click **Rename** on the shortcut menu.

<table>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;<i>OldName</i>&gt;</td><td>String</td><td>The current name of the design, which must be the design on which this command is invoked.</td></tr><tr><td>&lt;<i>NewName</i>&gt;</td><td>String</td><td>The new name for the design.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	RenameDesignInstance (<*OldName*>, <*NewName*>)
Python Example	`oDesign.RenameDesignInstance("Design1", "Design2")`

RenameSource [Interface Source]

Use: Renames an interface source

Syntax: RenameSource (STRING: Interface Source name)

Example: `oModule.RenameSource "OldName" "NewName"`

RevertAllToInitialCondition

Reverts all setups to their initial condition. Used for linked thermal designs to revert the target solution (clearing restart, thermal monitor, and field results).

UI Access	**Project Manager** > right-click **Analysis** > **Revert to Initial Condition**. or, with either *nothing* or with **Analysis** selected in the Project Manager: **Mechanical** > **Analysis** > **Revert to Initial Condition** (from the menu bar)
Parameters	None.

Return Value	None

Python Syntax	RevertAllToInitialCondition()
Python Example	`oModule.RevertAllToInitialCondition()`

SARSetup

Sets up for the specific absorption rate (SAR) computation. This command does not apply to HFSS-IE.

UI Access	**HFSS** > **Fields** > **SAR Setting**.
Parameters	<table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;<i>TissueMass</i>&gt;</td><td>Double</td><td>Value represents mass of tissue between 1 and 10 in grams.</td></tr><tr><td>&lt;<i>MaterialDensity</i>&gt;</td><td>Double</td><td>Density of material in gram/cm^3.</td></tr><tr><td>&lt;<i>VoxelSize</i>&gt;</td><td>Double</td><td>Size of a voxel in millimeters.</td></tr><tr><td>&lt;<i>AverageSARMethod</i>&gt;</td><td>Integer</td><td><b>0</b> - IEEE std 1528. <b>1</b> - Gridless, i.e. classical Ansoft method.</td></tr></table>
Return Value	None.

Python Syntax	SARSetup(<*TissueMass*>, <*MaterialDensity*>, <*TissueObjectListID*>, <*VoxelSize*>, <*AverageSARMethod*>)
Python Example	`oDesign.SARSetup(1.0, 1.0, 678, 1.0, 0)`

SetActiveEditor

Sets the active editor.

UI Access	N/A.
Parameters	Name | Type | Description *<EditorName>* | String | Text of the design's notes.
Return Value	Editor object.

Python Syntax	SetActiveEditor(<*EditorName*>)
Python Example	`oDesign.SetActiveEditor('3D Modeler')`

SetAllowMaterialOverride

Sets the option to allow material override.

UI Access	N/A.
Parameters	Name | Type | Description <*allowMaterialOverride*> | Integer | **1** - allow material override. **0** - does not allow material override.
Return Value	None.

Python Syntax	SetAllowMaterialOverride(<*allowMaterialOverride*>)
Python Example	`oDesign.SetAllowMaterialOverride(1)`

SetBackgroundMaterial

Sets the design's background material.

> **Important:**
>
> You can only use this script in 2D Extractor if there is no surface ground in the design *and* the problem type is "open".

<table>
<tr><td><b>UI Access</b></td><td>From the <b>Project Manager</b>, right-click the design and select <b>Set Background Material</b>.</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><i>&lt;matName&gt;</i></td><td>String</td><td>Material name.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	SetBackgroundMaterial(<*matName*>)
Python Example	`oDesign.SetBackgroundMaterial('vacuum')`

SetDesignMode

Switches between HFSS 3D Layout operating modes.

<table>
<tr><td><b>UI Access</b></td><td><b>Edit Layout...</b></td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><i>&lt;modeName&gt;</i></td><td>String</td><td>Enter "<b>IC</b>" or "<b>General</b>" to change operating modes.</td></tr></table></td></tr>
</table>

Return Value	None.

Python Syntax	SetDesignMode(<*modeName*>)
Python Example	`oDesign.SetDesignMode("IC")`

SetDesignSettings (Mechanical)

Sets the design settings for Mechanical designs. The available parameters depend on the current solution type (Modal, Steady-State Thermal, Transient Thermal, or Structural).

<table>
<tr><td><b>UI Access</b></td><td><b>Mechanical</b> > <b>Design Settings</b></td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><th>Name</th><th>Type</th><th>Solution Type*</th><th>Description</th></tr>
<tr><td><b>Mechanial Parameters Array:</b></td><td></td><td></td><td></td></tr>
<tr><td><i><NAME:MechanicalParams></i></td><td>string</td><td>1, 2, 3, 4</td><td>"NAME:Design Settings Data"</td></tr>
<tr><td><i><Allow Material Override></i></td><td>bool</td><td>2, 3</td><td>True | False</td></tr>
<tr><td><i><Perform Minimal Validation></i></td><td>bool</td><td>1, 2, 3, 4</td><td>True | False</td></tr>
<tr><td><i><AmbientTemperature></i></td><td>string</td><td>2, 3</td><td>Global ambient temperature for convection boundaries (with units)</td></tr>
<tr><td><i><EnvironmentTemperature></i></td><td>string</td><td>4</td><td>Global stress-free temperature for structural solutions (with units)</td></tr>
<tr><td><i><ExportOnSimulationComplete></i></td><td>bool</td><td>3</td><td>True | False</td></tr>
<tr><td><i><ExportDirectory></i></td><td>string</td><td>3</td><td>Path to export folder – If string is empty, files are saved to a subfolder:
"<<i>project_name</i>>.aedtresults," under the current project's folder</td></tr>
</table>
</td></tr>
</table>

	Model Parameters Array:			
	<NAME:ModelParams>	string	1, 2, 3, 4	Array "NAME:Model Validation Settings"
	<EntityCheckLevel>	string	1, 2, 3, 4	"None" | "WarningOnly" | "Basic" | "Strict"
	<IgnoreUnclassifiedObjects>	bool	1, 2, 3, 4	True | False
	<SkipIntersectionChecks>	bool	1, 2, 3, 4	True | False
Return Value	None			

Note: *

In the above table, the Solution Type numbers are defined as follows:

1. Modal
2. Steady-State Thermal
3. Transient Thermal
4. Structural

Python Syntax	SetDesignSettings ([<*NAME:MechanicalParams*>,<*Allow Material Override*>, <*Perform Minimal Validation*>, [<*AmbientTemperature*>, <*EnvironmentTemperature*>, <*ExportOnSimulationComplete*>, <*ExportDirectory*>], [<*NAME:ModelParams*>, <*EntityCheckLevel*>, <*IgnoreUnclassifiedObjects*>, <*SkipIntersectionChecks*>])

Python Example

[Transient Thermal]

```
oDesign.SetDesignSettings(
    [
        "NAME:Design Settings Data",
        "Allow Material Override:="      , True,
        "Perform Minimal validation:="   , False,
        "AmbientTemperature:="            , "20cel",
        "ExportOnSimulationComplete:="    , True,
        "ExportDirectory:="               , "D:/Ansys/Mech-TT/TransThermal5.ae-
dtexport/"
    ],
    [
        "NAME:Model Validation Settings" ,
        "EntityCheckLevel:="              , "Strict",
        "IgnoreUnclassifiedObjects:="     , False,
        "SkipIntersectionChecks:="        , False
    ])
```

Python Example

[Structural]

```
oDesign.SetDesignSettings(
    [
        "NAME:Design Settings Data",
        "Allow Material Override:="       , True,
        "Perform Minimal validation:="    , False,
        "EnvironmentTemperature:="        , "183cel"
    ],
    [
        "NAME:Model Validation Settings" ,
        "EntityCheckLevel:="              , "Strict",
        "IgnoreUnclassifiedObjects:="     , False,
        "SkipIntersectionChecks:="        , False
    ])
```

SetFastTransformationForLayoutComponent

Toggles whether layout components in show outlines of objects across layers during view changes. This improves the visualization performance.

UI Access	Fast Transformation

<table>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i><Layout ComponentName></i></td><td>String</td><td>Name of the Layout Component.</td></tr>
<tr><td>Boolean</td><td>String</td><td>True or False</td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	SetFastTransformationForLayoutComponent ("<LayoutComponentName>", <boolean>)
Python Example	`oDesign.SetFastTransformationForLayoutComponent ("LC1_1", False)`

SetLengthSettings

Sets the distributed and lumped lengths of the design, as well as the rise time.

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i><DistributedUnits></i></td><td>String</td><td>Length units used for post-processing.</td></tr>
<tr><td><i><LumpedLength></i></td><td>String</td><td>Length of design in post-processing.</td></tr>
<tr><td><i><RiseTime></i></td><td>String</td><td>Time used in post-processing.</td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	SetLengthSettings(<*DistributedUnits*>, <*LumpedLength*>, <*RiseTime*>)
Python Example	`oDesign.SetLengthSettings('mm', '7meter', '1s')`

SetObjectAttributesForLayoutComponent

Sets visualization attributes for layout components in for a named component for layers, nets, and objects.

<table>
<tr><th>UI Access</th><th>Object Attributes</th></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td>&lt;<i>Layout ComponentName</i>&gt;</td><td>String</td><td>Name of the Layout Component.</td></tr>
<tr><td>&lt;<i>ShowDielectric</i>&gt;</td><td>Boolean</td><td>True to show dielectrics; false to hide dielectrics</td></tr>
<tr><td>&lt;<i>DisplayMode</i>&gt;</td><td>Integer</td><td>0 (for layout mode), 1 (for net mode), and 2 (for object mode)</td></tr>
<tr><td>&lt;<i>ObjectAttributesinLayerMode</i>&gt;</td><td>Array</td><td>List of layers. The three arguments for each layer represent the “Show”, “Wire Frame”, “and “Transparency” options for that layer in the “Object Attributes” dialog box.</td></tr>
<tr><td>&lt;<i>ObjectAttributesinNetMode</i>&gt;</td><td>Array</td><td>List of nets. The three arguments for each net represent the “Show”, “Wire Frame”, “and “Transparency” options for that net in the “Object Attributes” dialog box.</td></tr>
<tr><td>&lt;<i>ObjectAttributesinObjectMode</i>&gt;</td><td>Array</td><td>List of objects. The three arguments for each object represent the “Show”, “Wire Frame”, “and “Transparency” options for that object in the “Object Attributes” dialog box.</td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>None</td></tr>
</table>

Python Syntax	SetObjectAttributesForLayoutComponent (<*LayoutComponentName*>, <*ShowDielectric*>, <*DisplayMode*>, <*ObjectAttributesinLayerMode*> <*ObjectAttributesinNetMode*>, <*ObjectAttributesinObjectMode*>)
Python Example	

```
oDesign.SetObjectAttributesForLayoutComponent (
  [
    "Name:="                , "LC1_1",
    "ShowDielectric:="      , False,
"DisplayMode:=" , 2,
[
  "NAME:ObjectAttributesInLayerMode",
  "GND_1_L2:="            , [True,True,0],
  "GND_2_L4:="            , [True,True,0],
  "GND_3_L6:="            , [True,True,0],
  "GND_4_L9:="            , [True,True,0],
  "GND_5_L11:="           , [True,True,0],
  "Inner_Layer_3_L10:="   , [True,True,0],
  "Top_L1:="              , [True,True,100],
  "VCC1_L7:="             , [True,True,0],
  "VCC2_L8:="             , [True,True,0]
],
[
  "NAME:ObjectAttributesInNetMode",
  "GND:="         , [True,True,29],
  "VCC:="         , [True,True,0],
  "neg:="                 , [True,True,67],
  "pos:="                 , [True,True,67]
],
    [
      "NAME:ObjectAttributesInObjectMode",
      "line_1:="              , [True,True,0],
      "line_2:="              , [True,True,0],
      "line_3:="              , [True,True,100],
      "line_4:="              , [True,True,100],
      "rect_6:="              , [True,True,0],
      "rect_17:="             , [True,True,0],
      "rect_18:="             , [True,True,0],
      "rect_19:="             , [True,True,0],
      "rect_20:="             , [True,True,0],
      "rect_21:="             , [True,True,0],
      "rect_22:="             , [True,True,0],
      "line_24:="             , [True,True,100],
      "via_0:="               , [True,True,100],
      "via_1:="               , [True,True,100],
      "via_2:="               , [True,True,100],
      "via_3:="               , [True,True,100]
    ]
  ])
```

SetPropValue [Design]

Sets the property value for the active property object.

<table>
<tr><td><b>UI Access</b></td><td>Edit properties on Project Tree objects.</td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;propPath&gt;</i></td><td>String</td><td>A child object's property path. See: Object Property Script Function Summary.</td></tr>
<tr><td>&lt;<i>Value</i>&gt;</td><td>String</td><td>New property value.</td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>Boolean:
• <b>True -</b> property found and the new value is valid.
• <b>False</b> - property not found.</td></tr>
</table>

Python Syntax	SetPropValue(<*propPath*>, <*Value*>)

Python Example	`oDesign.SetPropValue("offset", "12mm")`
	`oDesign.SetPropValue("Results/S Parameter Plot 1/Display Type", "Rectangular Plot")`

SetPropertyValue

Sets the value of a single propertybelonging 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.

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;propTab&gt;</i></td><td>String</td><td>One of the following, where tab titles are shown in parentheses:<ul><li>PassedParameterTab ("Parameter Values")</li><li>DefinitionParameterTab (Parameter Defaults")</li><li>LocalVariableTab ("Variables" or "Local Variables")</li><li>ProjectVariableTab ("Project variables")</li><li>ConstantsTab ("Constants")</li><li>BaseElementTab ("Symbol" or "Footprint")</li><li>ComponentTab ("General")</li><li>Component("Component")</li><li>CustomTab ("Intrinsic Variables")</li><li>Quantities ("Quantities")</li><li>Signals ("Signals")</li></ul></td></tr>
<tr><td><i>&lt;propServer&gt;</i></td><td>String</td><td>An object identifier, generally returned from another script method, such as CompInst@R;2;3</td></tr>
<tr><td><i>&lt;propName&gt;</i></td><td>String</td><td>Name of the property.</td></tr>
<tr><td><i>&lt;propValue&gt;</i></td><td>String</td><td>The value for the property</td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

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

SetShowLayoutForLayoutComponent

Layout visualization, rather than bounding box, for a Layout Component in design.

<table>
<tr><td><b>UI Access</b></td><td><b>Edit Layout...</b></td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><i>&lt;Layout ComponentName&gt;</i></td><td>String</td><td>Name of the Layout Component.</td></tr><tr><td>Boolean</td><td>String</td><td>True or False</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	SetShowLayoutForLayoutComponent ("<LayoutComponentName", <boolean>)
Python Example	`oDesign.SetShowLayoutForLayoutComponent ("LC1_1", True)`

SetShowAllLayoutComponents

This command will show/hide all layout components in design.

<table>
<tr><td><b>UI Access</b></td><td><b>Edit Layout... View > Visibility > Show All or Hide All</b></td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>Boolean</td><td>String</td><td>True or False</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>None</td></tr>
</table>

Python Syntax	SetShowAllLayoutComponents (<boolean>)

Python Example	`oDesign = oProject.GetActiveDesign()`
	`oDesign.SetShowAllComponentLayouts(True)`

SetSolutionType

Sets the solution type for the design.

<table>
<tr><td><b>UI Access</b></td><td><b>HFSS</b> &gt; <b>Solution Type</b>.</td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td>&lt;<i>SolutionType</i>&gt;</td><td>String</td><td>Possible values are: "SBR+", "HFSS [Modal | Terminal] [Network | Composite]","Transient [Network | Composite]", "Eigenmode", or "Characteristic".</td></tr>
<tr><td>EnableAutoOpen:=,</td><td>Boolean</td><td>Only .
• <b>True -</b> turn on auto open mode.
• <b>False -</b> turn off auto open mode.</td></tr>
<tr><td>&lt;<i>ModelExteriorAsIE</i>&gt;</td><td>String</td><td>Only Applies for Driven Solution Type with Auto Open Mode as True. Possible values are: Radiation, FEBI, PML</td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	SetSolutionType(<*SolutionType*>, <*AutoOpenMode*>, <*ModelExteriorAsIE*>)
Python Example	`oDesign.SetSolutionType("HFSS Modal Network",`

```
        [
                "NAME:Options",
                "EnableAutoOpen:="      , False
        ])

oDesign.SetSolutionType("Transient Network",
        [
                "NAME:Options",
                "EnableAutoOpen:="      , False
        ])

oDesign.SetSolutionType("HFSS Hybrid Modal Network",
        [
                "NAME:Options",
                "EnableAutoOpen:="      , False
        ])

oDesign.SetSolutionType("SBR+",
        [
        "NAME:Options",
```

<table>
<tr><td></td><td><pre>"EnableAutoOpen:="   , False
])</pre></td></tr>
</table>

SetSolveInsideThreshold

Sets the solve inside threshold to the specified double.

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><i>&lt;threshold&gt;</i></td><td>Double</td><td>Siemens/m.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	SetSolveInsideThreshold(<*threshold*>)
Python Example	`oDesign.SetSolveInsideThreshold(100000)`

SetSourceContexts

For Near or Far Field projects for Driven Modal or Driven Terminal Network Analysis Solutions, specifies the port name and all modes/terminals of that port to be enabled as Source Context.

<table>
<tr><td><b>UI Access</b></td><td><b>HFSS</b> &gt; <b>Fields</b> &gt; <b>Edit Sources</b>.</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><i>&lt;SourceId&gt;</i></td><td>Array</td><td>Name of modes/terminals to be set as source context.</td></tr></table></td></tr>
</table>

Return Value	None.

Python Syntax	SetSourceContexts(<*SourceId*>)

Python Example	`oModule.SetSourceContexts(`
	`[`
	`"Box1_T1", "Box1_T2", "Box1_T3", "Current1", "IncPWave1"`
	`])`

SetVariableValue

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

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

Python Syntax	SetVariableValue (<*VarName*>, <*VarValue*>)
Python Example	`oProject.SetVariableValue('$Var1', '3mm')`

Solve

Performs one or more simulation. The next script command will not be executed until the simulation(s) are complete.

<table>
<tr><td><b>UI Access</b></td><td>Select solution setup(s). Right-click and select <b>Analyze</b>.</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;<i>SimulationNames</i>&gt;</td><td>Array</td><td>Array containing string simulation names.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>Integer:
• <b>0</b> – Simulation(s) completed.
• <b>1</b> – Simulation error.
• <b>-1</b> – Command execution error.</td></tr>
</table>

Python Syntax	Solve <*SimulationNames*>
Python Example	`oDesign.Solve(['Setup1','Setup2','Setup3'])`

Undo [Design]

Cancels the last design-level command.

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

Python Syntax	Undo()
Python Example	`oDesign.Undo()`

ValidateDesign

Returns whether a design is valid.

UI Access	**Mechanical** > **Validation Check**.
Parameters	None.
Return Value	Integer: • **1 –** Validation passed. • **0 –** Validation failed.

Python Syntax	ValidateDesign()
Python Example	`oDesign.ValidateDesign()`

8 - 3D Modeler Editor Script Commands

3D Modeler commands should be executed by the "3D Modeler" editor:

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

```
oEditor.<CommandName>
```

Conventions Used in this Chapter:

<AttributesArray>

<AttributesArray> takes the following structure:

```
Array("NAME:Attributes",
     "Name:=", <string>,
     "Flags:=", <string>,
     "Color:=", <string>,
     "Transparency:=", <value>,
     "PartCoordinateSystem:=", <string>,
     "UDMId:=", <string>,
     "MaterialValue:=", <string>,
     "SurfaceMaterialValue:=", <string>,
     "Solveinside:=", <boolean>,
     "ShellElement:=", <boolean>,
     "ShellElementThickness:=", <string>,
     "ReferenceTemperature:=", <string>,
    "IsMaterialEditable:=", <boolean>,
    "UseMaterialAppearance:=", <boolean>,
    "IsLightweight:=", <boolean>)
```

Where:

- **Flags** – Takes a string containing "NonModel" and/or "Wireframe", separated by the # character. For example, "NonModel#Wireframe".
- **Color** – Takes a string containing an RGB triple, formatted as "(<*RGB*>)". For example, "(255 255 255)".
- **Transparency** – Takes a value between 0 and 1.
- **PartCoordinateSystem** – Orientation of the primitive. The name of one of the defined coordinate systems should be specified.
- **UDMId** – Takes a string containing an ID.
- **MaterialValue** – Takes a string of the material name.
- **SurfaceMaterialValue** – Takes a string of the surface material name.
- **Solveinside** – Takes a boolean value.
- **ShellElement** – Takes a boolean value specifying whether or not a shell element is present.
- **ShellElementThickness** – Takes a string containing the shell element thickness. If element is not present, pass empty string.
- **ReferenceTemperature** – String containing the temperature at which an object is in a stress and strain-free state (applicable to Mechanical–Structural solutions). The default value is "EnvTemp" (the environment temperature, which is a global design variable).
- **IsMaterialEditable** – Takes a boolean value.
- **IsLightweight** – Takes a boolean value.

<SelectionsArray>

<SelectionsArray> typically takes the following structure:

```
Array("NAME:Selections",
  "Selections:=", <string>)
```

Where:

- **Selections** – Takes a comma-separated list of parts on which to perform the action. For example, "Rect1, Rect2, Rect3".

In some cases, <SelectionsArray> takes additional parameters:

```
Array("NAME:Selections",
  "AllowRegionDependentPartSelectionForPMLCreation:=", <boolean>,
  "AllowRegionSelectionForPMLCreation:=", <boolean>,
  "Selections:=", <string>,
  "NewPartsModelFlag:=" , <string>,
  "UseCurrentCS:=", <boolean>)
```

Where:

- **AllowRegionDependentPartSelectionForPMLCreation** – Takes a boolean value. See individual script for whether this parameter is required.
- **AllowRegionSelectionForPMLCreation** – Takes a boolean value. See individual script for whether this parameter is required.
- **Selections** – Takes a comma-separated list of parts on which to perform the action. For example, "Rect1, Rect2, Rect3".
- **NewPartsModelFlag** – Takes either string "Model" or string "Nonmodel". See individual script for whether this parameter is required.
- **UseCurrentCS** – Takes a boolean value. See individual script for whether this parameter is required. Use GetActiveCoordinateSystem to determine the current CS.

> **Note:**
>
> *Selections* is the only parameter required in *all* 3D Modeler Editor scripts. See individual scripts for additional required parameters.

Organization

3D Modeler editor scripts are organized into the following categories:

Draw Menu Commands

Edit Menu Commands

Modeler Menu Commands

Other oEditor Commands

Draw Menu Commands

Create3DComponent

CreateBondwire

CreateBox

CreateCircle

CreateCone

CreateCutplane

CreateCylinder

CreateEllipse

CreateEquationCurve

CreateEquationSurface

CreateHelix

CreatePoint

CreatePolyline

CreateRectangle

CreateRegularPolygon

CreateRegularPolyhedron

CreateSphere

CreateSpiral

CreateTorus

CreateUserDefinedModel

CreateUserDefinedPart

Edit3DComponent

EditPolyline

EditNativeComponentDefinition [Beta – Layout Components in Mechanical]

Get3DComponentDefinitionNames

Get3DComponentInstanceNames

Get3DComponentMaterialNames

Get3DComponentMaterialProperties

Get3DComponentParameters

Insert3DComponent

InsertNativeComponentDefinition [Beta – Layout Components in Mechanical]

InsertPolylineSegment

SweepAlongPath

SweepAlongVector

SweepAroundAxis

SweepFacesAlongNormal

SweepFacesAlongNormalWithAttributes

UpdateComponentDefinition

Create3DComponent

Creates a 3D component.

<table>
<tr><th>UI Access</th><th>Draw > 3D Component Library > Create 3D Component.</th></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td><i>&lt;CreateData&gt;</i></td><td>Array</td><td>Structured array containing geometry data:
(Array["NAME:CreateData",
"ComponentName:=", "<string>",
"Company:=", "<string>",
"Company URL:=", "<string>",
"Model Number:=","<string>",
"Help URL:=" , "<string>",
"Version:=", "<string&gt;",</td></tr>
</table>
</td></tr>
</table>

			"Notes:=", "<string>", "IconType:=","<string>", "Owner:=", "<string>", "Email:=", "<string>", "Date:=", "<string>", #Component creation date "HasLabel:=", <boolean>, "IsEncrypted:=", <boolean>, "AllowEdit:=", <boolean>, "SecurityMessage:=" , "<string>", "Password:=" , "<string>", "EditPassword:=" , "<string>", "PasswordType:=" , "<string>", "HideContents:=" , <boolean>, "ReplaceNames:=" , <boolean>, #for objects and materials "ComponentOutline:=" , "<string>",#"None" or "Bounding Box" "IncludedParts:=" , [Array of partnames], "HiddenParts:=" , [Array of hidden parts], "IncludedCS:=" , [Array of included CS], "DefaultHandle:=" , "<reference CS>" "IncludedParameters:=" , [Array of included parameters], "IncludedDependentParameters:=", [Array of dependent

			`parameters],`
			`"ParameterDescription:=", <array>`
			`"IsLicensed:=" , <boolean>`
			`"LicensingDllName:=" , "<string>"`
			`"VendorComponentIdentifier:=", "<string>"`
			`"PublicKeyFile:=" , "<string>"`
			`]`

	<*DesignData*>	Array	Structured array containing design data:
			`Array["NAME:DesignData",`
			`"Boundaries:=", <array>,`
			`"Excitations:=", <array>,`
			`"MeshOperations:=", <array>]`

	<*FileName*>	String	Full file path to 3D component.

	<*ImageFile*>	Array	Optional. Structured array containing file path of 3D component image:
			`Array["NAME:ImageFile",`
			`"ImageFile:=", <string>])`

Return Value	None.		

Python Syntax	Create3DComponent(<*CreateData*>, <*DesignData*>, <*FileName*>, <*ImageFile*>)
Python	`oEditor.Create3DComponent(`

Example

```
[
  "NAME:CreateData",
  "ComponentName:=", "Connector",
  "Company:=", "",
  "Company URL:=" , ""
  "Version:=", "1.0",
  "Notes:=", "",
  "Owner:=", "",
  "IconType:=" , "",
  "Email:=", "",
  "Date:=", "11:41:01 AM Aug 28, 2024",
  "HasLabel:=", false,
  "IsEncrypted:="     , False,
  "AllowEdit:="       , False,
  "SecurityMessage:=" , "",
  "Password:="        , "",
  "EditPassword:="    , "",
  "PasswordType:="    , "UnknownPassword",
  "HideContents:="    , True,
  "ReplaceNames:="    , True,
  "ComponentOutline:="    , "None",
  "IncludedParts:=", ["Box1", "Cylinder1", "Cone1"],
  "IncludedCS:=", ["RelativeCS1"],
  "DefaultHandle:=", "Global",
  "IncludedParameters:=", ["htcone", "lr", "htcyl", "zs", "radcyl", "xs", "$rp",
  "$con"],
  "IncludedDependentParameters:=", [],
  "ParameterDescription:=", [],
  "IsLicensed:="          , False,
  "LicensingDllName:="    , "",
  "VendorComponentIdentifier:=", "",
  "PublicKeyFile:="       , ""
],
[
  "NAME:DesignData",
    "Boundaries:=", ["PerfE1", "FiniteCond1"],
    "Excitations:=", ["1"],
    "MeshOperations:=", []
  ],
  "C:/tmp/Connector.a3dcomp",
  [
    "NAME:ImageFile", "ImageFile:=", ""
  ])
```

Example with an External Circuit:

```
oDesign = oProject.SetActiveDesign("Maxwell3DDesign1")
oEditor = oDesign.SetActiveEditor("3D Modeler")
oEditor.Create3DComponent(
  [
    "NAME:CreateData",
    "ComponentName:=" , "Maxwell3DDesign1",
    "Company:=" , "",
    "Company URL:=" , "",
    "Model Number:=" , "",
    "Help URL:=" , "",
    "Version:=" , "1.0",
    "Notes:=" , "",
    "IconType:=" , "",
    "Owner:=" , "",
    "Email:=" , "",
    "Date:=" , "3:36:11 PM Feb 14, 2025",
    "HasLabel:=" , False,
    "IsEncrypted:=" , False,
    "AllowEdit:=" , False,
    "SecurityMessage:=" , "",
    "Password:=" , "",
    "EditPassword:=" , "",
    "PasswordType:=" , "UnknownPassword",
    "HideContents:=" , True,
    "ReplaceNames:=" , True,
    "ComponentOutline:=" , "None",
    "IncludedParts:=" , ["Box1"],
    "HiddenParts:=" , [],
    "IncludedCS:=" , [],
    "DefaultHandle:=" , "Global",
    "IncludedParameters:=" , [],
    "IncludedDependentParameters:=", [],
    "ParameterDescription:=", [],
    "IsLicensed:=" , False,
    "LicensingDllName:=" , "",
    "VendorComponentIdentifier:=", "",
    "PublicKeyFile:=" , ""
  ],
  [
    "NAME:DesignData",
    "Excitations:=" , ["CoilTerminal1","CoilTerminal2","Winding1","External Circuit”]
  ],
  "F:/Designs/ComponentExternalCircuit/test.a3dcomp",
  [
    "NAME:ImageFile",
    "ImageFile:=" , ""
  ])
```

CreateBondwire

Creates a bondwire.

UI Access	**Draw** > **Bondwire**.
Parameters	<table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><i>&lt;Parameters&gt;</i></td><td>Array</td><td>Structured array. <code>Array("NAME:BondwireParameters",</code></td></tr></table>

<table>
<tr><td></td><td>
<table>
<tr><td></td><td></td><td>"WireType:=", <string("JEDEC_4Points", "JEDEC_5Points", or "LOW")>,
"WireDiameter:=", <string>,
"NumSides:=", <value>,
"XPadPos:=", <value>,
"YPadPos:=", <value>,
"ZPadPos:=", <value>,
"XDir:=", <value>,
"YDir:=", <value>,
"ZDir:=", <value>,
"Distance:=", <value>,
"h1:=", <value>,
"h2:=", <value>,
"alpha:=", <value>,
"beta:=", <value>,
"WhichAxis:=", <string("X","Y", or "Z")>,
"ReverseDirection:=", <boolean&gt;)</td></tr>
<tr><td><i>&lt;AttributesArray&gt;</i></td><td>Array</td><td>Structured array. See: AttributesArray.</td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	CreateBondwire(<*Parameters*>, <*Attributes*>)

Python Example

```
oEditor.CreateBondwire(
["NAME:BondwireParameters",
  "WireType:="            , "JEDEC_4Points",
  "WireDiameter:="        , "0.025mm",
  "NumSides:="            , "6",
  "XPadPos:="             , "1.6mm",
  "YPadPos:="             , "-0.2mm",
  "ZPadPos:="             , "0mm",
  "XDir:="                , "-2.2mm",
  "YDir:="                , "-1.4mm",
  "ZDir:="                , "0mm",
  "Distance:="            , "2.60768096208106mm",
  "h1:="                  , "0.2mm",
  "h2:="                  , "0mm",
  "alpha:="               , "80deg",
  "beta:="                , "0",
  "WhichAxis:="           , "Z",
  "ReverseDirection:="    , True
  ],
["NAME:Attributes",
  "Name:="                , "Bondwire1",
  "Flags:="                 , "",
  "Color:="                 , "(143 175 143)",
  "Transparency:="          , 0,
  "PartCoordinateSystem:=", "Global",
  "UDMId:="                 , "",
  "MaterialValue:="         , "\"vacuum\"",
  "SurfaceMaterialValue:=", "\"\"",
  "SolveInside:="           , True,
  "ShellElement:="          , False,
  "ShellElementThickness:=", "0mm",
  "IsMaterialEditable:="    , True,
  "UseMaterialAppearance:=", False,
  "IsLightweight:="         , False
])
```

CreateBox

Creates a box.

<table>
<tr><th>UI Access</th><th><b>Draw</b> &gt; <b>Box</b>.</th></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><i>&lt;Parameters&gt;</i></td><td>Array</td><td>Structured array.</td></tr></table></td></tr>
</table>

		Array("NAME:BoxParameters", "XPosition:=", <string>, "YPosition:=", <string>, "ZPosition:=", <string>, "XSize:=" , <string>, "YSize:=" , <string>, "ZSize:=" , <string>)
<AttributesArray>	Array	Structured array. See: AttributesArray.

Return Value	None.

Python Syntax	CreateBox(<*Parameters*>, <*Attributes*>)
Python Example	oEditor.CreateBox(["NAME:BoxParameters", "XPosition:=" , "0.5mm", "YPosition:=" , "-6.5mm", "ZPosition:=" , "0mm", "XSize:=" , "2mm", "YSize:=" , "1.5mm", "ZSize:=" , "1.5mm"], ["NAME:Attributes",

```
  "Name:="                     , "Box3",
  "Flags:="                    , "",
  "Color:="                    , "(143 175 143)",
  "Transparency:="             , 0,
  "PartCoordinateSystem:=" , "Global",
  "UDMId:="                    , "",
  "MaterialValue:="            , "\"copper\"",
  "SurfaceMaterialValue:=" , "\"\"",
  "SolveInside:="              , False,
  "ShellElement:="             , False,
  "ShellElementThickness:=", "0mm",
  "IsMaterialEditable:="       , True,
  "UseMaterialAppearance:=", False,
  "IsLightweight:="            , False
])
```

CreateCircle

Creates a circle.

UI Access	**Draw** > **Circle**.

<table>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td><i>&lt;Parameters&gt;</i></td><td>Array</td><td>Structured array.
Array("NAME:CircleParameters",
  "IsCovered:=", <boolean>,
  "XCenter:=", <value>,
  "YCenter:=", <value>,
  "ZCenter:=", <value>,
  "Radius:=", <value>,
  "WhichAxis:=", <string>
  "NumSegments:=", <string containing integer&gt;)</td></tr>
<tr><td><i>&lt;AttributesArray&gt;</i></td><td>Array</td><td>Structured array. See: AttributesArray.</td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

<table>
<tr><td><b>Python Syntax</b></td><td>CreateCircle(<i>&lt;Parameters&gt;</i>, <i>&lt;Attributes&gt;</i>)</td></tr>
<tr><td><b>Python Example</b></td><td>oEditor.CreateCircle(
["NAME:CircleParameters",
  "IsCovered:="   , True,
  "XCenter:="   , "5.5mm",
  "YCenter:="   , "-3mm",
  "ZCenter:="   , "0mm",
  "Radius:="   , "0.707106781186548mm",
  "WhichAxis:="   , "Z",</td></tr>
</table>

```
  "NumSegments:="         , "0"
],
["NAME:Attributes",
  "Name:="                , "Circle1",
  "Flags:="               , "",
  "Color:="               , "(143 175 143)",
  "Transparency:="        , 0,
  "PartCoordinateSystem:=", "Global",
  "UDMId:="               , "",
  "MaterialValue:="       , "\"copper\"",
  "SurfaceMaterialValue:=", "\"\"",
  "SolveInside:="         , False,
  "ShellElement:="        , False,
  "ShellElementThickness:=", "0mm",
  "IsMaterialEditable:="  , True,
  "UseMaterialAppearance:=", False,
  "IsLightweight:="       , False
])
```

CreateCone

Creates a cone.

<table>
<tr><th>UI Access</th><th>Draw &gt; Cone.</th></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;Parameters&gt;</i></td><td>Array</td><td>Structured array.
<pre>
Array("NAME:ConeParameters",
   "XCenter:=", <string>,
   "YCenter:=", <string>,
   "ZCenter:=", <string>,
   "WhichAxis:=", <string>,
   "Height:=", <string>,
   "BottomRadius:=", <string>,
   "TopRadius:=", <string>)
</pre></td></tr>
<tr><td><i>&lt;AttributesArray&gt;</i></td><td>Array</td><td>Structured array. See: AttributesArray.</td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

<table>
<tr><th>Python Syntax</th><th>CreateCone(<i>&lt;Parameters&gt;</i>, <i>&lt;Attributes&gt;</i>)</th></tr>
<tr><td><b>Python Example</b></td><td>
<pre>
oEditor.CreateCone(
["NAME:ConeParameters",
   "XCenter:="   , "3mm",
   "YCenter:="   , "-4.5mm",
   "ZCenter:="   , "0mm",
   "WhichAxis:="   , "Z",
   "Height:="   , "2.5mm",
</pre></td></tr>
</table>

```
  "BottomRadius:="        , "2.82842712474619mm",
  "TopRadius:="           , "2.23606797749979mm"
],
["NAME:Attributes",
  "Name:="                , "Cone1",
  "Flags:="               , "",
  "Color:="               , "(143 175 143)",
  "Transparency:="        , 0,
  "PartCoordinateSystem:=", "Global",
  "UDMId:="               , "",
  "MaterialValue:="       , "\"copper\"",
  "SurfaceMaterialValue:=", "\"\"",
  "SolveInside:="         , False,
  "ShellElement:="        , False,
  "ShellElementThickness:=", "0mm",
  "IsMaterialEditable:="  , True,
  "UseMaterialAppearance:=", False,
  "IsLightweight:="       , False
])
```

CreateCutplane

Creates a cutplane.

<table>
<tr><td><b>UI Access</b></td><td><b>Draw</b> &gt; <b>Plane</b>.</td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;Parameters&gt;</i></td><td>Array</td><td>Structured array.
["NAME:PlaneParameters",
"PlaneBaseX:=", <string>,
"PlaneBaseY:=", <string>,
"PlaneBaseZ:=", <string>,
"PlaneNormalX:=", <string>,
"PlaneNormalY:=", <string>,
"PlaneNormalZ:=", <string&gt;]</td></tr>
<tr><td><i>&lt;AttributesArray&gt;</i></td><td>Array</td><td>See: AttributesArray. CreateCutplane only takes the Name and Color attributes.</td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	CreateCutplane(<*Parameters*>, <*Attributes*>)
Python Example	oEditor.CreateCutplane(["NAME:PlaneParameters", "PlaneBaseX:=" , "-0.6mm", "PlaneBaseY:=" , "-0.8mm",

<table>
<tr><td></td><td>"PlaneBaseZ:=" , "0mm",
"PlaneNormalX:=" , "1.2mm",
"PlaneNormalY:=" , "0.2mm",
"PlaneNormalZ:=" , "0mm"
],
["NAME:Attributes",
"Name:=" , "Plane1",
"Color:=" , "(143 175 143)"
])</td></tr>
</table>

CreateCylinder

Creates a cylinder.

<table>
<tr><th>UI Access</th><th><b>Draw</b> > <b>Cylinder</b>.</th></tr>
<tr><td><b>Parameters</b></td><td><table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i><Parameters></i></td><td>Array</td><td>Structured array.
Array("NAME:CylinderParameters",
"XCenter:=", <string>,
"YCenter:=", <string>,
"ZCenter:=", <string>,
"Radius:=", <string>,</td></tr>
</table></td></tr>
</table>

			`"Height:=", <string>,`
			`"WhichAxis:=", <string>,`
			`"NumSides:=", <string containing integer>)`

	<AttributesArray>	Array	Structured array. See: AttributesArray.
Return Value	None.		

<table>
<tr><td><b>Python Syntax</b></td><td>CreateCylinder(&lt;<i>Parameters</i>&gt;, &lt;<i>Attributes</i>&gt;)</td></tr>
<tr><td><b>Python Example</b></td><td><pre>
oEditor.CreateCylinder(
["NAME:CylinderParameters",
  "XCenter:="   , "6mm",
  "YCenter:="   , "-4.5mm",
  "ZCenter:="   , "0mm",
  "Radius:="   , "0.5mm",
  "Height:="   , "4.5mm",
  "WhichAxis:="   , "Z",
  "NumSides:="   , "0"
],
["NAME:Attributes",
  "Name:="   , "Cylinder1",
  "Flags:="   , "",
  "Color:="   , "(143 175 143)",
</pre></td></tr>
</table>

```
  "Transparency:="        , 0,
  "PartCoordinateSystem:=", "Global",
  "UDMId:="               , "",
  "MaterialValue:="       , "\"copper\"",
  "SurfaceMaterialValue:=", "\"\"",
  "SolveInside:="         , False,
  "ShellElement:="        , False,
  "ShellElementThickness:=", "0mm",
  "IsMaterialEditable:="  , True,
  "UseMaterialAppearance:=", False,
  "IsLightweight:="       , False
])
```

CreateEllipse

Creates an ellipse.

UI Access	**Draw** > **Ellipse**.

Parameters

Name	Type	Description

<Parameters>	Array	Structured array.
		`Array("NAME:EllipseParameters",`
		`  "IsCovered:=", <string>,`

			"XCenter:=", <string>, "YCenter:=", <string>, "ZCenter:=", <string>, "MajRadius:=", <string>, "Ratio:=", <string>, "WhichAxis:=", <string>, "NumSegments:=", <string>)
	<AttributesArray>	Array	Structured array. See: AttributesArray.
Return Value	None.		

Python Syntax	CreateEllipse(<*Parameters*>, <*Attributes*>)

Python Example	oEditor.CreateEllipse(
	["NAME:EllipseParameters",
	  "IsCovered:=" , True,
	  "XCenter:=" , "0.6mm",
	  "YCenter:=" , "-0.6mm",
	  "ZCenter:=" , "0mm",
	  "MajRadius:=" , "0.2mm",
	  "Ratio:=" , "7",
	  "WhichAxis:=" , "Z",
	  "NumSegments:=" , "0"

```
],
["NAME:Attributes",
  "Name:="                , "Ellipse1",
  "Flags:="               , "",
  "Color:="               , "(143 175 143)",
  "Transparency:="        , 0,
  "PartCoordinateSystem:=", "Global",
  "UDMId:="               , "",
  "MaterialValue:="       , "\"copper\"",
  "SurfaceMaterialValue:=", "\"\"",
  "SolveInside:="         , False,
  "ShellElement:="        , False,
  "ShellElementThickness:=", "0mm",
  "IsMaterialEditable:="  , True,
  "UseMaterialAppearance:=", False,
  "IsLightweight:="       , False
])
```

CreateEquationCurve

Creates an equation-based curve.

<table>
<tr><td><b>UI Access</b></td><td><b>Draw</b> > <b>Equation-Based Curve</b>.</td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i><Parameters></i></td><td>Array</td><td>Structured array.
Array("NAME:EquationBasedCurveParameters",
"XtFunction:=", <string>,
"YtFunction:=", <string>,
"ZtFunction:=", <string>,
"tStart:=", <string>,
"tEnd:=", <string>,
"NumOfPointsOnCurve:=", <string>,
"Version:=", <integer>),
<polylineArray(optional)>)</td></tr>
<tr><td><i><AttributesArray></i></td><td>Array</td><td>Structured array. See: AttributesArray.</td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

<table>
<tr><td><b>Python Syntax</b></td><td>CreateEquationCurve(<<i>Parameters</i>>, <<i>Attributes</i>>)</td></tr>
<tr><td><b>Python Example</b></td><td>oEditor.CreateEquationCurve(
["NAME:EquationBasedCurveParameters",
"XtFunction:=" , "1",
"YtFunction:=" , "3",
"ZtFunction:=" , "32",
"tStart:=" , "1",</td></tr>
</table>

```
  "tEnd:="                , "3",
  "NumOfPointsOnCurve:="  , "0",
  "Version:="             , 1,
  ["NAME:PolylineXSection",
    "XSectionType:="        , "None",
    "XSectionOrient:="      , "Auto",
    "XSectionWidth:="       , "0",
    "XSectionTopWidth:="    , "0",
    "XSectionHeight:="      , "0",
    "XSectionNumSegments:=" , "0",
    "XSectionBendType:="    , "Corner"
  ]
],
["NAME:Attributes",
  "Name:="                , "EquationCurve1",
  "Flags:="               , "",
  "Color:="               , "(143 175 143)",
  "Transparency:="        , 0,
  "PartCoordinateSystem:=", "Global",
  "UDMId:="               , "",
```

<table>
<tr><td></td><td><pre>
  "MaterialValue:="   , "\"copper\"",
  "SurfaceMaterialValue:=", "\"\"",
  "SolveInside:="   , False,
  "ShellElement:="   , False,
  "ShellElementThickness:=", "0mm",
  "IsMaterialEditable:=" , True,
  "UseMaterialAppearance:=", False,
  "IsLightweight:="   , False
])
</pre></td></tr>
</table>

CreateEquationSurface

Creates an equation-based surface.

<table>
<tr><th>UI Access</th><th><b>Draw</b> &gt; <b>Equation-Based Surface</b>.</th></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td><i>&lt;Parameters&gt;</i></td><td>Array</td><td>Structured array.<pre>
Array("NAME:EquationBasedSurfaceParameters",
  "XuvFunction:=" , <string equation containing Function, Operators and/or quantities _u, _v, or PI>,
  "YuvFunction:=" , <string equation containing Function, Operators and/or quantities _u, _v, or PI>,
  "ZuvFunction:=" , <string equation containing Function, Operators and/or quantities _u, _v, or PI>,
  "uStart:=" , <string>,
</pre></td></tr>
</table>
</td></tr>
</table>

		"uEnd:=" , <string>, "vStart:=" , <string>, "vEnd:=" , <string>, "Version:=" , <integer>)
<AttributesArray>	Array	Structured array. See: AttributesArray.

Return Value	None.

Python Syntax	CreateEquationSurface(<*Parameters*>, <*Attributes*>)

Python Example	oEditor.CreateEquationSurface(
	["NAME:EquationBasedSurfaceParameters",
	  "XuvFunction:=" , "_u",
	  "YuvFunction:=" , "_v",
	  "ZuvFunction:=" , "sin(_u)+cos(_v)",
	  "uStart:=" , "1",
	  "uEnd:=" , "10",
	  "vStart:=" , "1",
	  "vEnd:=" , "10",
	  "Version:=" , 1
	],
	["NAME:Attributes",

```
    "Name:="                   , "EquationSurface1",
    "Flags:="                  , "",
    "Color:="                  , "(143 175 143)",
    "Transparency:="           , 0,
    "PartCoordinateSystem:=", "Global",
    "UDMId:="                  , "",
    "MaterialValue:="          , "\"copper\"",
    "SurfaceMaterialValue:=", "\"\"",
    "SolveInside:="            , False,
    "ShellElement:="           , False,
    "ShellElementThickness:=", "0mm",
    "IsMaterialEditable:="     , True,
    "UseMaterialAppearance:=", False,
    "IsLightweight:="          , False
])
```

CreateHelix

Creates a helix based on a sweep of specified objects.

<table>
<tr><td><b>UI Access</b></td><td><b>Draw</b> &gt; <b>Helix</b>.</td></tr>
<tr><td><b>Parameters</b></td><td><table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;SelectionsArray&gt;</i></td><td>Array</td><td>Structured array. See: SelectionsArray.</td></tr>
<tr><td><i>&lt;Parameters&gt;</i></td><td>Array</td><td>Structured array.</td></tr>
</table></td></tr>
</table>

			Array("NAME:HelixParameters", "XCenter:=" , <string>, "YCenter:=" , <string>, "ZCenter:=" , <string>, "XStartDir:=" , <string>, "YStartDir:=" , <string>, "ZStartDir:=" , <string>, "NumThread:=" , <string>, "RightHand:=" , <boolean>, "RadiusIncrement:=" , <string>, "Thread:=" , <string>)
Return Value	None.		

Python Syntax	CreateHelix(<*SelectionsArray*>, <*Parameters*>)
Python Example	oEditor.CreateHelix(["NAME:Selections", "Selections:=" , "EquationSurface2", "NewPartsModelFlag:=" , "Model"], ["NAME:HelixParameters",

```
  "XCenter:="              , "10000mm",
  "YCenter:="              , "40000mm",
  "ZCenter:="              , "0mm",
  "XStartDir:="            , "0mm",
  "YStartDir:="            , "10000mm",
  "ZStartDir:="            , "0mm",
  "NumThread:="            , "1",
  "RightHand:="            , True,
  "RadiusIncrement:="      , "0mm",
  "Thread:="               , "1mm"
])
```

CreatePoint

Creates a point.

UI Access	**Draw** > **Point**.
Parameters	

Name	Type	Description

<Parameters>	Array	Structured array.
		`Array("NAME:PointParameters",`
		`"PointX:=", <value>,`
		`"PointY:=", <value>,`
		`"PointZ:=", <value>)`

<AttributesArray>	Array	Structured array. See: AttributesArray. CreatePoint takes only the Name

	and Color attributes.
Return Value	None.

Python Syntax	CreatePoint(<*Parameters*>, <*Attributes*>)

Python Example

```
oEditor.CreatePoint(

["NAME:PointParameters",
  "PointX:=" , "0.2mm",

  "PointY:=" , "-0.2mm",

  "PointZ:=" , "0mm"
    ],
["NAME:Attributes",

  "Name:="  , "Point1",

  "Color:=" , "(143 175 143)"

])
```

CreatePolyline

Creates a polyline.

<table>
<tr><th>UI Access</th><th><b>Draw</b> > <b>Line</b>.</th></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i><Parameters></i></td><td>Array</td><td>Structured array.
Array("NAME:PolylineParameters",
  "IsPolylineCovered:=", <bool>,
  "IsPolylineClosed:=", <bool>,
  <PolylinePointsArray>,
  <PolylineSegmentsArray>)</td></tr>
<tr><td><i><PolylinePointsArray></i></td><td>Array</td><td>Array("NAME:PolylinePoints", <OnePointArray>,
  <OnePointArray>, ...)</td></tr>
<tr><td><i><OnePointArray></i></td><td>Array</td><td>Array("NAME:PLPoint",
  "X:=", <value>,
  "Y:=", <value>,
  "Z:=", <value>))</td></tr>
<tr><td><i><PolylineSegmentsArray></i></td><td>Array</td><td><PolylineSegmentsArray>
  Array("NAME:PolylineSegments",
  <OneSegmentArray>, <OneSegmentArray>, ...)</td></tr>
<tr><td><i><OneSegmentArray></i></td><td>Array</td><td>Array("NAME:PLSegment",
  "SegmentType:=", <"Line", "Arc", "Spline", or "AngularArc">,
  "StartIndex:=", <value>,</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><td></td><td><table>
<tr><td></td><td></td><td>"NoOfPoints:=", <value&gt;)</td></tr>
<tr><td><i>&lt;AttributesArray&gt;</i></td><td>Array</td><td>Structured array. See: AttributesArray.</td></tr>
</table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	CreatePolyline(<*Parameters*>, <*Attributes*>)
Python Example	

```
oEditor.CreatePolyline(
["NAME:PolylineParameters",
  "IsPolylineCovered:="   , True,
  "IsPolylineClosed:="    , False,
["NAME:PolylinePoints",
  ["NAME:PLPoint",
  "X:="                   , "20000mm",
  "Y:="                   , "-20000mm",
  "Z:="                   , "0mm"
  ],
  ["NAME:PLPoint",
  "X:="                   , "-90000mm",
  "Y:="                   , "20000mm",
  "Z:="                   , "0mm"
  ],
  ["NAME:PLPoint",
  "X:="                    , "10000mm",
  "Y:="                    , "-140000mm",
  "Z:="                    , "0mm"
  ]
],
["NAME:PolylineSegments",
  ["NAME:PLSegment",
  "SegmentType:="          , "Line",
  "StartIndex:="           , 0,
  "NoOfPoints:="           , 2
  ],
  ["NAME:PLSegment",
  "SegmentType:="          , "Line",
  "StartIndex:="           , 1,
  "NoOfPoints:="           , 2
  ]
],
["NAME:PolylineXSection",
  "XSectionType:="         , "None",
  "XSectionOrient:="       , "Auto",
  "XSectionWidth:="       , "0mm",
  "XSectionTopWidth:="    , "0mm",
  "XSectionHeight:="      , "0mm",
  "XSectionNumSegments:=" , "0",
  "XSectionBendType:="    , "Corner"
]],
["NAME:Attributes",
  "Name:="                , "Polyline1",
  "Flags:="               , "",
  "Color:="               , "(143 175 143)",
  "Transparency:="        , 0,
  "PartCoordinateSystem:=", "Global",
  "UDMId:="               , "",
  "MaterialValue:="       , "\"copper\"",
  "SurfaceMaterialValue:=", "\"\"",
  "SolveInside:="         , False,
  "ShellElement:="        , False,
  "ShellElementThickness:=", "0mm",
  "IsMaterialEditable:="  , True,
  "UseMaterialAppearance:=", False,
```

	"IsLightweight:=" , False])

CreateRectangle

Creates a rectangle.

<table>
<tr><td><b>UI Access</b></td><td><b>Draw</b> &gt; <b>Rectangle</b>.</td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;Parameters&gt;</i></td><td>Array</td><td>Structured array.
Array("NAME:RectangleParameters",
"IsCovered:=" , <boolean>,
"XStart:=" , <string>,
"YStart:=" , <string>,
"ZStart:=" , <string>,
"Width:=" , <string>,
"Height:=" , <string>,
"WhichAxis:=" , <string "X", "Y", or "Z"&gt;)</td></tr>
<tr><td><i>&lt;AttributesArray&gt;</i></td><td>Array</td><td>Structured array. See: AttributesArray.</td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	CreateRectangle(<*Parameters*>, <*Attributes*>)
Python Example	oEditor.CreateRectangle(

```
["NAME:RectangleParameters",
  "IsCovered:="           , True,
  "XStart:="              , "-80000mm",
  "YStart:="              , "-90000mm",
  "ZStart:="              , "0mm",
  "Width:="               , "20000mm",
  "Height:="              , "30000mm",
  "WhichAxis:="           , "Z"
],
["NAME:Attributes",
  "Name:="                , "Rectangle1",
  "Flags:="               , "",
  "Color:="               , "(143 175 143)",
  "Transparency:="        , 0,
  "PartCoordinateSystem:=", "Global",
  "UDMId:="               , "",
  "MaterialValue:="       , "\"copper\"",
  "SurfaceMaterialValue:=", "\"\"",
  "SolveInside:="         , False,
  "ShellElement:="        , False,
```

<table>
<tr><td></td><td>"ShellElementThickness:=", "0mm",
"IsMaterialEditable:=" , True,
"UseMaterialAppearance:=", False,
"IsLightweight:=" , False
])</td></tr>
</table>

CreateRegularPolygon

Creates a regular polygon.

<table>
<tr><th>UI Access</th><th><b>Draw</b> &gt; <b>Regular Polygon</b>.</th></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td><i>&lt;Parameters&gt;</i></td><td>Array</td><td>Structured array.
Array("NAME:RegularPolygonParameters",
"IsCovered:=" , <boolean>,
"XCenter:=" , <string>,
"YCenter:=" , <string>,
"ZCenter:=" , <string>,
"XStart:=" , <string>,
"YStart:=" , <string>,
"ZStart:=" , <string>,
"NumSides:=" , <string containing number greater than 2>,
"WhichAxis:=" , <string "X", "Y", or "Z")</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><td></td><td><table><tr><td><i>&lt;AttributesArray&gt;</i></td><td>Array</td><td>Structured array. See: AttributesArray.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

<table>
<tr><td><b>Python Syntax</b></td><td>CreateRegularPolygon(&lt;<i>Parameters</i>&gt;, &lt;<i>Attributes</i>&gt;)</td></tr>
<tr><td><b>Python Example</b></td><td><pre>
oEditor.CreateRegularPolygon(
["NAME:RegularPolygonParameters",
  "IsCovered:="   , True,
  "XCenter:="   , "-70000mm",
  "YCenter:="   , "-100000mm",
  "ZCenter:="   , "0mm",
  "XStart:="   , "-50000mm",
  "YStart:="   , "-80000mm",
  "ZStart:="   , "0mm",
  "NumSides:="   , "12",
  "WhichAxis:="   , "Z"
],
["NAME:Attributes",
  "Name:="   , "Polygon1",
  "Flags:="   , "",
  "Color:="   , "(143 175 143)",
</pre></td></tr>
</table>

	"Transparency:=" , 0, "PartCoordinateSystem:=", "Global", "UDMId:=" , "", "MaterialValue:=" , "\"copper\"", "SurfaceMaterialValue:=", "\"\"", "SolveInside:=" , False, "ShellElement:=" , False, "ShellElementThickness:=", "0mm", "IsMaterialEditable:=" , True, "UseMaterialAppearance:=", False, "IsLightweight:=" , False])

CreateRegularPolyhedron

Creates a regular polyhedron.

<table>
<tr><th>UI Access</th><th><b>Draw</b> > <b>Regular Polyhedron</b>.</th></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td><i>&lt;Parameters&gt;</i></td><td>Array</td><td>Structured array.
Array("NAME:PolyhedronParameters",
"XCenter:=" , <string>,
"YCenter:=" , <string&gt;,</td></tr>
</table>
</td></tr>
</table>

			`"ZCenter:=" , <string>,`
			`"XStart:=" , <string>,`
			`"YStart:=" , <string>,`
			`"ZStart:=" , <string>,`
			`"Height:=" , <string>,`
			`"NumSides:=" , <string containing number greater than 2>,`
			`"WhichAxis:=" , <string "X", "Y", or "Z">)`

	<AttributesArray>	Array	Structured array. See: AttributesArray.
Return Value	None.		

Python Syntax	CreateRegularPolyhedron(<*Parameters*>, <*Attributes*>)

Python Example	`oEditor.CreateRegularPolyhedron(`
	`["NAME:PolyhedronParameters",`
	`  "XCenter:="             , "40000mm",`
	`  "YCenter:="             , "-80000mm",`
	`  "ZCenter:="             , "0mm",`
	`  "XStart:="              , "50000mm",`
	`  "YStart:="              , "-70000mm",`
	`  "ZStart:="              , "0mm",`

```
  "Height:="              , "50000mm",
  "NumSides:="            , "8",
  "WhichAxis:="           , "Z"
],
["NAME:Attributes",
  "Name:="                , "RegularPolyhedron1",
  "Flags:="               , "",
  "Color:="               , "(143 175 143)",
  "Transparency:="        , 0,
  "PartCoordinateSystem:=", "Global",
  "UDMId:="               , "",
  "MaterialValue:="       , "\"copper\"",
  "SurfaceMaterialValue:=", "\"\"",
  "SolveInside:="         , False,
  "ShellElement:="        , False,
  "ShellElementThickness:=", "0mm",
  "IsMaterialEditable:="  , True,
  "UseMaterialAppearance:=", False,
  "IsLightweight:="       , False
])
```

CreateSphere

Creates a sphere.

<table>
<tr><td><b>UI Access</b></td><td><b>Draw</b> &gt; <b>Sphere</b>.</td></tr>
<tr><td><b>Parameters</b></td><td><table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;Parameters&gt;</i></td><td>Array</td><td>Structured array.
Array("NAME:SphereParameters",
  "XCenter:=", <string>,
  "YCenter:=", <string>,
  "ZCenter:=", <string>,
  "Radius:=", <string&gt;)</td></tr>
<tr><td><i>&lt;AttributesArray&gt;</i></td><td>Array</td><td>Structured array. See: AttributesArray.</td></tr>
</table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

<table>
<tr><td><b>Python Syntax</b></td><td>CreateSphere(&lt;<i>Parameters</i>&gt;, &lt;<i>Attributes</i>&gt;)</td></tr>
<tr><td><b>Python Example</b></td><td>oEditor.CreateSphere(
["NAME:SphereParameters",
  "XCenter:="   , "-40000mm",
  "YCenter:="   , "-130000mm",
  "ZCenter:="   , "0mm",
  "Radius:="   , "22360.6797749979mm"</td></tr>
</table>

```
],
["NAME:Attributes",
  "Name:="                , "Sphere1",
  "Flags:="               , "",
  "Color:="               , "(143 175 143)",
  "Transparency:="        , 0,
  "PartCoordinateSystem:=", "Global",
  "UDMId:="               , "",
  "MaterialValue:="       , "\"copper\"",
  "SurfaceMaterialValue:=", "\"\"",
  "SolveInside:="         , False,
  "ShellElement:="        , False,
  "ShellElementThickness:=", "0mm",
  "IsMaterialEditable:="  , True,
  "UseMaterialAppearance:=", False,
  "IsLightweight:="       , False
])
```

CreateSpiral

Creates a spiral by sweeping the specified object(s).

UI Access	**Draw** > **Spiral**.

<table>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td>&lt;SelectionsArray&gt;</td><td>Array</td><td>Structured array. See: SelectionsArray.</td></tr>
<tr><td>&lt;<i>Parameters</i>&gt;</td><td>Array</td><td>Structured array.
Array("NAME:SpiralParameters",
"XCenter:=" , <string>,
"YCenter:=" , <string>,
"ZCenter:=" , <string>,
"YStartDir:=" , <string>,
"ZStartDir:=" , <string>,
"NumThread:=" , <string>,
"RightHand:=" , <boolean>,
"RadiusIncrement:=" , <string&gt;)</td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	CreateSpiral(<*Parameters*>, <*Attributes*>)
Python Example	oEditor.CreateSpiral(["NAME:Selections", "Selections:=" , "Polygon2", "NewPartsModelFlag:=" , "Model"],

	["NAME:SpiralParameters", "XCenter:=" , "-70000mm", "YCenter:=" , "50000mm", "ZCenter:=" , "0mm", "XStartDir:=" , "-60000mm", "YStartDir:=" , "-10000mm", "ZStartDir:=" , "0mm", "NumThread:=" , "1", "RightHand:=" , True, "RadiusIncrement:=" , "1mm"])

CreateTorus

Creates a torus.

<table>
<tr><th>UI Access</th><th><b>Draw</b> &gt; <b>Torus</b>.</th></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;Parameters&gt;</i></td><td>Array</td><td>Structured array.
Array("NAME:TorusParameters",
"XCenter:=" , <string>,
"YCenter:=" , <string>,
"ZCenter:=" , <string&gt;,</td></tr>
</table>
</td></tr>
</table>

		"MajorRadius:=" , <string>, "MinorRadius:=" , <string>, "WhichAxis:=" , <string "X", "Y", or "Z">)
<AttributesArray>	Array	Structured array. See: AttributesArray.

Return Value	None.

Python Syntax	CreateTorus(<*Parameters*>, <*Attributes*>)
Python Example	

```
oEditor.CreateTorus(
["NAME:TorusParameters",
  "XCenter:="                , "0.6mm",
  "YCenter:="                , "-0.6mm",
  "ZCenter:="                , "0mm",
  "MajorRadius:="            , "0.365028153987289mm",
  "MinorRadius:="            , "0.0821854415126694mm",
  "WhichAxis:="              , "Z"
],
["NAME:Attributes",
  "Name:="                   , "Torus1",
  "Flags:="                  , "",
  "Color:="                  , "(143 175 143)",
  "Transparency:="        , 0,
  "PartCoordinateSystem:=", "Global",
  "UDMId:="               , "",
  "MaterialValue:="       , "\"copper\"",
  "SurfaceMaterialValue:=", "\"\"",
  "SolveInside:="         , False,
  "ShellElement:="        , False,
  "ShellElementThickness:=", "0mm",
  "IsMaterialEditable:="  , True,
  "UseMaterialAppearance:=", False,
  "IsLightweight:="       , False
])
```

CreateUserDefinedModel

Creates a user-defined model.

> **Note:** This option can be used to create a UDM from a Discovery model (**Modeler** > Discovery Link).

<table>
<tr><td><b>UI Access</b></td><td><b>Draw</b> &gt; <b>User-Defined Model</b> &gt; [<b>Model</b>]</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><i>&lt;Parameters&gt;</i></td><td>Array</td><td>Structured array.
<code>["NAME:UserDefinedModelParameters",</code></td></tr></table></td></tr>
</table>

		<definitionArray>, <optionsArray>, <geometryParamsArray>, "DllName:=", <string filepath>, "Library:=", <string>, "Version:=", <string> "ConnectionID:=", <string>]
<definitionArray>	Array	Structured array containing string "NAME:Definition"
<optionsArray>	Array	Structured array containing string "NAME:Options"
<geometryParamsArray>	Array	Structured array containing arrays for individual parameters: ["NAME:GeometryParameters", ["NAME:UDMParam", "Name:=" , <string>, "Value:=" , <string>, "PropType2:=" , <integer>, "PropFlag2:=" , <integer>]] Required UDM parameters depend on the UDM being created. To see which properties apply to a UDM, right-click the UDM in the Project Tree and select **Properties**. Then select the **Parameters** tab. PropType2 can be any of the following: • **0** – Property takes a string value. • **1** – Property is a menu option.

			- **2** – Property takes a number (integer or double). - **3** – Property takes a value (numbers, variables, or expressions). - **4** – Property is a file name. - **5** – Property corresponds to a check box. - **6** – Property specifies a 3D position. `PropFlag2` can be any of the following: - **0** – No flags - **1** – Read-only - **2** – Must be integer - **4** – Must be real - **8** – Hidden `PropFlag2` values can be combined. For example, a read-only property that must be an integer would take the value 3. A hidden property that must be real would take the value 12. These values are further described in the `UserDefinedPrimitiveStructures.h` file included with the installation under "...\ANSYS Inc\v252\AnsysEM\UserDefinedPrimitives\Examples\Headers"
Return Value	None		

Python Syntax	CreateUserDefinedModel(<*Parameters*>)
Python Example	`])`

Python Example of Creating a UDM from a Discovery Model

```
oEditor.CreateUserDefinedModel(
[
  "NAME:UserDefinedModelParameters",
  [
    "NAME:Definition",
    [
      "NAME:UDMParam",
      "Name:=" , "GeometryFilePath",
      "Value:=" , "\"C:/Users/chenwan/Desktop/restored_files/tee.dsco\"",
      "DataType:=" , "String",
      "PropType2:=" , 0,
      "PropFlag2:=" , 1
    ],
    [
      "NAME:UDMParam",
      "Name:=" , "ProcessID",
      "Value:=" , "255988",
      "DataType:=" , "Int",
      "PropType2:=" , 2,
      "PropFlag2:=" , 8
    ]
  ],
  [
    "NAME:Options",
    [
      "NAME:UDMParam",
      "Name:=" , "IsDiscoveryLink",
      "Value:=" , "1",
      "DataType:=" , "Int",
      "PropType2:=" , 5,
      "PropFlag2:=" , 8
    ],
    [
      "NAME:UDMParam",
      "Name:=" , "Solid Bodies",
      "Value:=" , "1",
      "DataType:=" , "Int",
          "PropType2:=" , 5,
          "PropFlag2:=" , 0
        ],
        [
          "NAME:UDMParam",
          "Name:=" , "Surface Bodies",
          "Value:=" , "1",
          "DataType:=" , "Int",
          "PropType2:=" , 5,
          "PropFlag2:=" , 0
        ],
        [
          "NAME:UDMParam",
          "Name:=" , "Parameters",
          "Value:=" , "\"eCADImportParameterType_Independent,eCADImportParameterType_None,eCADImportParameterType_Independent,eCADImportParameterType_All\"",
          "DataType:=" , "String",
          "PropType2:=" , 1,
          "PropFlag2:=" , 0
        ],
        [
          "NAME:UDMParam",
          "Name:=" , "Parameter Key",
          "Value:=" , "\"\"",
          "DataType:=" , "String",
          "PropType2:=" , 0,
          "PropFlag2:=" , 0
        ],
        [
          "NAME:UDMParam",
          "Name:=" , "Named Selections",
          "Value:=" , "1",
          "DataType:=" , "Int",
          "PropType2:=" , 5,
          "PropFlag2:=" , 8
        ],
        [
          "NAME:UDMParam",
          "Name:=" , "Materials",
          "Value:=" , "\"None,None,Assignments,Assignments and Properties\"",
          "DataType:=" , "String",
          "PropType2:=" , 1,
          "PropFlag2:=" , 0
        ],
        [
          "NAME:UDMParam",
          "Name:=" , "Import As Lightweight",
          "Value:=" , "0",
          "DataType:=" , "Int",
          "PropType2:=" , 5,
          "PropFlag2:=" , 0
        ],
        [
          "NAME:UDMParam",
          "Name:=" , "Facet Level",
          "Value:=" , "\"3,1,2,3,4,5\"",
          "DataType:=" , "String",
          "PropType2:=" , 1,
          "PropFlag2:=" , 0
        ],
        [
          "NAME:UDMParam",
          "Name:=" , "Cleaning",
          "Value:=" , "0",
          "DataType:=" , "Int",
          "PropType2:=" , 5,
          "PropFlag2:=" , 0
        ],
        [
          "NAME:UDMParam",
          "Name:=" , "Use Parasolid For Transfer",
          "Value:=" , "1",
          "DataType:=" , "Int",
          "PropType2:=" , 5,
    "PropFlag2:=" , 8
  ],
  [
    "NAME:UDMParam",
    "Name:=" , "Tolerant Stitching",
    "Value:=" , "1",
    "DataType:=" , "Int",
    "PropType2:=" , 5,
    "PropFlag2:=" , 8
  ],
  [
    "NAME:UDMParam",
    "Name:=" , "Stitch Tolerance",
    "Value:=" , "0.1",
    "PropType2:=" , 3,
    "PropFlag2:=" , 8
  ],
  [
    "NAME:UDMParam",
    "Name:=" , "Tighten Gaps",
    "Value:=" , "1",
    "DataType:=" , "Int",
    "PropType2:=" , 5,
    "PropFlag2:=" , 8
  ],
  [
    "NAME:UDMParam",
    "Name:=" , "Tighten Gaps Tolerance",
    "Value:=" , "1e-06",
    "PropType2:=" , 3,
    "PropFlag2:=" , 8
  ],
  [
    "NAME:UDMParam",
    "Name:=" , "Use Associativity",
    "Value:=" , "1",
    "DataType:=" , "Int",
    "PropType2:=" , 5,
        "PropFlag2:=" , 8
      ],
      [
        "NAME:UDMParam",
        "Name:=" , "Attributes",
        "Value:=" , "1",
        "DataType:=" , "Int",
        "PropType2:=" , 5,
        "PropFlag2:=" , 8
      ],
      [
        "NAME:UDMParam",
        "Name:=" , "Import Coordinate Systems",
        "Value:=" , "1",
        "DataType:=" , "Int",
        "PropType2:=" , 5,
        "PropFlag2:=" , 8
      ],
      [
        "NAME:UDMParam",
        "Name:=" , "Decompose Disjoint Faces",
        "Value:=" , "1",
        "DataType:=" , "Int",
        "PropType2:=" , 5,
        "PropFlag2:=" , 8
      ],
      [
        "NAME:UDMParam",
        "Name:=" , "Import Using Instances",
        "Value:=" , "1",
        "DataType:=" , "Int",
        "PropType2:=" , 5,
        "PropFlag2:=" , 8
      ],
      [
        "NAME:UDMParam",
    "Name:=" , "Line Bodies",
    "Value:=" , "1",
    "DataType:=" , "Int",
    "PropType2:=" , 5,
    "PropFlag2:=" , 8
  ],
  [
    "NAME:UDMParam",
    "Name:=" , "Mixed Import Resolution",
    "Value:=" , "\"eMixedImport_None\"",
    "DataType:=" , "String",
    "PropType2:=" , 0,
    "PropFlag2:=" , 8
  ],
  [
    "NAME:UDMParam",
    "Name:=" , "Enclosure and Symmetry Processing",
    "Value:=" , "1",
    "DataType:=" , "Int",
    "PropType2:=" , 5,
    "PropFlag2:=" , 8
  ],
  [
    "NAME:UDMParam",
    "Name:=" , "Reader Mode Saves Updated File",
    "Value:=" , "0",
    "DataType:=" , "Int",
    "PropType2:=" , 5,
    "PropFlag2:=" , 8
  ],
  [
    "NAME:UDMParam",
    "Name:=" , "Smart CAD Update",
    "Value:=" , "0",
    "DataType:=" , "Int",
    "PropType2:=" , 5,
    "PropFlag2:=" , 8
  ],
    [
      "NAME:UDMParam",
      "Name:=" , "Import Work Points",
      "Value:=" , "1",
      "DataType:=" , "Int",
      "PropType2:=" , 5,
      "PropFlag2:=" , 8
    ]
  ],
  [
    "NAME:GeometryParams"
  ],
  "DllName:=" , "SACADIntegUDM",
  "Library:=" , "installLib",
  "Version:=" , "1.0",
  "ConnectionID:=" , ""
])
```

CreateUserDefinedPart

Creates a user-defined part.

<table>
<tr><th>UI Access</th><th><b>Draw</b> &gt; <b>User-Defined Primitive</b> &gt; [<b>Part</b>]</th></tr>
<tr><td><b>Parameters</b></td><td><table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;Parameters&gt;</i></td><td>Array</td><td>Structured array.
<code>["NAME:UserDefinedPrimitiveParameters",</code>
<code> "DllName:=" , <string&gt;,</code>
<code> "Version:=" , <string&gt;,</code>
<code> "NoOfParameters:=" , <integer&gt;,</code>
<code> "Library:=" , <string&gt;,</code></td></tr>
</table></td></tr>
</table>

		`<paramVectorArray>]`

<paramVectorArray>	Array	Structured array containing arrays for each pair:
		`["NAME:ParamVector",`
		`  <pair>, <pair>, <pair>,...`
		`]`
		`<pair>:`
		`["NAME:Pair",`
		`  "Name:=" , <string>,`
		`  "Value:=" , <string>]`

<Attributes>	Array	Structured array. See: AttributesArray.

Return Value	None.

Python Syntax	CreateUserDefinedPart(<*Parameters*>, <*paramVectorArray*>, <*Attributes*>)

Python Example	`oEditor.CreateUserDefinedPart(`
	`["NAME:UserDefinedPrimitiveParameters",`
	`"DllName:=" , "RMxprt/LapCoil.dll",`
	`"Version:=" , "16.0",`
	`"NoOfParameters:=" , 22,`
	`"Library:=" , "syslib",`
	`  ["NAME:ParamVector",`
	`    ["NAME:Pair",`

	"Name:=" , "DiaGap", "Value:=" , "100mm"], ["NAME:Pair", "Name:=" , "DiaYoke", "Value:=" , "20mm"], ["NAME:Pair", "Name:=" , "Length", "Value:=" , "100mm"], ["NAME:Pair", "Name:=" , "Skew", "Value:=" , "0deg"], ["NAME:Pair", "Name:=" , "Slots", "Value:=" , "18"], ["NAME:Pair",

	"Name:=" , "SlotType", "Value:=" , "1"], ["NAME:Pair", "Name:=" , "Hs0", "Value:=" , "1mm"], ["NAME:Pair", "Name:=" , "Hs1", "Value:=" , "1mm"], ["NAME:Pair", "Name:=" , "Hs2", "Value:=" , "10mm"], ["NAME:Pair", "Name:=" , "Bs0", "Value:=" , "2.5mm"], ["NAME:Pair", "Name:=" , "Bs1",

```
"Value:=" , "8mm"
],
["NAME:Pair",
"Name:=" , "Bs2",
"Value:=" , "5mm"
],
["NAME:Pair",
"Name:=" , "Rs",
"Value:=" , "0mm"
],
["NAME:Pair",
"Name:=" , "FilletType",
"Value:=" , "0"
],
["NAME:Pair",
"Name:=" , "Layers",
"Value:=" , "2"
],
["NAME:Pair",
"Name:=" , "CoilPitch",
"Value:=" , "4"
],
["NAME:Pair",
"Name:=" , "EndExt",
"Value:=" , "5mm"
],
["NAME:Pair",
"Name:=" , "SpanExt",
"Value:=" , "25mm"
],
["NAME:Pair",
"Name:=" , "BendAngle",
"Value:=" , "0deg"
],
["NAME:Pair",
"Name:=" , "SegAngle",
"Value:=" , "10deg"
],
["NAME:Pair",
"Name:=" , "LenRegion",
"Value:=" , "200mm"
```

```
    ],
    ["NAME:Pair",
    "Name:=" , "InfoCoil",
    "Value:=" , "0"
    ]
  ]
],
["NAME:Attributes",
  "Name:=" , "LapCoil1",
  "Flags:=" , "",
  "Color:=" , "(143 175 143)",
  "Transparency:=" , 0,
  "PartCoordinateSystem:=", "Global",
  "UDMId:=" , "",
  "MaterialValue:=" , "\"copper\"",
  "SurfaceMaterialValue:=", "\"\"",
  "SolveInside:=" , False,
  "ShellElement:=" , False,
  "ShellElementThickness:=", "0mm",
  "IsMaterialEditable:=" , True,
```

	`  "UseMaterialAppearance:=", False,`
	`  "IsLightweight:=" , False`
	`])`

Edit3DComponent

Edits a specified 3D component by saving it to a new name so that its properties are accessible. To change the properties on an existing component without changing its name, see ChangeProperty.

<table>
<tr><th>UI Access</th><th>N/A</th></tr>
<tr><td>Parameters</td><td>
<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td><em>&lt;compName&gt;</em></td><td>String</td><td>Component name.</td></tr>
<tr><td><em>&lt;Parameters&gt;</em></td><td>Array</td><td>Structured array.
<code>Array("NAME:EditComponentParametersData",</code>
<code> "NewComponentName:=", <string&gt;,</code>
<code> "GeometryParameters:=", <string&gt;,</code>
<code> "MaterialParameters:=", <string&gt;,</code>
<code> "DesignParameters:=", <string&gt;,</code>
<code> &lt;ComponentMeshing&gt;,</code>
<code> &lt;Excitations&gt;)</code></td></tr>
<tr><td><em>&lt;ComponentMeshing&gt;</em></td><td>Array</td><td>Structured array.
<code>Array("NAME:Component Meshing",</code>
<code> "MeshAssembly:=", <boolean&gt;)</code></td></tr>
<tr><td><em>&lt;Excitations&gt;</em></td><td>Array</td><td>Structured array containing array of suppressed excitations.</td></tr>
</table>
</td></tr>
</table>

			`Array("NAME:Excitations",`
			`  "Suppressed:=", <array>)`

Return Value	None.		

Python Syntax	Edit3DComponent (<*compName*>, <*Parameters*>)

Python Example	`oEditor.Edit3DComponent(`
	`"Connector1",`
	`  ["NAME:EditComponentParametersData",`
	`    "NewComponentName:=", "Connector2",`
	`    "GeometryParameters:=", "",`
	`    "MaterialParameters:=", "",`
	`    "DesignParameters:=", "",`
	`  ["NAME:Component Meshing",`
	`    "MeshAssembly:=", False],`
	`  ["NAME:Excitations",`
	`    "Suppressed:=", []]`
	`  ]`
	`)`

Edit3DComponentDefinition

Edits definitions of a specified 3D component.

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td><i>&lt;Parameters&gt;</i></td><td>Array</td><td>Structured array.
Array("NAME:EditComponentParametersData",
"OriginalComponentName:=", <string>,
"NewComponentName:=", <string>,
"GeometryParameters:=", <string>,
"MaterialParameters:=", <string>,
"DesignParameters:=", <string>,
<ComponentMeshing>,
<Excitations&gt;)</td></tr>
<tr><td><i>&lt;ComponentMeshing&gt;</i></td><td>Array</td><td>Structured array.
Array("NAME:Component Meshing",
"MeshAssembly:=", <boolean&gt;)</td></tr>
<tr><td><i>&lt;Excitations&gt;</i></td><td>Array</td><td>Structured array containing array of suppressed excitations.
Array("NAME:Excitations",
"Suppressed:=", <array&gt;)</td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
<tr><td><b>Python Syntax</b></td><td>Edit3DComponent (&lt;<i>Parameters</i>&gt;)</td></tr>
</table>

<table>
<tr><td><b>Python Example</b></td><td><pre>oEditor.Edit3DComponent(
  ["NAME:EditComponentParametersData",
  "OriginalComponentName:=", "Connector1",
  "NewComponentName:=", "Connector2",
  "GeometryParameters:=", "",
  "MaterialParameters:=", "",
  "DesignParameters:=", "",
  ["NAME:Component Meshing",
    "MeshAssembly:=", False],
  ["NAME:Excitations",
    "Suppressed:=", []]
  ]
)</pre></td></tr>
</table>

EditNativeComponentDefinition [Beta – Layout Components in Mechanical]

Use this command to edit the definition of an existing Layout Component in a Mechanical design. Layout components are supported for steady-state thermal, transient thermal, and structural solution types.

<table>
<tr><td><b>UI Access</b></td><td>In the Project Manager, expand <b>3D Components</b>, right-click on the component, and select <b>Edit Definition.</b>.</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><th>Name</th><th>Type</th><th>Description</th></tr><tr><td><i>&lt;LinkedParameters&gt;</i></td><td>Array</td><td>Structured array containing data of inserted layout component</td></tr></table></td></tr>
</table>

Return Value	None

<table>
<tr><th>Python Syntax</th><th>EditNativeComponentDefinition (&lt;<em>LinkedParameters</em>&gt;)</th></tr>
<tr><td><b>Python Example</b>
<b>Thermal</b> Solution
(with a custom grid resolution)</td><td>

```
oEditor.EditNativeComponentDefinition(
    [
        "NAME:EditNativeComponentDefinitionData",
        "DefinitionName:="              , "LC1",
        [
            "NAME:GeometryDefinitionParameters",
            [
                "NAME:VariableOrders"
            ]
        ],
        [
            "NAME:DesignDefinitionParameters",
            [
                "NAME:VariableOrders"
            ]
        ],
        [
            "NAME:MaterialDefinitionParameters",
            [
                "NAME:VariableOrders"
            ]
        ],
        "NextUniqueID:="                , 0,
        "MoveBackwards:="               , False,
        "DatasetType:="                 , "ComponentDatasetType",
```

</td></tr>
</table>

```
        [
            "NAME:DatasetDefinitions"
        ],
        [
            "NAME:NativeComponentDefinitionProvider",
            "Type:="                   , "Layout Component",
            "Unit:="                   , "mm",
            "Version:="                , 1.1,
            "EDBDefinition:="          , "RPS14_3",
            [
                "NAME:VariableMap"
            ],
            "ReferenceCS:="            , "",
            "CSToImport:="             , [],
            "BoardCutoutMaterial:="    , "air"
            "ViaHoleMaterial:="        , "FR4_epoxy",
            "ExtentsType:="            , "Polygon",
            "CustomResolution:="       , True,  # Use custom grid resolution:
            "CustomResolutionCol:="    , 1350,  # Number of columns (X-resolution)
            "CustomResolutionRow:="    , 375,   # Number of rows (Y-resolution)
            "UseThermalLink:="         , False,
            [
                "NAME:TopBoundary",
                "BoundaryType:="       , "Convection",
                "FilmCoefficient:="    , "20w_per_m2kel",
                "ReferenceTemperature:=", "AmbientTemp"
                "UseRadiation:="       , True,
                "ViewFactor:="         , 1,
                "Emissivity:="         , 0.8,
                "RadiationTemperature:=", "AmbientTemp"
            ],
            [
```

Python Example

Structural Solution

(with resolution set by the slider and with imported temperatures)

```
oEditor.EditNativeComponentDefinition(
    [
        "NAME:InsertNativeComponentData",
        "DefinitionName:="              , "LC1",
        [
            "NAME:GeometryDefinitionParameters",
            [
                "NAME:VariableOrders"
            ]
        ],
        [
            "NAME:DesignDefinitionParameters",
            [
                "NAME:VariableOrders"
            ]
        ],
        [
            "NAME:MaterialDefinitionParameters",
            [
                "NAME:VariableOrders"
            ]
        ],
        "NextUniqueID:="                , 0,
        "MoveBackwards:="               , False,
        "DatasetType:="                 , "ComponentDatasetType",
        [
            "NAME:DatasetDefinitions"
        ],
        [
            "NAME:NativeComponentDefinitionProvider",
            "Type:="                    , "Layout Component",
            "Unit:="                    , "mm",
            "Version:="                 , 1.1,
            "EDBDefinition:="           , "TraceMappingTest_EMDesign2",
            [
                "NAME:VariableMap"
            ],
            "ReferenceCS:="             , "",
            "BoardCutoutMaterial:="     , "air",  #      400, or 800, respectively,
along the short dimension
            "ViaHoleMaterial:="         , "FR4_epoxy",
            "ExtentsType:="             , "Polygon",
            "CSToImport:="              , [],
            "CustomResolution:="        , False,  # <--- Not custom; use resolution
slider:
            "SliderResolution:="        , 3,      # <--- Tick mark (0, 1, 2, 3, or
4) represents 50, 100, 200
            [
                "NAME:TopBoundary",
                "BoundaryType:="        , "Force",
                "ForceX:="              , "2.5newton",
                "ForceY:="              , "-4newton",
                "ForceZ:="              , "0newton"
            ],
            [
                "NAME:BottomBoundary",
                "BoundaryType:="        , "FrictionLessSupport"
            ],
            "EnableThermalCondtion:="   , True,                   #
            "BoundaryType:="            , "ThermalCondtion",      #
            "Uniform:="                 , False,                  #
```

```
"ComponentName:="                   , "LC1",
     "Company:="                         , "",
     "Company URL:="                     , "",
     "Model Number:="                    , "",
     "Help URL:="                        , "",
     "Version:="                         , "1.1",
     "Notes:="                           , "",
     "IconType:="                        , "Layout Component"
 ])
```

EditPolyline

Modifies a specified polyline. See: CreatePolyline.

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;SelectionsArray&gt;</i></td><td>Array</td><td>Structured array. See: SelectionsArray.</td></tr>
<tr><td><i>&lt;Parameters&gt;</i></td><td>Array</td><td>Structured array.
<code>Array("NAME:PolylineParameters",</code>
<code> "IsPolylineCovered:=", <bool&gt;,</code>
<code> "IsPolylineClosed:=", <bool&gt;,</code>
<code> &lt;PolylinePointsArray&gt;,</code>
<code> &lt;PolylineSegmentsArray&gt;)</code></td></tr>
<tr><td> <i>&lt;PolylinePointsArray&gt;</i></td><td>Array</td><td><code>Array("NAME:PolylinePoints", <OnePointArray&gt;,</code>
<code> &lt;OnePointArray>, ...)</code></td></tr>
<tr><td>   <i>&lt;OnePointArray&gt;</i></td><td>Array</td><td><code>Array("NAME:PLPoint",</code>
<code> "X:=", <value&gt;,</code></td></tr>
</table>
</td></tr>
</table>

		"Y:=", <value>, "Z:=", <value>))
<PolylineSegmentsArray>	Array	<PolylineSegmentsArray> Array("NAME:PolylineSegments", <OneSegmentArray>, <OneSegmentArray>, ...)
<OneSegmentArray>	Array	Array("NAME:PLSegment", "SegmentType:=", <"Line", "Arc", "Spline", or "AngularArc">, "StartIndex:=", <value>, "NoOfPoints:=", <value>)

Return Value	None.

Python Syntax	EditPolyline(<*SelectionsArray*>, <*Parameters*>)

Python Example	`oEditor.EditPolyline(`
	`["NAME:Selections",`
	`  "Selections:=", "Polyline1"]`
	`["NAME:PolylineParameters",`
	`  "IsPolylineCovered:="   , True,`
	`  "IsPolylineClosed:="    , False,`
	`  ["NAME:PolylinePoints",`
	`    ["NAME:PLPoint",`

```
    "X:="                    , "20000mm",
    "Y:="                    , "-20000mm",
    "Z:="                    , "0mm"
  ],
  ["NAME:PLPoint",
    "X:="                    , "-90000mm",
    "Y:="                    , "20000mm",
    "Z:="                    , "0mm"
  ],
  ["NAME:PLPoint",
    "X:="                    , "10000mm",
    "Y:="                    , "-140000mm",
    "Z:="                    , "0mm"
  ]
],
["NAME:PolylineSegments",
  ["NAME:PLSegment",
  "SegmentType:="          , "Line",
  "StartIndex:="           , 0,
  "NoOfPoints:="           , 2
  ],
  ["NAME:PLSegment",
  "SegmentType:="         , "Line",
  "StartIndex:="          , 1,
  "NoOfPoints:="          , 2
  ]
],
["NAME:PolylineXSection",
  "XSectionType:="        , "None",
  "XSectionOrient:="      , "Auto",
  "XSectionWidth:="       , "0mm",
  "XSectionTopWidth:="    , "0mm",
  "XSectionHeight:="      , "0mm",
  "XSectionNumSegments:=" , "0",
  "XSectionBendType:="    , "Corner"
]]
)
```

Get3DComponentDefinitionNames

Gets names of 3D component definitions.

UI Access	N/A

Parameters	None.
Return Value	Array of strings containing component definition names.

Python Syntax	Get3DComponentDefinitionNames()
Python Example	`oEditor.Get3DComponentDefinitionNames()`

Get3DComponentInstanceNames

Returns instance names of 3D component definitions.

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;<i>DefinitionName</i>&gt;</td><td>String</td><td>Definition name.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>Array containing instance names.</td></tr>
</table>

Python Syntax	Get3DComponentInstanceNames(<*DefinitionName*>)
Python Example	`oEditor.Get3DComponentInstanceNames("Connector")`

Get3DComponentMaterialNames

Returns material names for a specified *.a3dcomp format 3D component.

UI Access	N/A

<table>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;<i>InstanceName</i>&gt;</td><td>String</td><td>Component instance name.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>Array containing material names.</td></tr>
</table>

Python Syntax	Get3DComponentMaterialNames(<*InstanceName*>)
Python Example	`oEditor.Get3DComponentMaterialNames("Connector1.a3dcomp")`

Get3DComponentMaterialProperties

Returns material properties for a specified 3D component.

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;<i>MaterialName</i>&gt;</td><td>String</td><td>Material name.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>Array containing material properties.</td></tr>
</table>

Python Syntax	Get3DComponentMaterialProperties(<*MaterialName*>)
Python Example	`oEditor.Get3DComponentMaterialProperties('Connector1:Material01')`

Get3DComponentParameters

Returns parameters for a specified 3D component.

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;<i>compName</i>&gt;</td><td>String</td><td>3D component name.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>Array containing component parameters.</td></tr>
</table>

Python Syntax	Get3DComponentParameters(<*compName*>)
Python Example	`oEditor.Get3DComponentParameters('Connector')`

Get3DComponentPartNames

Returns part names for a specified 3D component.

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;<i>InstanceName</i>&gt;</td><td>String</td><td>3D component instance.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>Array containing part names.</td></tr>
</table>

Python Syntax	Get3DComponentParameters(<*InstanceName*>)
Python Example	`oEditor.Get3DComponentPartNames('Connector')`

Insert3DComponent

Inserts a 3D component in specified coordinate system with specified offset.

UI Access	**Draw** > **3D Component Library** > **Browse** > [**Component**].

	Name	Type	Description

Parameters	<*ComponentData*>	Array	Structured array.
			`Array("NAME:InsertComponentData",`
			`"TargetCS:=", <string>,`
			`"ComponentFile:=", <string filepath>),`
			`"IsLocal:", <Boolean>,`
			`"InstanceParameters:=", <string>,`
			`"XOffset:=" , "float with unit",`
			`"YOffset:=" , "float with unit",`
			`"ZOffset:=" , "float with unit")`

Return Value	None.		

Python Syntax	Insert3DComponent(<*ComponentData*>)

Python Example	`oEditor.Insert3DComponent(`
	`[`
	`"NAME:InsertComponentData",`
	`"TargetCS:=" , "Global",`
	`"ComponentFile:=" , "C:/Program Files/ANSYS Inc/v252/AnsysEM/syslib/3DComponents/HFSS/Rectangular Waveguide/Straight.a3dcomp",`
	`"IsLocal:=" , False,`

```
    "UniqueIdentifier:=" , "",
    [
      "NAME:InstanceParameters",
      "GeometryParameters:=" , "Flange=\'0.4in\' FlangeFillet=\'0.2in\'
      FlangeThickness=\'0.2in\' GuideHeight=\'0.4in\' GuideLength=\'4in\'
      GuideWidth=\'0.9in\' WallThickness=\'0.1in\'",
      "MaterialParameters:=" , "",
      "DesignParameters:=" , ""
    ],
    "XOffset:=" , "-70mm",
    "YOffset:=" , "45mm",
    "ZOffset:=" , "0mm"
  ])
```

InsertComponent

Inserts a component.

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;ComponentData&gt;</i></td><td>Array</td><td>Structured array.
Array("NAME:InsertComponentData",
  "Parameters:=", <string>,
  "TargetCS:=", <string&gt;,</td></tr>
</table>
</td></tr>
</table>

	"ComponentFile:=", <string filepath>)
Return Value	None.

Python Syntax	InsertComponent(<*ComponentData*>)

Python Example	`oEditor.InsertComponent(`
	`["NAME:InsertComponentData",`
	`        "Parameters:=", "",`
	`        "TargetCS:=", "Global",`
	`        "ComponentFile:=", "C:\tmp\Connector.a3dcomp"])`

InsertNativeComponent [Beta – Layout Component in Mechanical]

Use this command to insert a Layout Component into a Mechanical design. Layout components are supported for steady-state thermal, transient thermal, and structural solution types.

> **Note:**
>
> When the layout component being inserted into the Mechanical design is not already present in the project, the *oEditor.InsertNativeComponent* command must be preceded by the *oComponentManager.Add* command to add the component to the project's *Components* Library. (See the **Thermal Solution** Python example below.
>
> Similarly, when the source design contains geometry variables that you want to map to local mechanical design variables or AEDT project variables, one of the following commands must precede *oEditor.InsertNativeComponent*, depending on the preferred local variable type:
>
> - *oDesign.ChangeProperty* (see the **Thermal Solution** Python example below)
> - *oProject.ChangeProperty* (see the **Structural Solution** Python example below)

<table>
<tr><td><b>UI Access</b></td><td>In the Project Manager, right-click <b>3D Components</b> and select <b>Browse Layout Component</b>.</td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td><i>&lt;LinkedParameters&gt;</i></td><td>Array</td><td>Structured array containing data of inserted layout component.
See Python examples for details. Most parameters are self-explanatory.</td></tr>
<tr><td colspan="3">The following partial list includes parameters that need further explanation:</td></tr>
<tr><td><i>&lt;MapInstanceParameters&gt;</i></td><td>String</td><td>Local variable type: "DesignVariable" or "ProjectVariable"
This parameter is applicable when source geometry variables are mapped to layout component instance parameters and you have choosen to create local design or project variables of the instance parameters.</td></tr>
<tr><td><i>&lt;CustomResolution&gt;</i></td><td>Bool</td><td>Use slider position for trace mapping grid resolution when False
Specify X & Y grid divisions numerically when True</td></tr>
<tr><td><i>&lt;UseThermalLink&gt;</i></td><td>Bool</td><td>Import the power dissipation from a DCIR or AC analysis in the source HFSS 3D Layout design</td></tr>
</table>
</td></tr>
</table>

			(applicable to *Steady-State Thermal* and *Transient Thermal* Mechanical solutions)
	<*LayoutAnalysisType*>	String	Specify the type of solution in the source design for imported power dissipation (applicable when *UseThermalLink* is True. Choices are "DCIR" and "AC")
	<*SliderResolution*>	Integer	Slider position (tick mark number) used when <*CustomResolution*> = False (controls resolution along the shorter board dimension: Values 0–4 represent resolutions of 50, 100, 200, 400, and 800, respectively)
	<*CustomResolutionCol*>	Integer	Number of columns (X resolution) when <*CustomResolution*> = True
	<*CustomResolutionRow*>	Integer	Number of rows (Y resolution) when <*CustomResolution*> = True
Return Value	None		

Python Syntax	InsertNativeComponent (<*LinkedParameters*>)
Python Example **Thermal** Solution (with	`# Add an HFSS 3D Layout component (from TMT_03.aedb) to the current project:`

a custom grid resolution)

```
oComponentManager.Add(
   [
      "NAME:TMT_03",
      "Info:=",
      [
         "Type:="              , 29,
         "NumTerminals:="      , 0,
         "DataSource:="        , "",
         "ModifiedOn:="        , 1711743004,
         "Manufacturer:="      , "",
         "Symbol:="            , "TMT_03",
         "ModelNames:="        , "",
         "Footprint:="         , "",
         "Description:="       , "",
         "InfoTopic:="         , "",
         "InfoHelpFile:="      , "",
         "IconFile:="          , "BlueDot.bmp",
         "Library:="           , "",
         "OriginalLocation:=", "Project",
         "IEEE:="              , "",
         "Author:="            , "",
         "OriginalAuthor:="    , "",
         "CreationDate:="      , 1711743004,
         "ExampleFile:="       , "",
         "HiddenComponent:="  , 0,
     , 0,
         "GroupID:="           , 0
      ],
      "CircuitEnv:="           , 0,
      "Refbase:="              , "U",
      "NumParts:="             , 1,
      "ModSinceLib:="          , True,
      "CompExtID:="            , 9,
      "ModelEDBFilePath:="     , "D:\\Ansys\\2024R2\\TMT_03.aedb",
      "EDBCompPassword:="      , ""
   ])

# Add a design variable (L1_Thickness) for the mapped source parameter used to control the
thickness of the top PCB layer:

oDesign = oProject.SetActiveDesign("MechanicalDesign1")
oDesign.ChangeProperty(
   [
      "NAME:AllTabs",
      [
         "NAME:LocalVariableTab",
         [
            "NAME:PropServers",
            "LocalVariables"
         ],
            "NAME:NewProps",
            [
               "NAME:L1_Thickness",
               "PropType:=", "VariableProp",
               "UserDef:=" , True,
               "Value:="   , "4mm"
            ]
         ]
      ]
   ])

# Add the layout component to the current mechanical design - thermal solution:

oEditor.InsertNativeComponent(
   [
      "NAME:InsertNativeComponentData",
      "TargetCS:="                  , "Global",
      "SubmodelDefinitionName:="    , "LC1",
      [
         "NAME:ComponentPriorityLists"
      ],
      "NextUniqueID:="              , 0,
      "MoveBackwards:="             , False,
      "DatasetType:="               , "ComponentDatasetType",
      [
         "NAME:DatasetDefinitions"
      ],
      [
         "NAME:BasicComponentInfo",
         "ComponentName:="          , "LC1",
         "Company:="                , "",
   "NAME:VariableOrders"
       ]
    ],
    [
       "NAME:DesignDefinitionParameters",
       [
          "NAME:VariableOrders"
       ]
    ],
    [
       "NAME:MaterialDefinitionParameters",
       [
          "NAME:VariableOrders"
       ]
    ],
    "DefReferenceCSID:="          , 1,
    "MapInstanceParameters:="     , "DesignVariable",
    "UniqueDefinitionIdentifier:=", "13d6e53d-2f8c-4d0f-86ce-26b2583e17e4",
    "OriginFilePath:="            , "",
    "IsLocal:="                   , False,
    "ChecksumString:="            , "",
    "ChecksumHistory:="           , [],
    "VersionHistory:="            , [],
    [
       "NAME:NativeComponentDefinitionProvider",
       "Type:="                   , "Layout Component",
       "Unit:="                   , "mm",
       "Version:="                , 1.1,
       "EDBDefinition:="          , "TMT_03",
       [
```

<table>
<tr>
<td></td>
<td>

```
# if UseThermalLink = True.
      ],
      [
         "NAME:InstanceParameters",
         "GeometryParameters:="     , "",
         "MaterialParameters:="     , "",
         "DesignParameters:="       , ""
      ]
   ])
```

</td>
</tr>
<tr>
<td>

Python Example

Structural Solution

(with resolution set by the slider and with imported tem-

</td>
<td>

```
# Add a project variable ($L1_Thickness) for the mapped source parameter used to control
the thickness of the top PCB layer:
# Note: When specifying the variable name in the UI (Browse Layout Component: Variable Mapping),
do not include the dollar sign ($) prefix. It will be added automatically to the
beginning of the name you specify for project variables.

oProject = oDesktop.SetActiveProject("InstanceVariationTest")
oProject.ChangeProperty(
   [
      "NAME:AllTabs",
      [
         "NAME:ProjectVariableTab",
         [
            "NAME:PropServers",
            "ProjectVariables"
         ],
         [
            "NAME:NewProps",
```

</td>
</tr>
</table>

per-atur-es)

```
            [
               "NAME:$L1_Thickness",
               "PropType:="     , "VariableProp",
               "UserDef:="      , True,
               "Value:="        , "4mm"
            ]
         ]
      ]
   ])

# Add the layout component to the current mechanical design - structural solution:

oDesign = oProject.SetActiveDesign("MechanicalDesign2")
oEditor = oDesign.SetActiveEditor("3D Modeler")
oEditor.InsertNativeComponent(
   [
      "NAME:InsertNativeComponentData",
      "TargetCS:="                    , "Global",
      "SubmodelDefinitionName:="      , "LC1",
      [
         "NAME:ComponentPriorityLists"
      ],
      "NextUniqueID:="                , 0,
      "MoveBackwards:="               , False,
      "DatasetType:="                 , "ComponentDatasetType",
      [
         "NAME:DatasetDefinitions"
   ],
      [
         "NAME:GeometryDefinitionParameters",
         [
            "NAME:VariableOrders"
         ]
      ],
      [
         "NAME:DesignDefinitionParameters",
         [
            "NAME:VariableOrders"
         ]
      ],
      [
         "NAME:MaterialDefinitionParameters",
         [
            "NAME:VariableOrders"
         ]
      ],
      "DefReferenceCSID:="            , 1,
      "MapInstanceParameters:="       , "DesignVariable",
      "UniqueDefinitionIdentifier:=", "7bbea9cb-1775-4d61-82a0-8b386e89ee80",
      "OriginFilePath:="              , "",
      "IsLocal:="                     , False,
      "ChecksumString:="              , "",
      "ChecksumHistory:="             , [],
      "VersionHistory:="              , [],
      [
         "NAME:NativeComponentDefinitionProvider",
         "Type:="                     , "Layout Component",
         "Unit:="                     , "mm",
         "Version:="                  , 1.1,
:="     , True,                 #
         "PathRelativeTo:="         , "TargetProject"       #
      ],
      [
         "NAME:InstanceParameters",
         "GeometryParameters:="     , "",
         "MaterialParameters:="     , "",
         "DesignParameters:="       , ""
      ]
   ])
```

InsertPolylineSegment

Insets a polyline segment before or after a specified existing segment.

<table>
<tr><th>UI Access</th><th><b>Draw</b> > <b>Line Segment</b> > [<b>Selection</b>].</th></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;Parameters&gt;</i></td><td>Array</td><td>Structured array.
Array("NAME:Insert Polyline Segment",
"Selections:=" , <string>,
"Segment Indices:=" , <array containing integers>,
"At Start:=" , <boolean>,
"SegmentType:=" , <string "Line", "Arc", "Spline", or "AngularArc">,
<PolylinePointsArray&gt;</td></tr>
<tr><td><i>&lt;PolylinePointsArray&gt;</i></td><td>Array</td><td>Structured array. See: CreatePolyline.</td></tr>
</table>
</td></tr>
</table>

Return Value	None.

Python Syntax	InsertPolylineSegment(<*Parameters*>)
Python Example	

```
oEditor.InsertPolylineSegment(
  ["NAME:Insert Polyline Segment",
  "Selections:="          , "Polyline1:CreatePolyline:1",
  "Segment Indices:="     , [0],
  "At Start:="            , True,
  "SegmentType:="         , "Line",
  ["NAME:PolylinePoints",
    ["NAME:PLPoint",
    "X:="                    , "1.1mm",
    "Y:="                    , "0.8mm",
    "Z:="                    , "0mm"
    ],
    ["NAME:PLPoint",
    "X:="                    , "0.6mm",
    "Y:="                    , "-0.8mm",
    "Z:="                    , "0mm"
    ]
  ]
```

	`])`

SweepAlongPath

Sweeps the specified 1D or 2D parts along a path. The last 1D object specified is the path for the sweep.

UI Access	**Draw** > **Sweep** > **Along Path**.
Parameters	Name | Type | Description *<SelectionsArray>* | Array | Structured array. See: SelectionsArray. *<PathSweepParametersArray>* | Array | `Array("NAME:PathSweepParameters", "DraftAngle:=", <value>, "DraftType:=", <string>, "CheckFaceFaceIntersection:=", <bool>, "TwistAngle:=", <value>)` Possible values for DraftType are "Extended", "Round", and "Natural".
Return Value	None.

Python Syntax	SweepAlongPath(<*SelectionsArray*>, <*PathSweepParametersArray*>)

Python Example	`oEditor.SweepAlongPath(`
	`[`
	`"NAME:Selections",`

	"Selections:=" , "Rectangle1,Polyline1", "NewPartsModelFlag:=" , "Model"], ["NAME:PathSweepParameters", "DraftAngle:=" , "0deg", "DraftType:=" , "Round", "CheckFaceFaceIntersection:=", False, "TwistAngle:=" , "0deg"])

SweepAlongVector

Sweeps the specified 1D or 2D parts along a vector.

<table>
<tr><th>UI Access</th><th>Draw > Sweep > Along Vector.</th></tr>
<tr><td>Parameters</td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><SelectionsArray></td><td>Array</td><td>Structured array. See: SelectionsArray.</td></tr>
<tr><td><VecSweepParametersArray></td><td>Array</td><td>Array("NAME:VectorSweepParameters",
"DraftAngle:=", <value>,
"DraftType:=", <string>,
"CheckFaceFaceIntersection:=", <bool>,
"SweepVectorX:=", <value></td></tr>
</table>
</td></tr>
</table>

			"SweepVectorY:=", <value> "SweepVectorZ:=", <value>) Possible values for DraftType are "Extended", "Round", and "Natural".
Return Value	None.		

Python Syntax	SweepAlongVector(<SelectionsArray>, <VecSweepParametersArray>)

Python Example	oEditor.SweepAlongVector(
	  [
	    "NAME:Selections",
	    "Selections:=" , "Rectangle1",
	    "NewPartsModelFlag:=" , "Model"
	  ],
	  [
	    "NAME:VectorSweepParameters",
	    "DraftAngle:=" , "0deg",
	    "DraftType:=" , "Round",
	    "CheckFaceFaceIntersection:=", False,
	    "SweepVectorX:=" , "0mm"
	    "SweepVectorY:=" , "0mm"

	`"SweepVectorX:=" , "12mm"`
	`])`

SweepAroundAxis

Sweeps the specified 1D or 2D parts around an axis.

<table>
<tr><td><b>UI Access</b></td><td><b>Draw</b> &gt; <b>Sweep</b> &gt; <b>Around Axis</b>.</td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td>&lt;<i>SelectionsArray</i>&gt;</td><td>Array</td><td>Structured array. See: SelectionsArray.</td></tr>
<tr><td>&lt;<i>AxisSweepParametersArray</i>&gt;</td><td>Array</td><td><code>Array("NAME:AxisSweepParameters",</code>
<code> "DraftAngle:=", <value&gt;,</code>
<code> "DraftType:=", <string&gt;,</code>
<code> "CheckFaceFaceIntersection:=", <bool&gt;,</code>
<code> "SweepAxis:=", <value&gt;</code>
<code> "SweepAngle:=", <value&gt;</code>
<code> "NumOfSegments:=", <value&gt;)</code>
Possible values for DraftType are "Extended", "Round", and "Natural".
Possible values for SweepAxis are "X", "Y", and "Z".</td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	SweepAroundAxis(<*SelectionsArray*>, <*AxisSweepParametersArray*>)

Python Example	`oEditor.SweepAroundAxis(`
	`  [`
	`    "NAME:Selections",`
	`    "Selections:=" , "Rectangle1",`
	`    "NewPartsModelFlag:=" , "Model"`
	`  ],`
	`  [`
	`    "NAME:AxisSweepParameters",`
	`    "DraftAngle:=" , "0deg",`
	`    "DraftType:=" , "Round",`
	`    "CheckFaceFaceIntersection:=", False,`
	`    "SweepAxis:=" , "X"`
	`    "SweepAngle:=" , "360deg"`
	`    "NumOfSegments:=" , "12"`
	`  ])`

SweepFacesAlongNormal

Sweep the specified face(s) along normal.

UI Access	**Modeler** > **Surface** > **Sweep Faces Along Normal**

<table>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;SelectionsArray&gt;</i></td><td>Array</td><td>Structured array. See: SelectionsArray.</td></tr>
<tr><td><i>&lt;parameters&gt;</i></td><td>Array</td><td>Structured array.<pre>
Array("NAME:Parameters",
   "NAME:SweepFaceAlongNormalToParameters",
   "FacesToDetach:=", <faceIDarray>,
   "LengthOfSweep:=", "<value><units>")
</pre></td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>None</td></tr>
</table>

<table>
<tr><td><b>Python Syntax</b></td><td>SweepFacesAlongNormal(&lt;<i>SelectionsArray</i>&gt; &lt;<i>parameters</i>&gt;)</td></tr>
<tr><td><b>Python Example</b></td><td><pre>
oEditor.SweepFacesAlongNormal(
["NAME:Selections",
         "Selections:=", "Rectangle1",
         "NewPartsModelFlag:=", "Model"],
["NAME:Parameters",
         "NAME:SweepFaceAlongNormalToParameters",
         "FacesToDetach:=", [183],
         "LengthOfSweep:=", "0.1mm"])
</pre></td></tr>
</table>

SweepFacesAlongNormalWithAttributes

Sweep a face along normal, and specify attributes of the new object.

<table>
<tr><th>UI Access</th><th>Modeler > Surface > Sweep Faces Along Normal</th></tr>
<tr><td>Parameters</td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><em>&lt;SelectionsArray&gt;</em></td><td>Array</td><td>Structured array. See: SelectionsArray.</td></tr>
<tr><td><em>&lt;parameters&gt;</em></td><td>Array</td><td><code>Array("NAME:Parameters",</code>
<code>"NAME:SweepFaceAlongNormalToParameters",</code>
<code>"FacesToDetach:=", <faceIDarray&gt;,</code>
<code>"LengthOfSweep:=", "<value><units&gt;")</code></td></tr>
<tr><td><em>&lt;AttributesArray&gt;</em></td><td>Array</td><td>Structured array. See: AttributesArray.</td></tr>
</table>
</td></tr>
<tr><td>Return Value</td><td>None</td></tr>
</table>

Python Syntax	SweepFacesAlongNormalWithAttributes(*<SelectionsArray>*, *<parameters>*, *<AttributesArray>*)

Python Example	`oEditor.SweepFacesAlongNormalWithAttributes(`
	`["NAME:Selections",`
	`"Selections:=", "Rectangle1",`
	`"NewPartsModelFlag:=", "Model"],`
	`["NAME:Parameters",`
	`"NAME:SweepFaceAlongNormalToParameters",`
	`"FacesToDetach:=", [183],`

	"LengthOfSweep:=", "0.1mm"], ["NAME:Attributes", "Name:=", "Box3", "Flags:=", "", "Color:=", "(143 175 143)", "Transparency:=", 0, "PartCoordinateSystem:=", "Global", "UDMId:=", "", "MaterialValue:=", "\"copper\"", "SurfaceMaterialValue:=", "\"\"", "SolveInside:=", False, "ShellElement:=", False, "ShellElementThickness:=", "0mm", "IsMaterialEditable:=", True, "UseMaterialAppearance:=", False, "IsLightweight:=", False])

UpdateComponentDefinition

Updates a 3D component's definition.

UI Access	**Draw** > **3D Component Library** > **Definitions**.

<table>
<tr><td><b>Parameters</b></td><td><table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;data&gt;</i></td><td>Array</td><td>Structured array.
Array("NAME:UpdateDefinitionData",
"DefinitionNames:=", <string>,
"Passwords:=", <array of strings&gt;)</td></tr>
</table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	UpdateComponentDefinition(<*data*>)
Python Example	oEditor.UpdateComponentDefinition(['NAME:UpdateDefinitionData', 'DefinitionNames:=', 'Connector, Magic_Tee', 'Passwords:=', ['','']])

Edit Menu Commands

Copy

DeletePolylinePoint

DuplicateAlongLine

DuplicateAroundAxis

DuplicateMirror

Mirror

Move

OffsetFaces

Paste

Rotate

Scale

Copy

Copies specified part(s) to the clipboard.

UI Access	N/A
Parameters	Name | Type | Description *<SelectionsArray>* | Array | Structured array. See: SelectionsArray.
Return Value	None.

Python Syntax	Copy(<*SelectionsArray*>)

Python Example	`oEditor.Copy([`
	`  "NAME:Selections",`
	`  "Selections:=", "Box1"])`

DeletePolylinePoint

Deletes either a start point or an end point from an existing polyline segment.

<table>
<tr><td><b>UI Access</b></td><td><b>Edit</b> &gt; <b>Delete</b> [<b>Start</b>/<b>End</b>] <b>Point</b></td></tr>
<tr><td><b>Parameters</b></td><td><table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;DeletePointArray&gt;</i></td><td>Array</td><td>Structured array.
<code>Array("NAME:Delete Point",</code>
<code>"Selections:=", <string "<PolylineName>:<PolylineAction>:<int&gt;"&gt;,</code>
<code>"Segment Index:=", <integer&gt;,</code>
<code>"At Start:=", <bool True for start point; False for end point&gt;)</code></td></tr>
</table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	DeletePolylinePoint (<*DeletePointArray*>)

Python Example	`oEditor.DeletePolylinePoint(["NAME:Delete Point",`
	`"Selections:=", "Polyline1:CreatePolyline:1",`
	`"Segment Index:=", 1,`
	`"At Start:=", True])`

DuplicateAlongLine

Duplicates specified parts along a line.

UI Access	**Edit** > **Duplicate** > **Along Line**.

	Name	Type	Description

Parameters	<*SelectionsArray*>	Array	Structured array.
			`Array("NAME:Selections",`
			`"Selections:=" , <string>,`
			`"NewPartsModelFlag:=" , <string>)`

	<*ParametersArray*>	Array	Structured array.
			`Array("NAME:DuplicateToAlongLineParameters",`
			`"CreateNewObjects:=" , <boolean>,`
			`"XComponent:=" , <string>,`
			`"YComponent:=" , <string>,`
			`"ZComponent:=" , <string>,`
			`"NumClones:=" , <string containing number greater than 1>)`

	<*OptionsArray*>	Array	Structured array.
			`Array("NAME:Options",`
			`"DuplicateAssignments:=", <boolean>)`

	<*CreateGroup*>	Array	Optional. Structured array. `Array("CreateGroupsForNewObjects:=", <boolean>)`
Return Value	None.		

Python Syntax	DuplicateAlongLine (<*SelectionsArray*>, <*ParametersArray*>, <*OptionsArray*>, <*CreateGroup*>)
Python Example	`oEditor.DuplicateAlongLine(`

```
["NAME:Selections",
  "Selections:="          , "Box1",
  "NewPartsModelFlag:="   , "Model"
],
["NAME:DuplicateToAlongLineParameters",
  "CreateNewObjects:="    , False,
  "XComponent:="          , "1mm",
  "YComponent:="          , "-0.7mm",
  "ZComponent:="          , "0mm",
  "NumClones:="           , "2"
],
["NAME:Options",
  "DuplicateAssignments:=", False
],
["CreateGroupsForNewObjects:=", False
])
```

DuplicateAroundAxis

Duplicates specified parts around an axis.

<table>
<tr><td><b>UI Access</b></td><td><b>Edit</b> &gt; <b>Duplicate</b> &gt; <b>Around Axis</b>.</td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td>&lt;<i>SelectionsArray</i>&gt;</td><td>Array</td><td>Structured array.
<code>Array("NAME:Selections",</code>
<code>"Selections:=" , <string&gt;,</code>
<code>"NewPartsModelFlag:=" , <string&gt;)</code></td></tr>
<tr><td>&lt;<i>ParametersArray</i>&gt;</td><td>Array</td><td>Structured array.
<code>Array("NAME:DuplicateAroundAxisParameters",</code>
<code>"CreateNewObjects:=" , <boolean&gt;,</code>
<code>"WhichAxis:=" , <string&gt;,</code>
<code>"AngleStr:=" , <string&gt;,</code>
<code>"NumClones:=" , <string containing number greater than 1&gt;)</code></td></tr>
<tr><td>&lt;<i>OptionsArray</i>&gt;</td><td>Array</td><td>Structured array.
<code>Array("NAME:Options",</code>
<code>"DuplicateAssignments:=", <boolean&gt;)</code></td></tr>
<tr><td>&lt;<i>CreateGroup</i>&gt;</td><td>Array</td><td>Optional. Structured array.
<code>Array("CreateGroupsForNewObjects:=", <boolean&gt;)</code></td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	DuplicateAroundAxis (<*SelectionsArray*>, <*ParametersArray*>, <*OptionsArray*>, <*CreateGroup*>)
Python Example	`oEditor.DuplicateAroundAxis(`

```
["NAME:Selections",
  "Selections:="          , "Box1",
  "NewPartsModelFlag:="   , "Model"
],
["NAME:DuplicateAroundAxisParameters",
  "CreateNewObjects:="    , True,
  "WhichAxis:="           , "Z",
  "AngleStr:="            , "90deg",
  "NumClones:="           , "2"
],
["NAME:Options",
  "DuplicateAssignments:=", False
],
["CreateGroupsForNewObjects:=", False
])
```

DuplicateMirror

Duplicates specified parts according to a mirror plane.

UI Access	**Edit** > **Duplicate** > **Mirror**.

<table>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td>&lt;<i>SelectionsArray</i>&gt;</td><td>Array</td><td>Structured array.
<code>Array("NAME:Selections",</code>
<code>"Selections:=" , <string&gt;,</code>
<code>"NewPartsModelFlag:=" , <string&gt;)</code></td></tr>
<tr><td>&lt;<i>ParametersArray</i>&gt;</td><td>Array</td><td>Structured array.
<code>Array("NAME:DuplicateToMirrorParameters",</code>
<code>"DuplicateMirrorBaseX:=", <string&gt;,</code>
<code>"DuplicateMirrorBaseY:=", <string&gt;,</code>
<code>"DuplicateMirrorNormalX:=", <string&gt;,</code>
<code>"DuplicateMirrorNormalY:=", <string&gt;,</code>
<code>"DuplicateMirrorNormalZ:=", <string&gt;)</code></td></tr>
<tr><td>&lt;<i>OptionsArray</i>&gt;</td><td>Array</td><td>Structured array.
<code>Array("NAME:Options",</code>
<code>"DuplicateAssignments:=", <boolean&gt;)</code></td></tr>
<tr><td>&lt;<i>CreateGroup</i>&gt;</td><td>Array</td><td>Optional. Structured array.
<code>Array("CreateGroupsForNewObjects:=", <boolean&gt;)</code></td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	DuplicateMirror (<*SelectionsArray*>, <*ParametersArray*>, <*OptionsArray*>, <*CreateGroup*>)

Python Example	`oEditor.DuplicateMirror(`
	`["NAME:Selections",`

```
  "Selections:="          , "Box1",
  "NewPartsModelFlag:="    , "Model"
],
["NAME:DuplicateToMirrorParameters",
  "DuplicateMirrorBaseX:=", "-0.4mm",
  "DuplicateMirrorBaseY:=", "-1.2mm",
  "DuplicateMirrorBaseZ:=", "0mm",
  "DuplicateMirrorNormalX:=", "0.124034734589208mm",
  "DuplicateMirrorNormalY:=", "0.992277876713668mm",
  "DuplicateMirrorNormalZ:=", "0mm"
],
["NAME:Options",
  "DuplicateAssignments:=", False
],
["CreateGroupsForNewObjects:=", False
])
```

Mirror

Mirrors specified part(s).

<table>
<tr><th><b>UI Access</b></th><th><b>Edit</b> > <b>Arrange</b> > <b>Mirror</b>.</th></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td><i><SelectionsArray></i></td><td>Array</td><td>Structured array. See: SelectionsArray.</td></tr>
<tr><td><i><MirrorParameters></i></td><td>Array</td><td>Structured array.
Array("NAME:MirrorParameters",
  "MirrorBaseX:=" , <string>,
  "MirrorBaseY:=" , <string>,
  "MirrorBaseZ:=" , <string>,
  "MirrorNormalX:=" , <string>,
  "MirrorNormalY:=" , <string>,
  "MirrorNormalZ:=" , <string>)</td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

<table>
<tr><th><b>Python Syntax</b></th><th>Mirror(<<i>SelectionsArray</i>>, <<i>MirrorParameters</i>>)</th></tr>
<tr><td><b>Python Example</b></td><td>oEditor.Mirror(
["NAME:Selections",
  "Selections:="   , "Box1_1",
  "NewPartsModelFlag:="   , "Model"
],
["NAME:MirrorParameters",
  "MirrorBaseX:="   , "-0.2mm",
  "MirrorBaseY:="   , "-1.2mm",</td></tr>
</table>

	`"MirrorBaseZ:="          , "0mm",`
	`"MirrorNormalX:="        , "-0.316227766016838mm",`
	`"MirrorNormalY:="        , "0.948683298050514mm",`
	`"MirrorNormalZ:="        , "0mm"`
	`])`

Move

Moves specified part(s).

UI Access	**Edit** > **Arrange** > **Move**.
Parameters	<table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;<i>SelectionsArray</i>&gt;</td><td>Array</td><td>Structured array. See: SelectionsArray.</td></tr><tr><td>&lt;<i>TranslateParameters</i>&gt;</td><td>Array</td><td>Structured array. <code>Array(["NAME:TranslateParameters",</code> <code>"TranslateVectorX:=" , <string&gt;,</code> <code>"TranslateVectorY:=" , <string&gt;,</code> <code>"TranslateVectorZ:=" , <string&gt;)</code></td></tr></table>
Return Value	None.

Python Syntax	Move(<*SelectionsArray*>, <*TranslateParameters*>)
Python Example	`oEditor.Move(`

```
["NAME:Selections",
  "Selections:="           , "Box1_1",
  "NewPartsModelFlag:="    , "Model"
],
["NAME:TranslateParameters",
  "TranslateVectorX:="     , "-0.5mm",
  "TranslateVectorY:="     , "0.1mm",
  "TranslateVectorZ:="     , "0mm"
])
```

OffsetFaces

Offsets the faces of selected part(s).

UI Access	**Edit** > **Arrange** > **Offset**.

Parameters

Name	Type	Description
<*SelectionsArray*>	Array	Structured array. See: SelectionsArray.

<*OffsetParameters*>	Array	Structured array.
		`Array("NAME:OffsetParameters",`
		`  "OffsetDistance:=" , <string>)`

Return Value	None.

Python Syntax	OffsetFaces (<*SelectionsArray*>, <*OffsetParameters*>)

Python Example	`oEditor.OffsetFaces(`
	`["NAME:Selections",`
	`  "Selections:=", "Box1_1",`
	`  "NewPartsModelFlag:=", "Model"`
	`],`
	`["NAME:OffsetParameters",`
	`  "OffsetDistance:=", "16mm"`
	`])`

Paste (Model Editor)

Pastes previously copied object(s). See: Copy.

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

Python Syntax	Paste()

Python Example	`oEditor.Copy(`
	`["NAME:Selections",`
	`  "Selections:=" , "Box1_2"`
	`])`

	`oEditor.Paste()`

Rotate

Rotates specified object(s).

<table>
<tr><td><b>UI Access</b></td><td><b>Edit</b> &gt; <b>Arrange</b> &gt; <b>Rotate</b>.</td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td>&lt;<i>SelectionsArray</i>&gt;</td><td>Array</td><td>Structured array. See: SelectionsArray.</td></tr>
<tr><td>&lt;<i>RotateParameters</i>&gt;</td><td>Array</td><td>Structured array.
Array("NAME:RotateParameters",
  "RotateAxis:=" , <string "X", "Y", or "Z">,
  "RotateAngle:=" , <string&gt;)</td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	Rotate(<*SelectionsArray*>, <*RotateParameters*>)

Python Example	`oEditor.Rotate(`
	`["NAME:Selections",`
	`  "Selections:=", "Box1_1",`
	`  "NewPartsModelFlag:=", "Model"`
	`],`
	`["NAME:RotateParameters",`
	`  "RotateAxis:=", "Z",`

	"RotateAngle:=", "90deg"])

Scale

Scales specified object(s).

<table>
<tr><td><b>UI Access</b></td><td><b>Edit</b> &gt; <b>Scale</b>.</td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td>&lt;<i>SelectionsArray</i>&gt;</td><td>Array</td><td>Structured array. See: SelectionsArray.</td></tr>
<tr><td>&lt;<i>ScaleParameters</i>&gt;</td><td>Array</td><td>Structured array.
Array("NAME:ScaleParameters",
"ScaleX:=" , <string containing scale factor>,
"ScaleY:=" , <string containing scale factor>,
"ScaleZ:=" , <string containing scale factor&gt;)</td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	Scale (<*SelectionsArray*>, <*ScaleParameters*>)
Python Example	oEditor.Scale(["NAME:Selections", "Selections:=", "Box1", "NewPartsModelFlag:=", "Model"

	],
	["NAME:ScaleParameters",
	  "ScaleX:=", "2",
	  "ScaleY:=", "2",
	  "ScaleZ:=", "2"
	])

Modeler Menu Commands

AssignMaterial

Chamfer

Connect

CoverLines

CoverSurfaces

CreateEntityList

CreateFaceCS

CreateGroup

CreateObjectCS

CreateObjectFromEdges

CreateObjectFromFaces

CreateRelativeCS

DeleteEmptyGroups

AlignFaces

Aligns the adjacent selected faces of imported objects which have only one operation in their History Tree.

<table>
<tr><th>UI Access</th><th>Modeler > Model Preparation > Align Faces</th></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;argFaceList&gt;</i></td><td>Array</td><td>["NAME:<SpecifiedName>",
"BaseFaces:=" , <FaceNumberArray>,
"SnapFaces:=" , <FaceNumberArray&gt;]</td></tr>
<tr><td><i>&lt;FaceNumberArray&gt;</i></td><td>Array</td><td>Array of number represents specified faces.</td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

<table>
<tr><th>Python Syntax</th><th>AlignFaces(&lt;<i>argFaceList</i>&gt;)</th></tr>
<tr><td><b>Python Example</b></td><td><pre>oEditor.AlignFaces([

        "NAME:Entity List",

        "BaseFaces:="   , [9],

        "SnapFaces:="   , [18]

        ])</pre></td></tr>
</table>

AssignMaterial

Assigns a material to specified object(s).

<table>
<tr><th>UI Access</th><th>Modeler > Assign Material.</th></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;SelectionsArray&gt;</i></td><td>Array</td><td>Structured array. See: SelectionsArray.</td></tr>
<tr><td><i>&lt;AttributesArray&gt;</i></td><td>Array</td><td>Structured array. See: AttributesArray.</td></tr>
</table>
</td></tr>
</table>

			This script supports the following attributes: • MaterialValue • SolveInside • ShellElement • ShellElementThickness • IsMaterialEditable • UseMaterialAppearance • IsLightweight
Return Value	None.		

Python Syntax	AssignMaterial(<*SelectionsArray*>, <*AttributesArray*>)

Python Example	oEditor.AssignMaterial(["NAME:Selections", "AllowRegionDependentPartSelectionForPMLCreation:=", True, "AllowRegionSelectionForPMLCreation:=", True, "Selections:=" , "Box1"], ["NAME:Attributes", "MaterialValue:=" , "diamond", "SolveInside:=" , False, "ShellElement:=" , False, "ShellElementThickness:=" , "nan", "IsMaterialEditable:=" , True, "UseMaterialAppearance:=" , False, "IsLightweight:=" , False])

Chamfer

Creates a chamfer.

UI Access	**Modeler** > **Chamfer**.

<table>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;SelectionsArray&gt;</i></td><td>Array</td><td>Structured array. See: SelectionsArray.</td></tr>
<tr><td><i>&lt;Parameters&gt;</i></td><td>Array</td><td>Structured array.
Array("NAME:Parameters",
  Array("NAME:ChamferParameters",
  "Edges:=" , <array containing integer>,
  "Vertices:=" , <array>,
  "LeftDistance:=" , <string>,
  "RightDistance:=" , <string>,
  "ChamferType:=" , <string "Symmetric", "Left Distance-Angle", "Right Distance-Angle", or "Left Distance-Right Distance">)
)</td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

<table>
<tr><td><b>Python Syntax</b></td><td>Chamfer(&lt;<i>SelectionsArray</i>&gt;, &lt;<i>Parameters</i>&gt;)</td></tr>
<tr><td><b>Python Example</b></td><td>oEditor.Chamfer(
["NAME:Selections",
"Selections:="   , "Box2",
"NewPartsModelFlag:="   , "Model"
],
["NAME:Parameters",</td></tr>
</table>

	["NAME:ChamferParameters", "Edges:=" , [42], "Vertices:=" , [], "LeftDistance:=" , "0.1mm", "RightDistance:=" , "0.1mm", "ChamferType:=" , "Symmetric"]])

CleanUpModel

Cleans up history tree operations.

UI Access	**Modeler** > **Cleanup Model History**.
Parameters	None.
Return Value	None.

Python Syntax	CleanUpModel()
Python Example	`oEditor.CleanUpModel()`

Connect

Connects two or more 1D polyline objects or 2D sheet objects.

UI Access	**Modeler** > **Surface** > **Connect**.
Parameters	Name | Type | Description *<SelectionsArray>* | Array | Structured array. See: SelectionsArray.
Return Value	None.

Python Syntax	Connect(<*SelectionsArray*>)

Python Example	`oEditor.Connect(`
	`["NAME:Selections",`
	`"Selections:=", "Polyline2,Polyline1"])`

CoverLines

Covers two or more 1D objects to form a sheet.

UI Access	**Modeler** > **Surface** > **Cover Lines**.
Parameters	Name | Type | Description *<SelectionsArray>* | Array | Structured array. See: SelectionsArray.
Return Value	None.

Python Syntax	CoverLines(<*SelectionsArray*>)

Python Example	`oEditor.CoverLines([`
	`  "NAME:Selections",`
	`  "Selections:=", "Polyline3,Polyline4",`

	`"NewPartsModelFlag:=", "Model"])`

CoverSurfaces

Covers two or more faces to form a solid object.

<table>
<tr><th>UI Access</th><th><b>Modeler</b> &gt; <b>Surface</b> &gt; <b>Cover Faces</b>.</th></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;<i>SelectionsArray</i>&gt;</td><td>Array</td><td>Structured array. See: SelectionsArray.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	CoverSurfaces (<*SelectionsArray*>)

Python Example	`oEditor.CoverSurfaces(`
	`["NAME:Selections",`
	`  "Selections:=", "Obj1_Face1,Obj2_Face2",`
	`  "NewPartsModelFlag:=", "Model"`
	`])`

CreateEntityList

Creates a list of entities containing objects or faces (not both).

> **Warning:** As of the Ansys Electronics Desktop25r2 release, this command has been deprecated. It will be removed in a future release. Please use the CreateNamedSelection command instead.

UI Access	**Modeler** > **Named Selection** > **Create** > [**Object** / **Face**] **Selection**

Parameters	<table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>*&lt;Parameters&gt;*</td><td>Array</td><td>Structured array.
	`Array("NAME:GeometryEntityListParameters",`
	`  "EntityType:=" , <string "Object" or "Face">,`
	`  "EntityList:=" , <string of object names or IDs>)`
	See GetObjectIDByName for returning object IDs.</td></tr><tr><td>*&lt;AttributesArray&gt;*</td><td>Array</td><td>Structured array. See: AttributesArray. CreateEntityList takes only the "Name" parameter.</td></tr></table>

Return Value	None.

Python Syntax	CreateEntityList(<*Parameters*>,<*AttributesArray*>)

Python Example	`oEditor.CreateEntityList(`
	`["NAME:GeometryEntityListParameters",`
	`  "EntityType:=", "Object",`
	`  "EntityList:=", "Bondwire1,Bondwire2"`
	`],`
	`["NAME:Attributes",`
	`  "Name:=", "Objectlist1"`

	`])`

CreateFaceCS

Creates a Face Coordinate System from a selected face.

<table>
<tr><th>UI Access</th><th><b>Modeler</b> &gt; <b>Coordinate System</b> &gt; <b>Create</b> &gt; <b>FaceCS</b>.</th></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>&lt;<i>Parameters</i>&gt;</td><td>Array</td><td>Structured array.
Array("NAME:FaceCSParameters",
  <OriginArray>,
  "MoveToEnd:=", <boolean>,
  "FaceID:=", <integer>,
  <AxisPosnArray>,
  "WhichAxis:=" , <string "X", "Y", or "Z">,
  "ZRotationAngle:=" , <string>,
  "XOffset:=" , <string>,
  "YOffset:=" , <string>,
  "AutoAxis:=" , <boolean&gt;)</td></tr>
<tr><td>&lt;<i>OriginArray</i>&gt;</td><td>Array</td><td>Structured array.
Array("NAME:Origin",
  "IsAttachedToEntity:=" , <boolean&gt;,</td></tr>
</table>
</td></tr>
</table>

		"EntityID:=" , <integer>, "FacetedBodyTriangleIndex:=", <integer>, "TriangleVertexIndex:=" , <integer>, "PositionType:=" , <string "FaceCenter", "EdgeCenter", "OnVertex", "OnEdge", or "OnFace">, "UParam:=" , <float between 0 and 1 representing the relative position of the point on the edge or face>, "VParam:=" , <float between 0 and 1 representing the relative position of the point on the edge or face>, "XPosition:=" , <string>, "YPosition:=" , <string>, "ZPosition:=" , <string>) IsAttachedToEntity specifies whether the point is anchored to a vertex, edge, or face. If True, provide UParam and VParam. If False, provide XPosition, YPosition, and ZPosition to provide fixed position. Pass "0" for unused parameters.
<AxisPosnArray>	Array	Structured array. Array("NAME:AxisPosn", "IsAttachedToEntity:=" , <boolean>, "EntityID:=" , <integer>, "FacetedBodyTriangleIndex:=", <integer>, "TriangleVertexIndex:=" , <integer>, "PositionType:=" , <string "FaceCenter",

		`"EdgeCenter", "OnVertex", "OnEdge", or "OnFace">,`
		`"UParam:=" , <float>,`
		`"VParam:=" , <float>,`
		`"XPosition:=" , <string>,`
		`"YPosition:=" , <string>,`
		`"ZPosition:=" , <string>)`

<AttributesArray>	Array	Structured array. See: AttributesArray.

Return Value	None.

Python Syntax	CreateFaceCS(<*Parameters*>,<*AttributesArray*>)

Python Example	`oEditor.CreateFaceCS(`
	`["NAME:FaceCSParameters",`
	`  ["NAME:Origin",`
	`    "IsAttachedToEntity:="  , True,`
	`    "EntityID:="            , 46,`
	`    "FacetedBodyTriangleIndex:=", -1,`
	`    "TriangleVertexIndex:=" , -1,`
	`    "PositionType:="        , "FaceCenter",`
	`    "UParam:="              , 0,`
	`    "VParam:="              , 0,`

```
  "XPosition:="           , "0",
  "YPosition:="           , "0",
  "ZPosition:="           , "0"
],
"MoveToEnd:="           , False,
"FaceID:="              , 46,
["NAME:AxisPosn",
  "IsAttachedToEntity:="  , True,
  "EntityID:="            , 46,
  "FacetedBodyTriangleIndex:=", -1,
  "TriangleVertexIndex:=" , -1,
  "PositionType:="        , "OnFace",
  "UParam:="              , 0.487129134674319,
  "VParam:="              , 0.308528523557527,
  "XPosition:="           , "1292.27748080459mm",
  "YPosition:="           , "-814.882885865484mm",
  "ZPosition:="           , "0mm"
],
"WhichAxis:="           , "X",
"ZRotationAngle:="      , "0deg",
"XOffset:="             , "0mm",
  "YOffset:="              , "0mm",
  "AutoAxis:="             , False
],
["NAME:Attributes",
  "Name:="                 , "FaceCS1",
  "PartName:="             , "Rectangle1"
])
```

CreateGroup

Creates a group from objects specified in the history tree.

<table>
<tr><td><b>UI Access</b></td><td><b>Modeler</b> &gt; <b>Group</b> &gt; <b>Create</b>.</td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;Parameters&gt;</i></td><td>Array</td><td>Structured array.
Array("NAME:GroupParameter",
  "ParentGroupID:=" , <string>,
  "Parts:=" , <string>,
  "SubmodelInstances:=" , <string>,
  "Groups:=" , <string&gt;)</td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

<table>
<tr><th>Python Syntax</th><th>CreateGroup(&lt;<em>Parameters</em>&gt;)</th></tr>
<tr><td><strong>Python Example</strong></td><td><code>oEditor.CreateGroup(</code>
<code>["NAME:GroupParameter",</code>
<code> "ParentGroupID:="   , "Model",</code>
<code> "Parts:="   , "Box1,Box2,Box3",</code>
<code> "SubmodelInstances:="   , "",</code>
<code> "Groups:="   , ""</code>
<code>])</code></td></tr>
</table>

CreateNamedSelection

Creates a list of entities containing objects or faces (not both).

<table>
<tr><th>UI Access</th><th><strong>Modeler</strong> &gt; <strong>Named Selection</strong> &gt; <strong>Create</strong> &gt; [<strong>Object</strong> / <strong>Face</strong>] <strong>Selection</strong></th></tr>
<tr><td><strong>Parameters</strong></td><td>
<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>&lt;<em>Parameters</em>&gt;</td><td>Array</td><td>Structured array.
<code>Array("NAME:NamedSelectionParameters",</code>
<code> "Type:=" , <string "Object" or "Face"&gt;,</code>
<code> "Selection:=" , <string of object names or IDs&gt;)</code>
See GetObjectIDByName for returning object IDs.</td></tr>
<tr><td>&lt;<em>AttributesArray</em>&gt;</td><td>Array</td><td>Structured array. See AttributesArray. CreateNamedSelection takes only the "Name" parameter and UDM ID.</td></tr>
</table>
</td></tr>
<tr><td><strong>Return Value</strong></td><td>None.</td></tr>
</table>

Python Syntax	CreateNamedSelection(<*Parameters*>,<*AttributesArray*>)

Python Example	`namedSelectionName = oEditor.CreateNamedSelection(`
	`  [`
	`    "NAME:NamedSelectionParameters",`
	`    "Type:=" , "Face",`
	`    "Selection:=" , [7,9]`
	`  ],`
	`  [`
	`    "NAME:Attributes",`
	`    "Name:=" , "FaceSelection1",`
	`    "UDM ID:=" , -1`
	`  ])`

CreateObjectCS

Creates an object coordinate system from a selected object.

UI Access	**Modeler** > **Coordinate System** > **Create** > **Object** > [**Offset** / **Rotated** / **Both**].

Parameters	Name / Type / Description:
	<*Parameters*> / Array / Structured array.
	`Array("NAME:ObjectCSParameters",`
	`  <OriginArray>`
	`  "MoveToEnd:=" , <boolean>,`
	`  "ReverseXAxis:=" , <boolean>,`
	`  "ReverseYAxis:=" , <boolean>,`

		<xAxisArray / xAxisPosArray> <yAxisArray / yAxisPosArray>) **Note:** xAxisArray and xAxisPosArray differ. Use xAxisArray for absolute position and xAxisPosArray for relative position. Do the same for yAxisArray and yAxisPosArray.
<OriginArray>	Array	Structured array. Array("NAME:Origin", "IsAttachedToEntity:=" , <boolean>, "EntityID:=" , <integer>, "FacetedBodyTriangleIndex:=", <integer>, "TriangleVertexIndex:=" , <integer>, "PositionType:=" , <string "OnVertex", "EdgeCenter", "FaceCenter", "OnEdge", or "AbsolutePosition">, "UParam:=" , <float between 0 and 1 representing the relative position of the point on the edge or face>, "VParam:=" , <float between 0 and 1 representing the relative position of the point on the edge or face>, "XPosition:=" , <string>, "YPosition:=" , <string>, "ZPosition:=" , <string>) IsAttachedToEntity specifies whether the point is anchored. If True, provide UParam and VParam. If False, provide XPosition, YPosition, and

		ZPosition to provide fixed position. Pass "0" for unused parameters.

<xAxisArray>	Array	Structured array for absolute position:
		`Array("NAME:xAxis",`
		`"DirectionType:=" , "AbsoluteDirection",`
		`"EdgeID:=" , <integer>,`
		`"FaceID:=" , <integer>,`
		`"xDirection:=" , <string>,`
		`"yDirection:=" , <string>,`
		`"zDirection:=" , <string>,`
		`"UParam:=" , <float>,`
		`"VParam:=" , <float>)`

<xAxisPosArray>	Array	Structured array for relative position:
		`Array("NAME:xAxisPos",`
		`"IsAttachedToEntity:=" , <boolean>,`
		`"EntityID:=" , <integer>,`
		`"FacetedBodyTriangleIndex:=", <integer>,`
		`"TriangleVertexIndex:=" , <integer>,`
		`"PositionType:=" , <string "OnVertex", "EdgeCenter", "FaceCenter", or "OnEdge">,`
		`"UParam:=" , <float>,`
		`"VParam:=" , <float>,`

		"XPosition:=" , <string>, "YPosition:=" , <string>, "ZPosition:=" , <string>)
<*yAxisArray*>	Array	Structured array for absolute position: Array("NAME:yAxis", "DirectionType:=" , "AbsoluteDirection", "EdgeID:=" , <integer>, "FaceID:=" , <integer>, "xDirection:=" , <string>, "yDirection:=" , <string>, "zDirection:=" , <string>, "UParam:=" , <float>, "VParam:=" , <float>)
<*yAxisPosArray*>	Array	Structured array for relative position: Array("NAME:yAxisPos", "IsAttachedToEntity:=" , <boolean>, "EntityID:=" , <integer>, "FacetedBodyTriangleIndex:=", <integer>, "TriangleVertexIndex:=" , <integer>, "PositionType:=" , <string "OnVertex", "EdgeCenter", "FaceCenter", or "OnEdge">, "UParam:=" , <float>,

			"VParam:=" , <float>, "XPosition:=" , <string>, "YPosition:=" , <string>, "ZPosition:=" , <string>)
	<AttributesArray>	Array	Structured array. See: AttributesArray.
Return Value	None.		

Python Syntax	CreateObjectCS(<*Parameters*>,<*AttributesArray*>)

Python Example	oEditor.CreateObjectCS(
	["NAME:ObjectCSParameters",
	  ["NAME:Origin",
	    "IsAttachedToEntity:=" , True,
	    "EntityID:=" , 59,
	    "FacetedBodyTriangleIndex:=", -1,
	    "TriangleVertexIndex:=" , -1,
	    "PositionType:=" , "OnVertex",
	    "UParam:=" , 0,
	    "VParam:=" , 0,
	    "XPosition:=" , "0",
	    "YPosition:=" , "0",

```
  "ZPosition:="           , "0"
],
"MoveToEnd:="           , False,
"ReverseXAxis:="        , False,
"ReverseYAxis:="        , False,
["NAME:xAxis",
  "DirectionType:="       , "AbsoluteDirection",
  "EdgeID:="              , -1,
  "FaceID:="              , -1,
  "xDirection:="          , "1",
  "yDirection:="          , "0",
  "zDirection:="          , "0",
  "UParam:="              , 0,
  "VParam:="              , 0
],
["NAME:yAxis",
  "DirectionType:="       , "AbsoluteDirection",
  "EdgeID:="              , -1,
  "FaceID:="              , -1,
  "xDirection:="          , "0",
  "yDirection:="          , "1",
    "zDirection:="         , "0",
    "UParam:="             , 0,
    "VParam:="             , 0
  ]
],
["NAME:Attributes",
  "Name:="               , "ObjectCS1",
  "PartName:="           , "Box2"
])
```

CreateObjectFromEdges

Creates an object from the specified object edge.

<table>
<tr><th>UI Access</th><th>Modeler > Edge > Create Object From Edge</th></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;SelectionsArray&gt;</i></td><td>Array</td><td>Structured array. See: SelectionsArray.</td></tr>
<tr><td><i>&lt;ParametersArray&gt;</i></td><td>Array</td><td>Structured array.
Array("NAME:Parameters",
  Array("NAME:BodyFromEdgeToParameters",
  "Edges:=" , <array containing integer edges>
)</td></tr>
</table>
</td></tr>
</table>

			)
	<*CreateGroupsForNewObjects*>	Array	Structured array. `Array("CreateGroupsForNewObjects:=", <boolean True to create groups for new objects; else False>)`
Return Value	None.		

<table>
<tr><td><b>Python Syntax</b></td><td>CreateObjectFromEdges(&lt;<i>SelectionsArray</i>&gt;, &lt;<i>ParametersArray</i>&gt;, &lt;<i>CreateGroupsForNewObjects</i>&gt;)</td></tr>
<tr><td><b>Python Example</b></td><td><pre>
oEditor.CreateObjectFromEdges(
["NAME:Selections",
"Selections:=", "Box2",
"NewPartsModelFlag:=", "Model"
],
["NAME:Parameters",
        ["NAME:BodyFromEdgeToParameters",
        "Edges:=", [41]]
],
["CreateGroupsForNewObjects:=", False
])
</pre></td></tr>
</table>

CreateObjectFromFace

Creates 2D objects from specified face(s).

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td>&lt;<i>SelectionsArray</i>&gt;</td><td>Array</td><td>Structured array. See: SelectionsArray.</td></tr>
<tr><td>&lt;<i>Parameters</i>&gt;</td><td>Array</td><td>Structured array.
Array("NAME:Parameters",
  <BodyFromFaceToParameters&gt;)</td></tr>
<tr><td>&lt;<i>CreateGroupsForNewObjects</i>&gt;</td><td>Array</td><td>Optional. Structured array.
Array("CreateGroupsForNewObjects:=", <boolean&gt;)</td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	CreateObjectFromFace (<*SelectionsArray*>,<*Parameters*>, <*CreateGroupsForNewObjects*>)

Python Example

```
oEditor.CreateObjectFromFace(
["NAME:Selections",
"Selections:="          , "Box3",
"NewPartsModelFlag:="   , "Model"
],
["NAME:Parameters",
  ["NAME:BodyFromFaceToParameters",
```

	"FacesToDetach:=" , [68]]], ["CreateGroupsForNewObjects:=", False])

CreateObjectFromFaces

Creates 2D objects from specified face(s).

<table>
<tr><th>UI Access</th><th>Modeler > Surface > Create Object from Face</th></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;SelectionsArray&gt;</i></td><td>Array</td><td>Structured array. See: SelectionsArray.</td></tr>
<tr><td><i>&lt;Parameters&gt;</i></td><td>Array</td><td>Structured array.
Array("NAME:Parameters",
<BodyFromFaceToParameters&gt;)</td></tr>
<tr><td><i>&lt;CreateGroupsForNewObjects&gt;</i></td><td>Array</td><td>Structured array.
Array("CreateGroupsForNewObjects:=", <boolean&gt;)</td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	CreateObjectFromFaces (<*SelectionsArray*>,<*Parameters*>, <*CreateGroupsForNewObjects*>)
Python Example	oEditor.CreateObjectFromFaces(["NAME:Selections",

	"Selections:=" , "Box3", "NewPartsModelFlag:=" , "Model"], ["NAME:Parameters", ["NAME:BodyFromFaceToParameters", "FacesToDetach:=" , [68]]], ["CreateGroupsForNewObjects:=", False])

CreateRelativeCS

Creates a Relative Coordinate System.

<table>
<tr><th>UI Access</th><th>Modeler > Coordinate System > Create > Relative CS > [Offset / Rotated / Both].</th></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td><i>&lt;Parameters&gt;</i></td><td>Array</td><td>Structured array.
Array("NAME:RelativeCSParameters",
  "Mode:=" , "Axis/Position",
  "OriginX:=" , <string>,
  "OriginY:=" , <string&gt;,</td></tr>
</table>
</td></tr>
</table>

			"OriginZ:=" , <string>, "XAxisXvec:=" , <string>, "XAxisYvec:=" , <string>, "XAxisZvec:=" , <string>, "YAxisXvec:=" , <string>, "YAxisYvec:=" , <string>, "YAxisZvec:=" , <string>)
	<AttributesArray>	Array	Structured array. See: AttributesArray. CreateRelativeCS supports only the "Name" parameter.
Return Value	None.		

Python Syntax	CreateRelativeCS(*<Parameters>*,*<AttributesArray>*)

Python Example

```
oEditor.CreateRelativeCS(
["NAME:RelativeCSParameters",
  "Mode:="                , "Axis/Position",
  "OriginX:="             , "0.62mm",
  "OriginY:="             , "-0.7mm",
  "OriginZ:="             , "0mm",
  "XAxisXvec:="           , "1mm",
  "XAxisYvec:="           , "0mm",
  "XAxisZvec:="           , "0mm",
  "YAxisXvec:="           , "0mm",
```

	`"YAxisYvec:="            , "1mm",`
	`"YAxisZvec:="            , "0mm"`
	`],`
	`["NAME:Attributes",`
	`"Name:="                 , "RelativeCS1"`
	`])`

DeleteEmptyGroups

Deletes group(s) from the history tree.

<table>
<tr><td><b>UI Access</b></td><td><b>Modeler</b> > <b>Group</b> > <b>Delete Empty</b>.</td></tr>
<tr><td><b>Parameters</b></td><td><table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;Parameters&gt;</i></td><td>Array</td><td>Structured array.
<code>Array("Groups:=" , <array of string group IDs&gt;)</code></td></tr>
</table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	DeleteEmptyGroups(<*Parameters*>)

Python Example	`oEditor.DeleteEmptyGroups([`
	`"Groups:=", ["Group1","Group2","Group3"]`
	`])`

DeleteLastOperation

Deletes the last operation performed on the specified object(s).

<table>
<tr><td><b>UI Access</b></td><td><b>Modeler</b> &gt; <b>Delete Last Operation</b>.</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><i>&lt;SelectionsArray&gt;</i></td><td>Array</td><td>Structured array. See: SelectionsArray.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

<table>
<tr><td><b>Python Syntax</b></td><td>DeleteLastOperation(&lt;<i>SelectionsArray</i>&gt;)</td></tr>
<tr><td><b>Python Example</b></td><td><pre>oEditor.DeleteLastOperation([
  "NAME:Selections",
  "Selections:=", "Box3",
  "NewPartsModelFlag:=", "Model"
])</pre></td></tr>
</table>

DeleteOperation

Deletes specified operation performed on a selected object.

<table>
<tr><td><b>UI Access</b></td><td>Select an operation in the project tree, then press <b>Delete</b> on the keyboard.</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><i>&lt;ParamsArray&gt;</i></td><td>Array</td><td>Structured array.<pre>Array("NAME:Parameters",
  Array("NAME:PartOperations",</pre></td></tr></table></td></tr>
</table>

			Array("NAME:<PartName>", "OperationIndices:=", <array of operation indices>)), Array("NAME:UDMOperations"))
Return Value	None.		

Python Syntax	DeleteOperation(<*ParamsArray*>)

Python Example	oEditor.DeleteOperation(
	[
	  "NAME:Parameters",
	  [
	    "NAME:PartOperations",
	    [
	    "NAME:Coil_0",
	    "OperationIndices:=", [1]
	    ]
	  ],
	  ["NAME:UDMOperations"]
	])

DetachEdges

Detaches the specified edge(s) from an object.

<table>
<tr><th>UI Access</th><th><b>Modeler</b> &gt; <b>Edge</b> &gt; <b>Detach Edges</b>.</th></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;SelectionsArray&gt;</i></td><td>Array</td><td>Structured array. See: SelectionsArray.</td></tr>
<tr><td><i>&lt;Parameters&gt;</i></td><td>Array</td><td>Structured array.
<code>Array("NAME:Parameters",</code>
<code> &lt;DetachEdgesArray&gt;)</code></td></tr>
<tr><td><i>&lt;DetachEdgesArray&gt;</i></td><td>Array</td><td>Structured array.
<code>Array("NAME:DetachEdgesToParameters",</code>
<code> "EdgesToDetach:=" , <array containing integer edge IDs&gt;)</code></td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	DetachEdges(<*SelectionsArray*>, <*Parameters*>)

Python Example	`oEditor.DetachEdges(`
	`["NAME:Selections",`
	`"Selections:=", "Rectangle1",`
	`"NewPartsModelFlag:=", "Model"`
	`],`
	`["NAME:Parameters",`
	`   ["NAME:DetachEdgesToParameters",`

	"EdgesToDetach:=", [18,17]]])

DetachFaces

Detaches the specified face(s) from an object.

<table>
<tr><td><b>UI Access</b></td><td><b>Modeler</b> &gt; <b>Surface</b> &gt; <b>Detach Faces</b>.</td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td>&lt;<i>SelectionsArray</i>&gt;</td><td>Array</td><td>Structured array. See: SelectionsArray.</td></tr>
<tr><td>&lt;<i>Parameters</i>&gt;</td><td>Array</td><td>Structured array.
Array("NAME:Parameters",
<DetachFacesArray&gt;)</td></tr>
<tr><td>&lt;<i>DetachFacesArray</i>&gt;</td><td>Array</td><td>Structured array.
Array("NAME:DetachFacesToParameters",
"FacesToDetach:=" , <array containing integer face IDs&gt;)</td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	DetachFaces(<*SelectionsArray*>, <*Parameters*>)
Python Example	oEditor.DetachFaces(["NAME:Selections",

```
"Selections:=", "Box3",
"NewPartsModelFlag:=", "Model"
],
["NAME:Parameters",
  ["NAME:DetachFacesToParameters",
  "FacesToDetach:=", [68,67]
  ]
])
```

EditEntityList

Modifies an entity list.

> **Warning:** As of the Ansys Electronics Desktop25r2 release, this command has been deprecated. It will be removed in a future release. Please use the EditNamedSelection command instead.

<table>
<tr><td><b>UI Access</b></td><td><b>Modeler</b> &gt; <b>Named Selection</b> &gt; <b>Reassign</b>.</td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;SelectionsArray&gt;</i></td><td>Array</td><td>Structured array. See: SelectionsArray.</td></tr>
<tr><td><i>&lt;Parameters&gt;</i></td><td>Array</td><td>Structured array.
<code>Array("NAME:GeometryEntityListParameters",</code>
<code>"EntityType:=" , <string "Object" or "Face"&gt;,</code>
<code>"EntityList:=" , <string list&gt;</code></td></tr>
</table>
</td></tr>
</table>

<table>
<tr><td></td><td><table><tr><td></td><td></td><td>)</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

<table>
<tr><td><b>Python Syntax</b></td><td>EditEntityList(&lt;<i>SelectionsArray</i>&gt;, &lt;<i>Parameters</i>&gt;)</td></tr>
<tr><td><b>Python Example</b></td><td><pre>
oEditor.EditEntityList(
   ["NAME:Selections",
   "Selections:="   , "Objectlist1"
   ],
   ["NAME:GeometryEntityListParameters",
   "EntityType:="   , "Object",
   "EntityList:="   , "Box1, Box2, Box3"
   ])
</pre></td></tr>
</table>

EditFaceCS

Recreates an existing face coordinate system. See: CreateFaceCS.

<table>
<tr><td><b>UI Access</b></td><td><b>Modeler</b> &gt; <b>Coordinate System</b> &gt; <b>Edit</b>.</td></tr>
<tr><td><b>Parameters</b></td><td><table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td>&lt;<i>Parameters</i>&gt;</td><td>Array</td><td>Structured array.<pre>
Array("NAME:FaceCSParameters",
   <OriginArray>,
</pre></td></tr>
</table></td></tr>
</table>

		"MoveToEnd:=", <boolean>, "FaceID:=", <integer>, <AxisPosnArray>, "WhichAxis:=" , <string "X", "Y", or "Z">, "ZRotationAngle:=" , <string>, "XOffset:=" , <string>, "YOffset:=" , <string>, "AutoAxis:=" , <boolean>)
<OriginArray>	Array	Structured array. Array("NAME:Origin", "IsAttachedToEntity:=" , <boolean>, "EntityID:=" , <integer>, "FacetedBodyTriangleIndex:=", <integer>, "TriangleVertexIndex:=" , <integer>, "PositionType:=" , <string "FaceCenter", "EdgeCenter", "OnVertex", "OnEdge", or "OnFace">, "UParam:=" , <float between 0 and 1 representing the relative position of the point on the edge or face>, "VParam:=" , <float between 0 and 1 representing the relative position of the point on the edge or face>, "XPosition:=" , <string>,

		"YPosition:=" , <string>, "ZPosition:=" , <string>) IsAttachedToEntity specifies whether the point is anchored to a vertex, edge, or face. If True, provide UParam and VParam. If False, provide XPosition, YPosition, and ZPosition to provide fixed position. Pass "0" for unused parameters.
<AxisPosnArray>	Array	Structured array. Array("NAME:AxisPosn", "IsAttachedToEntity:=" , <boolean>, "EntityID:=" , <integer>, "FacetedBodyTriangleIndex:=", <integer>, "TriangleVertexIndex:=" , <integer>, "PositionType:=" , <string "FaceCenter", "EdgeCenter", "OnVertex", "OnEdge", or "OnFace">, "UParam:=" , <float>, "VParam:=" , <float>, "XPosition:=" , <string>, "YPosition:=" , <string>, "ZPosition:=" , <string>)
<AttributesArray>	Array	Structured array. See: AttributesArray. Use to select the coordinate system to edit.

Return Value	None.

Python Syntax	EditFaceCS (<*Parameters*>, <*AttributesArray*>)

Python Example

```
oEditor.EditFaceCS(
["NAME:FaceCSParameters",
  ["NAME:Origin",
    "IsAttachedToEntity:="  , True,
    "EntityID:="            , 12,
    "FacetedBodyTriangleIndex:=", -1,
    "TriangleVertexIndex:=" , -1,
    "PositionType:="        , "FaceCenter",
    "UParam:="              , 0,
    "VParam:="              , 0,
    "XPosition:="           , "0",
    "YPosition:="           , "0",
    "ZPosition:="           , "0"
  ],
  "MoveToEnd:="           , False,
  "FaceID:="              , 12,
  ["NAME:AxisPosn",
    "IsAttachedToEntity:="  , True,
    "EntityID:="            , 12,
    "FacetedBodyTriangleIndex:=", -1,
    "TriangleVertexIndex:=" , -1,
    "PositionType:=" , "OnFace",
    "UParam:="      , 0.62951717774066,
    "VParam:="      , 0.226514925559344,
    "XPosition:="   , "1200mm",
    "YPosition:="   , "-354.697014888131mm",
    "ZPosition:="   , "125.903435548132mm"
  ],
  "WhichAxis:="            , "X",
  "ZRotationAngle:="       , "0deg",
  "XOffset:="              , "0mm",
  "YOffset:="              , "0mm",
  "AutoAxis:="             , False
],
["NAME:Attributes",
"Name:="                , "FaceCS1",
"PartName:="            , "Box1"
])
```

EditNamedSelection

Modifies an entity list.

<table>
<tr><th>UI Access</th><th><b>Modeler</b> &gt; <b>Named Selection</b> &gt; <b>Reassign</b>.</th></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td>&lt;<i>SelectionsArray</i>&gt;</td><td>Array</td><td>Structured array. See: SelectionsArray.</td></tr>
<tr><td>&lt;<i>Parameters</i>&gt;</td><td>Array</td><td>Structured array.<pre>
Array("NAME:NamedSelectionParameters",

   "Type:=" , <string "Object" or "Face">,

   "Selection:=" , <string of object names or IDs>)

)
</pre></td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

<table>
<tr><th>Python Syntax</th><th>EditNamedSelection(&lt;<i>SelectionsArray</i>&gt;, &lt;<i>Parameters</i>&gt;)</th></tr>
<tr><td><b>Python Example</b></td><td><pre>
oEditor.EditNamedSelection(
   [
     "NAME:Selections",
     "Selections:=" , "FaceSelection1"
   ],
   [
     "NAME:NamedSelectionParameters",
     "Type:=" , "Face",
     "Selection:=" , [12]
   ])
</pre></td></tr>
</table>

EditObjectCS

Edits an existing object coordinate system. See: CreateObjectCS.

<table>
<tr><th>UI Access</th><th><b>Modeler</b> &gt; <b>Coordinate System</b> &gt; <b>Edit</b>.</th></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;Parameters&gt;</i></td><td>Array</td><td>Structured array.
<code>Array("NAME:ObjectCSParameters",</code>
<code> &lt;OriginArray&gt;</code>
<code> "MoveToEnd:=" , <boolean&gt;,</code>
<code> "ReverseXAxis:=" , <boolean&gt;,</code>
<code> "ReverseYAxis:=" , <boolean&gt;,</code>
<code> &lt;xAxisArray / xAxisPosArray&gt;</code>
<code> &lt;yAxisArray / yAxisPosArray&gt;)</code>
<b>Note:</b> xAxisArray and xAxisPosArray differ. Use xAxisArray for absolute position and xAxisPosArray for relative position. Do the same for yAxisArray and yAxisPosArray.</td></tr>
<tr><td>   <i>&lt;OriginArray&gt;</i></td><td>Array</td><td>Structured array.
<code>Array("NAME:Origin",</code>
<code> "IsAttachedToEntity:=" , <boolean&gt;,</code>
<code> "EntityID:=" , <integer&gt;,</code>
<code> "FacetedBodyTriangleIndex:=", <integer&gt;,</code>
<code> "TriangleVertexIndex:=" , <integer&gt;,</code></td></tr>
</table>
</td></tr>
</table>

		"PositionType:=" , <string "OnVertex", "EdgeCenter", "FaceCenter", "OnEdge", or "AbsolutePosition">, "UParam:=" , <float between 0 and 1 representing the relative position of the point on the edge or face>, "VParam:=" , <float between 0 and 1 representing the relative position of the point on the edge or face>, "XPosition:=" , <string>, "YPosition:=" , <string>, "ZPosition:=" , <string>) IsAttachedToEntity specifies whether the point is anchored. If True, provide UParam and VParam. If False, provide XPosition, YPosition, and ZPosition to provide fixed position. Pass "0" for unused parameters.
<xAxisArray>	Array	Structured array for absolute position: Array("NAME:xAxis", "DirectionType:=" , "AbsoluteDirection", "EdgeID:=" , <integer>, "FaceID:=" , <integer>, "xDirection:=" , <string>, "yDirection:=" , <string>, "zDirection:=" , <string>, "UParam:=" , <float>, "VParam:=" , <float>)

<xAxisPosArray>	Array	Structured array for relative position: Array("NAME:xAxisPos", "IsAttachedToEntity:=" , <boolean>, "EntityID:=" , <integer>, "FacetedBodyTriangleIndex:=", <integer>, "TriangleVertexIndex:=" , <integer>, "PositionType:=" , <string "OnVertex", "EdgeCenter", "FaceCenter", or "OnEdge">, "UParam:=" , <float>, "VParam:=" , <float>, "XPosition:=" , <string>, "YPosition:=" , <string>, "ZPosition:=" , <string>)
<yAxisArray>	Array	Structured array for absolute position: Array("NAME:yAxis", "DirectionType:=" , "AbsoluteDirection", "EdgeID:=" , <integer>, "FaceID:=" , <integer>, "xDirection:=" , <string>, "yDirection:=" , <string>, "zDirection:=" , <string>,

			"UParam:=" , <float>, "VParam:=" , <float>)
	<yAxisPosArray>	Array	Structured array for relative position: Array("NAME:yAxisPos", "IsAttachedToEntity:=" , <boolean>, "EntityID:=" , <integer>, "FacetedBodyTriangleIndex:=", <integer>, "TriangleVertexIndex:=" , <integer>, "PositionType:=" , <string "OnVertex", "EdgeCenter", "FaceCenter", or "OnEdge">, "UParam:=" , <float>, "VParam:=" , <float>, "XPosition:=" , <string>, "YPosition:=" , <string>, "ZPosition:=" , <string>)
	<AttributesArray>	Array	Structured array. See: AttributesArray. Use to select the coordinate system to edit.
Return Value	None.		

Python Syntax	EditObjectCS(<*Parameters*>,<*AttributesArray*>)
Python Example	oEditor.EditObjectCS(["NAME:ObjectCSParameters",

```
["NAME:Origin",
  "IsAttachedToEntity:="  , True,
  "EntityID:="            , 59,
  "FacetedBodyTriangleIndex:=", -1,
  "TriangleVertexIndex:=" , -1,
  "PositionType:="        , "OnVertex",
  "UParam:="              , 0,
  "VParam:="              , 0,
  "XPosition:="           , "0",
  "YPosition:="           , "0",
  "ZPosition:="           , "0"
],
"MoveToEnd:="           , False,
"ReverseXAxis:="        , False,
"ReverseYAxis:="        , False,
["NAME:xAxis",
  "DirectionType:="       , "AbsoluteDirection",
  "EdgeID:="              , -1,
  "FaceID:="              , -1,
  "xDirection:="          , "1",
    "yDirection:="          , "0",
    "zDirection:="          , "0",
    "UParam:="              , 0,
    "VParam:="              , 0
  ],
  ["NAME:yAxis",
    "DirectionType:="       , "AbsoluteDirection",
    "EdgeID:="              , -1,
    "FaceID:="              , -1,
    "xDirection:="          , "0",
    "yDirection:="          , "1",
    "zDirection:="          , "0",
    "UParam:="              , 0,
    "VParam:="              , 0
  ]
],
["NAME:Attributes",
  "Name:="                  , "ObjectCS1",
  "PartName:="              , "Box2"
])
```

EditRelativeCS

Edits an existing Relative Coordinate System. See: CreateRelativeCS.

<table>
<tr><td><b>UI Access</b></td><td><b>Modeler</b> > <b>Coordinate System</b> > <b>Edit</b>.</td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;Parameters&gt;</i></td><td>Array</td><td>Structured array.
Array("NAME:RelativeCSParameters",
  "Mode:=" , "Axis/Position",
  "OriginX:=" , <string>,
  "OriginY:=" , <string>,
  "OriginZ:=" , <string>,
  "XAxisXvec:=" , <string>,
  "XAxisYvec:=" , <string>,
  "XAxisZvec:=" , <string>,
  "YAxisXvec:=" , <string>,
  "YAxisYvec:=" , <string>,
  "YAxisZvec:=" , <string&gt;)</td></tr>
<tr><td><i>&lt;AttributesArray&gt;</i></td><td>Array</td><td>Structured array. See: AttributesArray. Use to select the coordinate system to edit.</td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	EditRelativeCS(<*Parameters*>,<*AttributesArray*>)

Python Example	`oEditor.EditRelativeCS(`
	`["NAME:RelativeCSParameters",`
	`  "Mode:="                , "Axis/Position",`
	`  "OriginX:="             , "0.62mm",`
	`  "OriginY:="             , "-0.7mm",`
	`  "OriginZ:="             , "0mm",`
	`  "XAxisXvec:="           , "1mm",`
	`  "XAxisYvec:="           , "0mm",`
	`  "XAxisZvec:="           , "0mm",`
	`  "YAxisXvec:="           , "0mm",`
	`  "YAxisYvec:="           , "1mm",`
	`  "YAxisZvec:="           , "0mm"`
	`],`
	`["NAME:Attributes",`
	`  "Name:="                , "RelativeCS1"`
	`])`

Export

Exports the model to a file.

> **Note:** This script does not export image file types or GDSII files. See: ExportModelImageToFile and ExportGDSII.

<table>
<tr><th>UI Access</th><th>Modeler > Export</th></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td><i>&lt;Parameters&gt;</i></td><td>Array</td><td>Structured array.
<code>Array("NAME:ExportParameters",</code>
<code>"AllowRegionDependentPartSelectionForPMLCreation:=", <boolean&gt;,</code>
<code>"AllowRegionSelectionForPMLCreation:=", <boolean&gt;,</code>
<code>"Selections:=" , <string list&gt;,</code>
<code>"File Name:=" , <string filepath&gt;,</code>
<code>"Major Version:=" , <integer (-1 if not applicable)&gt;,</code>
<code>"Minor Version:=" , <integer (-1 if not applicable)&gt;)</code></td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	Export(<*Parameters*>)

Python Example	`oEditor.Export(`
	`["NAME:ExportParameters",`

```
  "AllowRegionDependentPartSelectionForPMLCreation:=", True,
  "AllowRegionSelectionForPMLCreation:=", True,
  "Selections:="          , "Box1,Box2,Box3",
  "File Name:="           , "C:/Users/jdoe/Desktop/export.sab",
  "Major Version:="       , 25,
  "Minor Version:="       , 0
])
```

ExportModelImageToFile

Exports the model as an image file (*.avz, *.bmp, *.gif, *.jpeg, *.tiff, *.wrl). In Release 23.1, this command is fully supports -ng (non-graphical) mode. To export to Ensight use *.avz. For export to Ensight in -ng mode, the corresponding version of Ensight must be installed. On Linux, it might need manual set environment variable AWP_ROOT212 to its installation path, e.g. AWP_ROOT212-2=/installations/ansys_inc/v212/ for AnsysEDT v21.2 and Ensight 21.2.

ExportModelImageToFile exports a model image with background type and color that respect the AEDT color scheme by default. You can specify the background type and color with following parameters:

Parameter Name	Description	Parameter Value
BackgroundType	Choose one out of four types of background	Default Plain LinearGradient RadialGradient
BackgroundColor	Plain: Background color LinearGradient/RadialGradient: Background start color	3 integers with a range of [0,255] that represent red, green, and blue respectively
BackgroundContrastColor	Plain: Ignored LinearGradient/RadialGradient: Background end color	3 integers with a range of [0,255] that represent red, green, and blue respectively

Note: Current scripts will not be affected, the image will be exported with default background color as it always does

If no BackgroundType is specified, or BackgroundType is Default, BackgroundColor and BackgroundContrastColor will be ignored, and the image will be exported with default background color

ExportModelImageToFile supports export overlay of polar plot 3D with existing transformation (scaling, rotation and translation) in -ng (non-graphical) mode.

<table>
<tr><td><b>UI Access</b></td><td><b>Modeler</b> &gt; <b>Export</b>.</td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;path&gt;</i></td><td>String</td><td>Full file path including extension.</td></tr>
<tr><td><i>&lt;width&gt;</i></td><td>Integer</td><td>Width in pixels (use 0 for default).</td></tr>
<tr><td><i>&lt;height&gt;</i></td><td>Integer</td><td>Height in pixels (use 0 for default).</td></tr>
<tr><td><i>&lt;Parameters&gt;</i></td><td>Array</td><td>Structured array.
Array("NAME:SaveImageParams",
  "ShowAxis:=" , <string containing boolean>,
  "ShowGrid:=" , <string containing boolean>,
  "ShowRuler:=" , <string containing boolean>,
  "ShowRegion:=" , <string>,
  "Selections:=" , <string>,
  "FieldPlotSelections:=", <string>' # Comma delimited string. #Use to set which field plot to show.</td></tr>
</table>
</td></tr>
</table>

	`"FitToSelections:=" , "",`
	Note: "FitToSelections" specify geometry objects for the "Fit" operation.
	`"FitToFieldPlotSelection:=" , ""`
	Note: "FitToFieldPlotSelections" specifies field plots for the "Fit" operation.
	`"AutoFit:=" , "True",`
	Note: If FitToSlections or FitToFieldPlotSelections are used , then AutoFit is True, it makes sure color key does not overlap field plot. It is False by default. If neither is used, then "AutoFit" will "Fit" to full model.
	`"Orientation:=" , <string>`
	`"ShowOrientationGadget:=" , <False>`

Return Value	None.

Python Syntax	ExportModelImageToFile(<*path*> <*width*> <*height*> <*Parameters*>)

Python Example	`oEditor.ExportModelImageToFile`
	`  ("D:/Image.png", 1920, 1080,`

	`[`
	`  "NAME:SaveImageParams",`
	`    "ShowAxis:=" , "True",`
	`    "ShowGrid:=" , "True",`
	`    "ShowRuler:=" , "True",`
	`    "ShowRegion:=" , "Default",`
	`    "Selections:=" , "",`
	`    "FieldPlotSelections:=" , "",`
	`    "FitToSelections:=" , "",`
	`    "FitToFieldPlotSelections:=", "",`
	`    "AutoFit:=" , "True",`
	`    "Orientation:=" , ""`
	`])`

ExportModelMeshToFile

Exports geometry model to a 3D model file (e.g. *.obj, *.wrl, etc.).

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;filePath&gt;</i></td><td>String</td><td>Full file path, including extension *.obj, *.wrl, etc</td></tr>
<tr><td><i>&lt;selections&gt;</i></td><td>Array</td><td>Selected parts.</td></tr>
</table>
</td></tr>
</table>

Return Value	None.

Python Syntax	ExportModelMeshToFile <*filePath*>, <*selections*>)
Python Example	`oEditor.ExportModelMeshToFile("E:/MyDir/scriptRun/2Selected-ng.obj", ["BotCover-","AveragingVolumeAtPeakRMSEfieldLocation"])`

Fillet

Performs a fillet on specified edge(s).

<table>
<tr><td><b>UI Access</b></td><td><b>Modeler</b> > <b>Fillet</b>.</td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td>&lt;<i>SelectionsArray</i>&gt;</td><td>Array</td><td>Structured array. See: SelectionsArray.</td></tr>
<tr><td>&lt;<i>Parameters</i>&gt;</td><td>Array</td><td>Structured array.
Array("NAME:Parameters",
  <FilletParametersArray&gt;)</td></tr>
<tr><td> &lt;<i>FilletParametersArray</i>&gt;</td><td>Array</td><td>Structured array.
Array("NAME:FilletParameters",
  "Edges:=" , <array containing integer edge IDs>,
  "Vertices:=" , <empty array>,
  "Radius:=" , <string>,
  "Setback:=" , <string&gt;)</td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	Fillet(<*SelectionsArray*>, <*Parameters*>)

Python Example	`oEditor.Fillet(`
	`["NAME:Selections",`
	`  "Selections:="          , "Box1",`
	`  "NewPartsModelFlag:="   , "Model"`
	`],`
	`["NAME:Parameters",`
	`  ["NAME:FilletParameters",`
	`    "Edges:="               , [13],`
	`    "Vertices:="            , [],`
	`    "Radius:="              , "1mm",`
	`    "Setback:="             , "0mm"`
	`  ]`
	`])`

FlattenGroup

Flattens a specified history tree group.

<table>
<tr><th>UI Access</th><th><b>Modeler</b> &gt; <b>Group</b> &gt; <b>Flatten</b>.</th></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td>&lt;<i>GroupID</i>&gt;</td><td>Array</td><td>Structured array.</td></tr>
</table>
</td></tr>
</table>

			Array("Groups:=", Array(<string list of group IDs>))
Return Value	None.		

Python Syntax	FlattenGroup (<*GroupID*>)
Python Example	`oEditor.FlattenGroup(["Groups:=", ["Group1"]])`

GenerateHistory

Generates the history for specified 1D object(s).

UI Access	**Modeler** > **Generate History**.		
Parameters	Name	Type	Description
	<*SelectionsArray*>	Array	Structured array. See: SelectionsArray.
Return Value	None.		

Python Syntax	GenerateHistory (<*SelectionsArray*>)

Python Example	`oEditor.GenerateHistory(`
	`["NAME:Selections",`
	`"Selections:="           , "Polyline1",`
	`"NewPartsModelFlag:="    , "Model",`
	`"UseCurrentCS:="         , True`

	])

GetActiveCoordinateSystem

Returns the active coordinate system.

UI Access	None.
Parameters	None.
Return Value	String name of active coordinate system.

Python Syntax	GetActiveCoordinateSystem()
Python Example	`oEditor.GetActiveCoordinateSystem()`

GetActiveCoordinateSystemTransform

Returns transformation information for the active coordinate system (with respect to the global coordinate system), which consists of the affine matrix for rotation of the coordinate system and a 3D vector with the translation of the coordinate system's origin with respect to the global coordinate system's origin.

UI Access	None
Parameters	None
Return Value	Array of strings containing transformation information

Python Syntax	GetActiveCoordinateSystemTransform()
Python Example	`oEditor.GetActiveCoordinateSystemTransform()`

GetCoordinateSystems

Returns the names of coordinate systems in the design.

UI Access	None.
Parameters	None.
Return Value	Array containing string names of coordinate systems.

Python Syntax	GetCoordinateSystems()
Python Example	`oEditor.GetCoordinateSystems()`

HealObject

Heals an imported object.

<table>
<tr><td><b>UI Access</b></td><td><b>Modeler</b> &gt; <b>Model Preparation</b> &gt; <b>Heal</b>.</td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;SelectionsArray&gt;</i></td><td>Array</td><td>Structured array. See: SelectionsArray.</td></tr>
<tr><td><i>&lt;Parameters&gt;</i></td><td>Array</td><td>Structured array.
<code>Array("NAME:ObjectHealingParameters",</code>
<code> "Version:=" , <integer&gt;,</code>
<code> "AutoHeal:=" , <boolean&gt;,</code></td></tr>
</table>
</td></tr>
</table>

			"TolerantStitch:=" , <boolean>, "SimplifyGeom:=" , <boolean>, "TightenGaps:=" , <boolean>, "HealToSolid:=" , <boolean>, "StopAfterFirstStitchError:=", <boolean>, "MaxStitchTol:=" , <float>, "ExplodeAndStitch:=" , <boolean>, "GeomSimplificationTol:=", <integer>, "MaximumGeneratedRadiusForSimplification:=", <integer>, "SimplifyType:=" , <integer>, "TightenGapsWidth:=" , <float>, "RemoveSliverFaces:=" , <boolean>, "RemoveSmallEdges:=" , <boolean>, "RemoveSmallFaces:=" , <boolean>, "SliverFaceTol:=" , <integer>, "SmallEdgeTol:=" , <integer>, "SmallFaceAreaTol:=" , <integer>, "BoundingBoxScaleFactor:=", <integer>, "RemoveHoles:=" , <boolean>,

			"RemoveChamfers:=" , <boolean>, "RemoveBlends:=" , <boolean>, "HoleRadiusTol:=" , <integer>, "ChamferWidthTol:=" , <integer>, "BlendRadiusTol:=" , <integer>, "AllowableSurfaceAreaChange:=", <integer>, "AllowableVolumeChange:=", <integer>)
Return Value	None.		

Python Syntax	HealObject(<*SelectionsArray*>, <*Parameters*>)

Python Example	oEditor.HealObject(
	["NAME:Selections",
	  "Selections:=" , "Box1",
	  "NewPartsModelFlag:=" , "Model"
	],
	["NAME:ObjectHealingParameters",
	  "Version:=" , 1,
	  "AutoHeal:=" , True,
	  "TolerantStitch:=" , True,
	  "SimplifyGeom:=" , True,
	  "TightenGaps:=" , True,

```
"HealToSolid:="         , False,
"StopAfterFirstStitchError:=", False,
"MaxStitchTol:="        , 0.001,
"ExplodeAndStitch:="    , True,
"GeomSimplificationTol:=", -1,
"MaximumGeneratedRadiusForSimplification:=", -1,
"SimplifyType:="        , 2,
"TightenGapsWidth:="    , 1E-06,
"RemoveSliverFaces:="   , False,
"RemoveSmallEdges:="    , False,
"RemoveSmallFaces:="    , False,
"SliverFaceTol:="       , 0,
"SmallEdgeTol:="        , 0,
"SmallFaceAreaTol:="    , 0,
"BoundingBoxScaleFactor:=", 1250,
"RemoveHoles:="         , False,
"RemoveChamfers:="      , False,
"RemoveBlends:="        , False,
"HoleRadiusTol:="       , 0,
"ChamferWidthTol:="     , 0,
```

	"BlendRadiusTol:=" , 0, "AllowableSurfaceAreaChange:=", 5, "AllowableVolumeChange:=", 5])

Import

Imports a 3D model file.

> **Note:**
>
> This script does not import DXF or GDSII models. See: ImportDXF and ImportGDSII.

<table>
<tr><th>UI Access</th><th>Modeler > Import.</th></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;Parameters&gt;</i></td><td>Array</td><td>Structured array.
Array("NAME:NativeBodyParameters",
"HealOption:=" , <integer>,
"Options:=" , <string>,
"FileType:=" , <string "UnRecognized">,
"MaxStitchTol:=" , <integer>,
"ImportFreeSurfaces:=" , <boolean>,
"GroupByAssembly:=" , <boolean>,
"CreateGroup:=" , <boolean&gt;,</td></tr>
</table>
</td></tr>
</table>

			"STLFileUnit:=" , <string>, "MergeFacesAngle:=" , <float>, "HealSTL:=" , <boolean>, "ReduceSTL:=" , <boolean>, "ReduceMaxError:=" , <integer>, "ReducePercentage:=" , <integer>, "PointCoincidenceTol:=" , <float>, "CreateLightweightPart:=", <boolean>, "ImportMaterialNames:=" , <boolean>, "SeparateDisjointLumps:=", <boolean>, "SourceFile:=" , <string>)
Return Value	None.		

Python Syntax	Import(<*Parameters*>)
Python Example	oEditor.Import(["NAME:NativeBodyParameters", "HealOption:=" , 0, "Options:=" , "-1", "FileType:=" , "UnRecognized", "MaxStitchTol:=" , -1,

```
  "ImportFreeSurfaces:="  , False,
  "GroupByAssembly:="     , False,
  "CreateGroup:="         , True,
  "STLFileUnit:="         , "Auto",
  "MergeFacesAngle:="     , 0.02,
  "HealSTL:="             , False,
  "ReduceSTL:="           , False,
  "ReduceMaxError:="      , 0,
  "ReducePercentage:="    , 100,
  "PointCoincidenceTol:=" , 1E-06,
  "CreateLightweightPart:=", False,
  "ImportMaterialNames:=" , False,
  "SeparateDisjointLumps:=", False,
  "SourceFile:="          , "C:\\Users\\jdoe\\Desktop\\export.model"
])
```

ImportDXF [Modeler]

Imports a DXF model file.

<table>
<tr><td><b>UI Access</b></td><td><b>Modeler</b> &gt; <b>Import</b>.</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><i>&lt;Parameters&gt;</i></td><td>Array</td><td>Structured array.</td></tr></table></td></tr>
</table>

			Array("NAME:options", "FileName:=" , <string>, "Scale:=" , <float>, "AutoDetectClosed:=" , <boolean>, "SelfStitch:=" , <boolean>, "DefeatureGeometry:=" , <boolean>, "DefeatureDistance:=" , <integer>, "RoundCoordinates:=" , <boolean>, "RoundNumDigits:=" , <integer>, "WritePolyWithWidthAsFilledPoly:=", <boolean>, "ImportMethod:=" , <integer>, "2DSheetBodies:=" , <boolean>, <layersArray>)
	<layersArray>	Array	Structured array. Array("NAME:LayerInfo", <layer>, <layer>, <layer>,...)
	<layer>	Array	Structured array. Array("NAME:<layerName>", "source:=" , <string>, "display_source:=" , <string>, "import:=" , <boolean>,

			"dest:=" , <string>, "dest_selected:=" , <boolean>, "layer_type:=" , <string>)
Return Value	None.		

Python Syntax	ImportDXF(<*Parameters*>)

Python Example	oEditor.ImportDXF(
	  ["NAME:options",
	  "FileName:=", "C:/Users/jdoe/Desktop/export.dxf",
	  "Scale:=" , 0.001,
	  "AutoDetectClosed:=" , True,
	  "SelfStitch:=" , True,
	  "DefeatureGeometry:=" , False,
	  "DefeatureDistance:=" , 0,
	  "RoundCoordinates:=" , False,
	  "RoundNumDigits:=" , 4,
	  "WritePolyWithWidthAsFilledPoly:=", False,
	  "ImportMethod:=" , 1,
	  "2DSheetBodies:=" , False,
	  ["NAME:LayerInfo",
	    ["NAME:0",

```
"source:="              , "0",
"display_source:="      , "0",
"import:="              , True,
"dest:="                , "0",
"dest_selected:="       , False,
"layer_type:="          , "signal"
],
["NAME:LAYER_1",
"source:="              , "LAYER_1",
"display_source:="      , "LAYER_1",
"import:="              , True,
"dest:="                , "LAYER_1",
"dest_selected:="       , False,
"layer_type:="          , "signal"
],
["NAME:LAYER_2",
"source:="              , "LAYER_2",
"display_source:="      , "LAYER_2",
"import:="              , True,
"dest:="                , "LAYER_2",
```

	<code>"dest_selected:=" , False,</code> <code>"layer_type:=" , "signal"</code> <code>]</code> <code>]</code> <code>])</code>

ImportFromClipboard

Imports a model from clipboard.

UI Access	**Modeler** > **Import From Clipboard**.
Parameters	None.
Return Value	None.

Python Syntax	ImportFromClipboard ()
Python Example	`oEditor.ImportFromClipboard()`

ImportGDSII [Modeler]

Imports a GDSII model file.

UI Access	**Modeler** > **Import**.
Parameters	<table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><i>&lt;Parameters&gt;</i></td><td>Array</td><td>Structured array. <code>Array("NAME:options",</code></td></tr></table>

		`"FileName:=" , <string>,`
		`"FlattenHierarchy:=" , <boolean>,`
		`"ImportMethod:=" , <integer>,`
		`<layerMapArray>, <layerMapArray>, <layerMapArray>,...`
		`"OrderMap:=" ,`
		`[ "entry:=" ,`
		`<entry>, <entry>, <entry>,...`
		`]`
		`])`

<layerMapArray>	Array	Structured array.
		`Array("NAME:LayerMapInfo",`
		`"LayerNum:=" , <integer>,`
		`"DestLayer:=" , <string>,`
		`"layer_type:=" , <string>)`

<entry>	Array	Structured array. `Array("order:=" , <integer LayerNum>, "layer:=" , <string DestLayer>)`

Return Value	None.

Python Syntax	ImportGDSII(<*Parameters*>)
Python Example	`oEditor.ImportGDSII(`

```
["NAME:options",
  "FileName:=", "C:/Users/coil2.gds",
  "FlattenHierarchy:="    , True,
  "ImportMethod:="        , 1,
  ["NAME:LayerMap",
    ["NAME:LayerMapInfo",
    "LayerNum:="            , 12,
    "DestLayer:="           , "Signal12",
    "layer_type:="          , "signal"
    ],
    ["NAME:LayerMapInfo",
    "LayerNum:="            , 13,
    "DestLayer:="           , "Signal13",
    "layer_type:="          , "signal"
    ],
    ["NAME:LayerMapInfo",
    "LayerNum:="            , 14,
    "DestLayer:="           , "Signal14",
    "layer_type:="          , "signal"
    ]
  ],
"OrderMap:=",
  [
  "entry:=",
    ["order:=", 0,  "layer:=", "Signal12"],
  "entry:=",
    ["order:=", 1,  "layer:=", "Signal13"],
  "entry:=",
    ["order:=", 2,  "layer:=", "Signal14"]
  ]
])
```

Imprint

Imprints the geometry of one object upon another.

<table>
<tr><td><b>UI Access</b></td><td><b>Modeler</b> > <b>Boolean</b> > <b>Imprint...</b>.</td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td>&lt;<i>Selections</i>&gt;</td><td>Array</td><td>Structured array.
<code>Array("NAME:Selections",</code>
<code> "Blank Parts:=", <string, object name&gt;,</code>
<code> "Tool Parts:=", <string, object name&gt;)</code></td></tr>
<tr><td>&lt;<i>Parameters</i>&gt;</td><td>Array</td><td>Structured array.</td></tr>
</table>
</td></tr>
</table>

	<table><tr><td></td><td></td><td><code>Array("NAME:ImprintParameters",</code> <code> "KeepOriginals:=", <boolean&gt;)</code></td></tr></table>
Return Value	None.

Python Syntax	Imprint (<*Selections*>, <*Parameters*>)

Python Example	`oEditor.Imprint(`
	`["NAME:Selections",`
	`  "Blank Parts:=", "Cylinder1",`
	`  "Tool Parts:=", "Box1"],`
	`["NAME:ImprintParameters",`
	`  "KeepOriginals:=", False])`

ImprintProjection

Projects the form of a sheet object onto the face or faces of another object (either solid or sheet).

UI Access	**Modeler** > **Boolean** > **Imprint Projection**.
Parameters	<table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;<i>Selections</i>&gt;</td><td>Array</td><td>Structured array. <code>Array("NAME:Selections",</code> <code> "Selections:=", <string, selected objects&gt;)</code></td></tr><tr><td>&lt;<i>Parameters</i>&gt;</td><td>Array</td><td>Structured array.</td></tr></table>

			Array("NAME:ImprintProjectionParameters", "KeepOriginals:=", <boolean>, "NormalProjection:=", <boolean>, "Distance:=" , <float>, "DirectionX:=" , <float>, "DirectionY:=" , <float>, "DirectionZ:=" , <float>)
Return Value	None.		

Python Syntax	ImprintProjection (<*Selections*>, <*Parameters*>)
Python Example	oEditor.ImprintProjection(["NAME:Selections", "Selections:=", "Rectangle1,Cylinder1"], ["NAME:ImprintProjectionParameters", "KeepOriginals:=", False, "NormalProjection:=", True, "Distance:=", "1.36014705087354mm", "DirectionX:=", "0.882257546512569", "DirectionY:=", "0.294085848837523",

	`"DirectionZ:=", "0.367607311046904"])`

Intersect

Intersects specified objects.

<table>
<tr><td><b>UI Access</b></td><td><b>Modeler</b> &gt; <b>Boolean</b> &gt; <b>Intersect</b>.</td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;SelectionsArray&gt;</i></td><td>Array</td><td>Structured array. See: SelectionsArray.</td></tr>
<tr><td><i>&lt;Parameters&gt;</i></td><td>Array</td><td>Structured array.
<code>Array("NAME:IntersectParameters",</code>
<code>"KeepOriginals:=" , <boolean True to keep original objects; False to delete&gt;)</code></td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	Intersect(<*SelectionsArray*>, <*Parameters*>)

Python Example	`oEditor.Intersect(`
	`["NAME:Selections",`
	`  "Selections:=", "Rectangle1,Rectangle2"`
	`],`
	`["NAME:IntersectParameters",`
	`  "KeepOriginals:=", False`
	`])`

MoveCStoEnd

Moves a specified Object Coordinate System to the end of the History tree.

<table>
<tr><td><b>UI Access</b></td><td><b>Modeler</b> &gt; <b>Coordinate System</b> &gt; <b>Move CS to End</b>.</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;<i>SelectionsArray</i>&gt;</td><td>Array</td><td>Structured array. See: SelectionsArray.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

<table>
<tr><td><b>Python Syntax</b></td><td>MoveCStoEnd (&lt;<i>SelectionsArray</i>&gt;)</td></tr>
<tr><td><b>Python Example</b></td><td><pre>oEditor.MoveCSToEnd(
["NAME:Selections",
  "Selections:=" , "ObjectCS1"
])</pre></td></tr>
</table>

MoveEntityToGroup

Moves a specified entity or entities to a specified group.

<table>
<tr><td><b>UI Access</b></td><td>Drag item into the group in the history tree.</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;<i>Objects</i>&gt;</td><td>Array</td><td>Structured array. Selects the entity/entities to move.</td></tr></table></td></tr>
</table>

			Array("Objects:=" , <array containing string object IDs>)
	<*MoveEntityToGroup*>	Array	Structured array. Array("ParentGroup:=" , <string group name>)
Return Value	None.		

Python Syntax	MoveEntityToGroup (<*Objects*>, <*MoveEntityToGroup*>)
Python Example	oEditor.MoveEntityToGroup(["Objects:=", ["Box3"]], ["ParentGroup:=", "Box_Group"])

MoveFaces

Moves the specified faces along normal or along a vector.

UI Access	**Modeler** > **Surface** > **Move Faces** > **Along** [**Normal/Vector**].		
Parameters	Name	Type	Description
	<*SelectionsArray*>	Array	Structured array. See: SelectionsArray.
	<*MoveFacesParametersArray*>	Array	Structured array. Array("NAME:Parameters", <FacesOfOneObjToMove>, <FacesOfOneObjToMove>, ...)
	<*FacesOfOneObjToMove*>	Array	Structured array. Array("Name:MoveFacesParameters",

			"MoveAlongNormalFlag:=", <boolean>, "OffsetDistance:=", <string>, "MoveVectorX:=", <string>, "MoveVectorY:=", <string>, "MoveVectorZ:=", <string>, "FacesToMove:=", <array>) MoveAlongNormalFlag specifies whether to move along the face normal or along a vector. If false, provide the MoveVectorX, MoveVectorY, and MoveVectorZ parameters. FacesToMove is an array of integers (the IDs of the faces to move).
Return Value	None		

Python Syntax	MoveFaces (<*SelectionsArray*>, <*MoveFacesParametersArray*>)

Python Example

```
oEditor.MoveFaces(
    [
        "NAME:Selections",
        "Selections:="        , "Rectangle1",
        "NewPartsModelFlag:="    , "Model"
    ],
    [
        "NAME:Parameters",
        [
            "NAME:MoveFacesParameters",
                    "MoveAlongNormalFlag:="    , True,
                    "OffsetDistance:="     , "1mm",
                    "MoveVectorX:="        , "0mm",
                    "MoveVectorY:="        , "0mm",
                    "MoveVectorZ:="        , "0mm",
                    "FacesToMove:="        , [183]
                ]
            ])
```

ProjectSheet

Project a sheet object, typically for modeling thin conformal deposits. Typically followed by **Thicken Sheet**.

<table>
<tr><td><b>UI Access</b></td><td>Click <b>Modeler</b> &gt; <b>Surface</b> &gt; <b>Project Sheet</b>.</td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;SelectionsArray&gt;</i></td><td>Array</td><td>Structured array. See: SelectionsArray.</td></tr>
<tr><td><i>&lt;Parameters&gt;</i></td><td>Array</td><td>Array containing string.
<code>Array("NAME:ProjectSheetParameters")</code></td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	ProjectSheet (<*SelectionsArray*>, <*Parameters*>)

Python Example	`oEditor.ProjectSheet(`
	`['NAME:Selections',`
	`        'Selections:=','Box1,Box2,Polyline1'],`
	`['NAME:ProjectSheetParameters']`

	)

PurgeHistory

Purges the history of a specified object.

UI Access	**Modeler** > **Purge History**.
Parameters	
Return Value	None.

Name	Type	Description
<SelectionsArray>	Array	Structured array. See: SelectionsArray.

Python Syntax	PurgeHistory (<*SelectionsArray*>)

Python Example	`oEditor.PurgeHistory(`
	`["NAME:Selections",`
	`   "Selections:=", "Box2",`
	`   "NewPartsModelFlag:=", "Model"`
	`])`

ReplaceWith3DComponent

Replaces the selection with a 3D component.

<table>
<tr><td><b>UI Access</b></td><td>None.</td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td>&lt;<i>CreateData</i>&gt;</td><td>Array</td><td>Structured array.</td></tr>
<tr><td>&lt;<i>DesignData</i>&gt;</td><td>Array</td><td>Structured array.</td></tr>
<tr><td>&lt;<i>ImageFile</i>&gt;</td><td>Array</td><td>Structured array.</td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

<table>
<tr><td><b>Python Syntax</b></td><td>ReplaceWith3DComponent(&lt;<i>CreateData</i>&gt;, &lt;<i>DesignData</i>&gt;, &lt;<i>ImageFile</i>&gt;)</td></tr>
<tr><td><b>Python Example</b></td><td>
<pre>
oEditor.ReplaceWith3DComponent(
  ["NAME:CreateData",
    "ComponentName:=", "CoaxBend",
    "Company:=", "",
    "Company URL:=", "",
    "Model Number:=", "",
    "Help URL:=", "",
    "Version:=", "1.0",
    "Notes:=", "",
    "IconType:=", "",
    "Owner:=", "Jane Doe",
    "Email:=", "jdoe@email.com",
    "Date:=", "3:46:31 PM Jul 26, 2020",
</pre>
</td></tr>
</table>

```
    "HasLabel:=", False,
    "IncludedParts:=", ["outer","teflon","inner","teflon_1"],
    "HiddenParts:=", [],
    "IncludedCS:=", [],
    "DefaultHandle:=", "Global",
    "IncludedParameters:=", ["bend_angle"],
    "ParameterDescription:=", ["bend_angle:=", ""]
  ],
  ["NAME:DesignData",
    "Excitations:=", ["1","2"]
  ],
  ["NAME:ImageFile",
    "ImageFile:=", ""
  ]
)
```

Section

Creates a 2D cross-section of the selection in the specified plane.

UI Access	**Modeler** > **Surface** > **Section**.

<table>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td><i>&lt;SelectionsArray&gt;</i></td><td>Array</td><td>Structured array. See: SelectionsArray.</td></tr>
<tr><td><i>&lt;Parameters&gt;</i></td><td>Array</td><td>Structured array.
Array("NAME:SectionToParameters",
  "CreateNewObjects:=" , <boolean>,
  "SectionPlane:=" , <string "XY", "YZ", or "ZX">,
  "SectionCrossObject:=" , <boolean&gt;)</td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

<table>
<tr><td><b>Python Syntax</b></td><td>Section (&lt;<i>SelectionsArray</i>&gt;, &lt;<i>Parameters</i>&gt;)</td></tr>
<tr><td><b>Python Example</b></td><td>oEditor.Section(
["NAME:Selections",
  "Selections:="   , "Cone1",
  "NewPartsModelFlag:="   , "Model"
],
["NAME:SectionToParameters",
  "CreateNewObjects:="   , True,
  "SectionPlane:="   , "XY",
  "SectionCrossObject:=" , False
])</td></tr>
</table>

SeparateBody

Separates the bodies of specified multi-lump objects.

<table>
<tr><td><b>UI Access</b></td><td><b>Modeler</b> &gt; <b>Boolean</b> &gt; <b>Separate Bodies</b>.</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;<i>SelectionsArray</i>&gt;</td><td>Array</td><td>Structured array. See: SelectionsArray.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	SeparateBody (<*SelectionsArray*>)

Python Example	`oEditor.SeparateBody(`
	`["NAME:Selections",`
	`  "Selections:=", "Rectangle1,Circle1",`
	`  "NewPartsModelFlag:=", "Model"`
	`])`

SetModelUnits

Sets the model units.

UI Access	**Modeler** > **Units**.

<table>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;Parameters&gt;</i></td><td>Array</td><td>Structured array.
Array("NAME:Units Parameter",
"Units:=", <string>,
"Rescale:=", <boolean True to rescale model; else False>)
To see valid unit strings, select <b>Modeler</b> &gt; <b>Units</b>.</td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

<table>
<tr><td><b>Python Syntax</b></td><td>SetModelUnits (&lt;<i>Parameters</i>&gt;)</td></tr>
<tr><td><b>Python Example</b></td><td>oEditor.SetModelUnits(
["NAME:Units Parameter",
"Units:=", "km",
"Rescale:=", False
])</td></tr>
</table>

SetWCS

Sets the working coordinate system.

<table>
<tr><td><b>UI Access</b></td><td><b>Modeler</b> &gt; <b>Coordinate System</b> &gt; <b>Set Working CS</b>.</td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;Parameters&gt;</i></td><td>Array</td><td>Structured array.</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><td></td><td><table><tr><td></td><td></td><td>Array("NAME:SetWCS Parameter",
"Working Coordinate System:=", <string CS name>,
"RegionDepCSOk:=" , <boolean True if region-dependent; else False)</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	SetWCS (<*Parameters*>)

Python Example	`oEditor.SetWCS(`
	`["NAME:SetWCS Parameter",`
	`"Working Coordinate System:=", "Global",`
	`"RegionDepCSOk:=", False`
	`])`

ShowWindow

Opens the active 3D Modeler window.

UI Access	None.
Parameters	None.
Return Value	None.

Python Syntax	ShowWindow
Python Example	`oEditor.ShowWindow()`

Simplify

Converts a complex MCAD object into simpler primitives which are easy to mesh and solve.

<table>
<tr><td><b>UI Access</b></td><td><b>Modeler</b> &gt; <b>Model Preparation</b> &gt; <b>Simplify</b>.</td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td>&lt;<i>Selections</i>&gt;</td><td>Array</td><td>Structured array. See: SelectionsArray.</td></tr>
<tr><td>&lt;<i>Parameters</i>&gt;</td><td>Array</td><td>Structured array.
Array("NAME:SimplifyParameters",
"Type:=", <string fit type>,
"ExtrusionAxis:=", <string>,
"Cleanup:=", <boolean>,
"Splitting:=", <boolean>,
"SeparateBodies:=", <boolean>,
"CloneBody:=", <boolean>,
"Generate Primitive History:=", <boolean>,
"NumberPointsCurve:=", <integer>,
"LengthThresholdCurve:=", <integer&gt;)</td></tr>
<tr><td>&lt;<i>CreateGroup</i>&gt;</td><td>Array</td><td>Structured array.
Array("CreateGroupsForNewObjects:=",<boolean&gt;)</td></tr>
</table>
</td></tr>
</table>

Return Value	None.

Python Syntax	Simplify (<*Selections*>, <*Parameters*>, <*CreateGroup*>)
Python Example	oEditor.Simplify(["NAME:Selections", "Selections:=", "Cylinder1,Box2", "NewPartsModelFlag:=", "Model"], ["NAME:SimplifyParameters", "Type:=", "Polygon Fit", "ExtrusionAxis:=", "Auto", "Cleanup:=", True, "Splitting:=", True, "SeparateBodies:=", False, "CloneBody:=", False, "Generate Primitive History:=", True, "NumberPointsCurve:=", 3, "LengthThresholdCurve:=", 20], ["CreateGroupsForNewObjects:=", True])

Split

Splits the specified object(s) along a plane.

UI Access	**Modeler** > **Boolean** > **Split**.		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: SelectionsArray.

	<Parameters>	Array	Structured array.
			`Array("NAME:SplitToParameters",`
			`"SplitPlane:=" , <string "XY", "YZ", "ZX", or "Dummy">,`
			`"WhichSide:=" , <string "PositiveOnly", "NegativeOnly" or "Both">,`
			`"ToolType:=" , <string "PlaneTool" or "FaceTool">,`
			`"ToolEntityID:=" , <-1 if using SplitPlane; else FaceID>,`
			`"SplitCrossingObjectsOnly:=", <boolean>,`
			`"DeleteInvalidObjects:=", <boolean>)`
			You can split an object either along an existing plane , or by creating a plane using an edge.
			For existing plane:
			• `SplitPlane` is "XY", "YZ", or "ZX"
			• `ToolType` is "PlaneTool"
			• `ToolEntityID` is -1

<table>
<tr><td></td><td><table><tr><td></td><td></td><td>For an edge:
• SplitPlane is "Dummy"
• ToolType is "EdgeTool"
• ToolEntityID is the face ID</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	Split(<*SelectionsArray*>, <*Parameters*>)

Python Example

```
oEditor.Split(
["NAME:Selections",
  "Selections:="          , "Box1",
  "NewPartsModelFlag:="   , "Model"
],
["NAME:SplitToParameters",
  "SplitPlane:="          , "XY",
  "WhichSide:="           , "PositiveOnly",
  "ToolType:="            , "PlaneTool",
  "ToolEntityID:="        , -1,
  "SplitCrossingObjectsOnly:=", False,
  "DeleteInvalidObjects:=", True
])
```

Stitch

Stitches selected sheets.

<table>
<tr><th>UI Access</th><th><b>Modeler</b> > <b>Model Preparation</b> > <b>Stitch Sheets</b>.</th></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td>&lt;<i>Selections</i>&gt;</td><td>Array</td><td>Structured array. See: SelectionsArray.</td></tr>
<tr><td>&lt;<i>Parameters</i>&gt;</td><td>Array</td><td>Structured array.
Array("NAME:StitchParameters",
  "MaxTol:=", <integer maximum tolerance&gt;)</td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	Stitch (<*Selections*>, <*Parameters*>)
Python Example	oEditor.Stitch(["NAME:Selections", "Selections:=", "Rectangle3,Rectangle4", "NewPartsModelFlag:=", "Model"], ["NAME:StitchParameters", "MaxTol:=", -1])

Subtract

Subtracts the specified object(s).

<table>
<tr><td><b>UI Access</b></td><td><b>Modeler</b> &gt; <b>Boolean</b> &gt; <b>Subtract</b>.</td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td><i>&lt;Selections&gt;</i></td><td>Array</td><td>Structured array.
<code>Array("NAME:Selections",</code>
<code> "Blank Parts:=" , <string object name(s)&gt;,</code>
<code> "Tool Parts:=" , <string object name(s)&gt;)</code>
The <code>Tool Parts</code> object is the object being removed.
The <code>Blank Parts</code> object has any <code>Tool Parts</code> overlap removed.
Either string can contain more than one object.</td></tr>
<tr><td><i>&lt;Parameters&gt;</i></td><td>Array</td><td>Structured array.
<code>Array("NAME:SubtractParameters",</code>
<code> "KeepOriginals:=" , <boolean&gt;)</code></td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

<table>
<tr><td><b>Python Syntax</b></td><td>Subtract(&lt;<i>Selections</i>&gt;, &lt;<i>Parameters</i>&gt;)</td></tr>
<tr><td><b>Python Example</b></td><td><code>oEditor.Subtract(</code>
<code>["NAME:Selections",</code>
<code> "Blank Parts:=", "Rectangle1",</code></td></tr>
</table>

	"Tool Parts:=", "Rectangle2"], ["NAME:SubtractParameters", "KeepOriginals:=", False])

SweepFacesAlongNormal

Sweep the specified face(s) along normal.

<table>
<tr><th>UI Access</th><th><b>Modeler</b> &gt; <b>Surface</b> &gt; <b>Sweep Faces Along Normal</b></th></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;SelectionsArray&gt;</i></td><td>Array</td><td>Structured array. See: SelectionsArray.</td></tr>
<tr><td><i>&lt;parameters&gt;</i></td><td>Array</td><td>Structured array.
Array("NAME:Parameters",
"NAME:SweepFaceAlongNormalToParameters",
"FacesToDetach:=", <faceIDarray>,
"LengthOfSweep:=", "<value><units&gt;")</td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>None</td></tr>
</table>

Python Syntax	SweepFacesAlongNormal(<*SelectionsArray*> <*parameters*>)
Python Example	oEditor.SweepFacesAlongNormal(["NAME:Selections",

	"Selections:=", "Rectangle1", "NewPartsModelFlag:=", "Model"], ["NAME:Parameters", "NAME:SweepFaceAlongNormalToParameters", "FacesToDetach:=", [183], "LengthOfSweep:=", "0.1mm"])

ThickenSheet

Thickens a sheet object to convert it to a 3D object.

<table>
<tr><th>UI Access</th><th><b>Modeler</b> &gt; <b>Surface</b> &gt; <b>Thicken Sheet</b></th></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;SelectionsArray&gt;</i></td><td>Array</td><td>Structured array. See: SelectionsArray.</td></tr>
<tr><td><i>&lt;parameters&gt;</i></td><td>Array</td><td>Structured array.
Array("NAME:SheetThickenParameters",
"Thickness:=" , <string>,
"BothSides:=" , <boolean&gt;)</td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	ThickenSheet (<*SelectionsArray*> <*parameters*>)

Python Example	`oEditor.ThickenSheet(`
	`["NAME:Selections",`
	`  "Selections:="           , "Rectangle3",`
	`  "NewPartsModelFlag:="    , "Model"`
	`],`
	`["NAME:SheetThickenParameters",`
	`  "Thickness:="            , "0.01mm",`
	`  "BothSides:="            , False`
	`])`

UncoverFaces

Uncovers the specified face(s).

UI Access	**Modeler** > **Surface** > **Uncover Faces**
Parameters	

Name	Type	Description
<*SelectionsArray*>	Array	Structured array. See: SelectionsArray.

<*parameters*>	Array	Structured array.
		`Array("NAME:Parameters",`
		`  Array("NAME:UncoverFacesParameters",`
		`    "FacesToUncover:=" , <array of face IDs>))`

Return Value	None.

Python Syntax	UncoverFaces(<*SelectionsArray*> <*parameters*>)

Python Example	`oEditor.UncoverFaces(`
	`["NAME:Selections",`
	`  "Selections:="          , "Box1",`
	`  "NewPartsModelFlag:="   , "Model"`
	`],`
	`["NAME:Parameters",`
	`  ["NAME:UncoverFacesParameters",`
	`    "FacesToUncover:="      , [12,16,18]`
	`  ]`
	`])`

Unite

Unites the specified objects.

<table>
<tr><td><b>UI Access</b></td><td><b>Modeler</b> &gt; <b>Boolean</b> &gt; <b>Unite</b></td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;<i>SelectionsArray</i>&gt;</td><td>Array</td><td>Structured array. See: SelectionsArray.</td></tr></table></td></tr>
</table>

<table>
<tr><td></td><td><table><tr><td><em>&lt;parameters&gt;</em></td><td>Array</td><td>Structured array.
<code>Array("NAME:UniteParameters",</code>
<code> "KeepOriginals:=" , <boolean&gt;,</code>
<code> "TurnOnNBodyBoolean:=", <boolean&gt;)</code>
<strong>Note:</strong>
TurnOnNBodyBoolean is set to True by default. Enabled by Parasolid, this unites <em>n</em> bodies in a single step, rather than uniting them one by one.</td></tr></table></td></tr>
<tr><td><strong>Return Value</strong></td><td>None.</td></tr>
</table>

Python Syntax	Unite(<*SelectionsArray*>, <*parameters*>)

Python Example	`oEditor.Unite(`
	`["NAME:Selections",`
	`  "Selections:=", "Rectangle1,Rectangle2"`
	`],`
	`["NAME:UniteParameters",`
	`  "KeepOriginals:=", False,`
	`  "TurnOnNBodyBoolean:=", True`
	`])`

Ungroup

Ungroups a specified history tree group.

<table>
<tr><td><b>UI Access</b></td><td><b>Modeler</b> > <b>Group</b> > <b>Ungroup</b></td></tr>
<tr><td><b>Parameters</b></td><td><table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i><Groups></i></td><td>Array</td><td>Structured array.
<code>Array("Groups:=", <array of group names to ungroup>)</code></td></tr>
</table></td></tr>
<tr><td><b>Return Value</b></td><td>None</td></tr>
</table>

Python Syntax	Ungroup(<*Groups*>)
Python Example	`oEditor.Ungroup(["Groups:=", ["Group1"]] )`

WrapSheet

Wraps a sheet object to another object.

<table>
<tr><td><b>UI Access</b></td><td>None.</td></tr>
<tr><td><b>Parameters</b></td><td><table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i><SelectionsArray></i></td><td>Array</td><td>Structured array. See: SelectionsArray.</td></tr>
<tr><td><i><Parameters></i></td><td>Array</td><td>Structured array.
<code>Array("NAME:WrapSheetParameters",</code>
<code> "Imprinted:=", <boolean>)</code></td></tr>
</table></td></tr>
</table>

Return Value	None.

Python Syntax	WrapSheet (<*SelectionsArray*>, <*Parameters*>)

Python Example	`oEditor.WrapSheet(`
	`["NAME:Selections",`
	`  "Selections:=","Rectangle1,Box1"`
	`],`
	`["NAME:WrapSheetParameters",`
	`  "Imprinted:=", True`
	`]`
	`)`

Other oEditor Commands

AddDefinitionFromBlock

AddDefinitionFromLibFile

BreakUDMConnection

ChangeProperty

Delete

FitAll

GetBodyNamesByPosition

GetChildNames [Modeler]

GetChildOjbect [Modeler]

GetChildTypes [Modeler]

GetEdgeByPosition

GetEdgeIDsFromObject

GetEdgeIDsFromFace

GetEntityListIDByName

GetExtendedDefinitionObject

GetFaceArea

GetFaceByPosition

GetFaceCenter

GetFaceIDs

GetGeometryModelerMode

GetMatchedObjectName

GetModelBoundingBox

GetModelUnits

GetNumObjects

GetObjectIDByName

GetObjectName

GetObjectNameByFaceID

AddDefinitionFromBlock

Adds a material definition from block text (same definition format as would be contained in the material library file) by library type (using definition folder name). This scripting command directly supports the .AMAT (or .ASURF) definition formats.

UI Access	N/A

<table>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td>&lt;<i>defBlock</i>&gt;</td><td>String</td><td>Text of the new material definition in block form.</td></tr>
<tr><td>&lt;<i>defFolderName</i>&gt;</td><td>String</td><td>Library type (by definition folder name)</td></tr>
<tr><td>&lt;<i>newTimeStamp</i>&gt;</td><td>String</td><td>New timestamp (time_t as integer number of seconds since 1/1/1970 12:00am, as string), default is current time</td></tr>
<tr><td>&lt;<i>replaceExisting</i>&gt;</td><td>Boolean</td><td>True to replace existing, False to choose a new unique name if an existing definition is found</td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>A property scripting object for the definition.</td></tr>
</table>

Python Syntax	AddDefinitionFromBlock(<*defBlock*>,<*defFolderName*>, <*newTimeStamp*>, <*replaceExisting*>)

Python Example	`oProject = oDesktop.NewProject()`
	`oProject.InsertDesign("HFSS", "HFSSDesign1", "DrivenModal", "")`
	`oDesign = oProject.SetActiveDesign("HFSSDesign1")`
	`oEditor = oDesign.SetActiveEditor("3D Modeler")`
	`oEditor.CreateBox(`

```
[

  "NAME:BoxParameters",

  "XPosition:="           , "-0.4mm",
                "YPosition:="           , "-1mm",
                "ZPosition:="           , "0mm",
                "XSize:="               , "1.4mm",
                "YSize:="               , "1.6mm",
                "ZSize:="               , "0.6mm"
        ],

[

  "NAME:Attributes",
                "Name:="                , "Box1",

  "Flags:="               , "",

  "Color:="               , "(143 175 143)",

  "Transparency:="        , 0,

  "PartCoordinateSystem:=", "Global",
    "UDMId:="               , "",
    "MaterialValue:="       , "\"vacuum\"",
    "SurfaceMaterialValue:=", "\"\"",
    "SolveInside:="         , True,
            "ShellElement:="        , False,
    "ShellElementThickness:=", "0mm",
    "IsMaterialEditable:="  , True,
    "UseMaterialAppearance:=", False,
    "IsLightweight:="       , False
  ])
oDefinitionManager = oProject.GetDefinitionManager()
defBlock = "$begin 'vacuum2' $begin 'AttachedData' $begin 'MatAppearanceData' property_data-
='appearance_data' Red=230 Green=230 Blue=230 Transparency=0.95 $end 'MatAppearanceData'
$end 'AttachedData' simple('permittivity', 1) ModTime=1499970477 $end 'vacuum2'"
added = oDefinitionManager.AddDefinitionFromBlock(defBlock, "Materials", "10101010", True)
```

```
addedName = ''
  if isinstance(added, basestring):
    addedName = added
  elif isinstance(added, list):
    addedName = added[0]
else:

  addedName = added.GetName().replace("Materials:", "")
AddInfoMessage(os.path.basename(__file__) + " result: " + addedName)
materialNameInQuotes = "\"" + addedName + "\""
oEditor.ChangeProperty(
          [

     "NAME:AllTabs",

     [

       "NAME:Geometry3DAttributeTab",

       [

         "NAME:PropServers",

         "Box1"
```

```
        ],

        [

          "NAME:ChangedProps",

          [

          "NAME:Material",

          "Value:=", materialNameInQuotes

          ]
                             ]
                  ]
         ])
```

AddDefinitionFromLibFile

Adds a material definition from a library file (e.g. AMAT file), by name and library type (using definition folder name) . This scripting command directly supports the .AMAT (or .ASURF) definition formats.

UI Access	N/A		
Parameters	Name	Type	Description

<*FilePath*>	String	Path of the library file (i.e. AMAT file or ASURF file)
<*defName*>	String	Which definition to use, required because a lib file can have multiple definitions.
<*defFolderName*>	String	Library type (by definition folder name).
<*newTimeStamp*>	String	New timestamp string (time_t as integer, number of seconds since 1/1/1970 12:00am), default is current time
<*replaceExisting*>	Boolean	True to replace existing, False to choose a new unique name if an existing definition is found, default is False

Return Value	Property scripting object for the definition.

Python Syntax	AddDefinitionFromLibFile(<*FilePath*>, <*defName*>, <*defFolderName*>, <*newTimeStamp*>,<*replaceExisting*>)

Python Example

```
oProject = oDesktop.NewProject()

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

oDesign = oProject.SetActiveDesign("HFSSDesign1")

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

oEditor.CreateBox(

  [

    "NAME:BoxParameters",

    "XPosition:="           , "-0.4mm",
                    "YPosition:="           , "-1mm",
                  "ZPosition:="           , "0mm",
                  "XSize:="               , "1.4mm",
                  "YSize:="               , "1.6mm",
                  "ZSize:="               , "0.6mm"
        ],

[

  "NAME:Attributes",
                  "Name:="                , "Box1",

  "Flags:="               , "",

  "Color:="               , "(143 175 143)",

  "Transparency:="        , 0,

  "PartCoordinateSystem:=", "Global",

  "UDMId:="               , "",

  "MaterialValue:="       , "\"vacuum\"",
    "SurfaceMaterialValue:=", "\"\"",

    "SolveInside:="         , True,

            "ShellElement:="        , False,

    "ShellElementThickness:=", "0mm",

    "IsMaterialEditable:="  , True,

    "UseMaterialAppearance:=", False,

    "IsLightweight:="       , False

  ])
oDefinitionManager = oProject.GetDefinitionManager()
scriptDir = os.path.dirname(os.path.realpath(__file__))
amatFilePath = scriptDir + '/material0.txt'
materialName = "material0"
materialNameInQuotes = "\"" + materialName + "\""
added = oDefinitionManager.AddDefinitionFromLibFile(amatFilePath, materialName, "Materials", "")
addedName = ''
  if isinstance(added, basestring):
```

```
addedName = added
  elif isinstance(added, list):
addedName = added[0]
else:
  addedName = added.GetName().replace("Materials:", "")
AddInfoMessage(os.path.basename(__file__) + " result: " + addedName)
oEditor.ChangeProperty(
  [
    "NAME:AllTabs",
    [
      "NAME:Geometry3DAttributeTab",
      [
        "NAME:PropServers",
        "Box1"
      ],
      [
      "NAME:ChangedProps",
      [
        "NAME:Material",
        "Value:=", materialNameInQuotes
```

<table>
<tr><td></td><td><pre>
            ]
        ]
    ]
])</pre></td></tr>
</table>

AssignSurfaceMaterial

Assigns a material to specified surfaces.

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;SelectionsArray&gt;</i></td><td>Array</td><td>Structured array. See: SelectionsArray.</td></tr>
<tr><td><i>&lt;AttributesArray&gt;</i></td><td>Array</td><td>Structured array. See: AttributesArray.
This script supports the following attributes:<ul><li>MaterialValue</li><li>SolveInside</li><li>ShellElement</li><li>ShellElementThickness</li><li>IsMaterialEditable</li><li>UseMaterialAppearance</li><li>IsLightweight</li></ul></td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	AssignSurfaceMaterial(<*SelectionsArray*>, <*AttributesArray*>)

Python Example	`oEditor.AssignSurfaceMaterial(`
	`["NAME:Selections",`
	`"AllowRegionDependentPartSelectionForPMLCreation:=", True,`
	`"AllowRegionSelectionForPMLCreation:=", True,`
	`"Selections:=" , "Rectangle1"`
	`],`
	`["NAME:Attributes",`
	`"MaterialValue:=" , "diamond",`
	`"SolveInside:=" , False,`
	`"ShellElement:=" , False,`
	`"ShellElementThickness:=" , "nan",`
	`"IsMaterialEditable:=" , True,`
	`"UseMaterialAppearance:=" , False,`
	`"IsLightweight:=" , False`
	`])`

BreakUDMConnection

Breaks a current UDM connection to Discovery.

UI Access	Break Connection.

<table>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;<i>SelectionsArray</i>&gt;</td><td>Array</td><td>Structured array. See: SelectionsArray.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

<table>
<tr><td><b>Python Syntax</b></td><td>BreakUDMConnection (&lt;<i>SelectionsArray</i>&gt;)</td></tr>
<tr><td><b>Python Example</b></td><td>

```
oEditor.BreakUDMConnection(
              [
               "NAME:Selections",
               "Selections:="        , "Discovery1"
              ])
```

</td></tr>
</table>

ChangeProperty

Changes the properties of an object in the history tree.

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

Python Syntax	ChangeProperty(<*propertyArgs*>)

Python Example

**Example: Changing the Position of a Box
and the Reference Temperature (for Structural Solutions only)**

```
oEditor.ChangeProperty(
[
    "NAME:AllTabs",
    [
        "NAME:Geometry3DCmdTab",
        [
            "NAME:PropServers" ,
            "Box1:CreateBox:1"
        ],
        [
            "NAME:ChangedProps",
            [
                "NAME:Position",
                "X:="          , "0.35in",
                "Y:="          , "0.55in",
                "Z:="          , "0in"
            ],
            [
                "NAME:Reference Temperature",  # Applicable only to Mechanical-
                "Value:="      , "27cel"       # Structural solutions
            ]
        ]
    ]
])
```

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

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

```
         ]
       ]
     ]
  ])
```

Example: Changing a Box's Material and Wireframe Display

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

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

```
oEditor.ChangeProperty(

  [
     "NAME:AllTabs",
     [
       "NAME:General",
       [
         "NAME:PropServers",
         "LC1_1"
       ],
       [
         "NAME:ChangedProps",
         [
           "NAME:Handle",
           "Value:="             , " NotchOutY"
         ]
       ]
     ]
  ])


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

```
])
...
```

Example: Change Material Import for a Discovery Model

```
oEditor = oDesign.SetActiveEditor("3D Modeler")
oEditor.ChangeProperty(
   [
     "NAME:AllTabs",
     [
       "NAME:Options",
       [
         "NAME:PropServers",
         "Discovery1"
       ],
       [
         "NAME:ChangedProps",
         [
            "NAME:Materials",
            "Value:="    , "Assignments and Properties"
         ]
       ]
     ]
   ])
```

CloseAllWindows[Editor]

Closes all windows belong to current 3D Modeler editor.

UI Access	N/A
Parameters	None.

Return Value	None.

Python Syntax	CloseAllWindows()
Python Example	`oEditor.CloseAllWindows()`

Defeature

Removes irrelevant features from a primitive.

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td>&lt;<i>Selections</i>&gt;</td><td>Array</td><td>Structured array. See: SelectionsArray.</td></tr>
<tr><td>&lt;<i>Options</i>&gt;</td><td>Array</td><td>Structured array.
<code>Array("NAME:options",</code>
<code>"tolerance:=", <double containing tolerance value&gt;,</code>
<code>"fix:=", <boolean, if true, then self-intersections are fixed.&gt;)</code></td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

<table>
<tr><td><b>Python Syntax</b></td><td>Defeature(&lt;<i>Selections</i>&gt;, &lt;<i>Options</i>&gt;)</td></tr>
<tr><td><b>Python Example</b></td><td><code>oEditor.Defeature(</code>
<code>   ["NAME:Selections", "Selections:=", "Box1"],</code></td></tr>
</table>

	`["NAME:Options", "Tolerance:=", 1E-006, "Fix:=", True])`

Delete

Deletes the specified object(s).

<table>
<tr><td><b>UI Access</b></td><td><b>Edit</b> &gt; <b>Delete</b>.</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;<i>SelectionsArray</i>&gt;</td><td>Array</td><td>Structured array. See: SelectionsArray.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	Delete(<*SelectionsArray*>)

Python Example	`oEditor.Delete(`
	`["NAME:Selections",`
	`"Selections:=", "Rectangle1,Rectangle2"`
	`])`

FitAll

Fits the design to the modeling area.

UI Access	**View** > **Fit All** > **All Views**.

Parameters	None.
Return Value	None.

Python Syntax	FitAll()
Python Example	`oEditor.FitAll()`

GenerateAllUserDefinedModels

Generates all user defined models.

UI Access	N/A
Parameters	None.
Return Value	Array of models generated.

Python Syntax	GenerateAllUserDefinedModels ()
Python Example	`oEditor.GenerateAllUserDefinedModels()`

GenerateUserDefinedModel

Generates specified user defined model(s).

UI Access	N/A

<table>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><i>&lt;SelectionsArray&gt;</i></td><td>Array</td><td>Structured array. See: SelectionsArray.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>Array of models generated.</td></tr>
</table>

Python Syntax	GenerateUserDefinedModel (<*SelectionsArray*>)
Python Example	`oEditor.GenerateUserDefinedModel(["NAME:Selections", "Selections:=", "model1, model2"])`

GeometryCheckAndAutofix

Use: Runs Geometry Check and optionally applies autofixes.

Command: **HFSS 3D Layout > Geometry Check**

Syntax: GeometryCheckAndAutofix <ChecksArray>,

minimum_area_meters_squared,

<FixesArray>

Return Value: None

Parameters: **<ChecksArray> - Array("NAME:checks", <check 1>, <check 2>, …, <check n>)**

Specify the checks that should be included. Specifying fewer checks may speed up execution but may also result in less problems reported in the message manager (or the logfile) and consequently less problems that can be fixed by autofixes.

The following are valid checks that can be specified:

- "Self-Intersecting Polygons"
- "Disjoint Nets (Floating Nodes)"
- "DC-Short Errors"
- "Identical/Overlapping Vias"
- "Misaligments"

There may be no checks, all 5 of the checks, or anything in between. The order that checks are specified in is not relevant.

minimum_area_meters_squared

Specify a decimal value for the minimum area (e.g. .000015) optionally in scientific notation (e.g. 2E-006). Cutouts smaller than this minimum area may be ignored during validation check.

<FixesArray> - Array("NAME:fixes", <fix 1>, <fix 2>, …, <fix n>)

Specify the autofixes that should be applied if they are found by a check.

The following are valid fixes that can be specified:

- "Self-Intersecting Polygons"
- "Disjoint Nets",
- "Identical/Overlapping Vias"
- "Traces-Inside-Traces Errors"
- "Misalignments (Planes/Traces/Vias)"

There may be no fixes specified, all 5 fixes specified, or anything in between. The order that fixes are specified in is not relevant.

GetBodyNamesByPosition

Returns the names of objects that contact a specified point.

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td>&lt;<i>positionParameters</i>&gt;</td><td>Array</td><td>Structured array containing position coordinates for active coordinate system.
Array("NAME:Parameters",
"XPosition:=", <string>,
"YPosition:=", <string>,
"ZPosition:=", <string&gt;)</td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>Array containing string object names.</td></tr>
</table>

<table>
<tr><td><b>Python Syntax</b></td><td>GetBodyNamesByPosition (&lt;<i>positionParameters</i>&gt;)</td></tr>
<tr><td><b>Python Example</b></td><td>oEditor.GetBodyNamesByPosition(
["NAME:Parameters",
"XPosition:=","0mm",
"YPosition:=","15mm",
"ZPosition:=","0mm"
]</td></tr>
</table>

	)

GetChildNames [Modeler]

Returns the names of children for a specified input. For 3D Components and UDMs, these commands do not return parts, coordinate systems, plans, as top-level modeler children.

> **Note:**
>
> This command is not supported by the EMIT and Circuit design types.

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;category&gt;</i></td><td>String</td><td><i>Optional</i>. Passing no input returns the list of possible strings:
<ul>
<li>"AllParts" – Returns the names of all parts.</li>
<li>"CoordinateSystems" – Returns the names of all coordinate systems.</li>
<li>"Groups" – Returns the names of all groups.</li>
<li>"Lists" – Returns the names of all lists.</li>
<li>"ModelParts" – Returns names of model parts.</li>
<li>"NonModelParts" – Returns the names of non-model parts.</li>
<li>"Planes" – Returns the names of all planes.</li>
<li>"Points" – Returns the names of all points.</li>
<li>"SubmodelDefinitions" – Returns the names of sub-model definitions.</li>
</ul></td></tr>
</table>
</td></tr>
</table>

Return Value	Array containing object names in the selected category.

Python Syntax	GetChildNames(<*category*>)

Python Example	**Standalone Example:**
	`oEditor.GetChildNames("ModelParts")`
	`oEditor.GetChildNames("Points")`
	Example used in Conjunction with GetChildObject and GetPropNames:
	`oDesign = oProject.GetActiveDesign()`
	`oModel = oDesign.GetChildObject("3D Modeler")`
	`oModel.GetChildTypes()`
	• This returns an array containing strings: "ModelParts", "AllParts", "NonModelParts", "CoordinateSystems", "Points", "Planes", "SubmodelDefinitions", "Groups", and "Lists".
	`oModel.GetChildNames("ModelParts")`
	• This returns an array containing string model parts. For example: "WG_Interior", "WG", "MT_Interior", "MT", "Box2", "Cylinder1".
	`oWGi = oModel.GetChildObject("WG_Interior")`
	• This sets the object WG_Interior to variable oWGi.
	`oWGi.GetPropNames()`
	• This returns property names from child object assigned to oWGi. In this case: "Name", "Material", "Material/SIValue", "Material/EvaluatedValue", "Solve Inside", "Orientation", "Orientation/Choices", "Model", "Dis-

	play Wireframe", "Material Appearance", "Color", "Color/Red", "Color/Green", "Color/Blue", and "Transparent".

GetChildObject [Modeler]

Returns a 3D modeler child object, which can be assigned to a variable. Will return normally if there are no active objects. For 3D Components and UDMs, these commands do not return parts, coordinate systems, plans, as top-level modeler children.

> **Note:** This command is not supported by the EMIT and Circuit design types.

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;object&gt;</i></td><td>String</td><td>In this case, "3D Modeler".</td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>Returns the 3D Modeler object. In the examples below, it is assigned to the variable oEditor.</td></tr>
</table>

Python Syntax	GetChildObject(<object>)

Python Example	**Standalone Example:**
	`oDesign = oProject.GetActiveDesign()`
	`oEditor = oDesign.GetChildObject("3D Modeler")`
	Example used in Conjunction with GetChildNames and GetPropNames:
	`oDesign = oProject.GetActiveDesign()`
	`oModel = oDesign.GetChildObject("3D Modeler")`

	`oModel.GetChildTypes()`
	• This returns an array containing strings: "ModelParts", "AllParts", "NonModelParts", "CoordinateSystems", "Points", "Planes", "SubmodelDefinitions", "Groups", and "Lists".
	`oModel.GetChildNames("ModelParts")`
	• This returns an array containing string model parts. For example: "WG_Interior", "WG", "MT_Interior", "MT", "Box2", "Cylinder1".
	`oWGi = oModel.GetChildObject("WG_Interior")`
	• This sets the object WG_Interior to variable oWGi.
	`oWGi.GetPropNames()`
	• This returns property names from child object assigned to oWGi. In this case: "Name", "Material", "Material/SIValue", "Material/EvaluatedValue", "Solve Inside", "Orientation", "Orientation/Choices", "Model", "Display Wireframe", "Material Appearance", "Color", "Color/Red", "Color/Green", "Color/Blue", and "Transparent".

GetChildTypes [Modeler]

Gets child types of queried designs or editors obtained by GetActiveProject() and GetActiveDesign() commands. For 3D Components and UDMs, these commands do not return parts, coordinate systems, plans, as top-level modeler children.

> **Note:**
>
> This command is not supported by the EMIT and Circuit design types.

UI Access	N/A

Parameters	None.
Return Value	Array containing the names of child types.

Python Syntax	GetChildTypes()

Python Example	**Standalone Example:**
	`oDesign = oProject.GetActiveDesign()`
	`oEditor = oDesign.SetActiveEditor("3D Modeler")`
	`oEditor.GetChildTypes()`
	• This returns an array containing strings: "Part", "CoordinateSystem", "Group", "ComponentDefinition", "UserDefinedModel", "Point", "Plane", and "List".
	Note: Users can query the Part type to see if it is a model or nonmodel.
	Example Used in Conjunction with GetChildNames:
	`oProject = oDesktop.GetActiveProject()`
	`oDesign = oProject.GetActiveDesign()`
	`oDesign.GetChildNames()`
	• This returns an array containing names of child objects for the design. For example: "Boundaries", "Nets", "Analysis", "Optimetrics", "Radiation", "Results", and "3D Modeler".
	`oDesign.GetChildTypes()`
	• This returns an array containing the child types. For example: "Module", "Editor", and "Variable".

GetEdgeByPosition

Returns the ID for edge(s) that contact a specified point.

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td><i>&lt;positionParameters&gt;</i></td><td>Array</td><td>Structured array.
Array("NAME:EdgeParameters",
  "BodyName:=", <string object name>,
  "Xposition:=", <string>,
  "YPosition:=", <string>,
  "ZPosition:=", <string>)
<b>Note:</b>
For 2D XY Designs, ZPosition should be set to "0".
For 2D RZ Designs, YPosition should be set to "0".</td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>Array containing string edge IDs.</td></tr>
</table>

Python Syntax	GetEdgeByPosition (<*positionParameters*>)
Python Example	oEditor.GetEdgeByPosition(["NAME:EdgeParameters",

```
     "BodyName:=", "Box1",
     "Xposition:=", "10mm",
     "YPosition:=", "0mm",
     "ZPosition:=", "10mm"
   ]
)
```

GetEdgeIDFromNameForFirstOperation

Gets edge ID from first operation of a part.

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td><table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;PartName&gt;</i></td><td>String</td><td>Name of specified part.</td></tr>
<tr><td><i>&lt;EdgeName&gt;</i></td><td>String</td><td>Name of specified edge.</td></tr>
</table></td></tr>
<tr><td><b>Return Value</b></td><td>Integer edge ID</td></tr>
</table>

Python Syntax	GetEdgeIDFromNameForFirstOperation(<*PartName*>, <*EdgeName*>)
Python Example	`oEditor.GetEdgeIDFromNameForFirstOperation("Coil_1", "Edge_4510")`

GetEdgeIDsFromFace

Returns the edge IDs for a specified Face ID.

UI Access	N/A
Parameters	<table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><i>&lt;faceID&gt;</i></td><td>Integer</td><td>ID of the specified face.</td></tr></table>
Return Value	Array containing string edge IDs.

Python Syntax	GetEdgeIDsFromFace (<*faceID*>)
Python Example	`oEditor.GetEdgeIDsFromFace(20)`

GetEdgeIDsFromObject

Returns the edge IDs for a specified object.

UI Access	N/A
Parameters	<table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><i>&lt;object&gt;</i></td><td>String</td><td>Object name.</td></tr></table>
Return Value	Array containing string edge IDs.

Python Syntax	GetEdgeIDsFromObject (<*object*>)
Python Example	`oEditor.GetEdgeIDsFromObject("Box1")`

GetEdgeLength

Returns the length of edges for a specified Edge ID.

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><i>&lt;edgeID&gt;</i></td><td>Integer</td><td>ID of the specified edge.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>Integer containing the length of found edges.</td></tr>
</table>

Python Syntax	GetEdgeLength (<*EdgeID*>)
Python Example	`oEditor.GetEdgeLength(20)`

GetEntityIDsContainedByNamedSelection

Returns the list of entity IDs contained in the named selection. See CreateNamedSelection.

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><i>&lt;selectionName&gt;</i></td><td>String</td><td>selection name.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>String containing the list of entity IDs contained in the named selection.</td></tr>
</table>

Python Syntax	GetEntityIDsContainedByNamedSelection (<*selectionName*>)
Python Example	`entityIDs = oEditor.GetEntityIDsContainedByNamedSelection("FaceSelection1")`

GetEntityListIDByName

Returns the specified entity list's ID number. See CreateNamedSelection.

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><i>&lt;listName&gt;</i></td><td>String</td><td>List name.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>String containing the entity list ID number.</td></tr>
</table>

Python Syntax	GetEntityListIDByName (<*listName*>)
Python Example	`oEditor.GetEntityListIDByName("MyList")`

GetExtendedDefinitionObject

Get an object-oriented property scripting object by name and library type (using definition folder name) which also supports getting/setting mod time and getting/setting generic string attributes. This scripting command directly supports the .AMAT (or .ASURF) definition formats.

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><i>&lt;defName&gt;</i></td><td>String</td><td>Definition to retrieve.</td></tr></table></td></tr>
</table>

<table>
<tr><td></td><td><table><tr><td><i>&lt;defFolderName&gt;</i></td><td>String</td><td>Library type (by definition folder name).</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>An extended material wrapper script object (see material wrapper script object functions above).</td></tr>
</table>

GetFaceArea

Returns the area of a specified face.

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><i>&lt;faceID&gt;</i></td><td>Integer</td><td>ID of specified face.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>Long integer of face area.</td></tr>
</table>

Python Syntax	GetFaceArea (<*faceID*>)
Python Example	`oEditor.GetFaceArea(19)`

GetFaceCenter

Returns the center position of a specified face.

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><i>&lt;planarFaceID&gt;</i></td><td>Long</td><td>Face ID</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>Array containing string X, Y, Z coordinates.</td></tr>
</table>

Python Syntax	GetFaceCenter (<*planarFaceID*>)
Python Example	`oEditor.GetFaceCenter(12)`

GetFaceByPosition

Returns the face ID located at a specified position.

> **Note:**
>
> The coordinates must point to exactly one face, not a vertex or edge where two or more faces join.

UI Access	N/A
Parameters	<table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;<i>FaceParameters</i>&gt;</td><td>Array</td><td>Structured Array. <code>Array("NAME:FaceParameters",</code> <code>"BodyName:=", <string&gt;,</code> <code>"XPosition:=", <string&gt;,</code> <code>"YPosition:=", <string&gt;,</code> <code>"ZPosition:=", <string&gt;)</code></td></tr></table>
Return Value	Integer Face ID.

Python Syntax	GetFaceByPosition(<*FaceParameters*>)

Python Example	`oEditor.GetFaceByPosition(`
	`["NAME:FaceParameters",`
	`  "BodyName:=", "Box1",`
	`  "XPosition:=", "0.2mm",`
	`  "YPosition:=", "-0.2mm",`
	`  "ZPosition:=", "0.4mm"`
	`])`

GetFaceIDFromNameForFirstOperation

Gets face ID from first operation of a part.

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td><table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td>&lt;<i>PartName</i>&gt;</td><td>String</td><td>Name of specified part.</td></tr>
<tr><td>&lt;<i>FaceName</i>&gt;</td><td>String</td><td>Name of specified face.</td></tr>
</table></td></tr>
<tr><td><b>Return Value</b></td><td>Integer face ID</td></tr>
</table>

Python Syntax	GetFaceIDFromNameForFirstOperation(<*PartName*>, <*FaceName*>)
Python Example	`oEditor.GetFaceIDFromNameForFirstOperation("Rotor", "Face_14343"))`

GetFaceIDs

Returns the face IDs associated with a specified object.

UI Access	N/A
Parameters	<table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;*objectName*&gt;</td><td>String</td><td>Object name.</td></tr></table>
Return Value	Array containing string face IDs.

Python Syntax	GetFaceIDs (<*objectName*>)
Python Example	`oEditor.GetFaceIDs('Box1')`

GetFaceIDsOfSheet

Returns the face IDs associated with a specified sheet object.

UI Access	N/A
Parameters	<table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;*sheetName*&gt;</td><td>String</td><td>Name of specified sheet object.</td></tr></table>
Return Value	Array containing string face IDs.

Python Syntax	GetFaceIDsOfSheet(<*sheetName*>)
Python Example	`oEditor.GetFaceIDsOfSheet('Sheet1')`

GetMatchedObjectName

Returns all object names containing the input text string.

UI Access	N/A
Parameters	Name | Type | Description <*wildcardText*> | String | Text string present in object name(s).
Return Value	Array containing string object names.

Python Syntax	GetMatchedObjectName (<*wildcardText*>)

Python Example	`oEditor.GetMatchedObjectName('Box*')`
	`oEditor.GetMatchedObjectName('?ox?')`

GetModelBoundingBox

Returns the bounding box of the current model using the global coordinate system as reference. The outputs are in the defined modeler units. Only modeled objects are considered for this computation (non-modeled objects are ignored).

UI Access	N/A
Parameters	None.

Return Value	Array containing string Xmin, Ymin, Zmin, Xmax, Ymax, and Zmax values of the bounding box in the global coordinate system

Python Syntax	GetModelBoundingBox()
Python Example	`oEditor.GetModelBoundingBox()`

GetModelUnits

Returns the model's unit of measure.

UI Access	N/A
Parameters	None.
Return Value	String containing unit of measure.

Python Syntax	GetModelUnits()
Python Example	`oEditor.GetModelUnits()`

GetNumObjects

Returns the number of objects in a design.

UI Access	N/A
Parameters	None.
Return Value	Integer number of objects.

Python Syntax	GetNumObjects()
Python Example	`oEditor.GetNumObjects()`

GetObjectIDByName

Given an object's name, returns its ID. IDs are used with CreateEntityList.

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><i>&lt;objectName&gt;</i></td><td>String</td><td>Object name.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>Integer object ID.</td></tr>
</table>

Python Syntax	GetObjectIDByName(<*objectName*>)
Python Example	`oEditor.GetObjectIDByName('Box1')`

GetObjectName

Returns an object's name from its specified base index (creation order).

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;<i>index</i>&gt;</td><td>Integer</td><td>Base index (where '0' is the first item created)</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>String containing object name.</td></tr>
</table>

Python Syntax	GetObjectName(<*index*>)
Python Example	`oEditor.GetObjectName(3)`

GetObjectNameByEdgeID

Returns an object name given an edge ID.

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;<i>EdgeID</i>&gt;</td><td>Integer</td><td>The edge ID.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>String containing object name.</td></tr>
</table>

Python Syntax	GetObjectNameByEdgeID(<*EdgeID*>)
Python Example	`oEditor.GetObjectNameByEdgeID(88)`

GetObjectNameByFaceID

Returns an object name given a face ID.

UI Access	N/A
Parameters	Name | Type | Description *<FaceID>* | Integer | The face ID.
Return Value	String containing object name.

Python Syntax	GetObjectNameByFaceID (<*FaceID*>)
Python Example	`oEditor.GetObjectNameByFaceID(88)`

GetObjectNameByID

Returns an object name given an ID.

UI Access	N/A
Parameters	Name | Type | Description *<ObjID>* | Integer | The object ID.
Return Value	String containing object name.

Python Syntax	GetObjectNameByID(<*ObjID*>)
Python Example	`oEditor.GetObjectNameByID(88)`

GetObjectNameByVertexID

Returns an object name given a vertex ID.

UI Access	N/A
Parameters	Name | Type | Description *<VertexID>* | Integer | The vertex ID.
Return Value	String containing object name.

Python Syntax	GetObjectNameByVertexID(<*VertexID*>)
Python Example	`oEditor.GetObjectNameByVertexID(88)`

GetObjectsByMaterial

Returns a list of objects of a specified material.

UI Access	N/A
Parameters	Name | Type | Description *<material>* | String | Material name.
Return Value	Array containing string object names.

Python Syntax	GetObjectsByMaterial (<*material*>)
Python Example	`oEditor.GetObjectsByMaterial('copper')`

GetObjectShapeType

Returns the shape type of a specified object.

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td><table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td>&lt;<i>objName</i>&gt;</td><td>String</td><td>Name of specified object.</td></tr>
<tr><td>&lt;csName&gt;</td><td>String</td><td>Name of coordinate system of the object.</td></tr>
</table></td></tr>
<tr><td><b>Return Value</b></td><td>String containing shape type.</td></tr>
</table>

Python Syntax	GetObjectShapeType(<*objName*>, <*csName*>)
Python Example	`oEditor.GetObjectShapeType("box1", "Global")`

GetObjectsInGroup

Returns a list of objects in a specified group.

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td><table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td>&lt;<i>group</i>&gt;</td><td>String</td><td>Group name.
One of:</td></tr>
</table></td></tr>
</table>

<table>
<tr><td></td><td><table><tr><td></td><td></td><td>• "<<i>materialName</i>>"
• "<<i>assignmentName</i>>"
• "Non Model"
• "Solids"
• "Unclassified"
• "Sheets"
• "Lines"</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>Array containing string object names.</td></tr>
</table>

Python Syntax	GetObjectsInGroup (<*group*>)
Python Example	`oEditor.GetObjectsInGroup('Sheets')`

GetObjectVolume

Returns an object's volume from its name).

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><<i>name</i>></td><td>String</td><td>Object name</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>.Real</td></tr>
</table>

Python Syntax	GetObjectVolume(<*name*>)
Python Example	`oEditor.GetObjectVolume("Box1")`

GetObjPath [Editor]

Obtains the path to the 3D modeler.

UI Access	N/A
Parameters	None.
Return Value	String containing the path.

Python Syntax	GetObjPath()
Python Example	`oEditor.GetObjPath()`

GetPartsForUserDefinedModel

Obtains parts from a specified user defined model.

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><i>&lt;udmName&gt;</i></td><td>String</td><td>Name of user defined model.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>Array of strings containing associated parts.</td></tr>
</table>

Python Syntax	GetPartsForUserDefinedModel (<*udmName*>)
Python Example	`oEditor.GetPartsForUserDefinedModel("OnDieSpiralInductor1")`

GetPoints [3D Modeler Editor]

Returns all the points defined in current 3D modeler.

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

Python Syntax	GetPoints ()
Python Example	`oEditor.GetPoints()`

GetPropEvaluatedValue

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

> **Note:**
>
> This command is not supported by the EMIT and Circuit design types.

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><i>&lt;PropName&gt;</i></td><td>String</td><td>Name of the property.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>String value of the evaluated value.</td></tr>
</table>

Python Syntax	GetPropEvaluatedValue (<*PropName*>)

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

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.

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><i>&lt;PropTab&gt;</i></td><td>String</td><td>One of the following, where tab titles are shown in parentheses:
• PassedParameterTab ("Parameter Values")
• DefinitionParameterTab (Parameter Defaults")</td></tr></table></td></tr>
</table>

		• LocalVariableTab ("Variables" or "Local Variables") • ProjectVariableTab ("Project variables") • ConstantsTab ("Constants") • BaseElementTab ("Symbol" or "Footprint") • ComponentTab ("General") • Component("Component") • CustomTab ("Intrinsic Variables") • Quantities ("Quantities") • Signals ("Signals")
<PropServer>	String	An object identifier, generally returned from another script method, such as `CompInst@R;2;3`
<PropName>	String	Name of the property.

Return Value	String value of the property.

Python Syntax	GetPropertyValue (<*PropTab*>, <*PropServer*>, <*PropName*>)

Python Example	`selectionArray = oEditor.GetSelections()`
	`for k in selectionArray:`
	`val = oEditor.GetPropertyValue("PassedParameterTab", k, "R")`
	`...`

GetPropNames [Modeler]

Returns the property names for the active model object or specified property values.

> **Note:**
>
> This command is not supported by the EMIT and Circuit design types.

UI Access	N/A
Parameters	<table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;<i>IncludeReadOnly</i>&gt;</td><td>Boolean</td><td>Optional. • <b>True -</b> includes all properties. • <b>False -</b> returns only property names that can be changed. True by default.</td></tr></table>
Return Value	Returns property names of the current 3D Model object; includes the following: ["Name", "Material", "Solve Inside", "Orientation", "Orientation/Choices", "Model", "Group", "Display Wireframe", "Material Appearance", "Color", "Color/Red", "Color/Green", "Color/Blue", "Transparent"]

Python Syntax	GetPropNames(<*IncludeReadOnly*>)
Python Example	`oEditor.GetPropNames(ObjectName)`

GetPropSIValue

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

> **Note:**
>
> This command is not supported by the EMIT and Circuit design types.

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><i>&lt;PropName&gt;</i></td><td>String</td><td>Name of the property.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>Property value as a double floating value, or NAN if the property value cannot be converted to double floating point.</td></tr>
</table>

Python Syntax	GetPropSIValue (<*PropName*>)

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

GetPropValue [Modeler]

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

> **Note:**
>
> This command is not supported by the EMIT and Circuit design types.

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><i>&lt;propPath&gt;</i></td><td></td><td>Optional. Child object's property path. See: Object Property Function Summary.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>The property value.</td></tr>
</table>

Python Syntax	GetPropValue(<*propPath*>)

Python Example	`Rh1 = oDesign.GetChildObject('Box1')`
	`Rh1.GetPropValue('Orientation')`
	Returns the specified Property Value:
	`'Global'`

GetRelativeCoordinateSystems

Returns the relative coordinate systems in the current 3D modeler.

UI Access	N/A
Parameters	None.
Return Value	Array containing name of relative coordinate systems.

Python Syntax	GetRelativeCoordinateSystems()
Python Example	`oEditor.GetRelativeCoordinateSystems()`

GetSelections [Model Editor]

Returns an array of currently selected objects.

UI Access	N/A
Parameters	None.
Return Value	Array containing object IDs

Python Syntax	GetSelections()
Python Example	`oEditor.GetSelections()`

GetSubGroupsInGroup

Returns subgroup names in a specified group.

UI Access	N/A
Parameters	<table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>*<group>*</td><td>String</td><td>Group name. One of: • "*<materialName>*" • "*<assignmentName>*" • "Non Model" • "Solids" • "Unclassified" • "Sheets" • "Lines"</td></tr></table>
Return Value	Array containing subgroup names.

Python Syntax	GetSubGroupsInGroup(<*group*>)
Python Example	`oEditor.GetSubGroupsInGroup('Sheets')`

GetUserPosition

Returns a user's current coordinates in the 3D Modeler window.

UI Access	N/A
Parameters	<table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>*<prompt>*</td><td>String</td><td>"Enter a point." Then click a point in the 3D Modeler window.</td></tr></table>

Return Value	Array containing X, Y, Z coordinates.

Python Syntax	GetUserPosition(<*prompt*>)
Python Example	`oEditor.GetUserPosition("Enter a point.")`

GetVertexIDFromNameForFirstOperation

Returns vertex ID associated with a specified vertex belongs to the first operation of a part.

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td><table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;PartName&gt;</i></td><td>String</td><td>Name of specified part.</td></tr>
<tr><td><i>&lt;VertexName&gt;</i></td><td>String</td><td>Name of specified vertex.</td></tr>
</table></td></tr>
<tr><td><b>Return Value</b></td><td>Integer representing vertex ID.</td></tr>
</table>

Python Syntax	GetVertexIDFromNameForFirstOperation(<*PartName*>, <*VertexName*>)
Python Example	`oEditor.GetVertexIDFromNameForFirstOperation("Part1", "Vertex1")`

GetVertexIDsFromEdge

Returns vertex IDs associated with a specified edge.

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><i>&lt;edgeID&gt;</i></td><td>Integer</td><td>Edge ID.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>Array containing string vertex IDs.</td></tr>
</table>

Python Syntax	GetVertexIDsFromEdge(<*edgeID*>)
Python Example	`oEditor.GetVertexIDsFromEdge(10)`

GetVertexIDsFromFace

Returns vertex IDs associated with a specified face.

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><i>&lt;faceID&gt;</i></td><td>Integer</td><td>Face ID.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>Array containing string vertex IDs.</td></tr>
</table>

Python Syntax	GetVertexIDsFromFace(<*faceID*>)
Python Example	`oEditor.GetVertexIDsFromFace(10)`

GetVertexIDsFromObject

Returns vertex IDs associated with a specified object.

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><i>&lt;objectName&gt;</i></td><td>String</td><td>Object name.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>Array containing string vertex IDs.</td></tr>
</table>

Python Syntax	GetVertexIDsFromObject(<*objectName*>)
Python Example	`oEditor.GetVertexIDsFromObject('Box1')`

GetVertexPosition

Returns an array of coordinates for a specified vertex.

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><i>&lt;vertexID&gt;</i></td><td>Integer</td><td>Vertex ID.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>Array containing X, Y, Z coordinates.</td></tr>
</table>

Python Syntax	GetVertexPosition(<*vertexID*>)
Python Example	`oEditor.GetVertexPosition(1)`

GetWireBodyNames

Returns the wire body names in current 3D modeler.

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

Python Syntax	GetWireBodyNames()
Python Example	`oEditor.GetWireBodyNames()`

PageSetup

Specifies page settings for printing.

<table>
<tr><td><b>UI Access</b></td><td><b>File</b> > <b>Page Setup</b>.</td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;Parameters&gt;</i></td><td>Array</td><td>Structured array.
[
"NAME:PageSetupData",</td></tr>
</table>
</td></tr>
</table>

		"margins:=", <SetupArray>]
<*SetupArray*>	Array	Structured Array ["left:=", <string>, "right:=", <string>, "top:=", <string>, "bottom:=", <string>)]

Return Value	None.

Python Syntax	PageSetup(<*Parameters*>)
Python Example	oEditor.PageSetup(["NAME:PageSetupData", "margins:=", ["left:=", "10mm", "right:=", "10mm", "top:=", "10mm",

	`"bottom:=", "10mm"]`
	`])`

RemoveBadEdges

Removes bad edges from specified list.

UI Access	N/A
Parameters	Name | Type | Description *<EdgeList>* | Array | Array containing edge IDs.
Return Value	None.

Python Syntax	RemoveBadEdges (<*EdgeList*>)
Python Example	`oEditor.RemoveBadEdges([18,30])`

RemoveBadFaces

Removes bad faces from specified list.

UI Access	N/A
Parameters	Name | Type | Description *<FaceList>* | Array | Array containing face IDs.
Return Value	None.

Python Syntax	RemoveBadFaces (<*FaceList*>)
Python Example	`oEditor.RemoveBadFaces([328,333])`

RemoveBadVertices

Removes bad vertices from specified list.

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;<i>VertexList</i>&gt;</td><td>Array</td><td>Array containing vertex IDs.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	RemoveBadVertices (<*VertexList*>)
Python Example	`oEditor.RemoveBadVertices([324,325])`

RenamePart

Renames an object.

<table>
<tr><td><b>UI Access</b></td><td>Enter new name in <b>Name</b> field.</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;<i>renameParametersArray</i>&gt;</td><td>Array</td><td>Structured array.</td></tr></table></td></tr>
</table>

			Array("NAME:Rename Data",
			   "Old Name:=", <string>,
			   "New Name:=", <string>
			)

Return Value	None.		

Python Syntax	oEditor.RenamePart(<*renameParametersArray*>)
Python Example	oEditor.RenamePart(['NAME:Rename Data', 'Old Name:=', 'partname', 'New Name:=', 'newpartname',])

SetPropValue [Modeler]

Sets the property value for the active model property.

> **Note:**
>
> This command is not supported by the EMIT and Circuit design types.

<table>
<tr><td><b>UI Access</b></td><td>Edit <b>Properties</b> on History Tree objects</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><i>&lt;propPath&gt;</i></td><td>String</td><td>Child object's property path. See: Object Property Function Summary.</td></tr><tr><td><i>&lt;value&gt;</i></td><td>Varies</td><td>New property value.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>Boolean:
• <b>True</b> – property found.
• <b>False</b> – property not found.</td></tr>
</table>

Python Syntax	SetPropValue(<*propPath*>, <*value*>)
Python Example	`oEditor.SetPropValue("Color/r",111)`

SetTopDownViewDirectionForActiveView

Sets active view to top-down view direction.

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><i>&lt;RelativeCS&gt;</i></td><td>String</td><td>Name of relative coordinate system</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	SetTopDownViewDirectionForActiveView (<*RelativeCS*>)
Python Example	`oEditor.SetTopDownViewDirectionForActiveView("Global")`

SetTopDownViewDirectionForAllViews

Sets all views to top-down view direction.

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;<i>RelativeCS</i>&gt;</td><td>String</td><td>Name of relative coordinate system</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	SetTopDownViewDirectionForAllViews (<*RelativeCS*>)
Python Example	`oEditor.SetTopDownViewDirectionForAllViews("Global")`

UpdatePriorityList

Updates specified priority lists.

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;<i>Lists</i>&gt;</td><td>Array</td><td>Array of priority list names.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	UpdatePriorityList (<*Lists*>)
Python Example	`oEditor.UpdatePriorityList(["PriorityList1", "PriorityList2"])`

UpgradeVersion

Upgrades legacy geometry to current version.

UI Access	Right-click on an operation icon in the history tree, select **Upgrade Version**.
Parameters	<table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;<i>Selections</i>&gt;</td><td>Array</td><td>Structured array. <code>Array("NAME:Parameters",</code> <code> Array("NAME:PartOperations",</code> <code> Array("NAME:<string&gt;",),</code> <code> "OperationIndices:=", <array of integers&gt;),</code> <code> Array("NAME:UDMOperations"))</code></td></tr></table>
Return Value	None.

Python Syntax	UpgradeVersion (<*Selections*>)
Python Example	`oEditor.UpgradeVersion([`

```
        "NAME:Parameters",
        ["NAME:PartOperations",
                ["NAME:source",
                "OperationIndices:=", [0]]
        ],
        ["NAME:UDMOperations"]
])
```

Validate3DComponent

Validates a 3D component.

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td><table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td>&lt;<i>componentPath</i>&gt;</td><td>String</td><td>Path to a 3D component.</td></tr>
<tr><td>&lt;password&gt;</td><td>String</td><td>Optional. Password to access the component.</td></tr>
</table></td></tr>
<tr><td><b>Return Value</b></td><td>Boolean:
• <b>1</b> - component is valid.
• <b>0</b> - component is not valid.</td></tr>
</table>

Python Syntax	Validate3DComponent (<*componentPath*>, <*password*>)

Python Example	`oEditor.Validate3DComponent(`
	`"C:/temp/component.3dcomp","")`

WriteHistoryTreeLayoutForTest

Writes history tree layout to a file.

UI Access	N/A
Parameters	<table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>*<filePath>*</td><td>String</td><td>Path to the file.</td></tr><tr><td><*OrgByMaterial*></td><td>Integer</td><td>• **1** - organize objects by material. • **0** - do not organize by material.</td></tr><tr><td><*OrgByAssignment*></td><td>Integer</td><td>• **1** - organize sheets by assignment. • **0** - do not organize by assignment.</td></tr><tr><td><*OrgByCompDefinition*></td><td>Integer</td><td>• **1** - organize components by definition. • **0** - do not organize by definition.</td></tr><tr><td><*DonotOrgUnderGroup*></td><td>Integer</td><td>• **1** - do not organize within group. • **0** - organize within group.</td></tr><tr><td><*ShowGroup*></td><td>Integer</td><td>Optional. • **1** - show group. • **0** - do not show.</td></tr></table>
Return Value	None.

Python Syntax	WriteHistoryTreeLayoutForTest (<*filePath*>, <*OrgByMaterial*>, <*OrgByAssignment*>, <*OrgByCompDefinition*>, <*DonotOrgUnderGroup*>, <*ShowGroup*>)

Python Example	`oEditor.WriteHistoryTreeLayoutForTest`
	`('C:\temp', 1, 0, 0, 0, 1)`

This page intentionally left blank.

9 - Output Variable Script Commands

Output variable commands should be executed by the "OutputVariable" module.

First, obtain the output variable module from oDesign and use it for output variable commands:

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

```
oModule.<CommandName><args>
```

The commands are:

DeleteOutputVariable

DoesOutputVariableExist

EditOutputVariable

GetOutputVariableValue

CreateOutputVariable

Adds a new output variable. Different forms of this command are executed for different design types.

> **Note:**
>
> Output variables are associated with a name and an expression. The name is not permitted to collide with design variable, sim value, or other output variable names. It cannot have spaces or any arithmetic or other operators in it. Expression definitions cannot be cyclic. For example, A = 2*B, B=3*A is not allowed.

<table>
<tr><td><b>UI Access</b></td><td><b>Mechanical</b> > <b>Results</b> > <b>Output Variables</b>.</td></tr>
<tr><td><b>Parameters</b></td><td><table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;OutputVarName&gt;</i></td><td>String</td><td>Name of the new output variable.</td></tr>
<tr><td><i>&lt;Expression&gt;</i></td><td>Value</td><td>Value to assign to the variable.</td></tr>
</table></td></tr>
</table>

<*SolutionName*>	String	Name of the solution, as seen in the output variable UI.
<*reportType*>	String	The name of the report type as seen in the output variable UI.
<*ContextArray*>	Array	Structured array containing context for which the output variable expression is being evaluated. `Array("Context:=", <string>)`

Return Value	None.

Python Syntax	CreateOutputVariable (<*OutputVarName*>, <*Expression*>, <*SolutionName*>, <*reportType* | *solutionType*>, <*contextArray* | *domainArray*>)
Python Example	

DeleteOutputVariable

Deletes an existing output variable. The variable can only be deleted if it is not in use by any traces.

UI Access	**Mechanical** > **Results** > **Output Variables**. In the **Output Variables** window, click **Delete**.
Parameters	
Return Value	None.

Name	Type	Description
<*OutputVarName*>	String	Name of the output variable.

Python Syntax	DeleteOutputVariable (<*OutputVarName*>)

Python Example	`oModule = oDesign.GetModule("OutputVariable")`
	`oModule.DeleteOutputVariable ("testNew")`

DoesOutputVariableExist

Verifies whether or not a named output variable exists.

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><<i>outputVariableName</i>></td><td>String</td><td>The output variable name.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>Boolean True if the variable exists; False if it does not.</td></tr>
</table>

Python Syntax	DoesOutputVariableExist(<*outputVariableName*>)

Python Example	`oProject = oDesktop.GetActiveProject()`
	`oDesign = oProject.GetActiveDesign()`
	`oModule = oDesign.GetModule("OutputVariable")`
	`oModule.DoesOutputVariableExist("MyTestVar")`

EditOutputVariable

Changes the name or expression of an existing output variable.

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td>&lt;<i>OrigVarName</i>&gt;</td><td>String</td><td>Name of the original output variable.</td></tr>
<tr><td>&lt;<i>NewExpression</i>&gt;</td><td>String</td><td>New value to assign to the variable.</td></tr>
<tr><td>&lt;<i>NewVarName</i>&gt;</td><td>String</td><td>New name of the variable if any, else pass empty string.</td></tr>
<tr><td>&lt;<i>SolutionName</i>&gt;</td><td>String</td><td>Name of the solution as seen in the output variable UI.
For example, "<code>Setup1 : Last Adaptive</code>".</td></tr>
<tr><td>&lt;<i>ReportType</i>&gt;</td><td>String</td><td>The name of the report type as seen in the output variable UI.</td></tr>
<tr><td>&lt;<i>ContextArray</i>&gt;</td><td>Array</td><td>Structured array containing context for which the output variable expression is being evaluated.
<code>Array("Context:=", <string&gt;)</code></td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>None</td></tr>
</table>

Python Syntax	EditOutputVariable (<*OrigVarName*>, <*NewExpression*>, <*NewVarName*>, <*SolutionName*>, <*ReportType*>, <*ContextArray*>)

Python Example	`oModule = oDesign.GetModule("OutputVariable")`
	`oModule.EditOutputVariable ("test", "normalize(R1_0.V)", "testNew", "TR", "Standard", [])`

ExportOutputVariables

Exports output variables to a file.

<table>
<tr><td><b>UI Access</b></td><td>Click on <b>Export</b> in the <b>Output Variables</b> dialog.</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><i>&lt;FileName&gt;</i></td><td>String</td><td>Name of the file include path.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>Boolean:<ul><li><b>True</b> - output variable successfully exported.</li><li><b>False</b> - error when export output variables.</li></ul></td></tr>
</table>

Python Syntax	ExportOutputVariables(<*FileName*>)
Python Example	`oModule.ExportOutputVariables("C:/output_var.aoutvar")`

GetOutputVariables

Returns the list of output variables.

UI Access	N/A
Parameters	None.
Return Value	Array containing all output variables.

Python Syntax	GetOutputVariables()
Python Example	`oDesign.GetOutputVariables()`

GetOutputVariableValue

Returns the double value of an output variable. Only expressions that return a double value are supported. The expression is evaluated only for a single point.

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;OutputVarName&gt;</i></td><td>String</td><td>Name of the output variable.</td></tr>
<tr><td><i>&lt;IntrinsicVariation&gt;</i></td><td>String</td><td>A set of intrinsic variable value pairs to use when evaluating the output expression.</td></tr>
<tr><td><i>&lt;SolutionName&gt;</i></td><td>String</td><td>Name of the solution as listed in the output variable UI. For example, <code>"Setup1 : Last Adaptive"</code>.</td></tr>
<tr><td><i>&lt;ReportType&gt;</i></td><td>String</td><td>The name of the report type as seen in the output variable UI.</td></tr>
<tr><td><i>&lt;ContextArray&gt;</i></td><td>Array</td><td>Structured array containing context for which the output variable expression is being evaluated. Can be empty.
<code>Array("Context:=", <string&gt;)</code></td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>Double value of the output variable.</td></tr>
</table>

Python Syntax	GetOutputVariableValue(<*OutputVarName*>, <*IntrinsicVariation*>, <*SolutionName*>, <*ReportTypeName*>, <*ContextArray*>)
Python Example	`Val = oDesign.GetOutputVariableValue("test","Freq = '20Ghz' Theta='20deg'`

	`Phi='30deg'", "TR", "Standard", [])`

ImportOutputVariables

Imports output variables from a file.

UI Access	Click on **Import** in the **Output Variables** dialog.
Parameters	Name Type Description *<FileName>* String Name of the file include path.
Return Value	Boolean: • **True -** output variable successfully imported. • **False -** error when import output variables.

Python Syntax	ImportOutputVariables(<*FileName*>)
Python Example	`oModule.ImportOutputVariables("C:/output_var.aoutvar")`

SimValueContext

SimValueContext holds context information for a trace, and describes how data for a trace should be extracted from the simulation. SimValueContext contains a list of 14 required initial values:

```
SimValueContext (
Domain ID, Calculation Type, Number of Cycles, Rise Time,
Step, Impulse, Context ID, Window Width,
Window Type, TDR Kaiser Parameter, Hold Time, DeviceName,
TDR Step Time, DR Maximum Time )
```

For example, the following indicates a trace in the Time Domain, Standard Calculation with the number of cycles being 2:

```
"SimValueContext:=", Array(1, 0, 2, 0, false, false, -1, 1, 0, 1, 1, "", 0, 0)
```

Additional, context-specific values may follow the required values, as described in subsection 15 below.

1. Domain ID

No Domain	0
Time Domain	1
Spectrum Domain	2
Sweep Domain	3
Device Domain	4
SinglePt Domain	5
LoadPull Domain	6
Transient Domain	7
Budget Domain	8
NetworkFunction Domain	9
Oscillator Domain	55802
Noise Domain	55803
Transfer Function Domain	55807
Time Frequency Domain	55808

Transient Time Domain	55809
Periodic AC Domain	55818
UI Domain	55819
Eye Measurement Domain	55823
Initial Response Domain	55824
Peak Distortion Domain	55825

2. Calculation Type

Standard Calculation	0
Device2_DCIV	1
Device3_DCIV_Output	2
Device3_DCIV_Input	3
Device3_DCIV_Transfer	4
Device3_DCIV_Reverse	5
Device2_ACLoad	6
Device3_ACLoad_Output	7
Device3_ACLoad_Input	8
Device3_ACLoad_Transfer	9
Device3_ACLoad_Reverse	10
Constellation	11
EyeDiagram	12
FreeX (Statistic Report)	13

3. **Number of Cycles** — Used in Time Domain in HarmonicBalance analysis.

4. **Rise Time** — Not used by Designer/Nexxim.
5. **Step** — Not used by Designer/Nexxim.
6. **Impulse** — Not used by Designer/Nexxim.
7. **Context ID** — Not used by Designer/Nexxim.
8. **Window Width** — Not used by Designer/Nexxim.
9. **Window Type** — Not used by Designer/Nexxim.
10. **TDR Kaiser Parameter** — Not used by Designer/Nexxim.
11. **Hold Time** — Not used by Designer/Nexxim.
12. **DeviceName** — Not used by Designer/Nexxim.
13. **TDR Step Time** — Not used by Designer/Nexxim.
14. **TDR Maximum Time** — Not used by Designer/Nexxim.
15. **Context-specific values** — Used in Time Domain in HarmonicBalance analysis.

Context-specific values are entered in the format "key, true/false, keyvalue", where:

— **"key"** is the name of the key being set.

— **"true/false"** indicates whether the key is a string value.

— **"keyvalue"** is the value of the key.

— The order of the context keys is not significant.

— Context keys have software defaults that will be used if not provided in the script.

a. Plotting Range for Time domain in Transient and QuickEye analysis:

Description	Key Name	Is a string?	Key Value
Start Time	WS	False	0ns
Stop Time	WE	False	10ns
Minimum Time	WM	False	0ns
Maximum Time	WN	False	10ns
Is Thinning Enabled?	DE	False	0
Dy/dx Tolerance	DT	False	0.001
Number of points	DP	False	20000000

b.Transient report context for Spectral domain in Transient analysis:

Description	Key Name	Is a string?	Key Value
Start Time	TS	False	0ns
Stop Time	TE	False	10ns
Max Harmonics	MH	False	100
Max Frequency	MF	False	*
Window type	WT	False	0
Width Percentage	WW	False	100
Kaiser Parameter	KP	False	0
Adjust Coherent Gain	CG	False	0

* Script can specify either MH or MF. If neither is specified, Max Harmonics is set to 100. If both are specified, MF is used.

Window Type	ID
Rectangular	0
Bartlett	1
Blackman	2
Hamming	3
Hanning	4
Kaiser	5
Welch	6
Weber	7
Lanzcos	8

c. Eyeprobe index context for UI domain, Time domain, Eye Measuremant domain in VerifEye and QuickEye analysis:

Description	Key Name	Is a string?	Key Value
Eyeprobe compinst ID	PCID	False	0

d. Eyesource index context for Initial Response domain and Peak Distortion domain in VerifEye and QuickEye analysis:

Description	Key Name	Is a string?	Key Value
Eyesource compinst ID	SCID	False	0

e. UI domain context in VerifEye and QuickEye analysis:

Description	Key Name	Is a string?	Key Value
Use midpoint?	MIDPOINT	False	0 - Don't use midpoint. 1 - Use midpoint of amplitude. 2 - Use midpoint of UI.
Minimum latch overdrive	MLO	False	0

f. Distribution Context for UI Domain in VerifEye and QuickEye analysis:

Description	Key Name	Is a string?	Key Value
Use distribution?	USE_DIST	False	0 - No 1 - Yes
Distribution type	DIST	False	0 - Receiver Jitter 1 - Receiver Noise 2 - User Defined

Receiver Jitter Parameters

Description	Key Name	Is a string?	Key Value
DLL standard deviation	DSD	False	0
Distribution type	DIST	False	0
DLL taps	DMN	False	0
Static Offset	SOFF	False	0
Number of Gaussian data sets	NUMG	False	0
Gaussian std deviation	GS0,GS1...	False	0
Offset mean	GM1,GM1...	False	0
Number of Uniform data sets	NUMU	False	0
Uniform width	UW0,UW1...	False	0
Uniform mean	UM1,UM1...	False	0

Receiver Noise Parameters

Description	Key Name	Is a string?	Key Value
Number of Gaussian data sets	NUMG	False	0
Gaussian std deviation	GS0,GS1...	False	0
Number of Uniform data sets	NUMU	False	0

Uniform width	UW0,UW1...	False	0

User Defined Parameters

Description	Key Name	Is a string?	Key Value
Number of XY data pairs	NUMG	False	0
X data	X0,X1,X2...	False	0
Y data	Y0,Y1,Y2...	False	0
Cutoff probability	CP	False	0

This page intentionally
left blank.

10 - Reporter Editor Script Commands

Reporter commands should be executed by the oDesign object.

For example:

All Report and Trace properties can be edited using the **ChangeProperty** commands. This includes Title properties, General properties, and Background properties such as border color, fonts, X and Y axis scaling, and number display.

> **Note:**
>
> When you execute **Tools** > **Record Script**, operations performed in the Reporter are automatically recorded.

The list of commands is as follows:

AddCartesianLimitLine

AddCartesianXMarker

AddCartesianYMarker

AddCartesianYMarkerToStack

AddDeltaMarker

AddMarker

AddNote

AddTraces

ClearAllMarkers

ClearAllTraceCharacteristics

CopyReportDefinitions

CopyReportData

CopyTraceDefinitions

CopyTracesData

CreateReport [Mechanical]

CreateReportFromFile

CreateReportFromTemplate

CreateReportOfAllQuantities

DeleteAllReports

DeleteReports

DeleteTraces

ExportImageToFile

ExportToFile [Reporter]

GetAllCategories

GetAllQuantities

GetAllReportNames

GetAvailableDisplayTypes

GetAvailableReportTypes

GetAvailableSolutions

GetChildNames

GetChildObject

GetChildTypes

AddAllEyeMeasurements

Displays all the eye measurements in tabular format.

<table>
<tr><td><b>UI Access</b></td><td>Right-click the report and select <b>Trace Characteristics</b> &gt; <b>Add All Eye Measurements</b></td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><i>&lt;ReportName&gt;</i></td><td>String</td><td>Name of the report.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	AddAllEyeMeasurements(<*ReportName*>)
Python Example	`oModule.AddAllEyeMeasurements("DQS Eye")`

AddCartesianLimitLine

Adds a limit line to a report on the X axis.

<table>
<tr><td><b>UI Access</b></td><td><b>Report2D</b> &gt; <b>Add Limit Line</b>&gt; <b>Specify Points...</b></td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><i>&lt;ReportName&gt;</i></td><td>String</td><td>Name of the report.</td></tr><tr><td><i>&lt;Def&gt;</i></td><td>Array</td><td>Structured array:<pre>
Array("NAME:CartesianLimitLine",
  Array("NAME:XValues", <integer X values>),
  "XUnits:=", <string unit of measure for X>,
  Array("NAME:YValues", <integer Y values>),
  "YUnits:=", "<string unit of measure for Y>",
  "YAxis:=", <string name of associated Y axis>
</pre></td></tr></table></td></tr>
</table>

<table>
<tr><td></td><td><table><tr><td></td><td></td><td>)</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

<table>
<tr><td><b>Python Syntax</b></td><td>AddCartesianLimitLine (&lt;<i>ReportName</i>&gt;, &lt;<i>Def</i>&gt;)</td></tr>
<tr><td><b>Python Example</b></td><td><pre>
oModule.AddCartesianLimitLine("Project Outputs",
  ["NAME:CartesianLimitLine",
    ["NAME:XValues",0, 2, 5, 7, 10, 15],
    "XUnits:=", "s",
    ["NAME:YValues",0.05, 0.3, 0.65, 0.825, 0.95, 1],
    "YUnits:=", "mV", "YAxis:=", "Y1"
  ]
)
</pre></td></tr>
</table>

AddCartesianLimitLineFromCurve

Adds a limit line to a report from selected curve on the plot.

<table>
<tr><td><b>UI Access</b></td><td><b>Report2D</b> &gt; <b>Add Limit Line</b> &gt; <b>From Selected Curve...</b></td></tr>
<tr><td><b>Parameters</b></td><td><table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td>&lt;<i>ReportName</i>&gt;</td><td>String</td><td>Name of the report.</td></tr>
<tr><td>&lt;<i>LimitLineParams</i>&gt;</td><td>String</td><td>Structured array.<pre>
Array("NAME:CartesianLimitLineFromCurve",
  "TraceName:=", <string name of selected trace>,
</pre></td></tr>
</table></td></tr>
</table>

			"CurveName:=", <string name of selected curve>, "Start:=", "<value><unit>", "Stop:=", "<value><unit>", "YAxis:=", <integer Y-Axis number>, "YOffset:=", <value offset from Y-Axis>, "CreateMode:=", "<AboveCurve | BelowCurve | Above and Below Curve>", "YShiftPercent:=", <value percentage to shift from Y>)
Return Value	None.		

Python Syntax	AddCartesianLimitLineFromCurve(<*ReportName*>, <*LimitLineParams*>)
Python Example	oModule.AddCartesianLimitLineFromCurve("Variables Plot 2", ["NAME:CartesianLimitLineFromCurve", "TraceName:=" , "Phase", "CurveName:=" , "", "Start:=" , "0deg", "Stop:=" , "375deg", "YAxis:=" , 1, "YOffset:=" , 0,

	`"CreateMode:="           , "AboveCurve",`
	`"YShiftPercent:="        , 10`
	`])`

AddCartesianLimitLineFromEquation

Adds a limit line to a report from a specified equation.

<table>
<tr><th>UI Access</th><th><b>Report2D</b> &gt; <b>Add Limit Line</b> &gt; <b>Specify Equation...</b></th></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td>&lt;<i>ReportName</i>&gt;</td><td>String</td><td>Name of the report.</td></tr>
<tr><td>&lt;<i>LimitLineParams</i>&gt;</td><td>Array</td><td>Structured array.
<code>Array("NAME:CartesianLimitLineFromEquation",</code>
<code> "YAxis:=", <integer Y-Axis number&gt;,</code>
<code> "Start:=", <string start frequency with unit&gt;,</code>
<code> "Stop:=", <string end frequency with unit&gt;,</code>
<code> "Step:=", <string frequency resolution with unit&gt;,</code>
<code> "Equation:=", <string specified equation&gt;)</code>
<code> "XValuesUnit:=", "GHz"</code></td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	AddCartesianLimitLineFromEquation(<*ReportName*>, <*LimitLineParams*>)
Python Example	`oModule.AddCartesianLimitLineFromEquation("S Parameter Plot 1",`

```
[
  "NAME:CartesianLimitLineFromEquation",
  "YAxis:="       , 1,
  "Start:="       , "9GHz",
  "Stop:="        , "11GHz",
  "Step:="        , "0.2GHz",
  "Equation:="    , "x+1"
  "XValuesUnit:=" ,"GHz"
])
```

AddCartesianXMarker

Adds a marker to a report on the X axis.

<table>
<tr><td><b>UI Access</b></td><td><b>Report2D</b> &gt; <b>Marker</b> &gt; <b>Add X Marker</b>.</td></tr>
<tr><td><b>Parameters</b></td><td><table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td>&lt;<i>ReportName</i>&gt;</td><td>String</td><td>Name of report.</td></tr>
<tr><td>&lt;<i>MarkerName</i>&gt;</td><td>String</td><td>Marker name, including any trailing number.</td></tr>
<tr><td>&lt;<i>XValue</i>&gt;</td><td>Double</td><td>X coordinate.</td></tr>
</table></td></tr>
<tr><td><b>Return Value</b></td><td>None</td></tr>
</table>

Python Syntax	AddCartesianXMarker (<*ReportName*>, <*MarkerName*>, <*XValue*>)

Python Example	`oModule.AddCartesianXMarker ("XY Plot 1", "MX1", 0)`

AddCartesianYMarker

Adds a marker to a report on the Y axis.

<table>
<tr><td><b>UI Access</b></td><td><b>Report2D</b> &gt; <b>Marker</b> &gt; <b>Add Y Marker</b>.</td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;ReportName&gt;</i></td><td>String</td><td>Name of report.</td></tr>
<tr><td><i>&lt;MarkerName&gt;</i></td><td>String</td><td>Marker name, including any trailing number.</td></tr>
<tr><td><i>&lt;AxisName&gt;</i></td><td>String</td><td>Name of axis.</td></tr>
<tr><td><i>&lt;YValue&gt;</i></td><td>Double</td><td>Y coordinate.</td></tr>
<tr><td><i>&lt;CurveName&gt;</i></td><td>String</td><td>Name of curve.</td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>None</td></tr>
</table>

Python Syntax	AddCartesianYMarker (<*ReportName*>, <*MarkerName*>, <*AxisName*>, <*YValue*>, <*CurveName*>)

Python Example	`oModule.AddCartesianYMarker("XY Plot 1", "MY1", "Y1", 0,`
	`"dB() : Setup1 : Sweep1")`

AddCartesianYMarkerToStack

Adds a marker to a stacked report on the Y axis.

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;ReportName&gt;</i></td><td>String</td><td>Name of report.</td></tr>
</table>
</td></tr>
</table>

	<*MarkerName*>	String	Marker name, including any trailing number.
	<*AxisName*>	String	Name of axis.
	<*YValue*>	Double	Y coordinate.
	<*CurveName*>	String	Name of curve.
	<*StackOption*>	Array	"Current" to create marker on the current stack. "All" to create marker on all stack.
Return Value	None		

Python Syntax	AddCartesianYMarkerToStack (<*ReportName*>, <*MarkerName*>, <*AxisName*>, <*YValue*>, <*CurveName*>, <*StackOption*>)

Python Example	`oModule.AddCartesianYMarker("XY Plot 1", "MY1", "Y1", 0,`
	`"dB() : Setup1 : Sweep1", ["All"])`

AddDeltaMarker

Add markers to calculate differences between two trace points on a plot.

UI Access	**Report2D** > **Marker** > **Add Delta Marker**.		
Parameters	Name	Type	Description
	<*ReportName*>	String	Name of report.
	<*MarkerName1*>	String	Marker name, including any trailing number, for the first marker.
	<*CurveName1*>	String	Full trace name for the first marker.
	<*PrimarySweepValue1*>	String	
	<*MarkerName2*>	String	Marker name, including any trailing number, for the second marker.
	<*CurveName2*>	String	Full trace name for the second marker.

	<PrimarySweepValue2> | String |
Return Value	None

Python Syntax	AddDeltaMarker(<*ReportName*>, <*MarkerName1*>, <*CurveName1*>, <*PrimarySweepValue1*>, <*MarkerName2*>, <*CurveName2*>, <*PrimarySweepValue2*>)
Python Example	

AddMarker

Adds a marker to a trace on a report.

UI Access	**Report2D** > **Marker** > **Add Marker**.
Parameters	Name | Type | Description <*ReportName*> | String | Name of report. <*MarkerName*> | String | Marker name, including any trailing number. <*CurveName*> | String | Full trace name. <*PrimarySweepValue*> | String | Primary sweep value, including unit.
Return Value	None

Python Syntax	AddMarker(<*ReportName*>, <*MarkerName*>, <*CurveName*>, <*PrimarySweepValue*>)

Python Example	`oModule.AddMarker("XY Plot 1", "m5",`
	`"GS1.VAL : TR4 : Cartesian", "3.61599999999997s")`

AddNote

Adds a note at a specified location to a given report.

<table>
<tr><td><b>UI Access</b></td><td>Right-click on the plot and select <b>Add Note</b>.</td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;ReportName&gt;</i></td><td>String</td><td>Name of report</td></tr>
<tr><td><i>&lt;NoteDataArray&gt;</i></td><td>Array</td><td>Structured array.
<code>Array("NAME:<StringDataName>", <NoteArray&gt;)</code></td></tr>
<tr><td><i>&lt;NoteArray&gt;</i></td><td>Array</td><td>Structured array:
<code>Array("NAME:<NoteDataSourceName&gt;",</code>
<code>"SourceName:=", <string source name&gt;,</code>
<code>"HaveDefaultPos:=", <boolean True for position 0,0; False to specify below&gt;,</code>
<code>"DefaultXPos:=", <int X position for note; 0 for default&gt;,</code>
<code>"DefaultYPos:=", <int Y position for note; 0 for default&gt;,</code>
<code>"String:=", <string note text&gt;)</code></td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>None</td></tr>
</table>

Python Syntax	AddNote (<*ReportName*>, <*NoteDataArray*>)
Python Example	`oModule.AddNote("XY Plot1",`

<table>
<tr><td></td><td><pre> ["NAME:NoteDataSource",
    "SourceName:=", "Note1",
    "HaveDefaultPos:=", False,
    "DefaultXPos:=", 1996,
    "DefaultYPos:=", 3177,
    "String:=", "This is a note."
  ]
)</pre></td></tr>
</table>

AddTraceCharacteristics

Adds a trace characteristics field to the legend on a report.

<table>
<tr><th>UI Access</th><th><b>Report2D</b> > <b>Trace Characteristics</b> > <b>All</b>. This opens the <b>Add Trace Characteristics</b> dialog box.</th></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td>&lt;<i>ReportName</i>&gt;</td><td>String</td><td>Name of report.</td></tr>
<tr><td>&lt;<i>FunctionName</i>&gt;</td><td>String</td><td>The function name. See the <b>Functions</b> column of the <b>Add Trace Characteristics</b> dialog box.</td></tr>
<tr><td>&lt;<i>FunctionArgs</i>&gt;</td><td>Array</td><td>Array containing string values for any function arguments. Pass empty array if no arguments exist.
To see which argument(s) a function takes, see the bottom of the <b>Add Trace Characteristics</b> dialog box.
For function with one argument:
<code>Array(<value&gt;)</code></td></tr>
</table>
</td></tr>
</table>

		For function with multiple arguments: `Array(<value>, <value>, ...)`
<*RangeArgs*>	Array	Required. Array containing either string "Full" for a full sweep range, or "Specified" and strings containing the start and end values for the frequency range. For example: `Array("Specified", "19.5GHz", "24.4GHz")`

Return Value	None.

Python Syntax	AddTraceCharacteristics (<*ReportName*>, <*FunctionName*>, <*FunctionArgs*>, <*RangeArgs*>)

Python Exampl-e	`oModule.AddTraceCharacteristics("XY Plot 2", "delaytime", ["0"], ["Full"])`
	`oModule.AddTraceCharacteristics("Differential S-parameters", "prms", ["0", "0"], ["Full"])`
	`oModule.AddTraceCharacteristics("Rept2DRectFreq", "distortion", ["0"], ["Specified", "19.5GHz", "20.4GHz"])`

AddTraces

Creates a new trace and adds it to the specified report.

UI Access	**Modify Report** > **Add Trace**.
Parameters	Name | Type | Description

	<ReportName>	String	Name of Report
	<SolutionName>	String	Name of the solution as listed in the **Modify Report** dialog box.
	<ContextArray>	Array	Context for which the expression is being evaluated. This can be an empty string if there is no context. Array("Domain:=", <DomainType>) <DomainType> ex. "Sweep" or "Time" Array("Context:=", <GeometryType>) <GeometryType> ex. "Infinite Sphere*n*", "Sphere*n*", "Polyline*n*"
	<FamiliesArray>	Array	Contains sweep definitions for the report. Array("<VariableName>:= ", <ValueArray>) <ValueArray> Array("All") or Array("Value1", "Value2", ..."Value*n*") examples of <VariableName> "Freq", "Theta", "Distance"
	<ReportDataArray>	Array	This array contains the report quantity and X, Y, and (Z) axis definitions. Array("X Component:=", <VariableName>, "Y Component:=", <VariableName> | <ReportQuantityArray>) <ReportQuantityArray> ex. Array("dB(S(Port1, Port1))")
Return Value	None		

Python Syntax	Add Traces(<*ReportName*>, <*SolutionName*>, <*ContextArray* >, <*FamiliesArray*>, <*ReportDataArray*>)

Python Example	`oModule.AddTraces("XY Plot1", "Setup1 : Sweep1",`
	`["Domain:=", "Time", "HoldTime:=", 1, "RiseTime:=", 0,`
	`"StepTime:=", 6.24999373748E-012, "Step:=", False,`
	`"WindowWidth:=", 1,`
	`"WindowType:=", 0, "KaiserParameter:=", 1,`
	`"MaximumTime:=", 6.2437437437444E-009],`
	`["Time:=", ["All"], "OverridingValues:=", ["0s", "6.24999373748188e-012s", ... ]], ["X Component:=", "Time",`
	`"Y Component:=", ["TDRZ(WavePort1)"]], [])`

ApplyReportTemplate

Applies settings to a report from a template file.

<table>
<tr><th>UI Access</th><th>Right-click on a report, select <b>Report Templates</b> &gt; <b>Apply Settings</b>.</th></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td>&lt;<i>ReportName</i>&gt;</td><td>String</td><td>Name of the report to apply settings to.</td></tr>
<tr><td>&lt;<i>TemplateFile</i>&gt;</td><td>String</td><td>Template file name with path.</td></tr>
<tr><td>&lt;<i>PropertyType</i>&gt;</td><td>String</td><td>Property types to apply.
<code>("Graphical" | "Data" | "All")</code></td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	ApplyReportTemplate(<*ReportName*>, <*TemplateFile*>, <*PropertyType*>)
Python Example	`oModule.ApplyReportTemplate("Variables Plot 1", "C:/MyTemplate.rpt", "Graphical")`

ClearAllMarkers

Clears all markers from a report.

UI Access	**Report2D** > **Markers** > **Clear All**.
Parameters	Name / Type / Description <*ReportName*> / String / Name of Report
Return Value	None

Python Syntax	ClearAllMarkers(<*ReportName*>)
Python Example	`oModule.ClearAllMarkers("XY Plot 1")`

ClearAllTraceCharacteristics

Clears all trace characteristics from the legend in a report.

UI Access	**Report2D** > **Trace Characteristics** > **Clear All**.

Parameters	Name | Type | Description *<PlotName>* | String | Name of the plot
Return Value	None

Python Syntax	ClearAllTraceCharacteristics(<*PlotName*>)
Python Example	`oModule.ClearAllTraceCharacteristics("XY Plot 1")`

CloneReportsFromDatasetSolution

Clones a report for a solveed solution from a dataset solution.

UI Access	Right-click the Project tree on the report and choose **Clone from Dataset Solution** > [Dataset_SolutionName]
Parameters	Name | Type | Description <*ReportsToClone*> | Array | Array of report names to clone. <*SoluNameToUse*> | String | Name of the dataset solution.
Return Value	None.

Python Syntax	CloneReportsFromDatasetSolution(<*ReportsToClone*>, <*SoluNameToUse*>)
Python Example	`oModule.CloneReportsFromDatasetSolution(["Rectangular Plot1"], "DatasetSolution_rsm_5812")`

CopyPlotSettings

Copies settings of a specified plot.

<table>
<tr><td><b>UI Access</b></td><td>Right-click a report, select <b>Copy</b></td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><i>&lt;ReportName&gt;</i></td><td>String</td><td>Name of specified report.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	CopyPlotSettings(<*ReportName*>)
Python Example	`oModule.CopyPlotSettings("Plot 1")`

CopyReportDefinitions

Copy the definition of a report for paste operations.

<table>
<tr><td><b>UI Access</b></td><td>Select a report in the Project tree, right-click and select <b>Copy Definition</b></td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><i>&lt;ReportsArray&gt;</i></td><td>Array</td><td>Names of reports from which to copy the definitions</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>None</td></tr>
</table>

Python Syntax	CopyReportDefinitions(<*ReportsArray*>)
Python Example	`oModule.CopyReportDefinitions(["Transmission", "Reflection"])`

CopyReportsData

Copy all data corresponding to the specified reports.

<table>
<tr><td><b>UI Access</b></td><td>Select a report in the Project tree, right-click and select <b>Copy Data</b></td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><i>&lt;ReportsArray&gt;</i></td><td>Array</td><td>Names of reports from which to copy data</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>None</td></tr>
</table>

Python Syntax	CopyReportsData (<*ReportsArray*>)
Python Example	`oModule.CopyReportsData (["Transmission","Reflection"])`

CopyTraceDefinitions

Copy trace definitions for a paste operation.

<table>
<tr><td><b>UI Access</b></td><td>Select a trace in the Project tree, right-click and select <b>Copy Definition</b></td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr></table></td></tr>
</table>

	<*ReportName*>	String	Name of Report
	<*TracesArray*>	Array	Trace definitions to copy
Return Value	None		

Python Syntax	CopyTraceDefinitions(<*ReportName*>, <*TracesArray*>)

Python Example	`oModule.CopyTraceDefinitions ("Transmission",`
	`["mag(S(Port1,Port2))"])`

CopyTracesData

Copies trace data for a paste operation.

UI Access	Select a trace in the Project tree, right-click and select **Copy Data**.		
Parameters	Name	Type	Description
	<*ReportName*>	String	Name of Report.
	<*TraceArray*>	Array	Trace definitions from which to copy corresponding data.
Return Value	None.		

Python Syntax	CopyTracesData(<*ReportName*>, <*TracesArray*>)

Python Example	`oModule.CopyTracesData ("Transmission",`
	`["mag(S(Port1,Port2))"])`

CreateReport [Mechanical]

Creates a new report with one or more traces and adds it to the **Results** branch in the Project Manager.

<table>
<tr><td><b>UI Access</b></td><td>
<ul>
<li>From the <b>Results</b> ribbon tab: Click [<b>Fields Report</b> | <b>Modal Report</b> | <b>Monitor Report</b>] > <i>{Report_Type}</i></li>
<li>In the Project Manager: Right-click on <b>Results</b> > [<b>Create Fields Report</b> | <b>Create Modal Report</b> | <b>Create Monitor Report</b>] > <i>{Report_Type}</i></li>
<li>From the Menu Bar: Click <b>Mechanical</b> > <b>Results</b> > [<b>Fields Report</b> | <b>Modal Report</b> | <b>Monitor Report</b>] > <i>{Report_Type}</i></li>
</ul>
</td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td><i>&lt;ReportName&gt;</i></td><td>String</td><td>Name of report.</td></tr>
<tr><td><i>&lt;ReportType&gt;</i></td><td>String</td><td>Type of report (<code>"Modal"</code> or <code>"Fields"</code>).</td></tr>
<tr><td><i>&lt;DisplayType&gt;</i></td><td>String</td><td>Specifies the type of plot to create. Applicable choices for Mechanical designs include:
<code>"Rectangular Plot"</code>, <code>"Rectangular Stacked Plot"</code>, and <code>"Data Table"</code></td></tr>
<tr><td><i>&lt;SolutionName&gt;</i></td><td>String</td><td>Name of the solution on which the report is to be based</td></tr>
<tr><td><i>&lt;ContextArray&gt;</i></td><td>Array</td><td>Context where the expression is to be evaluated. The array consists of the geometry (&lt;<i>polyline_name</i>&gt; along which to report the quantity or the &lt;<i>point_name</i>&gt; at which to do so). Polylines can consist of any number of straight or curved segments. When specifying a polyline, the array also contains the number of points to calculate along its path. For example:
<code>Array("Context:=", "Polyline2", "PointCount:=", 101)</code>
This array can be empty if no context is required.</td></tr>
<tr><td><i>&lt;FamiliesArray&gt;</i></td><td>Array</td><td>Specifies Sweep and Families parameters.
When the Sweep is the <code>"Distance"</code> along a polyline, specify either <code>["All"]</code></td></tr>
</table>
</td></tr>
</table>

		for the full polyline path, or a range (`["<min_distance>,<max_distance>"]` for a portion of the polyline path. For example:
		`["Distance:=", ["All"]]`
		`["Distance:=", ["0.25mm","33.5mm"]]`
		For Modal reports, the Sweep is the vibration *"Mode"*. In this case, specify `["All"]` to report the quantities for all modes or ["<*start_mode*>","<*end_mode*>"] for a contiguous subset of the available modes. For example:
		`["Mode:=", ["All"]]`
		`["Mode:=", ["2","5"]]`
		For Modal fields reports, in which the Sweep is "Distance" along a polyline, the desired modes are specified in the *Families* tab of the UI's *Report* dialog box. In the script, you can list the desired modes, and they don't have to be contiguous, as shown in the first of the following two examples:
		`["Distance:=", ["All"], "Mode:=", ["1","2","4","5","7"]]`
		`["Distance:=", ["0.25mm", "33.5mm"], "Mode:=", ["All"]]`
		Selection of families is similar for all solution types when there are multiple sets of results (such as with parametric solutions). In such cases, a different variable would be substituted for "Mode."

<ReportDataArray>	Array	This array contains the X variable and Y variable (report quantities) definitions. For example: `["X Component:=", <VariableName>, "Y Component:=", [<ReportQuantityArray>]]`
<ReportQuantityArray>	Array	Report quantities are dependent on the solution type and include the following: • **All Solution Types:** ◦ "Volume(<*object_name*>)" – Volume of the named solid object

			• **Modal Solutions – Modal Report:** ◦ "ModeFreq" – Resonant frequency of the vibration modes ◦ "ParticipationFactor [1] " – Modal participation factors ◦ "EffectiveMass [1] " – Mass participating in the mode ◦ "EffectiveMassRatio [1] " – EffectiveMass / total mass ◦ "CummEMassFraction [1] " – Cumulative effective mass factor [2] • **Modal Solutions – Fields Report:** ◦ "Mag_Displacement" – Displacement magnitude [3] • **Thermal Solutions – Fields Report:** ◦ "Temperature" ◦ "Mag_HeatFlux" – Heat flux magnitude ◦ "Surface_Loss_Density" – Imported EM losses (surface-based) ◦ "Volume_Loss_Density" – Imported EM losses (volume-based) ◦ "Linked_Heat_Transfer_Coefficient" – Imported HPCs from an Icepak or Mechanical source solution (for Convection boundary film coefficients) ◦ "Thermal_Conductivity [4] " – Only applicable to designs with trace mapping • **Thermal Solutions – Monitor Report (Transient only):** ◦ "<*monitor_point_name*,Temperature" – Temperature vs. Time at the selected monitor points • **Structural Solutions:** ◦ "Mag_Displacement" – Displacement Magnitude ◦ "Equivalent_Stress – von Mises equivalent stress ◦ "Equivalent_Strain – von Mises equivalent strain

<table>
<tr>
<td></td>
<td>
<table>
<tr>
<td></td>
<td></td>
<td>
<ul><li>"Temperature" – Assigned or imported temperatures</li></ul>
The following four quantities are only applicable to designs with trace mapping:
<ul>
<li>Youngs_Modulus [4] "</li>
<li>"Poisson_ratio [4] "</li>
<li>"Thermal_Expansion [4] " – Coefficient of thermal expansion</li>
<li>Shear_Modulus [4] "</li>
</ul>
</td>
</tr>
</table>
</td>
</tr>
<tr>
<td><b>Return Value</b></td>
<td>None.</td>
</tr>
<tr>
<td><b>Notes</b></td>
<td>
[1] – Six different quantities are available for the indicated modal results—three translational ("Dir") and three rotational ("Rot"). After each <i>ReportQuantity</i> name, <b>_DirX</b>, <b>_DirY</b>, <b>_DirZ</b>, <b>_RotX</b>, <b>_RotY</b>, or <b>_RotZ</b> is appended to indicate the translational or rotational direction. For example, <b>"ParticipationFactor_RotZ"</b> is the modal participation factor for rotation about the Z axis.
[2] – For a complete definition of the <i>Cumulative Effective Mass Fraction</i> result, see the Results (Modal) help page.
[3] – See the note concerning displacement magnitudes for modal solutions on the <i>Results (Modal)</i> help page.
[4] – The indicated quantities are only applicable to Thermal or Structural solutions that include (for layout components). Three different quantities are available for each listed result to provide separate material properties in each of the three axis directions (based on the mapped metal fraction data). After each <i>ReportQuantity</i> name, <b>_X</b>, <b>_Y</b>, or <b>_Z</b> is appended to indicate the direction of the result. For example, <b>"Thermal_Conductivity_Y"</b> is the mapped thermal conductivity in the Y direction.
</td>
</tr>
</table>

<table>
<tr>
<td><b>Python Syntax</b></td>
<td>CreateReport(&lt;<i>ReportName</i>&gt;, &lt;<i>ReportType</i>&gt;, &lt;<i>DisplayType</i>&gt;, &lt;<i>SolutionName</i>&gt;, [&lt;<i>ContextArray</i>&gt;], [&lt;<i>FamiliesArray</i>&gt;], [&lt;<i>ReportDataArray</i>&gt;])</td>
</tr>
<tr>
<td><b>Python Example</b></td>
<td>

Five examples are given below covering each solution type and report type:

Example 1 – Modal (Modal Report):

```
Set oModule = oDesign.GetModule("ReportSetup")
oModule.CreateReport("Frequency Table 1", "Modal",
   "Data Table", "Setup1 : Solution", [],
   [
      "Mode:="       , ["All"]
   ],
   [
      "X Component:=", "Mode",
      "Y Component:=",
      [
         "ModeFreq",
         "ParticipationFactor_DirX",
         "ParticipationFactor_DirY",
         "ParticipationFactor_DirZ",
         "ParticipationFactor_RotX",
         "ParticipationFactor_RotY",
         "ParticipationFactor_RotZ"
      ]
   ])
```

Example 2 – Modal (Fields Report):

```
Set oModule = oDesign.GetModule("ReportSetup")
oModule.CreateReport("Calculator Expressions Plot 1",
   "Fields", "Rectangular Stacked Plot",
   "Setup1 : Solution",
```

</td>
</tr>
</table>

```
   [
      "Context:="    , "Polyline1",
      "PointCount:=" , 101
   ],
   [
      "Distance:="   , ["0.25mm","3.35mm"],
      "Mode:="       , ["1","2","4","5","7"]
   ],
   [
      "X Component:=", "Distance",
      "Y Component:=", ["Mag_Displacement"]
   ])
```

Example 3 – Steady-State Thermal (with Trace Mapping):

```
Set oModule = oDesign.GetModule("ReportSetup")
oModule.CreateReport("Calculator Expressions Plot 2",
   "Fields", "Rectangular Plot", "Setup1 : Solution",
   [
      "Context:="      , "Polyline1",
      "PointCount:="   , 101
   ],
   [
      "Distance:="     , ["All"]
   ],
   [
      "X Component:="  , "Distance",
      "Y Component:="  ,
      [
         "Temperature" ,
         "Mag_HeatFlux",
         "Volume_Loss_Density",
         "Thermal_Conductivity_X",
         "Thermal_Conductivity_Y",
         "Thermal_Conductivity_Z"
      ]
   ])
```

Example 4 – Transient Thermal (Monitor Report)

A monitor report example is given because fields reports for transient thermal solutions are the same as those for steady-state thermal solutions (see Example 3).

```
Set oModule = oDesign.GetModule("ReportSetup")
oModule.CreateReport "Monitor Plot 1", "Monitor",
   "Rectangular Plot", "Setup1 : Solution", [],
   [
      "Time:="       , ["All"]
   ]
   [
      "X Component:=", "Time",
      "Y Component:=",
      [
         "Global.Temperature",
         "HeatSinkCentroid.Temperature",
         "RodCentroid.Temperature"
      ]
   ])
```

Example 5 – Structural (with Trace Mapping)

```
Set oModule = oDesign.GetModule("ReportSetup")
oModule.CreateReport("Calculator Expressions Plot 3",
   "Fields", "Rectangular Plot", "Setup1 : Solution",
   [
      "Context:=", "Polyline1", "PointCount:=", 101
   ],
   [
      "Distance:=" , ["All"]
   ],
   [
      "X Component:=", "Distance",
      "Y Component:=",
      [
         "Equivalent_Stress",
         "Equivalent_Strain"
      ]
   ])
```

CreateReportFromFile

Creates a new report from an .rdat file.

<table>
<tr><td><b>UI Access</b></td><td>Right-click on <b>Results</b> &gt; <b>Create Report From File...</b></td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><i>&lt;FilePathName&gt;</i></td><td>String</td><td>Path to .rdat file.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	CreateReportFromFile(<*FilePathName*>)
Python Example	`oModule.CreateReportFromFile("C:/Users/MyDir/Documents/Return_Loss.rdat")`

CreateReportFromTemplate

Creates a report from a saved template.

<table>
<tr><td><b>UI Access</b></td><td><b>[product]</b> &gt; <b>Results</b> &gt; <b>Report Templates</b> &gt; <b>PersonalLib</b> &gt; [TemplateName]</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;<i>TemplatePath</i>&gt;</td><td>String</td><td>Path to report template</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>None</td></tr>
</table>

Python Syntax	CreateReportFromTemplate(<*TemplatePath*>)

Python Example	`oModule.CreateReportFromTemplate(`
	`"C:/MyHFSSProjects/PersonalLib/ReportTemplates/TestTemplate.rpt")`

CreateReportOfAllQuantities

Creates a report including all quantities in a category. Cannot create a report with expressions.

<table>
<tr><td><b>UI Access</b></td><td>NA</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;<i>ReportType</i>&gt;</td><td>String</td><td>Report type name as input parameter</td></tr><tr><td>&lt;<i>DisplayType</i>&gt;</td><td>String</td><td>Display type name as input parameter</td></tr><tr><td>&lt;<i>SolutionName</i>&gt;</td><td>String</td><td>Solution name as input parameter</td></tr><tr><td>&lt;<i>SimValueCtxt</i>&gt;</td><td>String</td><td>A context name, or array of string that encoded the context(I).</td></tr></table></td></tr>
</table>

<table>
<tr><td></td><td><table>
<tr><td><i>&lt;CategoryName&gt;</i></td><td>String</td><td>Category name as input parameter</td></tr>
<tr><td><i>&lt;PointSet&gt;</i></td><td>Array</td><td>Array of strings(II)</td></tr>
<tr><td><i>&lt;CommonComponentsOfTraces&gt;</i></td><td>Array</td><td>Array of strings(III)</td></tr>
<tr><td><i>&lt;ExtTraceInfo&gt;</i></td><td>Array</td><td>Array of strings(IV)</td></tr>
</table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	CreateReportOfAllQuantities(<*ReportNameArg*>, <*ReportType*>, <*DisplayType*>, <*SolutionName*>, <*SimValueCtxt*>, <*CategoryName*>, <*PointSet*>, <*CommonComponentsOfTraces*>, <*ExtTraceInfo*>)
Python Example	`oModule.CreateReportOfAllQuantities("Smith Chart all", "Modal Solution Data", "Smith Chart", "Setup1 : LastAdaptive", [],"S Parameter",["Freq:=", ["All"], "offset:=", ["All"],"a:=", ["Nominal"],"b:=", ["Nominal"]],[],[])`

DeleteMarker

*Use:*Deletes the specified marker.

<table>
<tr><td><b>UI Access</b></td><td>[product] &gt; <b>Fields</b> &gt; <b>Fields</b> &gt; <b>Marker</b> &gt; <b>Delete Marker</b>.</td></tr>
<tr><td><b>Parameters</b></td><td><table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;MarkerName&gt;</i></td><td>String</td><td>Name of the marker.</td></tr>
</table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	DeleteMarker(<*MarkerName*>)
Python Example	`oModule.DeleteMarker("m1")`

DeleteAllReports

Deletes all existing reports.

UI Access	Right-click the report to delete in the project tree, and then click **Delete All Reports** on the shortcut menu.
Parameters	None.
Return Value	None.

Python Syntax	DeleteAllReports()
Python Example	`oModule.DeleteAllReports()`

DeleteReports

Deletes an existing report or reports.

<table>
<tr><td><b>UI Access</b></td><td>Right-click the report to delete in the project tree, and then click <b>Delete</b> on the shortcut menu</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr></table></td></tr>
</table>

<table>
<tr><td></td><td><table><tr><td><*ReportNameArray*></td><td>Array</td><td>Array of report names to be deleted</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	DeleteReports(<*ReportNameArray*>)
Python Example	`oModule.DeleteReports (["Rept2DRectFreq"])`

DeleteTraceCharacteristics

Deletes a trace characteristics field from a report

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><*ReportName*></td><td>String</td><td>Name of the report to delete from.</td></tr><tr><td><*TraceCharsNames*></td><td>Array</td><td>Array of trace characteristics to delete.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	DeleteTraceCharacteristics(<*ReportName*>, <*TraceCharsNames*>)
Python Example	`oModule.DeleteTraceCharacteristics("Variables Plot 1", ["lowercutoff", "gain-crossover"])`

DeleteTraces

Deletes an existing traces or traces.

UI Access	Right-click the Trace to delete in the project tree, and then click **Delete** on the shortcut menu
Parameters	<table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;<i>TraceSelection</i>&gt;</td><td>Array</td><td>Structured array define selections. <code>Array("<ReportName>:=", <TracesArray>, <TracesArray&gt;,...)</code></td></tr><tr><td>&lt;<i>ReportName</i>&gt;</td><td>String</td><td>Name of Report</td></tr><tr><td>&lt;<i>TracesArray</i>&gt;</td><td>Array</td><td>Contains the traces to delete within a report <code>Array(<Trace>, <Trace>, ...)</code></td></tr><tr><td>&lt;<i>Trace</i>&gt;</td><td>String</td><td>A specific trace that the user wishes to delete</td></tr></table>
Return Value	None.

Python Syntax	DeleteTraces(<*TraceSelection*>)

Python Example	`oModule.DeleteTraces (["XY Plot 1:=",`
	`["dB(S(LumpPort1,LumpPort1))"],`
	`"XY Plot 2:=", ["Mag_E"]])`

DoesSupportTraceCharacteristics

Determines whether trace characteristics is supported in a specified display type.

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><i><DisplayType></i></td><td>String</td><td>Specified display type to check.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>Integer
• <b>1 -</b> trace characteristics is supported.
• <b>0 -</b> trace characteristics not supported.</td></tr>
</table>

Python Syntax	DoesSupportTraceCharacteristics(<*DisplayType*>)
Python Example	`oModule.DoesSupportTraceCharacteristics("Rectangular Plot")`

DumpAllReportsData

Dumps all reports data to an Ansoft report data file.

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><i><FileName></i></td><td>String</td><td>File name with path.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	DumpAllReportsData(<*FileName*>)
Python Example	`oModule.DumpAllReportsData("C:/ReportsData.rdat")`

EditCartesianXMarker

Edits an XMarker value.

UI Access	N/A
Parameters	<table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><i>&lt;ReportName&gt;</i></td><td>String</td><td>Name of report.</td></tr><tr><td><i>&lt;MarkerName&gt;</i></td><td>String</td><td>Marker name, including any trailing number.</td></tr><tr><td><i>&lt;XValue&gt;</i></td><td>Double</td><td>X coordinate.</td></tr></table>
Return Value	None.

Python Syntax	EditCartesianXMarker (<*ReportName*>, <*MarkerName*>, <*XValue*>)
Python Example	`oModule.EditCartesianXMarker ("XY Plot 1", "MX1", 0)`

EditCartesianYMarker

Edits a YMarker value.

UI Access	N/A
Parameters	<table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><i>&lt;ReportName&gt;</i></td><td>String</td><td>Name of report.</td></tr></table>

<table>
<tr><td></td><td><table>
<tr><td>&lt;<em>MarkerName</em>&gt;</td><td>String</td><td>Marker name, including any trailing number.</td></tr>
<tr><td>&lt;<em>AxisName</em>&gt;</td><td>String</td><td>Name of axis.</td></tr>
<tr><td>&lt;<em>YValue</em>&gt;</td><td>Double</td><td>Y coordinate.</td></tr>
<tr><td>&lt;<em>CurveName</em>&gt;</td><td>String</td><td>Name of curve.</td></tr>
</table></td></tr>
<tr><td><strong>Return Value</strong></td><td>None</td></tr>
</table>

Python Syntax	EditCartesianYMarker (<*ReportName*>, <*MarkerName*>, <*AxisName*>, <*YValue*>, <*CurveName*>)

Python Example	`oModule.EditCartesianYMarker("XY Plot 1", "MY1", "Y1", 0,`
	`"dB() : Setup1 : Sweep1")`

EditMarker

Edits a marker on a report.

<table>
<tr><td><strong>UI Access</strong></td><td>N/A</td></tr>
<tr><td><strong>Parameters</strong></td><td><table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td>&lt;<em>ReportName</em>&gt;</td><td>String</td><td>Name of report.</td></tr>
<tr><td>&lt;<em>MarkerName</em>&gt;</td><td>String</td><td>Marker name, including any trailing number.</td></tr>
<tr><td>&lt;<em>CurveName</em>&gt;</td><td>String</td><td>Full trace name.</td></tr>
<tr><td>&lt;<em>PrimarySweepValue</em>&gt;</td><td>String</td><td>Primary sweep value, including unit.</td></tr>
</table></td></tr>
<tr><td><strong>Return Value</strong></td><td>None.</td></tr>
</table>

Python Syntax	EditMarker(<*ReportName*>, <*MarkerName*>, <*CurveName*>, <*PrimarySweepValue*>)

Python Example	oModule.EditMarker("XY Plot 1", "m5", "GS1.VAL : TR4 : Cartesian", "3.61599999999997s")

ExportEyeMaskViolation

Exports eye mask violations to a file.

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;ReportName&gt;</i></td><td>String</td><td>Name of specified report.</td></tr>
<tr><td><i>&lt;ExportFileName&gt;</i></td><td>String</td><td>File name to export to.</td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>N/A</td></tr>
</table>

Python Syntax	ExportEyeMaskViolation(<*ReportName*>, <*ExportFileName*>)
Python Example	oModule.ExportEyeMaskViolation("Variables Plot 1", "C:/eyemask1.csv")

ExportImageToFile [Reporter]

Exports a report image in a specified format.This command is fully supports -ng (non-graphical) mode.

UI Access	N/A

<table>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;ReportName&gt;</i></td><td>String</td><td>Name of report to be exported.</td></tr>
<tr><td><i>&lt;FileName&gt;</i></td><td>String</td><td>Full path of the exported image file name; with extension of jpg, gif, tiff, tif, bmp, or wrl.</td></tr>
<tr><td><i>&lt;Width&gt;</i></td><td>Integer</td><td>Image width in pixels; if width or height is less or equal to zero, use the report window width, or 500 pixels if report window is closed.</td></tr>
<tr><td><i>&lt;Height&gt;</i></td><td>Integer</td><td>Image height in pixels; if width or height is less or equal to zero, use the report window height, or 500 pixels report window is closed.</td></tr>
<tr><td>[<i>&lt;SaveImageParameters2D&gt;</i>]</td><td>Boolean</td><td>Whether to ShowLegend, ShowMarkerTable, ShowDeltaMarkerTable. Values can be True, False or Default. Default is True.</td></tr>
<tr><td>[<i>&lt;SaveImageParameters3D&gt;</i>]</td><td></td><td>Whether to AutoFit [Default is "False"]
ShowLegend, ShowMarkerTable, ShowDeltaMarkerTable. Values can be True, False or Default. Default is True.</td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	ExportImageToFile(<*ReportName*>, <*FileName*>, <*Width*>, <*Height*>)

Python Example	#### Script Recorded for Saving 2D
	`oDesktop.OpenProject("E:/Ansoft/Tee2.aedt")`
	`oProject = oDesktop.SetActiveProject("Tee2")`
	`oDesign = oProject.SetActiveDesign("TeeModel")`
	`oModule = oDesign.GetModule("ReportSetup")`
	`oModule.ExportImageToFile("S Parameter Plot 1", "E:/Ansoft/S Parameter Plot 1.png", 1920, 1080)`
	`oModule.ExportImageToFile("S Parameter Plot 12", "E:/Ansoft/S Parameter Plot 12.png", 1920, 1080)`

```
oModule.ExportImageToFile("S Parameter Plot 1", "E:/Ansoft/S Parameter Plot 1.png",
1920, 1080,

  [

    "NAME:SaveImageParams",

    "ShowLegend:=" , "False",

    "ShowMarkerTable:=" , "False"

    "ShowDeltaMarkerTable:=" , "False"

  ])

oModule.ExportImageToFile("S Parameter Plot 1", "E:/Ansoft/S Parameter Plot 1.png",
1920, 1080,

  [

    "NAME:SaveImageParams",

    "ShowLegend:=" , "True",

    "ShowMarkerTable:=" , "True"
    "ShowDeltaMarkerTable:=" , "False"

  ])

oModule.ExportImageToFile("S Parameter Plot 12", "E:/Ansoft/S Parameter Plot 12.png",
1920, 1080,

  [

    "NAME:SaveImageParams",

    "ShowLegend:=" , "False",

    "ShowMarkerTable:=" , "Default"
    "ShowDeltaMarkerTable:=" , "False"
  ])
oModule.ExportImageToFile("S Parameter Plot 12", "E:/Ansoft/S Parameter Plot 12.png",
1920, 1080,
  [
    "NAME:SaveImageParams",
    "ShowLegend:=" , "True",
    "ShowMarkerTable:=" , "Default"
    "ShowDeltaMarkerTable:=" , "True"
  ])
  oDesktop.CloseProject("Tee2")
```

### Script Recorded for saving as 3D:

```
oDesktop.OpenProject("E:/Ansoft/Tee2.aedt")
oProject = oDesktop.SetActiveProject("Tee2")
oDesign = oProject.SetActiveDesign("TeeModel")
oModule = oDesign.GetModule("ReportSetup")
oModule.ExportImageToFile("S Parameter Plot 6", "E:/Ansoft/S Parameter Plot 11.png",
1335, 466,
  [
    "NAME:SaveImageParams",
    "AutoFit:=" , "False",
    "Orientation:=" , "",
    "ShowOrientationGadget:=", "Default"
  ])
oModule.ExportImageToFile("S Parameter Plot 6", "E:/Ansoft/S Parameter Plot 11.png",
1335, 466,
  [
    "NAME:SaveImageParams",
    "AutoFit:=" , "False",
    "Orientation:=" , "",
    "ShowOrientationGadget:=", "Default"
    "ShowLegend:=" , "True"
  ])
oModule.ExportImageToFile("S Parameter Plot 6", "E:/Ansoft/S Parameter Plot 10.png",
1335, 466,
  [
    "NAME:SaveImageParams",
    "AutoFit:=" , "False",
    "Orientation:=" , "",
    "ShowOrientationGadget:=", "Default",
    "ShowLegend:=" , "False"
```

	`    ])`

ExportModelImageToFile

Exports the model as an image file (*.avz, *.bmp, *.gif, *.jpeg, *.tiff, *.wrl). In Release 23.1, this command is fully supports -ng (non-graphical) mode. To export to Ensight use *.avz. For export to Ensight in -ng mode, the corresponding version of Ensight must be installed. On Linux, it might need manual set environment variable AWP_ROOT212 to its installation path, e.g. AWP_ROOT212-2=/installations/ansys_inc/v212/ for AnsysEDT v21.2 and Ensight 21.2.

ExportModelImageToFile exports a model image with background type and color that respect the AEDT color scheme by default. You can specify the background type and color with following parameters:

Parameter Name	Description	Parameter Value
BackgroundType	Choose one out of four types of background	Default Plain LinearGradient RadialGradient
BackgroundColor	Plain: Background color LinearGradient/RadialGradient: Background start color	3 integers with a range of [0,255] that represent red, green, and blue respectively
BackgroundContrastColor	Plain: Ignored LinearGradient/RadialGradient: Background end color	3 integers with a range of [0,255] that represent red, green, and blue respectively

Note: Current scripts will not be affected, the image will be exported with default background color as it always does

If no BackgroundType is specified, or BackgroundType is Default, BackgroundColor and BackgroundContrastColor will be ignored, and the image will be exported with default background color

ExportModelImageToFile supports export overlay of polar plot 3D with existing transformation (scaling, rotation and translation) in -ng (non-graphical) mode.

<table>
<tr><td><b>UI Access</b></td><td><b>Modeler</b> &gt; <b>Export</b>.</td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;path&gt;</i></td><td>String</td><td>Full file path including extension.</td></tr>
<tr><td><i>&lt;width&gt;</i></td><td>Integer</td><td>Width in pixels (use 0 for default).</td></tr>
<tr><td><i>&lt;height&gt;</i></td><td>Integer</td><td>Height in pixels (use 0 for default).</td></tr>
<tr><td><i>&lt;Parameters&gt;</i></td><td>Array</td><td>Structured array.
Array("NAME:SaveImageParams",
"ShowAxis:=" , <string containing boolean>,
"ShowGrid:=" , <string containing boolean>,
"ShowRuler:=" , <string containing boolean>,
"ShowRegion:=" , <string>,
"Selections:=" , <string>,
"FieldPlotSelections:=", <string>' # Comma delimited string. #Use to set which field plot to show.
"FitToSelections:=" , "",</td></tr>
</table>
</td></tr>
</table>

	Note: "FitToSelections" specify geometry objects for the "Fit" operation.
	`"FitToFieldPlotSelection:=" , ""`
	Note: "FitToFieldPlotSelections" specifies field plots for the "Fit" operation.
	`"AutoFit:=" , "True",`
	Note: If FitToSlections or FitToFieldPlotSelections are used , then AutoFit is True, it makes sure color key does not overlap field plot. It is False by default. If neither is used, then "AutoFit" will "Fit" to full model.
	`"Orientation:=" , <string>`
	`"ShowOrientationGadget:=" , <False>`

Return Value	None.

Python Syntax	ExportModelImageToFile(<*path*> <*width*> <*height*> <*Parameters*>)

Python Example	`oEditor.ExportModelImageToFile`
	`   ("D:/Image.png", 1920, 1080,`
	`[`

```
  "NAME:SaveImageParams",
    "ShowAxis:=" , "True",
    "ShowGrid:=" , "True",
    "ShowRuler:=" , "True",
    "ShowRegion:=" , "Default",
    "Selections:=" , "",
    "FieldPlotSelections:=" , "",
    "FitToSelections:=" , "",
    "FitToFieldPlotSelections:=", "",
    "AutoFit:=" , "True",
    "Orientation:=" , ""
])
```

ExportModelMeshToFile

Exports geometry model to a 3D model file (e.g. *.obj, *.wrl, etc.).

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td><table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;filePath&gt;</i></td><td>String</td><td>Full file path, including extension *.obj, *.wrl, etc</td></tr>
<tr><td><i>&lt;selections&gt;</i></td><td>Array</td><td>Selected parts.</td></tr>
</table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	ExportModelMeshToFile <*filePath*>, <*selections*>)
Python Example	`oEditor.ExportModelMeshToFile("E:/MyDir/scriptRun/2Selected-ng.obj", ["BotCover-","AveragingVolumeAtPeakRMSEfieldLocation"])`

ExportPlot3DToFile [Reporter]

Use: Exports 3D polar, spherical and rectangular plot to a case file. It works in both graphical and NG mode..

Command: None.

Syntax: ExportPlot3DToFile(<plotName>, <path>)

Return Value: A 3D plot file.

Parameters: <plotName>

Type: <string>

Plot name.

<Path>

Type: <string>

Path to file.

Python Syntax	ExportPlot3DToFile(<*name*> <*Path*>)

Python Example	`oModule = oDesign.GetModule("ReportSetup")`
	`oModule.UpdateReports(["rE Plot 1"])`
	`oModule.UpdateReports(["Directivity Plot 1"])`

	`oModule.UpdateReports(["Gain Plot 1"])`
	`oModule.ExportPlot3DToFile("rE Plot 1", "D:/projects/test2-output/rEPlot1.case")`
	`oModule.ExportPlot3DToFile("Directivity Plot 1", "D:/projects/test2-output/DirectivityPlot1.case")`
	`oModule.ExportPlot3DToFile("Gain Plot 1", "D:/projects/test2-output/GainPlot1.case")`

ExportReport

> **Note:**
>
> The ExportReport script command has been replaced by the script command ExportToFile. ExportReport remains in order to retain backward compatibility for existing scripts, but it is strongly recommended that you now use ExportToFile.

Export a report to a data file.

Command: None

Syntax: ExportReport <ReportName>, <FileName>, <FileExtension>

Return Value: None

Parameters: <ReportName>

Type: string

<Filename>

Type: string

<FileExtension>

Type: string

Python Syntax	ExportReport(<ReportName>, <FileName>)

Python Example	`oDesign.ExportReport(`
	`"Plot1", "c:\report1.dat")`

ExportReportDataToFile

Exports report data to a file.

UI Access	N/A
Parameters	<table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><*ReportName*></td><td>String</td><td>Name of specified report.</td></tr><tr><td><*ExportFileName*></td><td>String</td><td>Name of export file. File extenstion "rdat" is expected.</td></tr></table>
Return Value	None.

Python Syntax	ExportReportDataToFile(<*ReportName*>, <*ExportFileName*>)
Python Example	`oModule.ExportReportDataToFile("Plot 1", "C:/Plot1data.rdat")`

ExportTableToFile

Exports a marker table from a report to a file.

<table>
<tr><td><b>UI Access</b></td><td>Right-click on a plot, select <b>Marker</b> &gt; <b>Export Marker Table...</b> or <b>Export Delta Marker Table...</b></td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td>&lt;<i>ReportName</i>&gt;</td><td>String</td><td>Name of specified report.</td></tr>
<tr><td>&lt;<i>ExportFileName</i>&gt;</td><td>String</td><td>Name of export file.</td></tr>
<tr><td>&lt;<i>TableType</i>&gt;</td><td>String</td><td>Type of marker table to export. "Marker" or "DeltaMarker".</td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	ExportTableToFile(<*ReportName*>, <*ExportFileName*>, <*TableType*>)
Python Example	`oModule.ExportTableToFile("Plot 1", "C:/Marker.csv", "Marker")`

ExportToFile [Reporter]

From a data table or plot, generates text format, comma delimited, tab delimited, .dat, or .rdat type output files.

<table>
<tr><td><b>UI Access</b></td><td>Right-click on report name in the Project tree and select <b>Export</b>.</td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td>&lt;ReportName&gt;</td><td>String</td><td>Name of the report</td></tr>
<tr><td>&lt;FileName&gt;</td><td>String</td><td>Path and File Name
Supported formats
<table>
<tr><td>.txt</td><td>Post processor format file</td></tr>
<tr><td>.csv</td><td>Comma-delimited data file</td></tr>
</table>
</td></tr>
</table>
</td></tr>
</table>

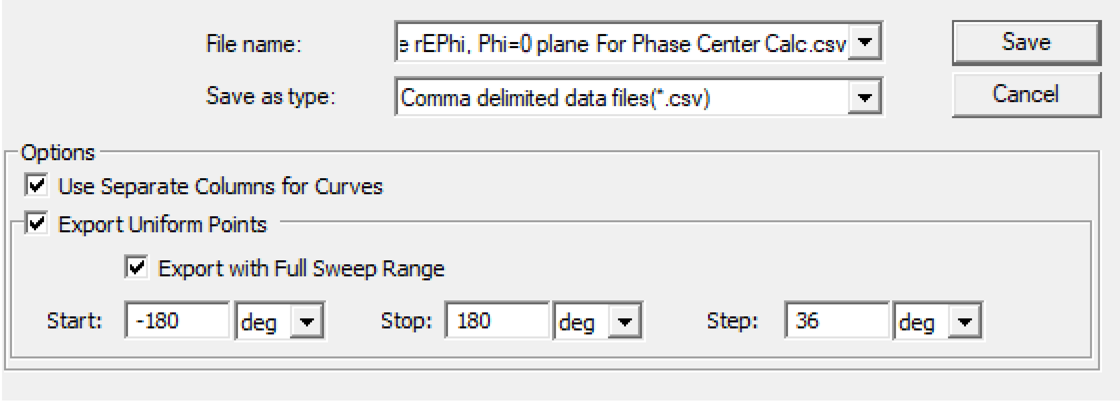
			.tab	Tab-separated file
			.dat	Ansys plot data file
			.rdat	Ansys report data file
Return Value	None			

Python Syntax	ExportToFile (<ReportName>, <FileName>)

Python Example	`oModule.ExportToFile(`
	`"Plot1", "c:\report1.dat")`

ExportUniformPointsToFile

Exports uniform points from a data table or plot report that includes the Export Uniform Points to File option enabled to text format, comma delimited, tab delimited, or .dat type output files.


File name: e rEPhi, Phi=0 plane For Phase Center Calc.csv

Save as type: Comma delimited data files(*.csv)

Options

☑ Use Separate Columns for Curves

☑ Export Uniform Points

☑ Export with Full Sweep Range

Start: -180 deg Stop: 180 deg Step: 36 deg

<table>
<tr><td><b>UI Access</b></td><td>Right-click on report name in the project tree and select <b>Export Data</b>.</td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;ReportName&gt;</i></td><td>String</td><td>Name of report to be exported.</td></tr>
<tr><td><i>&lt;FileName&gt;</i></td><td>String</td><td>Full path of the exported image file name; with extension of
.txt - Post processor format file
.csv - Comma-delimited data file
.tab - Tab-separated file
.dat - Ansys plot data file</td></tr>
<tr><td>&lt;SweepRange&gt;</td><td>String</td><td>Start, stop, and step range with units, for sweep.</td></tr>
<tr><td>&lt;UnitSpec&gt;</td><td>String</td><td>For example, “kV, Mhz, yd”</td></tr>
<tr><td>&lt;UseTraceNumberFormat&gt;</td><td>Boolean</td><td>"True", "False"</td></tr>
</table>
</td></tr>
</table>

Return Value	None.

Python Syntax	ExportUniformPointsToFile(<*ReportName*>, <*FileName*>, <*SweepRange*>)
Python Example	`oModule.ExportUniformPointsToFile("S Parameter Table 2", "D:/MyFiles/cftt.csv", "4GHz", "5GHz", "1GHz", False, " kV, MHz, yd ", False)`

GetAllCategories

Get all available category names (not including variable and output-variables) in a solution for a report type and display type, returned as an array of text strings.

<table>
<tr><td><b>UI Access</b></td><td>NA</td></tr>
<tr><td><b>Parameters</b></td><td><table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;ReportType&gt;</i></td><td>String</td><td>Report type name.</td></tr>
<tr><td><i>&lt;DisplayType&gt;</i></td><td>String</td><td>Name of display type.</td></tr>
<tr><td><i>&lt;SolutionName&gt;</i></td><td>String</td><td>Name of solution.</td></tr>
<tr><td><i>&lt;SimValueCtxt&gt;</i></td><td>Array</td><td>A context name, or array of strings that encode the contexts.</td></tr>
</table></td></tr>
<tr><td><b>Return Value</b></td><td>Array of text strings</td></tr>
</table>

Python Syntax	GetAllCategories(<*ReportType*>, <*DisplayType*>, <*SolutionName*>, <*SimValueCtxt*>)

Python Example	`oModule.GetAllCategories("Far Fields",`
	`"Rectangular Plot", "Setup1 : LastAdaptive", "Infinite Sphere1")`

GetAllQuantities

Gets all available quantity names in category, returned as an array of text strings.

<table>
<tr><td><b>UI Access</b></td><td>NA</td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;ReportType&gt;</i></td><td>String</td><td>Report type name.</td></tr>
<tr><td><i>&lt;DisplayType&gt;</i></td><td>String</td><td>Display type name.</td></tr>
<tr><td><i>&lt;SolutionName&gt;</i></td><td>String</td><td>Name of solution.</td></tr>
<tr><td><i>&lt;SimValueCtxt&gt;</i></td><td>Array</td><td>A context name, or array of string that encoded the contexts(I).</td></tr>
<tr><td><i>&lt;CategoryName&gt;</i></td><td>String</td><td>A category name as input parameter. a category name returned in GetAllCategories() or "Variables", or "Output Variables"</td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>Array of text strings</td></tr>
</table>

Python Syntax	GetAllQuantities(<*ReportType*>,<*DisplayType*>, <*SolutionName*>, <*SimValueCtxt*>, <*CategoryName*>)

Python Example	`oModule.GetAllQuantities(`
	`"Far Fields", "Rectangular Plot",`
	`"Setup1 : LastAdaptive",`
	`"Infinite Sphere1", "Gain")`

GetAllReportNames

Gets the names of existing reports in a design

UI Access	N/A
Parameters	None.
Return Value	Array of report names

Python Syntax	GetAllReportNames()
Python Example	`oModule.GetAllReportNames()`

GetAvailableDisplayTypes

Retrieves all supported display types in report type as an array of text strings.

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><i>&lt;ReportType&gt;</i></td><td>String</td><td>Report type name.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>Array of text strings</td></tr>
</table>

Python Syntax	GetAvailableDisplayTypes(<*ReportType*>)
Python Example	`oModule.GetAvailableDisplayTypes("Far Fields")`

GetAvailableReportTypes

Retrieves all available report types in the current Design as an array of text string.

UI Access	N/A
Parameters	None.
Return Value	Array of text strings

Python Syntax	GetAvailableReportTypes()
Python Example	`oModule.GetAvailableReportTypes()`

GetAvailableSolutions

Gets all available solutions in report type as an array of text strings.

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><i>&lt;ReportType&gt;</i></td><td>String</td><td>Report type name.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>Array of text strings</td></tr>
</table>

Python Syntax	GetAvailableSolutions(<*ReportType*>)
Python Example	`oModule.GetAvailableSolutions("Far Fields")`

GetChildNames [Report Setup]

Gets a list of report names.

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

Python Syntax	GetChildNames()
Python Example	`oModule.GetChildNames()`

GetChildObject [Report Setup]

Gets a report object or report child object; the module’s first level of child object is report. Report has trace, axis, header, Legend, and more children. Trace has curve as child etc. Those child objects can be accessed by calling all levels of parent object’s GetChildObject(path) function.

<table>
<tr><td><b>UI Access</b></td><td>NA</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><i>&lt;ObjectPath&gt;</i></td><td>String</td><td>Report Name or an object path beginning with a report name.</td></tr></table></td></tr>
</table>

Return Value	A ReportSetup(Results) Module Child Object, [ReportSetup(Results) Module Child Objects]

Python Syntax	GetChildObject(<*ObjectPath*>)

Python Example	`oRpt = oRptModule.GetChildObject("S Parameter Plot 1")`
	`oTrace = oRptModule.GetChildObject("S Parameter Plot 1/dB(S(Port1,Port1))")`
	`oAxisX = oRptModule.GetChildObject("S Parameter Plot 1/AxisX")`

GetChildTypes [ReportSetup]

Gets child types of queried Report module object.

UI Access	N/A
Parameters	None.
Return Value	Array of strings containing child object types.

Python Syntax	GetChildTypes ()
Python Example	`oModule.GetChildTypes()`

GetCurvePropServerName

Gets the PropServer (the owner of the properties, or the list containing them) name of a curve.

UI Access	N/A
Parameters	Name | Type | Description *<ReportName>* | String | Name of specified report. *<TraceName>* | String | Name of specified trace.
Return Value	Array of string containing the PropServer name.

Python Syntax	GetCurvePropServerName(<*ReportName*>, <*TraceName*>)
Python Example	`oModule.GetCurvePropServerName("Plot 1", "Phase")`

GetDisplayType

Gets the display type of a specified report.

UI Access	NA
Parameters	Name | Type | Description *<ReportName>* | String | Report name
Return Value	String containing display type.

Python Syntax	GetDisplayType(<*ReportName*>)
Python Example	`oModule.GetDisplayType("Design Plot 1")`

GetDynLinkIntrinsicVariables

Gets variable names from a trace included in dynamic link outputs.

UI Access	N/A
Parameters	<table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;<i>TraceName</i>&gt;</td><td>String</td><td>Trace name with its report name.</td></tr></table>
Return Value	Array of variable names

Python Syntax	GetDynLinkIntrinsicVariables(<*TraceName*>)
Python Example	`oModule.GetDynLinkIntrinsicVariables("Plot 1:Trace")`

GetDynLinkQtyValueState

Gets the state of the quantity values from a source dynamic linked trace.

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

	<TraceName> String Name of specified source trace. *<QtyName>* String Name of specified quantity value.
Return Value	Array of strings containing states.

Python Syntax	GetDynLinkQtyValueState(<*TraceName*>, <*QtyName*>)
Python Example	`oModule.GetDynLinkQtyValueState("Plot 1:Trace1", "")`

GetDynLinkTraces

Gets the names of the dynamic linked traces.

UI Access	N/A
Parameters	Name / Type / Description <*SoluName*> / String / Name of the source solution. If empty, refer to current solution.
Return Value	Array of strings containing trace names.

Python Syntax	GetDynLinkTraces(<*SoluName*>)
Python Example	`oModule.GetDynLinkTraces("")`

GetDynLinkVariableValues

Gets the values of a variable from a dynamic linked trace.

UI Access	N/A
Parameters	<table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;*TraceName*&gt;</td><td>String</td><td>Name of specified dynamic linked trace, with its report name.</td></tr><tr><td>&lt;*VarName*&gt;</td><td>String</td><td>Name of specified variable.</td></tr></table>
Return Value	Array of strings containing variable values.

Python Syntax	GetDynLinkVariableValues(<*TraceName*>, <*VarName*>)
Python Example	`oModule.GetDynLinkVariableValues("Plot 1:Trace", "Var1")`

GetName

Returns the design name of the active design, in that order separated by a semicolon.

UI Access	N/A
Parameters	None.
Return Value	String indicating the name of the active design.

Python Syntax	GetName()
Python Example	design_name = oDesign.GetName()

GetObjPath [Design]

Obtains the path to the design.

UI Access	N/A
Parameters	None.
Return Value	String containing the path to the design.

Python Syntax	GetObjPath()
Python Example	`oDesign.GetObjPath()`

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.

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;<i>PropTab</i>&gt;</td><td>String</td><td>One of the following, where tab titles are shown in parentheses:<ul><li>PassedParameterTab ("Parameter Values")</li><li>DefinitionParameterTab (Parameter Defaults")</li></ul></td></tr></table></td></tr>
</table>

			• LocalVariableTab ("Variables" or "Local Variables") • ProjectVariableTab ("Project variables") • ConstantsTab ("Constants") • BaseElementTab ("Symbol" or "Footprint") • ComponentTab ("General") • Component("Component") • CustomTab ("Intrinsic Variables") • Quantities ("Quantities") • Signals ("Signals")
	<PropServer>	String	An object identifier, generally returned from another script method, such as `CompInst@R;2;3`
	<PropName>	String	Name of the property.
Return Value	String value of the property.		

Python Syntax	GetPropertyValue (<*PropTab*>, <*PropServer*>, <*PropName*>)

Python Example	`selectionArray = oEditor.GetSelections()`
	`for k in selectionArray:`
	`val = oEditor.GetPropertyValue("PassedParameterTab", k, "R")`
	`...`

GetPropNames [Reporter]

Report setup module does not have its own property, this function always returns empty array.

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

Python Syntax	GetPropNames()
Python Example	`oModule.GetPropNames()`

GetPropValue [Report Setup]

Gets the property value for a Report, or reports' child object.

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><i>&lt;PropPath&gt;</i></td><td>String</td><td>A child object's property path. See property path discussion here.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>String containing the property value.</td></tr>
</table>

Python Syntax	GetPropValue(<*PropPath*>)
Python Example	`oModule.GetPropValue(“S Parameter Plot 1/Display Type”)`

GetQtyExpressionsForSourceTrace

Gets the quantity expressions from a specified source trace.

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;<i>SourceTraceName</i>&gt;</td><td>String</td><td>Name of specified source trace.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>Array of strings containing quantity expressions.</td></tr>
</table>

Python Syntax	GetQtyExpressionsForSourceTrace(<*SourceTraceName*>)
Python Example	`oModule.GetQtyExpressionsForSourceTrace("Plot 1:Trace1")`

GetReportTraceNames

Gets the names of existing trace names in a plot.

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;<i>PlotName</i>&gt;</td><td>String</td><td>Name of specified plot.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>Array of strings containing trace names.</td></tr>
</table>

Python Syntax	GetReportTraceNames(<*PlotName*>)
Python Example	`oModule.GetReportTraceNames("SParameter Plot 1")`

GetReportSummaryForRegressionTesting

Gets report summary from a dumped report file.

UI Access	N/A
Parameters	Name | Type | Description <*ReportName*> | String | Name of a specified dumped report.
Return Value	String containing report summary.

Python Syntax	GetReportSummaryForRegressionTesting(<*ReportName*>)
Python Example	`oModule.GetReportSummaryForRegressionTesting("C:/report.rdat")`

GetSolutionContexts

Gets all available solution context names in a solution as an array of text strings.

UI Access	N/A
Parameters	Name | Type | Description <*ReportType*> | String | Report type name. <*DisplayType*> | String | Display type name. <*SolutionName*> | String | Name of solution.

Return Value	Array of text strings

Python Syntax	GetSolutionContexts(<*ReportType*>, <*DisplayType*>, <*SolutionName*>)

Python Example	`oModule.GetSolutionContexts(`
	`"Far Fields", "Rectangular Plot", "Setup1:LastAdaptive")`

GetSolutionDataPerVariation

Obtains solution data for a given report type and solution. You must have already run a simulation.

UI Access	N/A
Parameters	
Return Value	ARRAY of ISolutionDataResultComInterface objects, containing: • GetSweepNames() • GetSweepUnits()

Name	Type	Description
<reportTypeArg>	String	Report type name as input parameter.
<solutionNameArg>	String	Solution name as input parameter.
<simValueCtxtArg>	Structured Array	Same as ContextArray values created in the relevant CreateReport script.
<familiesArg>	Array of Strings	Same as FamiliesArray values created in the relevant CreateReport script.
<expressionArg>	String or Array of Strings	Text string or array of text strings; valid expression, may validate it as the data-table Y-component.

	• GetSweepValues() • IsPerQuantityPrimarySweep() • GetPerQuantityPrimarySweepValues() • IsDataComplex() • GetDataUnits() • GetRealDataValues() • GetImagDataValues() • ReleaseData() • GetDesignVariableNames() • GetDesignVariableUnits() • GetDesignVariableValue() • GetDesignVariationKey()

Note:

- This command is *not* recordable from the UI, but its parameters are similar to CreateReport, so you may record a CreateReport script to get the parameter values.
- For the returned ISolutionDataResultComInterface object, some of its functions have an optional boolean parameter: SIValue. SIValue defaults to True. When the pass in value is True, return data values will be in Standard International values; when False, return data values will be in the current units.

  Example: Freq Sweep with [1GHz, 2GHz,3GHz], GetSweepUnits(“Freq”) return “GHz”; GetSweepValues(“Freq”, True) return [1000000,2000000,3000000]; GetSweepValues(“Freq”, False) return [1,2,3].

Python Syntax	GetSolutionDataPerVariation(reportTypeArg, solutionNameArg, simValueCtxtArg, familiesArg, expressionArg)
Python Example	`oModule = oDesign.GetModule("ReportSetup")`

	`arr = oModule.GetSolutionDataPerVariation('Modal Solution Data', 'Setup1 : Sweep', ['Domain:=', 'Sweep'], ['Freq:=', ['All'], 'offset:=', ['All']], ['S(Port1,Port1)', 'dB(S(Port1,Port3))'])`

GetDataUnits

Returns text string containing units.

> **Important:**
>
> This is a member function of ISolutionDataResultComInterface object, which is the element of the returned array from function GetSolutionDataPerVariation.

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;expressionString&gt;</td><td>String</td><td>Can be returned from <code>GetDataExpressions()</code></td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>Text string of units; empty if no units</td></tr>
</table>

Python Syntax	GetDataUnits(<expressionString>)
Python Example	`oModule.GetDataUnits(expressions)`

GetDesignVariableNames

Returns array of strings containing design variable names.

> **Important:**
>
> This is a member function of ISolutionDataResultComInterface object, which is the element of the returned array from function GetSolutionDataPerVariation.

UI Access	NA
Parameters	NA
Return Value	Array of strings

Python Syntax	GetDesignVariableNames()
Python Example	`names = oModule.GetDesignVariableNames()`

GetDesignVariableUnits

Returns array of strings containing design variable units.

> **Important:**
>
> This is a member function of ISolutionDataResultComInterface object, which is the element of the returned array from function GetSolutionDataPerVariation.

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;varName&gt;</td><td>String</td><td>Can be returned from GetDesignVariableNames()</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>Text string of units; empty if no units.</td></tr>
</table>

Python Syntax	GetDesignVariableUnits(<varName>)
Python Example	units = oModule.GetDesignVariableUnits('Variable Name')

GetDesignVariableValue

Returns a design variable's value.

> **Important:**
>
> This is a member function of ISolutionDataResultComInterface object, which is the element of the returned array from function GetSolutionDataPerVariation.

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;varName&gt;</td><td>String</td><td>Can be returned from GetDesignVariableNames()</td></tr><tr><td>&lt;siValue&gt;</td><td>Boolean</td><td>True to return SI-value; False to return by GetDesignVariableUnits()</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>Double value</td></tr>
</table>

Python Syntax	GetDesignVariableValue(<varName>, <siValue>)
Python Example	`value = oModule.GetDesignVariableValue('varName',1)`

GetDesignVariationKey

Returns a design's Variation Key.

> **Important:**
>
> This is a member function of ISolutionDataResultComInterface object, which is the element of the returned array from function GetSolutionDataPerVariation.

UI Access	N/A
Parameters	N/A
Return Value	String containing variation key.

Python Syntax	GetDesignVariationKey()
Python Example	`oModule.GetDesignVariationKey()`

GetImagDataValues

Returns array of imaginary data values.

> **Important:**
>
> This is a member function of ISolutionDataResultComInterface object, which is the element of the returned array from function GetSolutionDataPerVariation.

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td><table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td>&lt;expressionString&gt;</td><td>String</td><td>Can be returned from <code>GetDataExpressions()</code></td></tr>
<tr><td>&lt;siValue&gt;</td><td>Boolean</td><td>True to return SI-value; False to return with units returned in GetSweepUnits().</td></tr>
</table></td></tr>
<tr><td><b>Return Value</b></td><td>Array of doubles</td></tr>
</table>

Python Syntax	GetImagDataValues(<expressionString>,<siValue>)
Python Example	`imaginaryvalues = oModule.GetImagDataValues('expression',1)`

GetPerQuantityPrimarySweepValues

Returns per quantity primary sweep values.

> **Important:**
>
> This is a member function of ISolutionDataResultComInterface object, which is the element of the returned array from function GetSolutionDataPerVariation.

UI Access	N/A
Parameters	<table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;expressionString&gt;</td><td>String</td><td>Can be returned from `GetDataExpressions()`</td></tr><tr><td>&lt;siValue&gt;</td><td>Boolean</td><td>True to return SI-value; False to return by GetSweepUnits().</td></tr></table>
Return Value	Array of doubles if IsPerQuantityPrimarySweep() returned True; error if returned False

Python Syntax	GetPerQuantitySweepValues(<expressionString>, <siValue>)
Python Example	`sweepvalues = oModule.GetPerQuantitySweepValues('0.111,0.201,0.345,0.231', 1)`

GetRealDataValues

Returns array of real data values.

> **Important:**
>
> This is a member function of `ISolutionDataResultComInterface` object, which is the element of the returned array from function GetSolutionDataPerVariation.

UI Access	N/A
Parameters	<table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;expressionString&gt;</td><td>String</td><td>Can be returned from `GetDataExpressions()`</td></tr><tr><td>&lt;siValue&gt;</td><td>Boolean</td><td>True to return SI-value; False to return with units returned in GetSweepUnits().</td></tr></table>
Return Value	Array of doubles

Python Syntax	GetRealDataValues(<expressionString>,<siValue>)
Python Example	`realvalues = oModule.GetRealDataValues('expression',1)`

GetSweepNames

Returns array of text strings containing primary sweep name(s).

> **Important:**
>
> This is a member function of ISolutionDataResultComInterface object, which is the element of the returned array from function GetSolutionDataPerVariation.

UI Access	N/A
Parameters	N/A
Return Value	Array of text strings

Python Syntax	GetSweepNames()
Python Example	`sweepnames = arr[0].GetSweepNames()`

GetSweepUnits

Returns text string containing units.

> **Important:**
>
> This is a member function of ISolutionDataResultComInterface object, which is the element of the returned array from function GetSolutionDataPerVariation.

UI Access	N/A
Parameters	Name | Type | Description <sweepName> | String | Primary sweep name
Return Value	Text string containing units

Python Syntax	GetSweepUnits(<sweepName>)
Python Example	`sweepunits = oModule.GetSweepUnits('Sweep 1')`

GetSweepValues

Returns sweep values.

> **Important:**
>
> This is a member function of ISolutionDataResultComInterface object, which is the element of the returned array from function GetSolutionDataPerVariation.

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td><table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td>&lt;sweepName&gt;</td><td>String</td><td>Primary sweep name</td></tr>
<tr><td>&lt;siValue&gt;</td><td>Boolean</td><td>True to return SI-value; False to return by GetSweepUnits().</td></tr>
</table></td></tr>
<tr><td><b>Return Value</b></td><td>Array of doubles</td></tr>
</table>

Python Syntax	GetSweepValues(<sweepName>, <siValue>)
Python Example	`sweepvalues = oModule.GetSweepValues('Sweep 1', True)`

IsDataComplex

Returns whether an expression is complex.

> **Important:**
>
> This is a member function of ISolutionDataResultComInterface object, which is the element of the returned array from function GetSolutionDataPerVariation.

UI Access	NA
Parameters	<table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;expressionString&gt;</td><td>String</td><td>Can be returned from `GetDataExpressions()`.</td></tr></table>
Return Value	Boolean (True if expression is Complex data; False if not)

Python Syntax	IsDataComplex(<expressionString>)
Python Example	`oModule.IsDataComplex('.001,.234,.455,.434')`

IsPerQuantityPrimarySweep

Returns whether data expressions have different primary sweep values.

> **Important:**
>
> This is a member function of ISolutionDataResultComInterface object, which is the element of the returned array from function GetSolutionDataPerVariation.

UI Access	N/A

Parameters	N/A
Return Value	Boolean (True if data expressions have different primary sweep values)

Python Syntax	IsPerQuantityPrimarySweep()
Python Example	`var = oModule.IsPerQuantityPrimarySweep()`

Release Data

Releases all cached data. After this function is called, all subsequent function calls to the object will fail.

> **Important:**
>
> This is a member function of ISolutionDataResultComInterface object, which is the element of the returned array from function GetSolutionDataPerVariation.

UI Access	N/A
Parameters	N/A
Return Value	N/A

Python Syntax	ReleaseData()

Python Example	`oModule.ReleaseData()`

GroupPlotCurvesByGroupingStrategy

Groups curves in a Stacked Plot automatically based on a curve grouping strategy.

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td><table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;ReportName&gt;</i></td><td>String</td><td>Name of report.</td></tr>
<tr><td>&lt;GroupStrategy&gt;</td><td>String</td><td>Strategy for grouping, "Single", "By Trace" or "By Units".</td></tr>
</table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	GroupPlotCurvesByGroupingStrategy(<*ReportName*>, <*GroupStrategy*>)
Python Example	`oModule.GroupPlotCurvesByGroupingStrategy("Transient Plot 1", "By Trace")`

ImportIntoReport

Imports .tab, .csv, and .dat format files into a report.

<table>
<tr><td><b>UI Access</b></td><td>Right-click on report name in the Project tree and select <b>Import...</b>.</td></tr>
<tr><td><b>Parameters</b></td><td><table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;ReportName&gt;</i></td><td>String</td><td>Name of the Report</td></tr>
<tr><td><i>&lt;FileName&gt;</i></td><td>String</td><td>Path and File Name</td></tr>
</table></td></tr>
</table>

			.csv	Comma-delimited data file
			.tab	Tab-separated file
			.dat	Ansys plot data file
Return Value	None			

Python Syntax	ImportIntoReport (<*ReportName*>, <*FileName*>)
Python Example	`oDesign.ImportIntoReport ("Plot1", "c:\report1.dat")`

ImportReportDataIntoReport

Imports report data from a file into a specified report.

UI Access	N/A		
Parameters	Name	Type	Description
	<*ReportName*>	String	Name of specified report.
	<*FileName*>	String	Name of specified data file. File extenstion "rdat" is expected.
Return Value	None.		

Python Syntax	ImportReportDataIntoReport(<*ReportName*>, <*FileName*>)

Python Example	`oModule.ImportReportDataIntoReport("Plot 1", "C:/Plot1data.rdat")`

MovePlotCurvesToGroup

In a Stacked Plot move curve(s) from its stack(s) to an existing stack. Here term ‘group’ is synonymous to ‘stack’ in the context of cartesian stacked plot.

UI Access	N/A
Parameters	

Name	Type	Description
<*ReportName*>	String	Name of report.
<*CurveArray*>	Array	Array of curve names to move.
<*StackName*>	String	Name of stack to move to.

Return Value	None.

Python Syntax	MovePlotCurvesToGroup(<*ReportName*>, <*CurveArray*>, <*StackName*>)
Python Example	`oModule.MovePlotCurvesToGroup("XY Stacked Plot 1", ["R2.V : TR", "R2.I : TR"], "Stack 2")`

MovePlotCurvesToNewGroup

Move curve(s) from its stack(s) to a new stack. Here term ‘group’ is synonymous to ‘stack’ in the context of Cartesian stacked plot.

UI Access	N/A

Parameters	<table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><i>&lt;ReportName&gt;</i></td><td>String</td><td>Name of report.</td></tr><tr><td><i>&lt;CurveArray&gt;</i></td><td>Array</td><td>Array of curve names to move.</td></tr></table>
Return Value	None.

Python Syntax	MovePlotCurvesToNewGroup (<*ReportName*>, <*CurveArray*>)

Python Example	`oModule.MovePlotCurvesToNewGroup(`
	`"XY Stacked Plot 1",`
	`["R2.V : TR", "R2.I : TR"])`

OpenWindowForAllReports

Opens windows for all reports belong to current design.

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

Python Syntax	OpenWindowForAllReports()
Python Example	`oModule.OpenWindowForAllReports()`

OpenWindowForReports

Opens windows for specified reports.

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;<i>ReportNames</i>&gt;</td><td>Array</td><td>Array of strings containing report names.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	OpenWindowForReports(<*ReportNames*>)
Python Example	`oModule.OpenWindowForReports(["Report1", "Report2"])`

PastePlotSettings

Paste plot settings to a specified report.

<table>
<tr><td><b>UI Access</b></td><td>Right-click a report, select <b>Paste</b></td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;<i>ReportName</i>&gt;</td><td>String</td><td>Name of specified report.</td></tr><tr><td>&lt;<i>PropTypeToApply</i>&gt;</td><td>String</td><td>Property type to paste. "Graphical", "Data", or "All".</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	PastePlotSettings(<*ReportName*>, <*PropTypeToApply*>)
Python Example	`oModule.PastePlotSettings("Plot 1", "Graphical")`

PasteReports

Paste copied reports to results in the current project.

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

Python Syntax	PasteReports ()
Python Example	`oModule.PasteReports()`

PasteReportsWithLegacyNames

Pastes copied reports to results in the current project with legacy name definitions.

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

Python Syntax	PasteReportsWithLegacyNames ()
Python Example	`oModule.PasteReportsWithLegacyNames()`

PasteTraces

Pastes copied traces to a named plot.

UI Access	Paste
Parameters	Name | Type | Description *<ReportName>* | String | Name of plot
Return Value	None

Python Syntax	PasteTraces (<*ReportName*>)
Python Example	`oModule.PasteTraces ("XY Plot1")`

PasteTracesWithLegacyNames

Pastes copied traces to a named plot using legacy name definitions.

UI Access	N/A

<table>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;<i>ReportName</i>&gt;</td><td>String</td><td>Name of plot</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>None</td></tr>
</table>

Python Syntax	PasteTracesWithLegacyNames (<*ReportName*>)
Python Example	`oModule.PasteTracesWithLegacyNames("XY Plot1")`

RenameReport

Renames an existing report.

<table>
<tr><td><b>UI Access</b></td><td>Select a report on the Project tree, right-click and select <b>Rename</b></td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;<i>OldReportName</i>&gt;</td><td>String</td><td>Old Report Name</td></tr><tr><td>&lt;<i>NewReportName</i>&gt;</td><td>String</td><td>New Report Name</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	RenameReport (<*OldReportName*>, <*NewReportName*>)
Python Example	`oModule.RenameReport("XY Plot1", "Reflection")`

RenameTrace

To rename a trace in a plot

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td><table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;ReportName&gt;</i></td><td>String</td><td>Name of report.</td></tr>
<tr><td><i>&lt;TraceName&gt;</i></td><td>String</td><td>Name of Trace</td></tr>
<tr><td><i>&lt;NewName&gt;</i></td><td>String</td><td>New trace name.</td></tr>
</table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	RenameTrace(<*ReportName*>, <*TraceName*>, <*NewName*>)

Python Example	`oModule.RenameTrace ("XY Plot1",`
	`"dB(S(WavePort1,WavePort1))1", "Port1dbS")`

ResetPlotSettings

Resets plot settings to defaults.

<table>
<tr><td><b>UI Access</b></td><td>Right-click on a plot, select <b>Edit</b> &gt; <b>Reset Plot Settings</b></td></tr>
<tr><td><b>Parameters</b></td><td><table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;PlotName&gt;</i></td><td>String</td><td>Name of specified plot.</td></tr>
</table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	ResetPlotSettings(<*PlotName*>)
Python Example	`oModule.ResetPlotSettings("Differential S-parameters")`

SavePlotSettingsAsDefault

Saves report plot settings as default.

UI Access	**Report Templates** > **Save Settings as Default**
Parameters	<table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><*PlotName*></td><td>String</td><td>Name of plot to use for plot defaults.</td></tr></table>
Return Value	None.

Python Syntax	SavePlotSettingsAsDefault ("<*PlotName*>")
Python Example	`oModule.SavePlotSettingsAsDefault ("XY Plot1")`

SetLinkOutputTraces

Specifies dynamic link output traces from the current design.

UI Access	Right-click the **Results**, select **Link Output...**

<table>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;<i>TraceArray</i>&gt;</td><td>Array</td><td>Array of traces to set.
<code>Array("<ReportName>:=", <array of trace names&gt;,</code>
<code>"&lt;ReportName>:=", <array of trace names&gt;,...)</code></td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

<table>
<tr><td><b>Python Syntax</b></td><td>SetLinkOutputTraces(&lt;<i>TraceArray</i>&gt;)</td></tr>
<tr><td><b>Python Example</b></td><td><code>oModule.SetLinkOutputTraces(</code>
<code>[</code>
<code> "Plot 1:=", ["Trace1"],</code>
<code> "Plot 2:=", ["Trace1"]</code>
<code>])</code></td></tr>
</table>

SetPropValue [Report Setup]

Sets the property value for report module child object.

<table>
<tr><td><b>UI Access</b></td><td>Select <b>Edit Properties</b> on Report objects.</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;<i>PropPath</i>&gt;</td><td>String</td><td>A child object's property path. See property path discussion here.</td></tr><tr><td>&lt;<i>NewValue</i>&gt;</td><td>String, Number, or Boolean</td><td>New value data type is depending on the property type,</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>True if the property is found and the new value is valid. Otherwise return False.</td></tr>
</table>

Python Syntax	SetPropValue(<*PropPath*>, <*NewValue*>)

Python Example	`oRptModule.SetPropValue(“S Parameter Plot 1/Display Type”, “DataTable”)`
	`oRptModule.SetPropValue(“S Parameter Plot 1/db(S(port1,port2)/Primary sweep”, “Freq”)`

UnGroupPlotCurvesInGroup

From a Stacked Plot, ungroups curves in a stack.

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td>&lt;<i>ReportName</i>&gt;</td><td>String</td><td>Name of report.</td></tr>
<tr><td>&lt;<i>GroupName</i>&gt;</td><td>String</td><td>Stack group name.</td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	UnGroupPlotCurvesInGroup(<*ReportName*>, <*GroupName*>)
Python Example	`oModule.UngroupPlotCurvesInGroup("S Parameter Plot 3", "Stack 1")`

UpdateAllReports

Updates all reports in the **Results** branch in the project tree.

UI Access	Right-click on **Results** in the project tree, select **Update All Reports**
Parameters	None
Return Value	None

Python Syntax	UpdateAllReports()
Python Example	`oModule.UpdateAllReports()`

UpdateReports

Updates specified reports.

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;<i>ReportNames</i>&gt;</td><td>Array</td><td>Array of strings containing report names.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	UpdateReports(<*ReportNames*>)
Python Example	`oModule.UpdateReports (["XY Plot 1", "XY Plot 4"])`

UpdateTraces

Update the traces in a report for which traces are not automatically updated by the Report Traces dialog box, Update Report, Real Time selection.

<table>
<tr><th>UI Access</th><th>In <b>Report</b> dialog, click <b>Apply Traces</b> button</th></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td>&lt;<i>ReportName</i>&gt;</td><td>String</td><td>Name of Report.</td></tr>
<tr><td>&lt;<i>TraceNames</i>&gt;</td><td>Array</td><td>Array of strings containing trace names.</td></tr>
<tr><td>&lt;<i>SolutionName</i>&gt;</td><td>String</td><td>Name of the solution.</td></tr>
<tr><td>&lt;<i>ContextArray</i>&gt;</td><td>Array</td><td>Context for which the expression is being evaluated. This can be an empty string if there is no context.
<code>Array("Domain:=", <DomainType&gt;)</code>
<code>&lt;DomainType> ex. "Sweep" or "Time"</code>
<code>Array("Context:=", <GeometryType&gt;)</code>
<code>&lt;GeometryType> ex. "Infinite Sphere<i>n</i>", "Sphere<i>n</i>", "Polyline<i>n</i>"</code></td></tr>
<tr><td>&lt;<i>FamiliesArray</i>&gt;</td><td>Array</td><td>Contains sweep definitions for the report.
<code>Array("<VariableName>:= ", <ValueArray&gt;)</code>
<code>&lt;ValueArray&gt;</code>
<code>Array("All") or Array("Value1", "Value2", ..."Value<i>n</i>")</code>
<code>examples of <VariableName> "Freq", "Theta", "Distance"</code></td></tr>
<tr><td>&lt;<i>ReportDataArray</i>&gt;</td><td>Array</td><td>This array contains the report quantity and X, Y, and (Z) axis definitions.</td></tr>
</table>
</td></tr>
</table>

			`Array("X Component:=", <VariableName>,`
			`"Y Component:=", <VariableName> | <ReportQuantityArray>)`
			`<ReportQuantityArray> ex. Array("dB(S(Port1, Port1))")`

	<ExtTraceInfo>	Array	Optional. Array defines extended trace information.
Return Value	None.		

Python Syntax	UpdateTraces(<*ReportName*>, <*SolutionName*>, <*ContextArray* >, <*FamiliesArray*>, <*ReportDataArray*>)

Python Example	`oModule.UpdateTraces("XY Plot 1", ["NEG1.VAL"], "TR4",`
	`[`
	`"NAME:Context",`
	`"SimValueContext:=", [2,0,2,0,False,False,`
	`-1,1,0,1,1,"",0,0,"CG",False,"0","KP",False,"0","MH",False,`
	`"100","TE",False,"100s","TH",False,"40",`
	`"TS",False,"0ns","UF",False,`
	`"0","WT",False,"0","WW",False,"100"]`
	`],`
	`[`
	`"Spectrum:=", ["All"]`
	`],`

	`[`
	`"X Component:=", "Spectrum",`
	`"Y Component:=", ["mag(NEG1.VAL)"]`
	`], [])`

UpdateTracesContextAndSweeps

Edits sweeps and context of multiple traces without affecting their component expressions.

UI Access	**Modify Report** with multiple traces selected.
Parameters	<table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;<i>ReportName</i>&gt;</td><td>String</td><td>Name of Report.</td></tr><tr><td>&lt;<i>TraceNames</i>&gt;</td><td>Array</td><td>Array of strings containing trace names.</td></tr><tr><td>&lt;<i>SolutionName</i>&gt;</td><td>String</td><td>Name of the solution as listed in the <b>Modify Report</b> dialog box. For example: <code>"Setup1 : Last Adaptive"</code></td></tr><tr><td>&lt;<i>ContextArray</i>&gt;</td><td>Array</td><td>Context for which the expression is being evaluated. This can be an empty string if there is no context. ex. "Sweep" or "Time"</td></tr><tr><td>&lt;<i>PointSet</i>&gt;</td><td>Array</td><td>Point set for the selected traces, for example, X and Y values for the plot.</td></tr></table>
Return Value	None

Python Syntax	UpdateTracesContextAndSweeps(<*ReportName*>, <*TraceNames*>, <*SolutionName*>, <*ContextArray*>,

	<*PointSet*>)

Python Example	`oModule.UpdateTracesContextAndSweeps_`
	`("Active S Parameter Quick Report",`
	`["dB(ActiveS(Port1:1))", "dB(ActiveS(Port2:1))"],`
	`"Setup1 : Sweep1", [],`
	`["Freq:=", ["9GHz", "9.05GHz", "9.1GHz",`
	`  "9.15GHz", "9.2GHz",`
	`  "9.25GHz", "9.3GHz", "9.35GHz",`
	`  "9.4GHz", "9.45GHz", "9.5GHz",`
	`  "9.55GHz",`
	`  "9.6GHz", "9.65GHz", "9.7GHz",`
	`  "9.9GHz", "9.95GHz", "10GHz"],`
	`"offset:=",["All"]])`

This page intentionally left blank.

11 - Boundary and Excitation Module Script Commands

Boundary and excitation commands should be executed by the "BoundarySetup" module.

Set oModule = oDesign.GetModule("BoundarySetup")

Conventions Used in this Chapter

`<BoundName>`

Type: string.

Name of a boundary.

`<AssignmentObjects>`

Type: Array of strings.

An array of object names.

`<AssignmentFaces>`

Type: Array of integers.

An array of face IDs. The ID of a face can be determined through the user interface using the **3D Modeler> Measure> Area** command. The face ID is given in the **Measure Information** dialog box.

`<LineEndPoint>`

```
  Array(<double>, <double>, <double>)
```

The topics for this section include:

General Commands Recognized by the Boundary/Excitations Module

AddAssignmentToBoundary

DeleteAllBoundaries

DeleteAllExcitations

DeleteBoundaries

GetBoundaryAssignment

GetBoundaries

GetBoundariesOfType

GetExcitations

GetExcitationsOfType

GetHybridRegions

GetHybridRegionsOfType

GetNumBoundaries

GetNumBoundariesOfType

GetNumExcitations

GetNumExcitationsOfType

ReassignBoundary

RemoveAssignmentFromBoundary

RenameBoundary

ReprioritizeBoundaries

AddAssignmentToBoundary

Adds a new geometry assignment to a boundary.

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;Assignment&gt;</i></td><td>Array</td><td>Structured array.
<code>Array("Name:<BoundName&gt;",</code>
<code>   "Objects:=", <AssignmentObjects&gt;,</code>
<code>   "Faces:=", <AssignmentFaces&gt;)</code></td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	AddAssignmentToBoundary(<*Assignment*>)

Python Example	`oModule.AddAssignmentToBoundary((`
	`["NAME:PerfE1",`
	`"Faces:=", [12]])`

AutoIdentifyLatticePair

Automatically identifies coupled lattice pair within specified object.

<table>
<tr><td><b>UI Access</b></td><td><b>Boundaries</b> &gt; <b>Assign</b> &gt; <b>Coupled...</b> &gt; <b>Auto Identify Lattice Pair...</b></td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;PlaneName&gt;</i></td><td>String</td><td>Name of relative plane.</td></tr>
<tr><td><i>&lt;ObjectName&gt;</i></td><td>String</td><td>Name of specified object.</td></tr>
</table>
</td></tr>
</table>

Return Value	None.

Python Syntax	AutoIdentifyLatticePair (<*PlaneName*>, <*ObjectName*>)
Python Example	`oModule.AutoIdentifyLatticePair("Global:XY", "Cylinder1")`

AutoIdentifyNets

Use: Automatically identifies nets.

Command: **Q3D Extractor>Nets>Auto Identify Nets**

Syntax: AutoIdentifyNets

Return Value: None

Command: oModule.AutoIdentifyNets

AutoIdentifyPorts

Automatically identifies ports in a terminal design.

UI Access	N/A		
Parameters	Name	Type	Description

	<*FaceIDArray*>	Array	Array of face IDs.
			`Array("NAME:Faces",`
			`<FaceID>, <FaceID>, ...)`

	<*IsWavePort*>	Boolean	• **True** - waveport

			• **False** - lumped port.

	<*ReferenceConductorsArray*>	Array	Structured array.
			`Array("NAME:ReferenceConductors",`
			`<ConductorName>, <ConductorName>, ...)`

	<*BaseNameforCreatedPorts*>	String	Optional. Base name to use for created ports
	<UseConductorNames>	Boolean	Optional. • **True** - use conductor names. • **False** - use port object name as base name
Return Value	None.		

Python Syntax	AutoIdentifyPorts (<*FaceIDArray*>, <*IsWavePort*>, <*ReferenceConductorsArray*>)

Python Example	`oModule.AutoIdentifyPorts(`
	`["NAME:Faces", 52],`
	`True,`
	`["NAME:ReferenceConductors", "Conductor1"],`
	`True)`

AutoIdentifyTerminals

Automatically identifies the terminals within the given ports.

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;ConductorsArray&gt;</i></td><td>Array</td><td>Structured array.
<code>Array("NAME:ReferenceConductors",</code>
<code>&lt;ConductorName>, <ConductorName>, ...)</code></td></tr>
<tr><td><i>&lt;PortNames&gt;</i></td><td>String</td><td>Names of given ports.</td></tr>
<tr><td><i>&lt;UseConductorNames&gt;</i></td><td>Boolean</td><td>• <b>True</b> - use conductor names.
• <b>False</b> - use port object name as base name.</td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	AutoIdentifyTerminals (<*ConductorsArray*>, <*PortNames*>, <*UseConductorNames*>)

Python Example	`oModule.AutoIdentifyTerminals([`
	`"NAME:ReferenceConductors", "Conductor1"],`
	`"WavePort1", True)`

ChangeImpedanceMult

Modifies the port impedance multiplier.

UI Access	**HFSS** > **Excitations** > **Edit Port Impedance Multiplier**.

Parameters	Name	Type	Description
	<MultVal>	Double	New value for the impedance multiplier.
Return Value	None.		

Python Syntax	ChangeImpedanceMult (<*MultVal*>)
Python Example	`oModule.ChangeImpedanceMult(0.5)`

ConvertNportCircuitElementsToPorts

Converts one or more HFSS Circuit Element to one or more ports.

UI Access	Right-click on element and select **Convert to HFSS Ports**.		
Parameters	Name	Type	Description
	<NportName>	Array	Array of Nport names.
Return Value	None.		

Python Syntax	ConvertNportCircuitElementToPorts(<*NportName*>)
Python Example	`oModule.ConvertNportCircuitElementToPorts(["Nport2"])`

CreateNportCircuitElements

Creates an HFSS Circuit Element from one or more ports.

<table>
<tr><td><b>UI Access</b></td><td>Right-click <b>Circuit Elements</b> &gt; <b>Create Single Port Model...</b> or <b>Circuit Elements</b> &gt; <b>Create Multi-Terminal Model...</b></td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><i>&lt;NPortArray&gt;</i></td><td>Array</td><td>Structured array.
Array("NAME:<NPortName>",
"Definition:=", <string>,
<AssignmentArray&gt;)</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	CreateNportCircuitElement (<*NPortArray*>)
Python Example	oModule.CreateNportCircuitElement(["NAME:Nport1", "Definition:=", "Model", ["NAME:Assignments", ["NAME:Model", "Assign:=", "1"]]]) oModule.CreateNportCircuitElement(["NAME:Nport2",

```
   "Definition:=", "Model2",
   ["NAME:Assignments",
     ["NAME:P01__NET179__T1",
     "Assign:=", "2"],
     ["NAME:P02__NET178__T1",
     "Assign:=", "3"],
     ["NAME:P03__NET178__T1",
     "Assign:=", "4"],
     ["NAME:P04__NET179__T1",
     "Assign:=", "5"],
   ]
])
```

DeleteAllBoundaries

Deletes all boundaries.

UI Access	[product] > **Boundaries** > **Delete All**
Parameters	None.
Return Value	None.

Python Syntax	DeleteAllBoundaries()

Python Example	`oModule.DeleteAllBoundaries()`

DeleteAllExcitations

Deletes all excitations.

UI Access	[product] > **Excitations** > **Delete All**
Parameters	None.
Return Value	None.

Python Syntax	DeleteAllExcitations()
Python Example	`oModule.DeleteAllExcitations()`

DeleteBoundaries

Deletes the specified boundaries and excitations.

<table>
<tr><td><b>UI Access</b></td><td><b>Delete</b> command in the <b>List</b> dialog box. Click <b>[product]</b> > <b>List</b> to open the <b>List</b> dialog box.</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><i><NameArray></i></td><td>Array</td><td>Array of boundary condition names.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	DeleteBoundaries(<*NameArray*>)
Python Example	`oModule.DeleteBoundaries(["PerfE1", "WavePort1"])`

EditNportCircuitElement

Edits an HFSS Circuit Element.

UI Access	N/A
Parameters	<table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><i>&lt;NPortArray&gt;</i></td><td>Array</td><td>Structured array. <code>Array("NAME:<NPortName&gt;",</code> <code> "Definition:=", <string&gt;,</code> <code> <AssignmentArray&gt;)</code></td></tr></table>
Return Value	None.

Python Syntax	EditNportCircuitElement(<*NPortArray*>)

Python Example	`oModule.EditNportCircuitElement(`
	`["NAME:Nport1",`
	`  "Definition:=", "Model",`
	`  ["NAME:Assignments",`
	`    ["NAME:Model",`
	`    "Assign:=", "1"`

	`    ]`
	`  ]`
	`])`

GetBoundaries

Gets boundary names in the current design.

UI Access	N/A
Parameters	None.
Return Value	Array of boundary names.

Python Syntax	GetBoundaries()
Python Example	`oModule.GetBoundaries()`

GetBoundariesOfType

Gets boundary names of the given type.

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><i>&lt;BoundaryType&gt;</i></td><td>String</td><td>Name of boundary type.</td></tr></table></td></tr>
</table>

Return Value	Array of boundary names of the given type.

Python Syntax	GetBoundariesOfType(<*BoundaryType*>)
Python Example	`oModule.GetBoundariesOfType("PerfectE")`

GetBoundaryAssignment

Gets a list of face IDs associated with the given boundary or excitation assignment.

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;<i>BoundaryName</i>&gt;</td><td>String</td><td>Name of the specified boundary or excitation.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>Array of face IDs or object IDs.</td></tr>
</table>

Python Syntax	GetBoundaryAssignment(<*BoundaryName*>)
Python Example	`oModule.GetBoundaryAssignment("Rad1")`

GetDefaultBaseName

Gets the default base name for boundaries for a project.

UI Access	N/A

Parameters	<table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>*<BoundaryType>*</td><td>String</td><td>Name of legal boundary type.</td></tr></table>
Return Value	String of boundary default base name.

Python Syntax	GetDefaultBaseName(<*BoundaryType*>)
Python Example	`oModule.GetDefaultBaseName("Radiation")`

GetDiffPairs

Gets the names of differential pairs defined.

UI Access	N/A
Parameters	None.
Return Value	Array of differential pair names.

Python Syntax	GetDiffPairs()
Python Example	`oModule.GetDiffPairs()`

GetExcitations

Gets excitation port and terminal names for a model.

UI Access	N/A
Parameters	None.
Return Value	Array of excitation name paired with excitation type.

Python Syntax	GetExcitations()
Python Example	`oModule.GetExcitations()`

GetExcitationsOfType

Gets excitation names of the given type.

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><i>&lt;ExcitationType&gt;</i></td><td>String</td><td>Name of excitation type.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>Array of excitation names of the given type.</td></tr>
</table>

Python Syntax	GetExcitationsOfType(<*ExcitationType*>)
Python Example	`oModule.GetExcitationsOfType("Wave Port")`

GetHybridRegions

Gets hybrid region names for a project.

UI Access	N/A
Parameters	None.
Return Value	Array of hybrid region names.

Python Syntax	GetHybridRegions ()
Python Example	`oModule.GetHybridRegions()`

GetHybridRegionsOfType

Gets hybrid region names of the given type.

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><i>&lt;HybridRegionType&gt;</i></td><td>String</td><td>Name of legal hybrid region type.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>Array of hybrid region names of the given type.</td></tr>
</table>

Python Syntax	GetHybridRegionsOfType(<*HybridRegionType*>)
Python Example	`oModule.GetHybridRegionsOfType("FE-BI")`

GetNumBoundaries

Gets the number of boundaries in a design.

UI Access	N/A
Parameters	None.
Return Value	Integer number of boundaries.

Python Syntax	GetNumBoundaries()
Python Example	`oModule.GetNumBoundaries()`

GetNumBoundariesOfType

Gets the number of boundaries of the given type.

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><i>&lt;BoundaryType&gt;</i></td><td>String</td><td>Specified boundary type.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>Integer number of boundaries.</td></tr>
</table>

Python Syntax	GetNumBoundariesOfType(<*BoundaryType*>)
Python Example	`oModule.GetNumBoundariesOfType("PerfectE")`

GetNumExcitations

Gets the number of excitations in a design.

UI Access	N/A
Parameters	None.
Return Value	Integer number of excitations.

Python Syntax	GetNumExcitations()
Python Example	`oModule.GetNumExcitations()`

GetNumExcitationsOfType

Gets the number of excitations of the given type, including all defined modes and terminals of ports.

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;<i>ExcitationType</i>&gt;</td><td>String</td><td>Specified type of excitation.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>Integer number of excitations.</td></tr>
</table>

Python Syntax	GetNumExcitationsOfType(<*ExcitationType*>)

Python Example	`oModule.GetNumExcitationsOfType("Voltage")`

GetNumHybridRegions

Gets number of hybrid regions in a design.

UI Access	N/A
Parameters	None.
Return Value	Integer number of hybrid regions.

Python Syntax	GetNumHybridRegions()
Python Example	`oModule.GetNumHybridRegions()`

GetNumHybridRegionsOfType

Gets number of hybrid regions of the given type.

<table>
<tr><td><b>UI Access</b></td><td>placeholder</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;<i>HybridRegionType</i>&gt;</td><td>String</td><td>Name of legal hybrid region type.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>Integer number of hybrid regions.</td></tr>
</table>

Python Syntax	GetNumHybridRegionsOfType(<*HybridRegionType*>)

Python Example	`oModule.GetNumHybridRegionsOfType("FE-BI")`

GetPortExcitationCounts

Gets all port names and corresponding number of modes/terminals for each port excitation.

UI Access	N/A
Parameters	None.
Return Value	Array of port names and corresponding mode/terminal counts.

Python Syntax	GetPortExcitationCounts()
Python Example	`oModule.GetPortExcitationCounts()`

ReassignBoundary

Specifies a new geometry assignment for a boundary.

<table>
<tr><td><b>UI Access</b></td><td>Right-click <b>Boundaries</b> &gt; <b>Reassign</b> or <b>Excitations</b> &gt; <b>Reassign</b></td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><i>&lt;AssignmentArray&gt;</i></td><td>Array</td><td>Structured array.
<code>Array("Name:<BoundName&gt;",</code>
<code> "Objects:=", <AssignmentObjects&gt;,</code></td></tr></table></td></tr>
</table>

<table>
<tr><td></td><td><table><tr><td></td><td></td><td>"Faces:=", <AssignmentFaces&gt;)</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	ReassignBoundary(<*AssignmentArray*>)

Python Example	`oModule.ReassignBoundary(`
	`[`
	`"NAME:PerfH12",`
	`"Faces:=", [17]`
	`])`

RenameBoundary

Renames a boundary or excitation.

<table>
<tr><td><b>UI Access</b></td><td>Right-click a boundary in the project tree, and then click <b>Rename</b> on the shortcut menu.</td></tr>
<tr><td><b>Parameters</b></td><td><table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td>&lt;<i>OldName</i>&gt;</td><td>String</td><td>Name of the boundary to be renamed.</td></tr>
<tr><td>&lt;<i>NewName</i>&gt;</td><td>String</td><td>New name for the boundary.</td></tr>
</table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	RenameBoundary(<*OldName*>, <*NewName*>)

Python Example	`oModule.RenameBoundary("Rad2", "Rad3")`

ReprioritizeBoundaries

Specifies the order in which the boundaries and excitations are recognized by the solver. The first boundary in the list has the highest priority.

Note: this command is only valid if all defined boundaries and excitations appear in the list. All ports must be listed before any other boundary type.

<table>
<tr><td><b>UI Access</b></td><td>[<b>product</b>] &gt; <b>Boundaries</b> &gt; <b>Reprioritize</b></td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;<i>NewOrderArray</i>&gt;</td><td>Array</td><td>Structured array.
<code>Array("NAME:NewOrder",</code>
<code>&lt;BoundName>, <BoundName&gt;, ...)</code></td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	ReprioritizeBoundaries(<*NewOrderArray*>)

Python Example	`oModule.ReprioritizeBoundaries(`
	`  ["NAME:NewOrder",`
	`  "Imped1",`
	`  "PerfE1",`
	`  "PerfH1"])`

SetDefaultBaseName

Sets the default base name for boundaries for a project.

UI Access	N/A
Parameters	<table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><i><BoundaryType></i></td><td>String</td><td>Name of legal boundary type.</td></tr><tr><td><i><DefaultName></i></td><td>String</td><td>Default name for boundaries of specified type.</td></tr></table>
Return Value	None.

Python Syntax	SetDefaultBaseName(<*BoundaryType*>, <*DefaultName*>)
Python Example	`oModule.SetDefaultBaseName("Radiation", "RadBnd")`

Script Commands for Creating and Modifying Boundaries

Following are script commands for creating and modifying boundaries that are recognized by the BoundarySetup module. In the following commands, all named data can be included or excluded as desired and may appear in any order.

AssignRadiation

AssignRadiation

AssignScreeningImpedance

AssignSymmetry

AssignTerminal

AssignVoltage

AssignWavePort

AutoCreatePECCapForWavePort

CircuitPortToLumpedPort

EditCircuitPort [HFSS]

EditCurrent

EditDiffPairs

EditFiniteCond

EditGradientSurfaceRoughness

EditHalfSpace

EditHybridRegion

EditImpedance

EditIncidentWave

EditInteriorNearFieldSource

EditLayeredImpedance

EditPrimary

EditPerfectE

EditPerfectH

EditLumpedPort

EditLumpedRLC

EditMagneticBias

EditNPortCircuitElement

EditRadiation

EditRFDischargeDCBias

EditSymmetry

EditTerminal

EditVoltage

EditWavePort

LumpedPortToCircuitPort

SetHybridRegionCoupledGroup

SetSBRSources

SetSBRSourcesBlockage

SetSBRWedgeSettings

SetTerminalReferenceImpedances

SwapCircuitPortAssignment

AssignCircuitPort

Assigns a circuit port for a driven terminal or driven modal design in HFSS.

<table>
<tr><th>UI Access</th><th><b>HFSS</b> &gt; <b>Excitations</b> &gt; <b>Assign</b> &gt; <b>Circuit Port...</b></th></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td><i>&lt;CircuitPortArray&gt;</i></td><td>Array</td><td>Structured array.
Array("NAME:<PortName>",
  "Edges:=", [n1, n2],
  "Impedance:=", "valueohm",
  "DoDeembed:=", <boolean>
  "RenormalizeAllTerminals:=", <boolean>
  "TerminalIDLit:=", Array()
)</td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	AssignCircuitPort (<*CircuitPortArray*>)
Python Example	oModule.AssignCircuitPort(["NAME:1", "Edges:=" , [50,56], "Impedance:=" , "50ohm",

	"DoDeembed:=" , False, "RenormalizeAllTerminals:=", True, "TerminalIDList:=" , []])

AssignCurrent

Creates a current source.

<table>
<tr><th>UI Access</th><th>HFSS > Excitations > Assign > Current</th></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;CurrentArray&gt;</i></td><td>Array</td><td>Structured array.
Array("NAME:<BoundName>",
"Objects:=", <AssignmentObjects>,
"Current:=", <value>,
<DirectionArray>,
"Faces:=", <AssignmentFaces>)
"TerminalIDLit:=", Array())</td></tr>
<tr><td><i>&lt;DirectionArray&gt;</i></td><td>Array</td><td>Structured array.
Array("NAME:Direction",
"Start:=", <LineEndPoint>,
"End:=", <LineEndPoint&gt;)</td></tr>
</table>
</td></tr>
</table>

Return Value	None.

Python Syntax	AssignCurrent (<*CurrentArray*>)

Python Example	`oModule.AssignCurrent(`
	`["NAME:Current1",`
	`  "Current:=", "1000mA",`
	`  ["NAME:Direction",`
	`    "Start:=", [-0.4, 0.4, -1.6],`
	`    "End:=", [-0.4, 0.4, 0]],`
	`  "Faces:=", [12]])`

AssignCylindricalWave

Creates an incident Cylindrical wave

<table>
<tr><th>UI Access</th><th><b>HFSS</b> &gt; <b>Excitations</b> &gt; <b>Assign</b> &gt; <b>Incident Wave</b> &gt; <b>Cylindrical Wave</b>.</th></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;Parameters&gt;</i></td><td>Array</td><td>Structured array.
<code> Array("NAME:IncCWave1",</code>
<code> "Objects:=" , <Array of objects&gt;,</code>
<code> "IsCartesian:=" , <boolean&gt;,</code>
<code> "EoX:=" , <string of integer value&gt;,</code></td></tr>
</table>
</td></tr>
</table>

			"EoY:=" , <string of integer value>, "EoZ:=" , <string of integer value>, "kX:=" , <string of integer value>, "kY:=" , <string of integer value>, "kZ:=" , <string of integer value>, "OriginX:=" , <string of integer value>, "OriginY:=" , <string of integer value>, "OriginZ:=" , <string of integer value>, "CylinderRadius:=" , <string of float value>)
Return Value	None.		

Python Syntax	AssignCylindricalWave (<*Parameters*>)
Python Example	oModule.AssignCylindricalWave(["NAME:IncCWave1", "Objects:=" , ["Cylinder1"], "IsCartesian:=" , True, "EoX:=" , "1", "EoY:=" , "0", "EoZ:=" , "0",

	"kX:=" , "0", "kY:=" , "0", "kZ:=" , "1", "OriginX:=" , "0mm", "OriginY:=" , "0mm", "OriginZ:=" , "0mm", "CylinderRadius:=" , "0.25in"])

AssignDielectricCavity

Assigns a Hybrid Region as a Dielectric Cavity.

<table>
<tr><td><b>UI Access</b></td><td><b>HFSS</b> &gt; <b>Hybrid</b> &gt; <b>Assign Hybrid</b> &gt; <b>Dielectric Cavity</b>.</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><i>&lt;Parameters&gt;</i></td><td>Array</td><td>Structured array.
Array("NAME:<cavity name>",
"Objects:=", <array of objects&gt;)</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	AssignDielectricCavity(<*Parameters*>)
Python Example	oModule.AssignDielectricCavity(

	`["NAME:Cavity1",`
	`"Objects:=", ["Cylinder1"]])`

AssignFEBI

Assigns a Hybrid Region as a FEBI.

UI Access	**HFSS** > **Hybrid** > **Assign Hybrid** > **FE-BI...**.
Parameters	
Return Value	None.

Name	Type	Description

<Parameters>	Array	Structured array.
		`Array("NAME:<name of the FEBI Hybrid Region.>",`
		`"Objects:=", <array of names for the geometry assigned as FEBI Hybrid Region>)`

Python Syntax	AssignFEBI(<*Parameters*>)

Python Example	`oModule.AssignFEBI(`
	`["NAME:FE-BI1",`
	`"Objects:=", ["RegularPolyhedron1"]])`

AssignFiniteCond

Assigns a single finite conductivity boundary on selected edges.

<table>
<tr><th>UI Access</th><th><b>2D Extractor</b> > <b>Boundary</b> > <b>Assign</b> > <b>Finite Conductivity</b>.</th></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><Parameters></td><td>Array</td><td>Structured array.
Array("NAME:<name of the boundary.>",
"Edges:=", <array of edge ids>,
"Roughness:=", "<string of double with units of length>",
"UseCoating:=", <boolean>,
"LayerThickness:=", "<string of double with units of length>",
"UseMaterial:=", <boolean>,
"Material:=", "<string material name for coating>",
"Radius:=", <string of double with units of length>,
"Ratio:=", <string of double>)</td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	AssignFiniteCond(<*Parameters*>)
Python Example	Example for the Hammerstad-Jensen surface roughness model. `oModule.AssignFiniteCond`

	["NAME:FiniteCond1", "Edges:=", [7,9], "Roughness:=", "2um", "UseCoating:=", True, "LayerThickness:=", "1.2um", "UseMaterial:=", True, "Material:=", "Copper"] Example for the Hurray surface roughness model oModule.AssignFiniteCond ["NAME:FiniteCond1", "Edges:=", [7], "UseCoating:=", False, "Radius:=", "0.5um", "Ratio:=", "2.9"]

AssignFloquet

Creates a Floquet port.

<table>
<tr><th>UI Access</th><th><b>HFSS</b> > <b>Excitations</b> > <b>Assign</b> > <b>Floquet</b>.</th></tr>
<tr><td><b>Parameters</b></td><td><table><tr><th>Name</th><th>Type</th><th>Description</th></tr><tr><td><i>&lt;FloquetPortArray&gt;</i></td><td>Array</td><td>Structured array.</td></tr></table></td></tr>
</table>

		Array("NAME:<BoundName>", "Faces:=", <array of face IDs>, "NumModes:=", <integer>, "RenormalizeAllTerminals:=", <boolean>, "DoDeembed:=", <boolean>, <ModesArray>, "ShowReporterFilter:=", <boolean>, "UseScanAngles:=", <boolean>, "Phi:=", "<numdeg>", "Theta:=", "<numdeg>", <LatticeAVector>, <LatticeBVector>, <ModesCalculator>, <ModesList>)
<ModesArray>	Array	Structured array. Array("NAME:Modes", Array("NAME:<ModeName>", "ModeNum:=", <integer>, "UseIntLine:=", <boolean>), ...)
<LatticeAVector>	Array	Structured array. Array("NAME:LatticeAVector", "Start:=", Array("<num><units>", "<num><units>",

		"<num><units>"), "End:=", Array("<num><units>", "<num><units>", "<num><units>"))
<LatticeBVector>	Array	Structured array. Array("NAME:LatticeBVector", "Start:=", Array("<num><units>", "<num><units>", "<num><units>"), "End:=", Array("<num><units>", "<num><units>", "<num><units>"))
<ModesCalculator>	Array	Structured array. Array("NAME:ModesCalculator", "Frequency:=", "<Value>GHz", "FrequencyChanged:=", <Boolean>, "PhiStart:=", "<num>deg", "PhiStop:=", "<num>deg", "PhiStep:=", "<num>deg", "ThetaStart:=", "<num>deg", "ThetaStop:=", "<num>deg", "ThetaStep:=", "<num>deg")
<ModesList>	Array	Structured array. Array("NAME:ModesList", Array("NAME:Mode",

			"ModeNumber:=", <ModeID>, "IndexM:=", <integer index>, "IndexN:=", <integer index>, "KC2:=", <integer value>, "PropagationState:=", "Propagating", "Attenuation:=", <integer value>, "PolarizationState:=", <TE or TM>, "AffectsRefinement:=", <boolean>), ...)
Return Value	None.		

Python Syntax	AssignFloquetPort (<*FloquetPortArray*>)
Python Example	oModule.AssignFloquetPort(["NAME:FloquetPort1", "Faces:=", [7], "NumModes:=", 2, "RenormalizeAllTerminals:=", True, "DoDeembed:=", False, ["NAME:Modes", ["NAME:Mode1", "ModeNum:=",1,

```
        "UseIntLine:=", False],
        ["NAME:Mode2",
        "ModeNum:=", 2,
        "UseIntLine:=", False]],
"ShowReporterFilter:=", False,
"UseScanAngles:=", True, "Phi:=", "0deg", "Theta:=", "0deg",
["NAME:LatticeAVector",
        "Start:=", ["0mm", "0mm", "0.8mm"],
        "End:=", ["0mm", "0.6mm", "0.8mm"]],
["NAME:LatticeBVector",
        "Start:=", ["0mm", "0mm", "0.8mm"],
        "End:=", ["0.8mm", "0mm", "0.8mm"]],
["NAME:ModesCalculator",
        "Frequency:=", "1GHz",
        "FrequencyChanged:=", False,
        "PhiStart:=", "0deg",
        "PhiStop:=", "0deg",
        "PhiStep:=","0deg",
        "ThetaStart:=", "0deg",
        "ThetaStop:=", "0deg",
        "ThetaStep:=", "0deg"],
["NAME:ModesList",
        ["NAME:Mode",
        "ModeNumber:=", 1,
        "IndexM:=", 0,
        "IndexN:=", 0,
        "KC2:=", 0,
        "PropagationState:=", "Propagating",
        "Attenuation:=", 0,
        "PolarizationState:=", "TE",
        "AffectsRefinement:=", False],
["NAME:Mode",
        "ModeNumber:=", 2,
        "IndexM:=", 0,
        "IndexN:=", 0,
        "KC2:=", 0,
        "PropagationState:=", "Propagating",
        "Attenuation:=", 0,
        "PolarizationState:=", "TM",
        "AffectsRefinement:=", False]
]])
```

AssignFresnel

Assigns a Fresnel boundary condition, allows specifying either a perfect absorber or to import a Fresnel table file that describes reflection and transmission coefficients.

<table>
<tr><th>UI Access</th><th><b>HFSS</b> > <b>Boundaries</b> > <b>Assign</b> > <b>Fresnel (SBR+)...</b></th></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;ArgArray&gt;</i></td><td>Array</td><td>Structured array.
<code>Array("NAME:<FresnelName&gt;",</code>
<code>"Faces:=", <array of face IDs&gt;,</code>
<code>"Fresnel Boundary Type:=", <PerfectAbsorber or ImportFromTableFile&gt;,</code>
<code>"RTTable Path:=", <string path to table file&gt;)</code></td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	AssignFresnel(<*ArgArray*>)

Python Example	`oModule.AssignFresnel(["NAME:Fresnel2",`
	`"Faces:=", [8],`
	`"Fresnel Boundary Type:=", "ImportFromTableFile",`
	`"RTTable Path:=", "C:\\temp\reflection-transmission-table.rttbl"`
	`])`

AssignGaussianBeam

Assigns a Gaussian Beam type of incident wave excitation.

<table>
<tr><td><b>UI Access</b></td><td><b>HFSS</b> &gt; <b>Excitations</b> &gt; <b>Assign</b> &gt; <b>Incident Wave</b> &gt; <b>Gaussian Beam...</b></td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td><i>&lt;IncidentWaveArray&gt;</i></td><td>Array</td><td>Structured array.
Array("NAME:<Name of incident wave>",
"Faces:=", <array of face IDs>,
"IsCartesian:=", <boolean>,
"EoX:=" , "<numeric value>",
"EoY:=" , "<numeric value>",
"EoZ:=" , "<numeric value>",
"kX:=" , "<numeric value>",
"kY:=" , "<numeric value>",
"kZ:=" , "<numeric value>",
"PhiStart:=",<value>,
"PhiStop:=", <value>,
"PhiPoints:=", <int>,
"ThetaStart:=", <value>,
"ThetaStop:=", <value>,
"ThetaPoints:=", <int&gt;,</td></tr>
</table>
</td></tr>
</table>

			"EoPhi:=", <value>, "EoTheta:=", <value>, "OriginX:=" , "<numUnit>", "OriginY:=" , "<numUnit>", "OriginZ:=" , "<numUnit>", "BeamWidth:=" , "<numUnit>") IsCartesian If true, provide the EoX, EoY, EoZ, kX, kY, kZ parameters. If false, provide the PhiStart, PhiStop, PhiPoints, ThetaStart, ThetStop, ThetaPoints, EoPhi, EoTheta parameters.
Return Value	None.		

Python Syntax	AssignGaussianBeam(<*IncidentWaveArray*>)
Python Example	oModule.AssignGaussianBeam(["NAME:IncGBeam1", "Faces:=" , [9], "IsCartesian:=" , True, "EoX:=" , "1", "EoY:=" , "0",

```
"EoZ:="                 , "0",
"kX:="                  , "0",
"kY:="                  , "0",
"kZ:="                  , "1",
"OriginX:="             , "0mm",
"OriginY:="             , "0mm",
"OriginZ:="             , "0mm",
"BeamWidth:="           , "10mm"
])
oModule.AssignGaussianBeam(
["NAME:IncGBeam2",
"Faces:="               , [9],
"IsCartesian:="         , False,
"PhiStart:="            , "0deg",
"PhiStop:="             , "0deg",
"PhiPoints:="           , 1,
"ThetaStart:="          , "0deg",
"ThetaStop:="           , "0deg",
"ThetaPoints:="         , 1,
"EoPhi:="               , "1",
"EoTheta:="             , "0",
```

	`"OriginX:=" , "0mm",`
	`"OriginY:=" , "0mm",`
	`"OriginZ:=" , "0mm",`
	`"BeamWidth:=" , "10mm"`
	`])`

AssignGradientSurfaceRoughness

Assigns a Gradient Surface Roughness boundary.

UI Access	**HFSS** > **Boundaries** > **Assign** > **Gradient Surface Roughness...**
Parameters	Name | Type | Description

<table>
<tr><td></td><td><i><ArgArray></i></td><td>Array</td><td>Structured array.
Array(NAME:<Name of GradientSurfaceBoundary>",
[
"NAME:<Name of GradientSurfaceBoundary>",
"Objects:=" , ["<geometryID>"],
"RMS Rougheness:=" , "<value><units>",
"Bulk Conductivity:=" , "<value>",
"Thickness:=" , ""<value><units>",
"Max Frequency:=" , ""<value><units>",
"Surface Type:=" , "[Low Profile | High Profile]"</td></tr>
<tr><td><b>Return Value</b></td><td colspan="3">None.</td></tr>
</table>

Python Syntax	AssignGradientSurfaceRoughness(<*ArgArray*>)
Python Example	oModule = oDesign.GetModule("BoundarySetup") oModule.AssignGradientSurfaceRoughness(["NAME:GradientImped1", "Objects:=" , ["Box1"], "RMS Rougheness:=" , "1um", "Bulk Conductivity:=" , "58000000",

	"Thickness:=" , "15um",
	    "Max Frequency:=" , "100GHz",
	    "Surface Type:=" , "Low Profile"
	  ])

AssignHalfSpace

Assigns a Half Space boundary, dividing the background material at a specified Z axis point. You also assign a material, typically to the lower half.

<table>
<tr><th>UI Access</th><th><b>HFSS</b> &gt; <b>Boundaries</b> &gt; <b>Assign</b> &gt; <b>Half Space...</b></th></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td><i>&lt;ArgArray&gt;</i></td><td>Array</td><td>Structured array.
<code>Array("NAME:<Name of Half Space&gt;",</code>
<code>"ZLocation:=", "<intUnits&gt;",</code>
<code>"Material:=", "<string material name&gt;")</code></td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	AssignHalfSpace(<*ArgArray*>)

Python Example	`oModule.AssignHalfSpace(["NAME:HalfSpace1",`
	`"ZLocation:=", "2mm",`

	"Material:=", "water_sea"])

AssignHertzianDipoleWave

Assigns an incident Hertzian-Dipole wave as excitation.

<table>
<tr><td><b>UI Access</b></td><td><b>HFSS</b> > <b>Excitations</b> > <b>Assign</b> > <b>Incident Wave</b> > <b>Hertzian-Dipole Wave...</b></td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;ArgArray&gt;</i></td><td>Array</td><td>Structured array.
Array("NAME:<Name of incident wave>",
"Faces:=", <array of face IDs>,
"IsCartesian:=", <boolean>,
"EoX:=" , "<numeric value>",
"EoY:=" , "<numeric value>",
"EoZ:=" , "<numeric value>",
"kX:=" , "<numeric value>",
"kY:=" , "<numeric value>",
"kZ:=" , "<numeric value>",
"OriginX:=" , "<numUnit>",
"OriginY:=" , "<numUnit>",
"OriginZ:=" , "<numUnit>",
"SphereRadius:=" , "<numUnit>",
"IsElectricDipole:=" , <boolean&gt;</td></tr>
</table>
</td></tr>
</table>

	<table><tr><td></td><td></td><td>)</td></tr></table>
Return Value	None.

<table>
<tr><td><b>Python Syntax</b></td><td>AssignHertzianDipoleWave(&lt;<i>ArgArray</i>&gt;)</td></tr>
<tr><td><b>Python Example</b></td><td><pre>
oModule.AssignHertzianDipoleWave(
["NAME:IncHDWave2",
"Faces:="   , [9],
"IsCartesian:="   , True,
"EoX:="   , "0",
"EoY:="   , "0",
"EoZ:="   , "1",
"kX:="   , "0",
"kY:="   , "0",
"kZ:="   , "1",
"OriginX:="   , "0mm",
"OriginY:="   , "0mm",
"OriginZ:="   , "0mm",
"SphereRadius:="   , "10mm",
"IsElectricDipole:="   , True
</pre></td></tr>
</table>

	])

AssignHybridRegion

Assigns a Hybrid Region to a conductor, one of IE, PO, or SBR.

<table>
<tr><th>UI Access</th><th><b>HFSS</b> &gt; <b>Hybrid</b> &gt; <b>Assign Hybrid</b> &gt; <b>IE Region...</b> or <b>PO Region...</b> or <b>SBR+ Region...</b>.</th></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;ArgArray&gt;</i></td><td>Array</td><td>Structured array.
Array("NAME:<name of hybrid region>",
"Objects:=", <array of names for the geometry assigned as Hybrid Region>,
"Type:=" , <string one of "IE", "PO" or "SBR">,
"IsLinkedRegion:=" , <boolean>,
"ConductivityThreshold:=", "<numUnit>"<br>)</td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	AssignHybridRegion(<*ArgArray*>)
Python Example	oModule.AssignHybridRegion(["NAME:Hybrid1", "Objects:=" , ["Cylinder1"],

	"Type:=" , "IE", "IsLinkedRegion:=" , False, "ConductivityThreshold:=", "20000S_per_m"])

AssignInteriorNearFieldSource

Assigns an interior near field source as an external data file and associated coordinate system.

<table>
<tr><th>UI Access</th><th>HFSS > Boundaries > Assign > Linked Field>Near Field...</th></tr>
<tr><td><b>Parameters</b></td><td><table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;ArgArray&gt;</i></td><td>Array</td><td>Structured array.
("NAME:<Name of Near Field Source data>",
"Type:=" , "Incident",
"ExternalDataFile:=" , "<filePath>.and",
"SourceCoordSystem:=" , "<CS_name&gt;"</td></tr>
</table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	AssignInteriorNearFieldSource(<*ArgArray*>)

Python Example	oDesign = oProject.SetActiveDesign("internal")
	oModule = oDesign.GetModule("BoundarySetup")
	oModule.AssignInteriorNearFieldSource(

	`[`
	`"NAME:ExtNFData2",`
	`"Type:=" , "Incident",`
	`"ExternalDataFile:=" , "D:\\Ansoft\\Sphere_EH.and",`
	`"SourceCoordSystem:=" , "Global"`
	`])`

AssignImpedance

Creates an impedance boundary for an HFSS design.

UI Access	**HFSS** > **Boundaries** > **Assign** > **Impedance...**

Parameters	<table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>*<ImpedanceArray>*</td><td>Array</td><td>Structured array.
	`Array("NAME:<BoundName>",`
	`"Resistance:=", <value>,`
	`"Reactance:=", <value>,`
	`"InfGroundPlane:=", <boolean>,`
	`"Objects:=", <AssignmentObjects>,`
	`"Faces:=", <AssignmentFaces>)`
	`)`</td></tr></table>

Return Value	None.

Python Syntax	AssignImpedance(<*ImpedanceArray*>)

Python Example	`oModule.AssignImpedance(["NAME:Imped1",`
	`"Resistance:=", "50",`
	`"Reactance:=", "50",`
	`"InfGroundPlane:=", False,`
	`"Faces:=", [12]]`
	`)`

AssignIncidentWave

Creates an incident wave excitation.

<table>
<tr><th>UI Access</th><th>N/A</th></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td><i>&lt;IncidentWaveArray&gt;</i></td><td>Array</td><td>Structured array.
Array("NAME:<BoundName>",
"Faces:=", <array of face IDs>,
"IsCartesian:=",<boolean>,
"EoX:=", <value>,
"EoY:=", <value>,
"EoZ:=", <value&gt;,</td></tr>
</table>
</td></tr>
</table>

			"kX:=", <value>, "kY:=", <value>, "kZ:=", <value> "PhiStart:=",<value>, "PhiStop:=", <value>, "PhiPoints:=", <int>, "ThetaStart:=", <value>, "ThetaStop:=", <value>, "ThetaPoints:=", <int>, "EoPhi:=", <value>, "EoTheta:=", <value>, "IsPropagating:=", <boolean>, "IsEvanescent:=", <boolean>, "IsEllipticallyPolarized:=", <boolean>) IsCartesian If true, provide the EoX, EoY, EoZ, kX, kY, kZ parameters. If false, provide the PhiStart, PhiStop, PhiPoints, ThetaStart, ThetStop, ThetaPoints, EoPhi, EoTheta parameters.
Return Value	None.		

Python Syntax	AssignIncidentWave (<*IncidentWaveArray*>)

Python Example	`oModule.AssignIncidentWave(["NAME:IncWave1",`
	`"Faces:=", [9],`
	`"IsCartesian:=", True,`
	`"EoX:=", "1", "EoY:=", "0", "EoZ:=", "0",`
	`"kX:=", "0", "kY:=", "0", "kZ:=", "1",`
	`"IsPropagating:=", True,`
	`"IsEvanescent:=", False,`
	`"IsEllipticallyPolarized:=", False]`
	`)`
	`oModule.AssignIncidentWave(["NAME:IncWave2",`
	`"Faces:=", [9],`
	`"IsCartesian:=", False,`
	`"PhiStart:=","0deg",`
	`"PhiStop:=", "90deg",`
	`"PhiPoints:=", 2,`
	`"ThetaStart:=", "0deg",`
	`"ThetaStop:=", "180deg",`
	`"ThetaPoints:=", 3,`
	`"EoPhi:=", "1", "EoTheta:=", "0",`

	"IsPropagating:=", True, "IsEvanescent:=", False, "IsEllipticallyPolarized:=", False])

AssignLatticePair

Assigns coupled Lattice Pair boundaries.

<table>
<tr><td><b>UI Access</b></td><td><b>HFSS</b> > <b>Boundaries</b> > <b>Assign</b> > <b>Coupled...</b> > <b>Lattice Pair...</b></td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;BoundaryArray&gt;</i></td><td>Array</td><td>Structured array.
Array("NAME:<BoundName>",
"Faces:=", <Array of face IDs>,
"ReverseV:=", <boolean>,
"PhaseDelay:=", <UseScanAngle | InputPhaseDelay>,
"Phi:=", <numdeg>,
"Theta:=", <numdeg>,
"Phase:=", <numdeg&gt;)</td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	AssignLatticePair(<*BoundaryArray*>)

Python Example

```
oModule.AssignLatticePair(
["NAME:LatticePair1",
"Faces:="                , [9,8],
"ReverseV:="             , False,
"PhaseDelay:="           , "UseScanAngle",
"Phi:="                  , "0deg",
"Theta:="                , "0deg"]
)
oModule.AssignLatticePair(
["NAME:LatticePair2",
"Faces:="                , [8,9],
"ReverseV:="             , False,
"PhaseDelay:="           , "InputPhaseDelay",
"Phase:="                , "10deg"]
)
```

AssignLayeredImp

Creates a layered impedance boundary.

UI Access	**HFSS** > **Boundaries** > **Assign** > **Layered Impedance...**

	Name	Type	Description
Parameters	*<LayeredImpArray>*	Array	Structured array. Array("NAME:<BoundName>", "Frequency:=", <value>, "Roughness:=", <value>, "IsInternal:=", <bool>, "IsTwoSided:=", <bool>, "IsShellElement:=", <bool>, <LayersArray>, "Objects:=", <AssignmentObjects>, "Faces:=", <AssignmentFaces>, "InfGroundPlane:=", <boolean>)
	<LayersArray>	Array	Structured array. Array("NAME:Layers", <OneLayerArray>, <OneLayerArray>, ...)
	<OneLayerArray>	Array	Structured array. Array("NAME:<LayerName>", "LayerType:=", <LayerType>, "Thickness:=",<value>, "Material:=", <string>) Thickness Thickness of the layer. Should be specified for all layers except the last layer.

		`Material` Material assigned on the layer. For the last layer, do not specify a material if the LayerType is "PerfectE" or "PerfectH".
<*LayerName*>	String	Specifies the layer number, such as "Layer1" or "Layer2"
<*LayerType*>	String	Should be specified for the last layer only. Possible values: "Infinite", "PerfectE", or "PerfectH"

Return Value	None.

Python Syntax	AssignLayeredImp(<*LayeredImpArray*>)
Python Example	see below

```
oModule.AssignLayeredImp(

["NAME:Layered1",

"Objects:=", ["Rectangle1"],

"Frequency:=", "0GHz",

"Roughness:=", "0um",

"IsTwoSided:=", True,

"IsShellElement:=", True,

["NAME:Layers", ["NAME:Layer1",

"Thickness:=", "1um", "Material:=", "vacuum"]],

"InfGroundPlane:=", False])
```

AssignLinearAntennaWave

Creates an incident linear antenna wave excitation.

UI Access	**HFSS** > **Excitations** > **Assign** > **Incident Wave** > **Linear Antenna Wave...**
Parameters	

Name	Type	Description

<*IncidentWaveArray*>	Array	Structured array.
		`Array("NAME:<Name of incident wave>",`
		`"Faces:=", <array of face IDs>,`
		`"IsCartesian:=", <boolean>,`
		`"EoX:=" , "<numeric value>",`
		`"EoY:=" , "<numeric value>",`
		`"EoZ:=" , "<numeric value>",`
		`"kX:=" , "<numeric value>",`
		`"kY:=" , "<numeric value>",`
		`"kZ:=" , "<numeric value>",`
		`"PhiStart:=",<value>,`
		`"PhiStop:=", <value>,`
		`"PhiPoints:=", <int>,`
		`"ThetaStart:=", <value>,`
		`"ThetaStop:=", <value>,`
		`"ThetaPoints:=", <int>,`
		`"EoPhi:=", <value>,`

	"EoTheta:=", <value>, "OriginX:=" , "<numUnit>", "OriginY:=" , "<numUnit>", "OriginZ:=" , "<numUnit>", "AntennaRadius:=" , "<numUnit>", "AntennaLength:=" , "<numUnit>") IsCartesian If true, provide the EoX, EoY, EoZ, kX, kY, kZ parameters. If false, provide the PhiStart, PhiStop, PhiPoints, ThetaStart, ThetStop, ThetaPoints, EoPhi, EoTheta parameters.
Return Value	None.

Python Syntax	AssignLinearAntennaWave(<*IncidentWaveArray*>)
Python Example	oModule.AssignLinearAntennaWave(["NAME:IncLWave1", "Faces:=" , [9], "IsCartesian:=" , True, "EoX:=" , "1", "EoY:=" , "0", "EoZ:=" , "0",

```
"kX:="                  , "0",
"kY:="                  , "0",
"kZ:="                  , "1",
"OriginX:="             , "0mm",
"OriginY:="             , "0mm",
"OriginZ:="             , "0mm",
"AntennaRadius:="       , "10mm",
"AntennaLength:="       , "10mm"
])
oModule.AssignLinearAntennaWave(
["NAME:IncLWave2",
"Faces:="               , [9],
"IsCartesian:="         , False,
"PhiStart:="            , "0deg",
"PhiStop:="             , "0deg",
"PhiPoints:="           , 1,
"ThetaStart:="          , "0deg",
"ThetaStop:="           , "0deg",
"ThetaPoints:="         , 1,
"EoPhi:="               , "1",
"EoTheta:="             , "0",
```

	"OriginX:=" , "0mm", "OriginY:=" , "0mm", "OriginZ:=" , "0mm", "AntennaRadius:=" , "10mm", "AntennaLength:=" , "10mm"])

AssignLinkedImpedance

Assigns a linked Impedance boundary.

<table>
<tr><th>UI Access</th><th>HFSS &gt; Boundaries > Assign > Linked Impedance...</th></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;LinkedImpArray&gt;</i></td><td>Array</td><td>Structured array.
Array("NAME:<BoundName>",
"Faces:=", <array of face IDs>,
"UseInfiniteGroundPlane:=", <boolean>,
"UseShellElement:=" , <boolean>,
<LinkDataArray&gt;)</td></tr>
<tr><td><i>&lt;LinkDataArray&gt;</i></td><td>Array</td><td>Structured array.
Array("NAME:<LinkName>",
"Project:=", <string file path&gt;,</td></tr>
</table>
</td></tr>
</table>

			"Product:=", <string product type>, "Design:=", <string linked source design name>, "Soln:=", <string linked source solution name>, <array solution parameters>, "ForceSourceToSolve:=", <boolean>, "PreservePartnerSoln:=" , <boolean>, "PathRelativeTo:=" , <string target project name>)
Return Value	None.		

Python Syntax	AssignLinkedImpedance(<*LinkedImpArray*>)
Python Example	oModule.AssignLinkedImpedance(["NAME:Linked1", "Faces:=", [9], "UseInfiniteGroundPlane:=", True, "UseShellElement:=", True, ["NAME:XLink", "Project:=", "C://temp/linkedproject.aedt", "Product:=", "HFSS", "Design:=", "Source_Project_Solver", "Soln:=", "1000MHz : LastAdaptive", ["NAME:Params", "xfactor:=", "1.2", "yfactor:=", "1.6"],

	`"ForceSourceToSolve:=", True,`
	`"PreservePartnerSoln:=", False,`
	`"PathRelativeTo:=", "TargetProject"]`
	])

AssignLinkedRegion

Assigns a Hybrid Region as a Linked Region.

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;LinkedRegionArray&gt;</i></td><td>Array</td><td>Structured array.
<code>Array("NAME:<name of linked region&gt;",</code>
<code>"Objects:=", <array, names for the objects assigned as Hybrid Region&gt;,</code>
<code>"Type:=" , <string, one of "IE", "PO" or "SBR"&gt;,</code>
<code>"IsLinkedRegion:=" , <boolean, true for linked region&gt;)</code></td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	AssignLinkedRegion (<*LinkedRegionArray*>)
Python Example	`oModule.AssignLinkedRegion (`

	["NAME:Linked1", "Objects:=", ["RegularPolyhedron1"], "Type:=", "PO", "IsLinkedRegion:=", True])

AssignLumpedPort

Creates a lumped port excitation.

<table>
<tr><th>UI Access</th><th><b>HFSS</b> &gt; <b>Excitations</b> &gt; <b>Assign</b> &gt; <b>Port</b> &gt; <b>Lumped Port...</b></th></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;LumpedPortArray&gt;</i></td><td>Array</td><td>Structured array.
Array("NAME:<BoundName>",
"Faces:=", <FaceIDArray>,
"RenormalizeAllTerminals:=", <boolean>,
"DoDeembed:=", <boolean>,
<ModesArray>,
"ShowReporterFilter:=", <boolean>,
"ReporterFilter:=", <array of boolean>,
"Impedance:=" , <value&gt;)</td></tr>
<tr><td><i>&lt;ModesArray&gt;</i></td><td>Array</td><td>Structured array.
Array("NAME:<ModesArrayName&gt;",</td></tr>
</table>
</td></tr>
</table>

			<OneModeArray>, <OneModeArray>,...)
	<OneModeArray>	Array	Structured array. Array("NAME:<ModeName>", "ModeNum:=", <integer>, "UseIntLine:=", <boolean>, <IntegerationLineArray>, "AlignmentGroup:=", <integer, group id>, "CharImp:=", <string, characteristic impedance>, "RenormImp:=" , <value, renormalize impedance to>)
	<IntegerationLineArray>	Array	Structured array. Array("NAME:<LineName>", "Coordinate System:=", <string, relative coordinate system>, "Start:=" , <array, start location coordinates>, "End:=" , <array, end location coordinates>)
Return Value	None.		

Python Syntax	AssignLumpedPort (<*LumpedPortArray*>)
Python Example	oModule.AssignLumpedPort(["NAME:LumpledPort1",

```
"Faces:=", [9],
"DoDeembed:=", False,
"RenormalizeAllTerminals:=", True,
["NAME:Modes",
        ["NAME:Mode1",
        "ModeNum:=", 1,
        "UseIntLine:=", True,
        ["NAME:IntLine",
                "Coordinate System:=", "Global",
                "Start:=", ["-0.4mm","-1mm","0.8mm"],
                "End:=" , ["-0.3mm","-1.2mm","0.8mm"]
        ],
        "AlignmentGroup:=", 0,
        "CharImp:=", "Zpi",
        "RenormImp:=", "50ohm"]],
"ShowReporterFilter:=", False,
"ReporterFilter:=", [True],
"Impedance:=", "50ohm"
])
```

AssignLumpedRLC

Creates a lumped RLC boundary.

<table>
<tr><th>UI Access</th><th>HFSS > Boundaries > Assign > Lumped RLC...</th></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td><i>&lt;LumpedRLCArray&gt;</i></td><td>Array</td><td>Structured array.
<pre>
Array("NAME:<BoundName>",
  "RLC Type:=" , <"Parallel" | "Serial" >,
  "UseResist:=",<boolean>,
  "Resistance:=", <value>,
  "UseInduct:=", <boolean>,
  "Inductance:=", <value>,
  "UseCap:=", <boolean>,
  "Capacitance:=", <value>,
  <CurrentLineArray>,
  "Objects:=", <AssignmentObjects>,
  "Faces:=", <AssignmentFaces>)
</pre></td></tr>
<tr><td><i>&lt;CurrentLineArray&gt;</i></td><td>Array</td><td>Structured array.
<pre>
Array("NAME:CurrentLine", _
  "Start:=", <LineEndPoint>,
  "End:=", <LineEndPoint>)
</pre></td></tr>
</table>
</td></tr>
</table>

Return Value	None.

Python Syntax	AssignLumpedRLC (<*LumpedRLCArray*>)

Python Example

```
oModule.AssignLumpedRLC(
  [
    "NAME:LumpRLC1",
    "Objects:=", ["Box1"],
    [
      "NAME:CurrentLine",
      "Start:=", ["0.15mm","-0.2mm","0mm"],
      "End:=" , ["0.15mm","0.6mm","0mm"]
    ],
    "RLC Type:=", "Parallel",
    "UseResist:=", True,
    "Resistance:=", "100ohm",
    "UseInduct:=", True,
    "Inductance:=", "10nH",
    "UseCap:=", True,
    "Capacitance:=", "10pF"
  ])
```

AssignMagneticBias

Creates a magnetic bias source.

<table>
<tr><td><b>UI Access</b></td><td><b>HFSS</b> &gt; <b>Excitations</b> &gt; <b>Assign</b> &gt; <b>Magnetic Bias...</b></td></tr>
<tr><td><b>Parameters</b></td><td><table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td>&lt;<i>MagneticBiasArray</i>&gt;</td><td>Array</td><td>Structured array.
<pre> Array("NAME:<BoundName>",
    "IsUniformBias:=", <boolean>,
    "Bias:=", <value>,
    "XAngle:=", <value>,
    "YAngle:=", <value>,
    "ZAngle:=", <value>,
    "Project:=",<string>,
    "Objects:=", <AssignmentObjects&gt;)</pre><code>IsUniformBias</code>
If true, supply the Bias, XAngle, YAngle, and ZAngle parameters.
If false, supply the Project parameter.</td></tr>
</table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	AssignMagneticBias (<*MagneticBiasArray*>)

Python Example	`oModule.AssignMagneticBias(`
	`["NAME:MagBias1",`
	`  "IsUniformBias:=", True,`
	`  "Bias:=", "1",`
	`  "XAngle:=", "10deg",`
	`  "YAngle:=", "10deg",`
	`  "ZAngle:=", "10deg",`
	`  "Objects:=", ["Box2"]])`
	`oModule.AssignMagneticBias(`
	`["NAME:MagBias2",`
	`  "IsUniformBias:=", False,`
	`  "Project:=","D:/Maxwell/testing/m3dfs.pjt",`
	`  "Objects:=", ["Box2"]])`

AssignMultipactionChargeRegion

Creates a Multipaction Charge Region for HFSS multipaction analysis.

<table>
<tr><td><b>UI Access</b></td><td><b>HFSS</b> &gt; <b>Excitations</b> &gt; <b>Assign</b> &gt; <b>Multipaction Charge Region...</b></td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;ChargeRegionArray&gt;</i></td><td>Array</td><td>Structured array.
<code>Array("NAME:MultipactionChargeRegion1",</code>
<code>"Objects:=", <array of objects&gt;,</code></td></tr>
</table>
</td></tr>
</table>

<table>
<tr><td></td><td><table><tr><td></td><td></td><td>"Faces:=", <array of face IDs>,
"NumParticles:=", "<integer, number of particles>",
"ParticleCharge:=", "<num><unit>",
"ParticleMass:=", "<num><unit>",
"Vx:=", "<num><unit>",
"Vy:=", "<num><unit>",
"Vz:=", "<num><unit&gt;")</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	AssignMultipactionChargeRegion(<*ChargeRegionArray*>)

Python Example	`oModule.AssignMultipactionChargeRegion(`
	`["NAME:MultipactionChargeRegion1",`
	`"Objects:="            , ["Cylinder1"],`
	`"NumParticles:="       , "100",`
	`"ParticleCharge:="     , "-1.60217662e-19Coulomb",`
	`"ParticleMass:="       , "9.10938356e-31kg",`
	`"Vx:="                 , "0.1m_per_sec",`
	`"Vy:="                 , "0.3m_per_sec",`
	`"Vz:="                 , "0.15m_per_sec"])`

AssignMultipactionDCBias

Set up DC electric or magnetic bias fields for a multipaction analysis

<table>
<tr><td><b>UI Access</b></td><td><b>HFSS</b> &gt; <b>Excitations</b> &gt; <b>Assign</b> &gt; <b>Multipaction DC Bias...</b></td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td><i>&lt;DCBiasArray&gt;</i></td><td>Array</td><td>Structured array.
<pre>
Array(NAME:MultipactionDCBias1",
"Objects:=", <array of objects>,
"UniformE:=", <boolean>,
"Ex:=", "<value><unit>",
"Ey:=", "<value><unit>",
"Ez:=", "<value><unit>",
<LinkedField>,
"UniformH:=" , <boolean>,
"Hx:=", "<value><unit>",
"Hy:=", "<value><unit>",
"Hz:=", "<value><unit>",
<LinkedField>)
</pre>
UniformE or Uniform H:
<ul>
<li><b>True</b> - provide filed values.</li>
<li><b>False</b> - provide field through linked Maxwell analysis.</li>
</ul>
</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><td></td><td><i>&lt;LinkedField&gt;</i></td><td>Array</td><td>Structured array.
Array("NAME:<EField | HField>",
"Project:=", <string, path to linked project file>,
"Product:=", "Maxwell",
"Design:=", <string, name of source design>,
"Soln:=", <string, name of linked solution>,
<Parameters for linked solution>,
"ForceSourceToSolve:=", <boolean>,
"PreservePartnerSoln:=", <boolean>,
"PathRelativeTo:=" , "TargetProject")</td></tr>
<tr><td><b>Return Value</b></td><td colspan="3">None.</td></tr>
</table>

Python Syntax	AssignMultipactionDCBias (<*DCBiasArray*>)
Python Example	oModule.AssignMultipactionDCBias(["NAME:MultipactionDCBias1", "Objects:=", ["Cylinder1"], "UniformE:=", False, ["NAME:EField", "Project:=", "C://projects/Maxwell/HallSensor.aedt",

```
        "Product:=", "Maxwell",
        "Design:=", "3_Maxwell3D",
        "Soln:=", "Setup1 : Transient",
        ["NAME:Params",
                "angle:=", "0deg",
                "cell_spacing:=", "2.5mm",
                "die_thickness:=", "2mm",
                "gap:=", "1.5mm",
                "magnet_center_y:=", "0mm",
                "magnet_center_z:=", "0mm",
                "magnet_x:=", "3mm",
                "magnet_y:=", "6mm",
                "magnet_z:=", "5mm",
                "move_x:=", "-1.5mm",
                "sensor_offset_x:=", "0mm",
                "sensor_offset_y:=", "0mm",
                "sensor_offset_z:=", "0mm",
                "sensor_pitch:=", "0deg",
                "sensor_roll:=", "0deg",
                "sensor_yaw:=", "0deg",
                "target_dia:=", "80mm"
        ],
        "ForceSourceToSolve:=", False,
        "PreservePartnerSoln:=", False,
        "PathRelativeTo:=", "TargetProject"
],
"UniformH:=", True,
"Hx:=", "0A_per_m",
"Hy:=", "0A_per_m",
"Hz:=", "0A_per_m"
])
```

AssignMultipactionSEE

Creates Secondary Electron Emission (SEE) boundaries for a Multipaction analysis.

<table>
<tr><th>UI Access</th><th>HFSS &gt; Boundaries > Assign > Multipaction SEE...</th></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;SEEArray&gt;</i></td><td>Array</td><td>Structured array.
Array("NAME:<SEEName>",
"Faces:=", <array of face IDs>,
"AlphaMax:=", <value>,
"Alpha0:=", <value&gt;,</td></tr>
</table>
</td></tr>
</table>

			"E0:=", <value>, "E1:=", <value>, "E2:=", <value>, "Em:=", <value>, "DielectricSurface:=", <boolean>)
Return Value	None.		

Python Syntax	AssignMultipactionSEE (<*SEEArray*>)
Python Example	oModule.AssignMultipactionSEE(["NAME:SEE1", "Faces:=", [342, 7, 341, 11, 8, 175, 35], "AlphaMax:=", "2.25", "Alpha0:=", "0", "E0:=", "12.5", "E1:=", "25", "E2:=", "5000", "Em:=", "175", "DielectricSurface:=", false])

AssignPerfectE

Creates a perfect E boundary.

<table>
<tr><td><b>UI Access</b></td><td><b>HFSS</b> &gt; <b>Boundaries</b> &gt; <b>Assign</b> &gt; <b>Perfect E...</b></td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;PerfectEArray&gt;</i></td><td>Array</td><td>Structured array.
Array("NAME:<BoundName>",
"InfGroundPlane:=", <boolean>,
"Objects:=", <AssignmentObjects>,
"Faces:=", <AssignmentFaces&gt;)</td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	AssignPerfectE (<*PerfectEArray*>)
Python Example	oModule.AssignPerfectE(["NAME:PerfE1", "InfGroundPlane:=", False, "Faces:=", [12]])

AssignPerfectH

Creates a perfect H boundary.

UI Access	**HFSS** > **Boundaries** > **Assign** > **Perfect H...**

<table>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><i><PerfectHArray></i></td><td>Array</td><td>Structured array.
Array("NAME:<BoundName>",
"Objects:=", <AssignmentObjects>,
"Faces:=", <AssignmentFaces>)</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	AssignPerfectH (<*PerfectHArray*>)

Python Example	`oModule.AssignPerfectH([`
	`"NAME:PerfH1",`
	`"Faces:=", [12]])`

AssignPlaneWave

Creates an incident plane wave excitation.

<table>
<tr><td><b>UI Access</b></td><td><b>HFSS</b> > <b>Excitations</b> > <b>Assign</b> > <b>Incident Wave</b> > <b>Plane Wave...</b></td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><i><PlaneWaveArray</i>
></td><td>Array</td><td>Structured array.
Array("NAME:<Name of incident wave>",
"Faces:=", <array of face IDs>,
"IsCartesian:=", <boolean>,</td></tr></table></td></tr>
</table>

			"EoX:=", "<numeric value>", "EoY:=", "<numeric value>", "EoZ:=", "<numeric value>", "kX:=", "<numeric value>", "kY:=", "<numeric value>", "kZ:=", "<numeric value>", "PhiStart:=",<value>, "PhiStop:=", <value>, "PhiPoints:=", <int>, "ThetaStart:=", <value>, "ThetaStop:=", <value>, "ThetaPoints:=", <integer>, "EoPhi:=", <value>, "EoTheta:=", <value>, "OriginX:=", "<numUnit>", "OriginY:=", "<numUnit>", "OriginZ:=", "<numUnit>", "IsPropagating:=", <boolean>, "IsEvanescent:=", <boolean>, "IsEllipticallyPolarized:=", <boolean>,

<table><tr><td></td><td><table><tr><td></td><td></td><td>"RealPropConst:=", "<value>",
"ImagPropConst:=", "<value>",
"PolarizationAngle:=", "<value>deg",
"PolarizationRatio:=" , "<value>")
IsCartesian:
• **True -** provide the EoX, EoY, EoZ, kX, kY, kZ parameters.
• **False -** provide the PhiStart, PhiStop, PhiPoints, ThetaStart, ThetStop, ThetaPoints, EoPhi, EoTheta parameters.
IsEvanescent:
• **True** - provide the RealPropConst, ImagPropConst.
IsEllipticallyPolarized:
• **True -** provide the PolarizationAngle, PolarizationRatio.</td></tr></table></td></tr><tr><td>**Return Value**</td><td>None.</td></tr></table>

Python Syntax	AssignPlaneWave(<*PlaneWaveArray*>)
Python Example	oModule.AssignPlaneWave(["NAME:IncPWave1", "Faces:=" , [9], "IsCartesian:=" , True, "EoX:=" , "1", "EoY:=" , "0",

```
"EoZ:="                 , "0",
"kX:="                  , "0",
"kY:="                  , "0",
"kZ:="                  , "1",
"OriginX:="             , "0mm",
"OriginY:="             , "0mm",
"OriginZ:="             , "0mm",
"IsPropagating:="       , True,
"IsEvanescent:="        , False,
"IsEllipticallyPolarized:=", False
])
oModule.AssignPlaneWave(
["NAME:IncPWave2",
"Faces:="               , [8],
"IsCartesian:="         , False,
"PhiStart:="            , "0deg",
"PhiStop:="             , "0deg",
"PhiPoints:="           , 1,
"ThetaStart:="          , "0deg",
"ThetaStop:="           , "0deg",
```

	`"ThetaPoints:="         , 1,`
	`"EoPhi:="               , "1",`
	`"EoTheta:="             , "0",`
	`"OriginX:="             , "0mm",`
	`"OriginY:="             , "0mm",`
	`"OriginZ:="             , "0mm",`
	`"IsPropagating:="       , False,`
	`"IsEvanescent:="        , True,`
	`"IsEllipticallyPolarized:=", False,`
	`"RealPropConst:="       , "0",`
	`"ImagPropConst:="       , "1"`
	`])`

AssignRadiation

Creates a radiation boundary.

UI Access	**HFSS** > **Boundaries** > **Assign** > **Radiation...**
Parameters	<table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><i>&lt;RadiationArray&gt;</i></td><td>Array</td><td>Structured array. <code>Array("NAME:<BoundName&gt;",</code> <code> "Objects:=", <AssignmentObjects&gt;,</code> <code> "Faces:=", <AssignmentFaces&gt;,</code></td></tr></table>

	"IsFssReference:=", <boolean>, "IsForPML:=", <boolean>) "IsFssReference" and "IsForPML" are only used to support legacy designs. In the current version, these should always be set to "False".
Return Value	None.

Python Syntax	AssignRadiation(<*RadiationArray*>)
Python Example	oModule.AssignRadiation(["NAME:Rad1", "Faces:=",[6, 7], "IsFssReference:=", False, "IsForPML:=", False])

AssignRFDischargeDCBias

Creates the DC bias for an RF Discharge simulation.

UI Access	**HFSS** > **Excitations** > **Assign** > **RF Discharge DC Bias...**

<table>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;DCBiasParameters&gt;</i></td><td>Array</td><td>Structured array.
<code>Array("NAME:<DCBiasName&gt;",</code>
<code>"UniformH:=", <boolean&gt;,</code>
<code>"Hx:=", "<real>A_per_m",</code>
<code>"Hy:=", "<real>A_per_m",</code>
<code>"Hz:=", "<real>A_per_m")</code></td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	AssignRFDischargeDCBias (<*DCBiasParameters*>)

Python Example	`oModule.AssignRFDischargeDCBias(`
	`[`
	`"NAME:RFDischargeDCBias1",`
	`"UniformH:=", True,`
	`"Hx:=", "795774.71546A_per_m",`
	`"Hy:=", "0A_per_m",`
	`"Hz:=", "0A_per_m"`
	`])`

AssignScreeningImpedance

Creates a screening impedance boundary.

Command: **HFSS>Boundaries>Assign>Screening Impedance**

Syntax: AssignScreeningImpedance <ScreeningArray>

Return Value: None.

Parameters: <ScreeningArray>

Array("NAME:<name>",

"Objects:=", Array("<name>"),

"IsAnisotropic:=", <Boolean>,

If true, you need to specify the coordinate system

"CoordSystem:=", <integer or name>,

"HasExternalLink:=", <Boolean>,

true or false. If False, specify XResistence and XReactance values. Also see the first example.

"XResistance:=", "<value>",

"XReactance:=", "<value>"

If true, then specify the external link array with the project and solution to use. Also see the second example.

Array("NAME:XLink",

"Project:=", "<projectName>.aedt",

"Design:=", "<DesignName>",

"Soln:=", "Setup1 : LastAdaptive",

Array("NAME:Params", "<variable>:=", "<value>"),

"ForceSourceToSolve:=", <Boolean>,

"PreservePartnerSoln:=", <Boolean>,

"PathRelativeTo:=", "TargetProject"),

Array("NAME:YLink",

"Project:=", "<projectName>.aedt",

"Design:=", "HFSSDesign1",

"Soln:=", "Setup1 : LastAdaptive",

Array("NAME:Params", "<variable>:=", "<value>"),

"ForceSourceToSolve:=", <Boolean>

"PreservePartnerSoln:=", <Boolean>,

"PathRelativeTo:=", "TargetProject"))

AssignSymmetry

Creates a symmetry boundary.

UI Access	**HFSS** > **Boundaries** > **Assign** > **Symmetry**.
Parameters	<table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><i>&lt;SymmetryArray&gt;</i></td><td>Array</td><td>Structured array. <code>Array("NAME:<BoundName&gt;",</code> <code>"IsPerfectE:=", <boolean&gt;,</code> <code>"Objects:=", <AssignmentObjects&gt;,</code> <code>"Faces:=", <AssignmentFaces&gt;)</code></td></tr></table>
Return Value	None.

Python Syntax	AssignSymmetry (<*SymmetryArray*>)

Python Example	`oModule.AssignSymmetry(`
	`["NAME:Sym1",`
	`  "IsPerfectE:=", True,`
	`  "Faces:=", [12]])`

AssignTerminal

UI Access	
Parameters	Name | Type | Description *<TerminalArray>* | Array | Structured array.
Return Value	None.

Python Syntax	AssignTerminal (<*TerminalArray*>)
Python Example	

AssignVoltage

Creates a voltage source.

UI Access	**HFSS** > **Excitations** > **Assign** > **Voltage...**

<table>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;VoltageArray&gt;</i></td><td>Array</td><td>Structured array.
<code>Array("NAME:<BoundName&gt;",</code>
<code>"Voltage:=", <value&gt;,</code>
<code>&lt;DirectionArray&gt;,</code>
<code>"Objects:=", <AssignmentObjects&gt;,</code>
<code>"Faces:=", <AssignmentFaces&gt;)</code></td></tr>
<tr><td><i>&lt;DirectionArray&gt;</i></td><td>Array</td><td>Structured array.
<code>Array("NAME:Direction",_</code>
<code>"Start:=",<LineEndPoint&gt;,</code>
<code>"End:=", <LineEndPoint&gt;)</code></td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	AssignVoltage (<*VoltageArray*>)

Python Example	`oModule.AssignVoltage(["NAME:Voltage1",`
	`"Voltage:=", "1000mV",`
	`["NAME:Direction",`
	`"Start:=", [-0.4, -1.2, 0],`
	`"End:=", [-1.4, -1.2, 0]],`
	`"Faces:=", [7])`

AutoCreatePECCapforWavePort

Use: If the HFSS Option for HFSS>Boundary Assignment>Automatic PEC cap creation for wave port is enabled, HFSS automatically creates a PEC cap under the following conditions: The general rule is that the dialog for creating wave port PEC backing appears if the wave port is internal to the design and does not touch any non-solve-inside component. If a project has open region defined, all geometries are considered internal.

Command: None.

Syntax: AutoCreatePECCapForWavePort ([<PECCapParameters>])

Return Value: None

```
Parameters: <PECCapParameters>

  "NAME:AutoCreatePECCapForWavePort",

  "Wave Port Name:=" , "<string>",

  "Face ID:=" , <intID>,

  "Flip Side:=" , Boolean,

  "Thickness:=" , "<real><units>"
```

Python Syntax	AutoCreatePECCapforWavePort(<*PECCapParameters*>)

Python Example	`AutoCreatePECCapForWavePort(`
	`[`
	`"NAME:AutoCreatePECCapForWavePort",`
	`"Wave Port Name:="      , "2",`
	`"Face ID:="             , 280,`
	`"Flip Side:="           , False,`
	`"Thickness:="           , "0.18mm"`
	`])`

CircuitPortToLumpedPort

Converts a circuit port to a lumped port for a driven terminal or driven modal design in HFSS.

<table>
<tr><td><b>UI Access</b></td><td>Right-click on a circuit port, then select <b>Convert to Lumped Port</b>.</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><i>&lt;PortName&gt;</i></td><td>String</td><td>Name of specified circuit port.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	CircuitPortToLumpedPort (<*PortName*>)
Python Example	`oModule.CircuitPortToLumpedPort("1")`

EditAnisotropicImpedance

Modifies an Anisotropic Impedance boundary condition.

<table>
<tr><td><b>UI Access</b></td><td>Double-click the boundary condition in the project tree to modify its settings.</td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i><BondName></i></td><td>String</td><td>Name of boundary condition to be edited.</td></tr>
<tr><td><i><AnisotropicImpArray></i></td><td>Array</td><td>Structured array.
Array("NAME:<string name of anisotropic impedance>",
"Faces:=" , <array of face IDs>,
"UseInfiniteGroundPlane:=", <boolean>,
"CoordSystem:=" , <string coordinate system name>,
"HasExternalLink:=" , <boolean>,
"ZxxResistance:=" , <string of an integer value>,
"ZxxReactance:=" , <string of an integer value>,
"ZxyResistance:=" , <string of an integer value>,
"ZxyReactance:=" , <string of an integer value>,
"ZyxResistance:=" , <string of an integer value>,
"ZyxReactance:=" , <string of an integer value>,
"ZyyResistance:=" , <string of an integer value>,
"ZyyReactance:=" , <string of an integer value>)</td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	EditAnisotropicImpedance (<*BondName*>, <*AnisotropicImpArray*>)
Python Example	oModule.EditAnisotropicImpedance("Anistropic10", ["NAME:Anisotropic1", "Faces:=" , [17215], "UseInfiniteGroundPlane:=", False, "CoordSystem:=" , "Global", "HasExternalLink:=" , False, "ZxxResistance:=" , "377", "ZxxReactance:=" , "0", "ZxyResistance:=" , "0", "ZxyReactance:=" , "0", "ZyxResistance:=" , "0", "ZyxReactance:=" , "0", "ZyyResistance:=" , "377", "ZyyReactance:=" , "0"])

EditAperture

Modifies an aperture boundary.

UI Access	Double-click the boundary condition in the project tree to modify its settings.

<table>
<tr><td><b>Parameters</b></td><td><table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;BondName&gt;</i></td><td>String</td><td>Name of boundary condition to be edited.</td></tr>
<tr><td><i>&lt;ApertureArray&gt;</i></td><td>Array</td><td>Array defines selected objects.</td></tr>
</table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	EditAperture(<*BondName*>, <*ApertureArray*>)

Python Example	oModule.EditAperture("Aperture1"
	    ["NAME:Aperture2",
	    "Objects:=", ["Rectangle1"]
	])

EditCircuitPort[HFSS]

Edits a circuit port for a driven terminal or driven modal design in HFSS.

<table>
<tr><td><b>UI Access</b></td><td>Double-click the excitation in the project tree to modify its settings.</td></tr>
<tr><td><b>Parameters</b></td><td><table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;ExcitationName&gt;</i></td><td>String</td><td>Name of the excitation to be edited.</td></tr>
<tr><td><i>&lt;CircuitPortArray&gt;</i></td><td>Array</td><td>Structured array.
<code>Array("NAME:<PortName&gt;",</code>
<code> "Impedance:=", "valueohm",</code>
<code> "DoDeembed:=", <boolean&gt;</code></td></tr>
</table></td></tr>
</table>

			"RenormalizeAllTerminals:=", <boolean> "TerminalIDLit:=", Array())
Return Value	None.		

Python Syntax	EditCircuitPort (<*ExcitationName*>, <*CircuitPortArray*>)

Python Example	oModule.EditCircuitPort ("5"
	["NAME:1",
	  "Impedance:=" , "50ohm",
	  "DoDeembed:=" , False,
	  "RenormalizeAllTerminals:=", True,
	  "TerminalIDList:=" , []
	])

EditDielectricCavity

Modifies definitions of a Dielectric Cavity.

UI Access	Double-click the cavity under **Hybrid Regions** in the project tree to modify its settings.		
Parameters	Name	Type	Description
	<*CavityName*>	String	Name of the cavity to be edited.
	<*CavityArray*>	Array	Structured array. Array("NAME:<new name>")

Return Value	None.

Python Syntax	EditDielectricCavity(<*CavityName*>, <*CavityArray*>)

Python Example	`oModule.EditDielectricCavity("Cavity1"`
	`[`
	`  "NAME:Cavity2"`
	`])`

EditDiffPairs

Edits the properties of differential pairs defined from terminal excitations on wave ports.

<table>
<tr><th>UI Access</th><th><b>HFSS</b> &gt; <b>Excitations</b> &gt; <b>Differential Pairs</b>.</th></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td><i>&lt;DifferentialPairsArray&gt;</i></td><td>Array</td><td>Structured array.<pre> Array("NAME:EditDiffPairs",
    <OneDiffPairArray>, <OneDiffPairArray>,...)</pre></td></tr>
<tr><td><i>&lt;OneDiffPairArray&gt;</i></td><td>Array</td><td>Structured array.<pre> Array("NAME:Pair1",_
    "PosBoundary:=", <string, name of the terminal to
    use as the positive terminal.>,
    "NegBoundary:=", <string, name of the terminal to</pre></td></tr>
</table>
</td></tr>
</table>

			use as the negative terminal.>, "CommonName:=", <string, name for the common mode.>, "CommonRefZ:=", <value, reference impedance for the common mode.>, "DiffName:=", <string, name for the differential mode.>, "DiffRefZ:=", <value, reference impedance for the differential mode.>, "IsActive:=", <boolean>)
Return Value	None.		

Python Syntax	EditDiffPairs(<*DifferentialPairsArray*>)

Python Example	oModule.EditDiffPairs(["NAME:EditDiffPairs",
	["NAME:Pair1",
	  "PosBoundary:=", "Rectangle1_T1",
	  "NegBoundary:=", "Rectangle2_T1",
	  "CommonName:=", "Comm1",
	  "CommonRefZ:=", "25ohm",
	  "DiffName:=", "Diff1",
	  "DiffRefZ:=", "100ohm",
	  "IsActive:=", True]]

	)

EditFEBI

Modifies a FE-BI hybrid region.

UI Access	Double-click the FE-BI under **Hybrid Regions** in the project tree to modify its settings.
Parameters	<table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>*&lt;FEBIName&gt;*</td><td>String</td><td>Name of the FE-BI region to be edited.</td></tr><tr><td>*&lt;FEBIArray&gt;*</td><td>Array</td><td>Structured array. `Array("NAME:<new name>")`</td></tr></table>
Return Value	None.

Python Syntax	EditFEBI (<*FEBIName*>, <*FEBIArray*>)

Python Example	`oModule.EditFEBI("FE-BI1"`
	`[`
	`"NAME:FE-BI2"`
	`])`

EditFiniteCond

Modifies parameters of single finite conductivity boundary.

<table>
<tr><td><b>UI Access</b></td><td>Double-click finite conductivity boundary in project tree.</td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;BoundaryName&gt;</i></td><td>String</td><td>Name of the finite conductivity boundary to be edited.</td></tr>
<tr><td><i>&lt;FiniteCondArray&gt;</i></td><td>Array</td><td>Structured array.
Array("NAME:<Name of the boundary>",
"Roughness:=", "<string, double value with units of length>",
"UseCoating:=", <boolean>,
"LayerThickness:=", "<string, double value with units of length>",
"UseMaterial:=", <boolean>,
"Material:=", "<string, material name for coating.&gt;")</td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

<table>
<tr><td><b>Python Syntax</b></td><td>EditFiniteCond (&lt;<i>BoundaryName</i>&gt;, &lt;<i>FiniteCondArray</i>&gt;)</td></tr>
<tr><td><b>Python Example</b></td><td>oModule.EditFiniteCond("FiniteCond1",
["NAME:FiniteCond1",
"Roughness:=", "2um",
"UseCoating:=", false]
)</td></tr>
</table>

EditFloquetPort

Modifies a Floquet port excitation.

<table>
<tr><td><b>UI Access</b></td><td>Double-click on a floquet port under <b>Excitations</b> in history tree.</td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;FloquetPortName&gt;</i></td><td>String</td><td>Name of the floquet port excitation to be edited.</td></tr>
<tr><td><i>&lt;FloquetPortArray&gt;</i></td><td>Array</td><td>Structured array.
Array("NAME:<BoundName>",
"NumModes:=", <integer>,
"RenormalizeAllTerminals:=", <boolean>,
"DoDeembed:=", <boolean>,
<ModesArray>,
"ShowReporterFilter:=", <boolean>,
"UseScanAngles:=", <boolean>,
"Phi:=", "<numdeg>",
"Theta:=", "<numdeg>",
<LatticeAVector>,
<LatticeBVector>,
<ModesCalculator>,
<ModesList&gt;)</td></tr>
<tr><td><i>&lt;ModesArray&gt;</i></td><td>Array</td><td>Structured array.</td></tr>
</table>
</td></tr>
</table>

		Array("NAME:Modes", Array("NAME:<ModeName>", "ModeNum:=", <integer>, "UseIntLine:=", <boolean>), ...)
<LatticeAVector>	Array	Structured array. Array("NAME:LatticeAVector", "Start:=", Array("<num><units>", "<num><units>", "<num><units>"), "End:=", Array("<num><units>", "<num><units>", "<num><units>"))
<LatticeBVector>	Array	Structured array. Array("NAME:LatticeBVector", "Start:=", Array("<num><units>", "<num><units>", "<num><units>"), "End:=", Array("<num><units>", "<num><units>", "<num><units>"))
<ModesCalculator>	Array	Structured array. Array("NAME:ModesCalculator", "Frequency:=", "<Value>GHz", "FrequencyChanged:=", <Boolean>, "PhiStart:=", "<num>deg", "PhiStop:=", "<num>deg", "PhiStep:=", "<num>deg", "ThetaStart:=", "<num>deg",

			"ThetaStop:=", "<num>deg", "ThetaStep:=", "<num>deg")
	<ModesList>	Array	Structured array. Array("NAME:ModesList", Array("NAME:Mode", "ModeNumber:=", <ModeID>, "IndexM:=", <integer index>, "IndexN:=", <integer index>, "KC2:=", <integer value>, "PropagationState:=", "Propagating", "Attenuation:=", <integer value>, "PolarizationState:=", <TE or TM>, "AffectsRefinement:=", <boolean>), ...)
Return Value	None.		

Python Syntax	EditFloquetPort(<*FloquetPortName*>, <*FloquetPortArray*>)
Python Example	oModule.EditFloquetPort("FloquetPort1", ["NAME:FloquetPort1After",

```
"NumModes:=", 2,
"RenormalizeAllTerminals:=", True,
"DoDeembed:=", False,
["NAME:Modes",
        ["NAME:Mode1",
        "ModeNum:=",1,
        "UseIntLine:=", False],
        ["NAME:Mode2",
        "ModeNum:=", 2,
        "UseIntLine:=", False]],
"ShowReporterFilter:=", False,
"UseScanAngles:=", True, "Phi:=", "0deg", "Theta:=", "0deg",
["NAME:LatticeAVector",
        "Start:=", ["0mm", "0mm", "0.8mm"],
        "End:=", ["0mm", "0.5mm", "0.8mm"]],
["NAME:LatticeBVector",
        "Start:=", ["0mm", "0mm", "0.8mm"],
        "End:=", ["0.8mm", "0mm", "0.5mm"]],
["NAME:ModesCalculator",
        "Frequency:=", "1GHz",
        "FrequencyChanged:=", False,
        "PhiStart:=", "0deg",
        "PhiStop:=", "0deg",
        "PhiStep:=","0deg",
        "ThetaStart:=", "0deg",
        "ThetaStop:=", "0deg",
        "ThetaStep:=", "0deg"],
["NAME:ModesList",
        ["NAME:Mode",
        "ModeNumber:=", 1,
        "IndexM:=", 0,
        "IndexN:=", 0,
        "KC2:=", 0,
        "PropagationState:=", "Propagating",
        "Attenuation:=", 0,
        "PolarizationState:=", "TE",
        "AffectsRefinement:=", False],
["NAME:Mode",
        "ModeNumber:=", 2,
        "IndexM:=", 0,
        "IndexN:=", 0,
```

	`"KC2:=", 0,`
	`"PropagationState:=", "Propagating",`
	`"Attenuation:=", 0,`
	`"PolarizationState:=", "TM",`
	`"AffectsRefinement:=", False]`
	`]])`

EditFresnel

Modifies a Fresnel boundary condition.

<table>
<tr><td><b>UI Access</b></td><td>Double-click on a fresnel under <b>Boundaries</b> in project history tree.</td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;FresnelName&gt;</i></td><td>String</td><td>Name of the fresnel boundary to be edited.</td></tr>
<tr><td><i>&lt;FresnelArray&gt;</i></td><td>Array</td><td>Structured array.
<code>Array("NAME:<FresnelName&gt;",</code>
<code>"Fresnel Boundary Type:=", <PerfectAbsorber or ImportFromTableFile&gt;,</code>
<code>"RTTable Path:=", <string path to table file&gt;)</code></td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	EditFresnel(<*FresnelName*>, <*FresnelArray*>)
Python Example	`oModule.AssignFresnel("Fresnel1"`

	`["NAME:Fresnel2",`
	`  "Fresnel Boundary Type:=", "PerfectAbsorber"`
	`])`

EditGlobalMatEnv

Modifies global material environment setting to tell HFSS what material properties to use when calculating far fields.

UI Access	**HFSS** > **Boundaries** > **Edit Global Material Environment...**
Parameters	Name | Type | Description *<MatEnvName>* | String | Name of the global material environment, default is "vacuum".
Return Value	None.

Python Syntax	EditGlobalMatEnv(<*MatEnvName*>)
Python Example	`oModule.EditGlobalMatEnv("water_distilled")`

EditGradientSurfaceRoughness

Edit an existing Gradient Surface Roughness boundary.

UI Access	**Double-click** on a half space under **Boundaries** in **history tree**
Parameters	Name | Type | Description

<table>
<tr><td></td><td><table><tr><td><i>&lt;ArgArray&gt;</i></td><td>Array</td><td>Structured array.
[
"NAME:<Name of GradientSurfaceBoundary>",
"Objects:=" , ["<geometryID>"],
"RMS Rougheness:=" , "<value><units>",
"Bulk Conductivity:=" , "<value>",
"Thickness:=" , ""<value><units>",
"Max Frequency:=" , ""<value><units>",
"Surface Type:=" , "[Low Profile | High Profile]"</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	AssignGradientSurfaceRoughness(<*ArgArray*>)
Python Example	oModule = oDesign.GetModule("BoundarySetup") oModule.AssignGradientSurfaceRoughness(["NAME:GradientImped1", "Objects:=" , ["Box1"], "RMS Rougheness:=" , "1um", "Bulk Conductivity:=" , "58000000", "Thickness:=" , "15um",

	"Max Frequency:=" , "100GHz", "Surface Type:=" , "Low Profile"])

EditHalfSpace

Modifies a Half Space boundary name, Z location, and or materials.

<table>
<tr><td><b>UI Access</b></td><td>Double-click on a half space under <b>Boundaries</b> in history tree.</td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;HalfSpaceName&gt;</i></td><td>String</td><td>Name of the half space boundary to be edited.</td></tr>
<tr><td><i>&lt;HalfSpaceArray&gt;</i></td><td>Array</td><td>Structured array.
Array("NAME:<Name of Half Space>",
"ZLocation:=", "<intUnits>",
"Material:=", "<string material name&gt;")</td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	EditHalfSpace(<*HalfSpaceName*>, <*HalfSpaceArray*>)
Python Example	oModule.EditHalfSpace("HalfSpace1" ["NAME:HalfSpace1After", "ZLocation:=", "2mm",

	"Material:=", "tungsten"])

EditHybridRegion

Modifies settings of an assigned hybrid region.

UI Access	Double-click on an assignment under **Hybrid Regions** in project history tree.

Parameters	<table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>*<HybridRegionName>*</td><td>String</td><td>Name of the hybrid region to be edited.</td></tr><tr><td>*<HybridRegionArray>*</td><td>Array</td><td>Structured array.
	`Array("NAME:<new name of hybrid region>",`
	`"Type:=", <string one of "IE", "PO" or "SBR">,`
	`"IsLinkedRegion:=", <boolean>)`</td></tr></table>

Return Value	None.

Python Syntax	EditHybridRegion(<*HybridRegionName*>, <*HybridRegionArray*>)

Python Example	`oModule.EditHybridRegion("Hybrid1"`
	`["NAME:Hybrid1After",`
	`  "Type:=", "IE",`
	`  "IsLinkedRegion:=", False`
	`])`

EditImpedance

Modifies an impedance boundary definitions.

<table>
<tr><th>UI Access</th><th>Double-click on an impedance under **Boundaries** in the project history tree.</th></tr>
<tr><td>Parameters</td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td>*<BoundaryName>*</td><td>String</td><td>Name of the impedance boundary to be edited.</td></tr>
<tr><td>*<ImpedanceArray>*</td><td>Array</td><td>Structured array.
Array("NAME:<New name>",
"Resistance:=", <value>,
"Reactance:=", <value>,
"InfGroundPlane:=", <boolean>)</td></tr>
</table>
</td></tr>
<tr><td>Return Value</td><td>None.</td></tr>
</table>

Python Syntax	EditImpedance(*<BoundaryName>*, *<ImpedanceArray>*)
Python Example	oModule.AssignImpedance("Imped1" ["NAME:Imped1After", "Resistance:=", "50", "Reactance:=", "50", "InfGroundPlane:=", False])

EditIncidentWave

Modifies an incident wave excitation.

<table>
<tr><td><b>UI Access</b></td><td>Double-click the excitation in the project tree to modify its settings.</td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;BoundaryName&gt;</i></td><td>String</td><td>Name of the incident wave excitation to be edited.</td></tr>
<tr><td><i>&lt;IncidentWaveArray&gt;</i></td><td>Array</td><td>Structured array.
Array("NAME:<BoundName>",
"IsCartesian:=",<boolean>,
"EoX:=", <value>,
"EoY:=", <value>,
"EoZ:=", <value>,
"kX:=", <value>,
"kY:=", <value>,
"kZ:=", <value>
"PhiStart:=",<value>,
"PhiStop:=", <value>,
"PhiPoints:=", <int>,
"ThetaStart:=", <value>,
"ThetaStop:=", <value>,
"ThetaPoints:=", <int&gt;,</td></tr>
</table>
</td></tr>
</table>

	`"EoPhi:=", <value>,`
	`"EoTheta:=", <value>,`
	`"IsPropagating:=", <boolean>,`
	`"IsEvanescent:=", <boolean>,`
	`"IsEllipticallyPolarized:=", <boolean>)`
	`IsCartesian`
	If true, provide the EoX, EoY, EoZ, kX, kY, kZ parameters.
	If false, provide the PhiStart, PhiStop, PhiPoints, ThetaStart, ThetStop, ThetaPoints, EoPhi, EoTheta parameters.

Return Value	None.

Python Syntax	EditIncidentWave(<*BoundaryName*>, <*IncidentWaveArray*>)

Python Example	`oModule.EditIncidentWave("IncWave1",`
	`["NAME:IncWave1After",`
	`"IsCartesian:=", True,`
	`"EoX:=", "1", "EoY:=", "0", "EoZ:=", "0",`
	`"kX:=", "0", "kY:=", "0", "kZ:=", "1",`
	`"IsPropagating:=", True,`
	`"IsEvanescent:=", False,`
	`"IsEllipticallyPolarized:=", False])`

EditInteriorNearFieldSource

Edits an interior near field source as an external data file and associated coordinate system.

<table>
<tr><th>UI Access</th><th><b>HFSS</b> &gt; <b>Boundaries</b> &gt; <b>Assign</b> &gt; <b>Linked Field</b>&gt;<b>Near Field...</b></th></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td><i>&lt;ArgArray&gt;</i></td><td>Array</td><td>Structured array.
<pre>
("NAME:<Name of Near Field Source data>",
"Type:=" , "Incident",
"ExternalDataFile:=" , "<filePath>.and",
"SourceCoordSystem:=" , "<CS_name>"
</pre></td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

<table>
<tr><th>Python Syntax</th><th>AssignInteriorNearFieldSource(&lt;<i>ArgArray</i>&gt;)</th></tr>
<tr><td><b>Python Example</b></td><td>
<pre>
oDesign = oProject.SetActiveDesign("internal")
oModule = oDesign.GetModule("BoundarySetup")
oModule.AssignInteriorNearFieldSource(
  [
    "NAME:ExtNFData2",
    "Type:=" , "Incident",
    "ExternalDataFile:=" , "D:\\Ansoft\\Elliptical_EH.and",
</pre></td></tr>
</table>

	`"SourceCoordSystem:=" , "Global"`
	`])`

EditLatticePair

Modifies coupled Lattice Pair boundaries.

UI Access	Double-click on a lattice pair under **Boundaries** in the project history tree.

Parameters	Name / Type / Description
	<BoundaryName> / String / Name of the lattice pair boundary to be edited.
	<BoundaryArray> / Array / Structured array.
	`Array("NAME:<BoundName>",`
	`"ReverseV:=", <boolean>,`
	`"PhaseDelay:=", <UseScanAngle | InputPhaseDelay>,`
	`"Phi:=", <numdeg>,`
	`"Theta:=", <numdeg>,`
	`"Phase:=", <numdeg>)`

Return Value	None.

Python Syntax	EditLatticePair(*<BoundaryName>*, *<BoundaryArray>*)
Python Example	`oModule.EditLatticePair("LatticePair1",`

	["NAME:LatticePair1After",
	  "ReverseV:=", False,
	  "PhaseDelay:=", "UseScanAngle",
	  "Phi:=", "10deg",
	  "Theta:=", "0deg"]
	)

EditLayeredImp

Modifies a layered impedance boundary.

<table>
<tr><td><b>UI Access</b></td><td>Double-click on a layered impedance under <b>Boundaries</b> in the project history tree.</td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;BoundaryName&gt;</i></td><td>String</td><td>Name of the layered impedance boundary.</td></tr>
<tr><td><i>&lt;LayeredImpArray&gt;</i></td><td>Array</td><td>Structured array.
Array("NAME:<BoundName>",
  "Frequency:=", <value>,
  "Roughness:=", <value>,
  "IsInternal:=", <bool>,
  "IsTwoSided:=", <bool>,
  "IsShellElement:=", <bool>,
  <LayersArray>,
  "InfGroundPlane:=", <boolean&gt;)</td></tr>
</table>
</td></tr>
</table>

	<LayersArray>	Array	Structured array.
			`Array("NAME:Layers",`
			`<OneLayerArray>, <OneLayerArray>, ...)`

	<OneLayerArray>	Array	Structured array.
			`Array("NAME:<LayerName>",`
			`"LayerType:=", <LayerType>,`
			`"Thickness:=",<value>,`
			`"Material:=", <string>)`
			`Thickness`
			Thickness of the layer. Should be specified for all layers except the last layer.
			`Material`
			Material assigned on the layer. For the last layer, do not specify a material if the LayerType is "PerfectE" or "PerfectH".

	<LayerName>	String	Specifies the layer number, such as "Layer1" or "Layer2"
	<LayerType>	String	Should be specified for the last layer only. Possible values: "Infinite", "PerfectE", or "PerfectH"
Return Value	None.		

Python Syntax	EditLayeredImp(<*BoundaryName*>, <*LayeredImpArray*>)
Python Example	`oModule.EditLayeredImp("Layered1",`

	["NAME:Layered1After", "Frequency:=", "10GHz", "Roughness:=", "0um", "IsTwoSided:=", True, "IsShellElement:=", True, ["NAME:Layers", ["NAME:Layer1", "Thickness:=", "1um", "Material:=", "vacuum"]], "InfGroundPlane:=", False])

EditLinkedImpedance

Edits a linked Impedance boundary.

<table>
<tr><td><b>UI Access</b></td><td>Double-click on a linked impedance under <b>Boundaries</b> in project history tree.</td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;BoundaryName&gt;</i></td><td>String</td><td>Name of the linked impedance boundary to be edited.</td></tr>
<tr><td><i>&lt;LinkedImpArray&gt;</i></td><td>Array</td><td>Structured array.
  Array("NAME:<BoundName>",
    "UseInfiniteGroundPlane:=", <boolean>,
    "UseShellElement:=" , <boolean>,
    <LinkDataArray&gt;)</td></tr>
<tr><td><i>&lt;LinkDataArray&gt;</i></td><td>Array</td><td>Structured array.
  Array("NAME:<LinkName&gt;",</td></tr>
</table>
</td></tr>
</table>

			"Project:=", <string file path>, "Product:=", <string product type>, "Design:=", <string linked source design name>, "Soln:=", <string linked source solution name>, <array solution parameters>, "ForceSourceToSolve:=", <boolean>, "PreservePartnerSoln:=" , <boolean>, "PathRelativeTo:=" , <string target project name>)
Return Value	None.		

Python Syntax	EditLinkedImpedance(<*BoundaryName*>, <*LinkedImpArray*>)
Python Example	oModule.EditLinkedImpedance("Linked1", ["NAME:Linked1After", "UseInfiniteGroundPlane:=", True, "UseShellElement:=", True, ["NAME:XLink", "Project:=", "C://temp/linkedproject.aedt", "Product:=", "HFSS", "Design:=", "Source_Project_Solver", "Soln:=", "1000MHz : LastAdaptive",

	`["NAME:Params", "xfactor:=", "1.2", "yfactor:=", "1.6"],`
	`"ForceSourceToSolve:=", True,`
	`"PreservePartnerSoln:=", False,`
	`"PathRelativeTo:=", "TargetProject"]`
	])

EditLinkedRegion

Edits a Linked Region.

UI Access	N/A

Parameters	<table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;*BoundaryName*&gt;</td><td>String</td><td>Name of linked region.</td></tr><tr><td>&lt;*LinkedRegionArray*&gt;</td><td>Array</td><td>Structured array.
	`Array("NAME:<New name>",`
	`"Type:=" , <string, one of "IE", "PO" or "SBR">,`
	`"IsLinkedRegion:=" , <boolean, true for linked region>)`</td></tr></table>

Return Value	None.

Python Syntax	EditLinkedRegion(<*BoundaryName*>, <*LinkedRegionArray*>)

Python Example	`oModule.EditLinkedRegion("Linked1",`
	`["NAME:Linked1After",`

	"Type:=", "PO", "IsLinkedRegion:=", True])

EditLumpedPort

Modifies a lumped port.

<table>
<tr><td><b>UI Access</b></td><td>Double-click the excitation in the project tree to modify its settings.</td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;BoundaryName&gt;</i></td><td>String</td><td>Name of lumped port excitation to be edited.</td></tr>
<tr><td><i>&lt;LumpedPortArray&gt;</i></td><td>Array</td><td>Structured array.
Array("NAME:<BoundName>",
"RenormalizeAllTerminals:=", <boolean>,
"DoDeembed:=", <boolean>,
<ModesArray>,
"ShowReporterFilter:=", <boolean>,
"ReporterFilter:=", <array of boolean>,
"Impedance:=" , <value&gt;)</td></tr>
<tr><td><i>&lt;ModesArray&gt;</i></td><td>Array</td><td>Structured array.
Array("NAME:<ModesArrayName>",
<OneModeArray>, <OneModeArray&gt;,...)</td></tr>
</table>
</td></tr>
</table>

	<OneModeArray>	Array	Structured array.
			`Array("NAME:<ModeName>",`
			`"ModeNum:=", <integer>,`
			`"UseIntLine:=", <boolean>,`
			`<IntegerationLineArray>,`
			`"AlignmentGroup:=", <integer, group id>,`
			`"CharImp:=", <string, characteristic impedance>,`
			`"RenormImp:=" , <value, renormalize impedance to>)`

	<IntegerationLineArray>	Array	Structured array.
			`Array("NAME:<LineName>",`
			`"Coordinate System:=", <string, relative coordinate system>,`
			`"Start:=" , <array, start location coordinates>,`
			`"End:=" , <array, end location coordinates>)`

Return Value	None.		

Python Syntax	EditLumpedPort(*<BoundaryName>*, *<LumpedPortArray>*)

Python Example	`oModule.EditLumpedPort("LumpledPort1",`
	`["NAME:LumpledPort1After",`
	`"DoDeembed:=", False,`
	`"RenormalizeAllTerminals:=", True,`

```
["NAME:Modes",
        ["NAME:Mode1",
        "ModeNum:=", 1,
        "UseIntLine:=", True,
        ["NAME:IntLine",
                "Coordinate System:=", "Global",
                "Start:=", ["-0.4mm","-1mm","0.8mm"],
                "End:=" , ["-0.3mm","-1.2mm","0.8mm"]
        ],
        "AlignmentGroup:=", 0,
        "CharImp:=", "Zpi",
        "RenormImp:=", "50ohm"]],
"ShowReporterFilter:=", False,
"ReporterFilter:=", [True],
"Impedance:=", "50ohm"
])
```

EditLumpedRLC

Modifies a lumped RLC boundary.

<table>
<tr><td><b>UI Access</b></td><td>Double-click the boundary in the project tree to modify its settings.</td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;BoundaryName&gt;</i></td><td>String</td><td>Name of the boundary to be edited.</td></tr>
<tr><td><i>&lt;LumpedRLCArray&gt;</i></td><td>Array</td><td>Structured array.
Array("NAME:<NewBoundName>",
"RLC Type:=", <"Parallel" | "Serial" >,
"UseResist:=", <boolean>,
"Resistance:=", <value>,
"UseInduct:=", <boolean>,
"Inductance:=", <value>,
"UseCap:=", <boolean>,
"Capacitance:=", <value>,
<CurrentLineArray&gt;)</td></tr>
<tr><td><i>&lt;CurrentLineArray&gt;</i></td><td>Array</td><td>Structured array.
Array("NAME:CurrentLine",
"Start:=", <LineEndPoint>,
"End:=", <LineEndPoint&gt;)</td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	EditLumpedRLC(<*BoundaryName*>, <*LumpedRLCArray*>)
Python Example	oModule.AssignLumpedRLC("LumpRLC1",

	["NAME:LumpRLC1After", ["NAME:CurrentLine", "Start:=", ["0.15mm","-0.2mm","0mm"], "End:=", ["0.15mm","0.6mm","0mm"]], "RLC Type:=", "Parallel", "UseResist:=", True, "Resistance:=", "100ohm", "UseInduct:=", True, "Inductance:=", "10nH", "UseCap:=", True, "Capacitance:=", "10pF"])

EditMagneticBias

Modifies a magnetic bias excitation.

<table>
<tr><th>UI Access</th><th>Double-click on the magnetic bias under <b>Excitations</b> in the project tree.</th></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;BoundaryName&gt;</i></td><td>String</td><td>Name of the magnetic bias excitation to be edited.</td></tr>
<tr><td><i>&lt;MagneticBiasArray&gt;</i></td><td>Array</td><td>Structured array.
Array("NAME:<NewBoundName>",
  "IsUniformBias:=", <boolean&gt;,</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><td></td><td><table><tr><td></td><td></td><td>"Bias:=", <value>,
"XAngle:=", <value>,
"YAngle:=", <value>,
"ZAngle:=", <value>,
"Project:=",<string>)
IsUniformBias
If true, supply the Bias, XAngle, YAngle, and ZAngle parameters.
If false, supply the Project parameter.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	EditMagneticBias(<*BoundaryName*>, <*MagneticBiasArray*>)

Python Example	`oModule.EditMagneticBias("MagBias1",`
	`["NAME:MagBias1After",`
	`"IsUniformBias:=", True,`
	`"Bias:=", "1",`
	`"XAngle:=", "10deg",`
	`"YAngle:=", "10deg",`
	`"ZAngle:=", "10deg"])`

EditMultipactionChargeRegion

Modifies settings for a Multipaction Charge Region.

<table>
<tr><td><b>UI Access</b></td><td>Double-click on the excitation in project tree to edit its settings.</td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;BoundaryName&gt;</i></td><td>String</td><td>Name of the multipaction charge region to be edited.</td></tr>
<tr><td><i>&lt;ChargeRegionArray&gt;</i></td><td>Array</td><td>Structured array.
Array("NAME:MultipactionChargeRegion1",
"NumParticles:=", "<integer, number of particles>",
"ParticleCharge:=", "<num><unit>",
"ParticleMass:=", "<num><unit>",
"Vx:=", "<num><unit>",
"Vy:=", "<num><unit>",
"Vz:=", "<num><unit&gt;")</td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	EditMultipactionChargeRegion (<*BoundaryName*>, <*ChargeRegionArray*>)

Python Example	`oModule.EditMultipactionChargeRegion("MultipactionChargeRegion1",`
	`["NAME:MultipactionChargeRegion1After",`
	`  "NumParticles:=", "100",`

	"ParticleCharge:=", "-1.60217662e-19Coulomb", "ParticleMass:=", "9.10938356e-31kg", "Vx:=", "0.1m_per_sec", "Vy:=", "0.3m_per_sec", "Vz:=", "0.15m_per_sec"])

EditMultipactionDCBias

Edits settings for a multipaction DC bias boundary.

<table>
<tr><td><b>UI Access</b></td><td>Double-click on the excitation in the project tree to edit its settings.</td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;BoundaryName&gt;</i></td><td>String</td><td>Name of the multipaction DC bias to be edited.</td></tr>
<tr><td><i>&lt;DCBiasArray&gt;</i></td><td>Array</td><td>Structured array.
Array(NAME:MultipactionDCBias1",
"UniformE:=", <boolean>,
"Ex:=", "<value><unit>",
"Ey:=", "<value><unit>",
"Ez:=", "<value><unit>",
<LinkedField>,
"UniformH:=" , <boolean>,
"Hx:=", "<value><unit>",
"Hy:=", "<value><unit&gt;",</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><td></td><td>
<table>
<tr><td></td><td></td><td>"Hz:=", "<value><unit>",
<LinkedField>)
UniformE or Uniform H:
• <b>True</b> - provide filed values.
• <b>False</b> - provide field through linked Maxwell analysis.</td></tr>
<tr><td><i>&lt;LinkedField&gt;</i></td><td>Array</td><td>Structured array.
Array("NAME:<EField | HField>",
"Project:=", <string, path to linked project file>,
"Product:=", "Maxwell",
"Design:=", <string, name of source design>,
"Soln:=", <string, name of linked solution>,
<Parameters for linked solution>,
"ForceSourceToSolve:=", <boolean>,
"PreservePartnerSoln:=", <boolean>,
"PathRelativeTo:=" , "TargetProject")</td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	EditMultipactionDCBias(<*BoundaryName*>, <*DCBiasArray*>)
Python Example	`oModule.EditMultipactionDCBias("MultipactionDCBias1",`

```
["NAME:MultipactionDCBias1After",
"UniformE:=", False,
["NAME:EField",
        "Project:=", "C://projects/Maxwell/HallSensor.aedt",
        "Product:=", "Maxwell",
        "Design:=", "3_Maxwell3D",
        "Soln:=", "Setup1 : Transient",
        ["NAME:Params",
                "angle:=", "0deg",
                "cell_spacing:=", "2.5mm",
                "die_thickness:=", "2mm",
                "gap:=", "1.5mm",
                "magnet_center_y:=", "0mm",
                "magnet_center_z:=", "0mm",
                "magnet_x:=", "3mm",
                "magnet_y:=", "6mm",
                "magnet_z:=", "5mm",
                "move_x:=", "-1.5mm",
                "sensor_offset_x:=", "0mm",
                "sensor_offset_y:=", "0mm",
                "sensor_offset_z:=", "0mm",
                "sensor_pitch:=", "0deg",
                "sensor_roll:=", "0deg",
                "sensor_yaw:=", "0deg",
                "target_dia:=", "80mm"
        ],
        "ForceSourceToSolve:=", False,
        "PreservePartnerSoln:=", False,
        "PathRelativeTo:=", "TargetProject"
],
"UniformH:=", True,
"Hx:=", "0A_per_m",
"Hy:=", "0A_per_m",
"Hz:=", "0A_per_m"
])
```

EditMultipactionSEE

Edits a Secondary Electron Emission (SEE) boundary settings.

<table>
<tr><td><b>UI Access</b></td><td>Double-click on the boundary in the project tree to edit its settings.</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><i><BoundaryName></i></td><td>String</td><td>Name of the boundary to be edited.</td></tr></table></td></tr>
</table>

	<*SEEArray*>	Array	Structured array.
			`Array("NAME:<NewSEEName>",`
			`"AlphaMax:=", <value>,`
			`"Alpha0:=", <value>,`
			`"E0:=", <value>,`
			`"E1:=", <value>,`
			`"E2:=", <value>,`
			`"Em:=", <value>,`
			`"DielectricSurface:=", <boolean>)`

Return Value	None.		

Python Syntax	EditMultipactionSEE(<*BoundaryName*>, <*SEEArray*>)

Python Example	`oModule.EditMultipactionSEE("SEE1",`
	`["NAME:SEE1After",`
	`"AlphaMax:=", "2.25", "Alpha0:=", "0",`
	`"E0:=", "12.5", "E1:=", "25", "E2:=",  "5000",`
	`"Em:=", "175",`
	`"DielectricSurface:=", false]`
	`)`

EditRFDischargeDCBias

Edits settings for an RF discharge DC bias excitation.

<table>
<tr><td><b>UI Access</b></td><td>Double-click on the excitation in the project tree to edit its settings.</td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;BoundaryName&gt;</i></td><td>String</td><td>Name of the excitation to be edited.</td></tr>
<tr><td><i>&lt;DCBiasParameters&gt;</i></td><td>Array</td><td>Structured array.
Array("NAME:<DCBiasName>",
"UniformH:=", <boolean>,
"Hx:=", "<real>A_per_m",
"Hy:=", "<real>A_per_m",
"Hz:=", "<real>A_per_m")</td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	EditRFDischargeDCBias(<BoundaryName>, <*DCBiasParameters*>)

Python Example	`oModule.EditRFDischargeDCBias("DCBias1",`
	`[`
	`"NAME:RFDischargeDCBias1",`
	`"UniformH:=", True,`
	`"Hx:=", "795774.71546A_per_m",`

	`"Hy:=", "0A_per_m",`
	`"Hz:=", "0A_per_m"`
	`])`

EditPerfectE

Modifies a perfect E boundary.

<table>
<tr><td><b>UI Access</b></td><td>Double-click on the boundary in project tree to edit its settings.</td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td>&lt;<i>BoundaryName</i>&gt;</td><td>String</td><td>Name of the boundary to be edited.</td></tr>
<tr><td>&lt;<i>PerfectEArray</i>&gt;</td><td>Array</td><td>Structured array.
<code>Array("NAME:<NewBoundName&gt;",</code>
<code>"InfGroundPlane:=", <boolean&gt;)</code></td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	EditPerfectE(<*BoundaryName*>, <*PerfectEArray*>)

Python Example	`oModule.EditPerfectE("PerfE1",`
	`["NAME:PerfE1After",`
	`"InfGroundPlane:=", False,`
	`])`

EditPerfectH

Modifies a perfect H boundary.

UI Access	Double-click on the boundary in the project tree to edit its settings.
Parameters	<table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>*<BoundaryName>*</td><td>String</td><td>Name of the boundary to be edited.</td></tr><tr><td>*<PerfectHArray>*</td><td>Array</td><td>Structured array. `Array("NAME:<NewBoundName>")`</td></tr></table>
Return Value	None.

Python Syntax	EditPerfectH(<*BoundaryName*>, <*PerfectHArray*>)

Python Example	`oModule.EditPerfectH("PerfH1",`
	`["NAME:PerfH1After"])`

EditRadiation

Modifies a radiation boundary.

UI Access	Double-click the boundary in the project tree to modify its settings.
Parameters	<table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>*<BoundaryName>*</td><td>String</td><td>Name of the boundary to be edited.</td></tr><tr><td>*<RadiationArray>*</td><td>Array</td><td>Structured array.</td></tr></table>

	`Array("NAME:<BoundName>",`
	`"IsIncidentField:=", <boolean>,`
	`"IsEnforcedHField:=", <boolean>,`
	`"IsEnforcedEField:=", <boolean>,`
	`"IsFssReference:=", <boolean>,`
	`"IsForPML:=", <boolean>,`
	`"UseAdaptiveIE:=", <boolean>,`
	`"IncludeInPostproc:=", <boolean>)`

Return Value	None.

Python Syntax	EditRadiation(<*BoundaryName*>, <*RadiationArray*>)

Python Example	`oModule.EditRadiation("Rad1",`
	`["NAME:Rad1After",`
	`"IsIncidentField:=", True,`
	`"IsEnforcedField:=", False,`
	`"IsFssReference:=", False,`
	`"IsForPML:=", False,`
	`"UseAdaptiveIE:=", False,`
	`"IncludeInPostproc:=", Frue]`
	`)`

EditSymmetry

Modifies a symmetry boundary.

UI Access	Double-click the symmetry boundary in the project tree to edit its settings.
Parameters	<table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>*<BoundaryName>*</td><td>String</td><td>Name of the symmetry boundary to be edited.</td></tr><tr><td>*<SymmetryArray>*</td><td>Array</td><td>Structured array. `Array("NAME:<BoundName>",` `"IsPerfectE:=", <boolean>)`</td></tr></table>
Return Value	None.

Python Syntax	EditSymmetry(*<BoundaryName>*, *<SymmetryArray>*)

Python Example	`oModule.EditSymmetry("Sym1",`
	`["NAME:Sym1After",`
	`"IsPerfectE:=", True`
	`])`

EditTerminal

Modifies properties of a terminal

UI Access	**Edit Properties** for a selected terminal.

Parameters	<table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;*TerminalName*&gt;</td><td>String</td><td>Name of the terminal to be edited.</td></tr><tr><td>&lt;*TerminalArray*&gt;</td><td>Array</td><td>Structured array.
	`Array("NAME: <TerminalName>",`
	`"ParentBndID:=", "<PortName>",`
	`"TerminalResistance:=," <Value and units of resistance>")`</td></tr></table>

Return Value	None.

Python Syntax	EditTerminal (<*TerminalName*>, <*TerminalArray*>)

Python Example	`oModule.EditTerminal("Rectangle2_T1",`
	`["NAME:Rectangle2_T1",`
	`"ParentBndID:=", "WavePort1",`
	`"TerminalResistance:=", "75ohm"]`
	`)`

EditVoltage

Modifies a voltage source.

UI Access	Double-click the excitation in the project tree to modify its settings.
Parameters	Name | Type | Description

<BoundaryName>	String	Name of the voltage source to be edited.

<VoltageArray>	Array	Structured array.
		`Array("NAME:<BoundName>",`
		`"Voltage:=", <value>,`
		`<DirectionArray>)`

<DirectionArray>	Array	Structured array.
		`Array("NAME:Direction",`
		`"Start:=",<LineEndPoint>,`
		`"End:=", <LineEndPoint>)`

Return Value	None.

Python Syntax	EditVoltage(<*BoundaryName*>, <*VoltageArray*>)

Python Example	`oModule.EditVoltage("Voltage1",`
	`["NAME:Voltage1After",`
	`"Voltage:=", "1000mV",`
	`["NAME:Direction",`
	`"Start:=", [-0.4, -1.2, 0],`
	`"End:=", [-1.4, -1.2, 0]`
	`]`
	`)`

EditVoltageDrop

Use: Edits a voltage drop excitation.

Command: Double-click the excitation in the project tree to edit it.

Syntax:EditVoltageDrop <BoundName> <VoltageDropArray>

Return Value: None

EditWavePort

Modifies a wave port.

UI Access	Double-click the excitation in the project tree to modify its settings.		
Parameters	Name	Type	Description
	<*BoundaryName*>	String	Name of the wave port to be edited.

	<*WavePortArray*>	Array	Structured array.
			`Array("NAME:<BoundName>",`
			`"NumModes:=", <integer>,`
			`"PolarizeEField:=",<boolean>,`
			`"DoDeembed:=", <boolean>,`
			`"DeembedDist:=", <value>,`
			`"DoRenorm:=", <boolean>,`
			`"RenormValue:=",<value>,`
			`<ModesArray>,`
			`"TerminalIDList:=", <TerminalsArray>)`

			`NumModes` Number of modes for modal problems. Number of terminals for terminal problems.

	<ModesArray>	Array	Structured array.
			`Array("NAME:Modes",`
			`<OneModeArray>, <OneModeArray>, ...)`

	<OneModeArray>	Array	Structured array.
			`Array("NAME:<ModeName>",`
			`"ModeNum:=", <integer>,`
			`"UseIntLine:=", <boolean>,`
			`<IntLineArray>)`

	<IntLineArray>	Array	Structured array.
			`Array("NAME:IntLine",`
			`"Start:=", <LineEndPoint>,`
			`"End:=", <LineEndPoint>,`
			`"CharImp:=", <string, Characteristic impedance of the mode. Possible values are "Zpi", "Zpv", or "Zvi">)`

Return Value	None.		

Python Syntax	EditWavePort(<*BoundaryName*>, <*WavePortArray*>)

Python Example

```
oModule.EditWavePort("WavePort1",
["NAME:WavePort1After",
  "NumModes:=", 2,
  "PolarizeEField:=",False,
  "DoDeembed:=", True,
  "DeembedDist:=", "10mil",
  "DoRenorm:=", True,
  "RenormValue:=","50Ohm",
  ["NAME:Modes",
    ["NAME:Mode1",
    "ModeNum:=", 1,
    "UseIntLine:=", True,
    ["NAME:IntLine",
    "Start:=", [-0.4, -1.2, 0),
    "End:=", [-1.4, 0.4, 0]],
    "CharImp:=", "Zpi"),
    ["NAME:Mode2",
    "ModeNum:=", 2,
    "UseIntLine:=", False]
    ]
])
```

LumpedPortToCircuitPort

Converts a lumped port to a circuit port for a driven terminal or driven modal design in HFSS.

<table>
<tr><td><b>UI Access</b></td><td>Right-click on a lumped port excitation in the project tree, then select <b>Convert to Circuit Port</b>.</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><i>&lt;PortName&gt;</i></td><td>String</td><td>Name of the lumped port to be converted.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	LumpedPortToCircuitPort(<PortName>)
Python Example	`oModule.LumpedPortToCircuitPort("Port1")`

SetAllHybridRegionsToOneWayCoupled

Specifies one-way coupling for all hybrid regions.

UI Access	Right-click **Hybrid Regions** > **Set Regions** > **All Regions Are One Way Coupled**.
Parameters	None.
Return Value	None.

Python Syntax	SetAllHybridRegionsToOneWayCoupled()
Python Example	`oModule.SetAllHybridRegionsToOneWayCoupled()`

SetAllHybridRegionsToTwoWayCoupled

Specifies two-way coupling for all hybrid regions.

UI Access	Right-click **Hybrid Regions** > **Set Regions** > **All Regions Are Two Way Coupled**.
Parameters	None.
Return Value	None.

Python Syntax	SetAllHybridRegionsToTwoWayCoupled()
Python Example	`oModule.SetAllHybridRegionsToTwoWayCoupled()`

SetHybridRegionCoupledGroup

Use: To set coupling for hybrid regions.

Command: **HFSS>HybridRegions>Set Coupling**

Syntax: SetHybridRegionCoupledGroup <value>

Return Value: None

`Parameters: <value>`

Type: <string>

"OneWayCoupled" or "TwoWayCoupled" for all regions or "Advanced", Array("One Way:= <hybridregionname>, Array(Two Way:=", Array("<hybridregionname>" ,)))

Example:

Setting All Hybrid Regions to One Way Coupled

```
oProject = oDesktop.SetActiveProject("Project35")

oDesign = oProject.SetActiveDesign("HFSSDesign1")

oModule = oDesign.GetModule("BoundarySetup")

oModule.SetHybridRegionCoupledGroup "OneWayCoupled"
```

Setting All Hybrid Regions to Two Way Coupled

```
oProject = oDesktop.SetActiveProject("Project35")

oDesign = oProject.SetActiveDesign("HFSSDesign1")

oModule = oDesign.GetModule("BoundarySetup")

oModule.SetHybridRegionCoupledGroup "TwoWayCoupled"
```

Setting Hybrid Region Groupings

```
oProject = oDesktop.SetActiveProject("Project35")

oDesign = oProject.SetActiveDesign("HFSSDesign1")

oModule = oDesign.GetModule("BoundarySetup")

oModule.SetHybridRegionCoupledGroup "Advanced", ["One Way:=", ["Two Way:=",

["Hybrid1", "Hybrid2"]]]
```

SetHybridRegionsCoupling

Sets coupling for hybrid regions.

<table>
<tr><td><b>UI Access</b></td><td>Right-click <b>Hybrid Regions</b> &gt; <b>Set Couling</b> &gt; <b>Advanced</b>.</td></tr>
<tr><td><b>Parameters</b></td><td><table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;CouplingArray&gt;</i></td><td>Array</td><td>Structured array.
<code>Array(</code>
<code>"One Way:=", <array of region names&gt;,</code>
<code>"Two Way:=", <array of region names&gt;)</code>
Note: At least two regions must be specified in a group.</td></tr>
</table></td></tr>
<tr><td><b>Return Value</b></td><td>None</td></tr>
</table>

Python Syntax	SetHybridRegionsCoupling(<*CouplingArray*>)

Python Example	`oModule.SetHybridRegionsCoupling(`
	`[`
	`"One Way:=", ["FE-BI1","FE-BI2"],`
	`"Two Way:=", ["FE-BI3","FE-BI4","FE-BI5"]`
	`])`

SetScatteredFieldFormulation

Sets incident wave to scattered field formulation. **Note**: Incident field formulation is only applicable in a driven terminal/modal design with incident wave.

UI Access	Right-click on **Excitations**, then select **Set Incident Field Formulation** > **Scattered**.
Parameters	None.
Return Value	None.

Python Syntax	SetScatteredFieldFormulation()
Python Example	`oModule.SetScatteredFieldFormulation()`

SetSBRCreepingWaveSettings

Sets creeping wave settings for SBR+ solutions.

<table>
<tr><td><b>UI Access</b></td><td>Right-click on <b>Hybrid Regions</b>, then select <b>Creeping Wave Settings...</b></td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;CWArray&gt;</i></td><td>Array</td><td>Structured array.
<code>Array("NAME:SBRCreepingWaveSettings",</code>
<code>"CWRaySampleDensity:=" , <value&gt;,</code>
<code>"CWRayCutoffDb:=" , <value&gt;,</code>
<code>"CWCurvatureSensitivity:=", <value&gt;,</code></td></tr>
</table>
</td></tr>
</table>

	"CWAngularRayInterval:=", <value>)
Return Value	None.

Python Syntax	SetSBRCreepingWaveSettings(<*CWArray*>)

Python Example	`oModule.SetSBRCreepingWaveSettings([`
	`  "NAME:SBRCreepingWaveSettings",`
	`  "CWRaySampleDensity:="  , 10,`
	`  "CWRayCutoffDb:="       , 40,`
	`  "CWCurvatureSensitivity:=", 50,`
	`  "CWAngularRayInterval:=", 2`
	`])`

SetSBRTxRxSettings

Assigns Transmit and Receive assignments for antennas in SBR+ solutions.

<table>
<tr><td><b>UI Access</b></td><td>Right-click on <b>Excitations</b> > <b>Select Tx/Rx...</b></td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i><TxRxArray></i></td><td>Array</td><td>Structured array.
<code>Array("NAME:SBRTxRxSettings",</code>
<code> <AssignmentList>, <AssignmentList>,...)</code></td></tr>
<tr><td><i><AssignmentList></i></td><td>Array</td><td>Structured array.</td></tr>
</table>
</td></tr>
</table>

			Array("NAME:<ListName>", "Tx Antenna:=", <string, antenna components>, "Rx Antennas:=", <string, antenna components>)
Return Value	None.		

Python Syntax	SetSBRTxRxSettings(<*TxRxArray*>)
Python Example	oModule.SetSBRTxRxSettings(["NAME:SBRTxRxSettings", ["NAME:Tx/Rx List 0", "Tx Antenna:=", "cHorn1_1_p1", "Rx Antennas:=", "Beam1_1_p1,pHorn1_1_p1,sDipole1_1_p1"], ["NAME:Tx/Rx List 1", "Tx Antenna:=", "Beam1_1_p1", "Rx Antennas:=", "Beam1_1_p1,pHorn1_1_p1,sDipole1_1_p1"], [

	`"NAME:Tx/Rx List 2",`
	`"Tx Antenna:=", "sDipole1_1_p1",`
	`"Rx Antennas:=", "Beam1_1_p1,pHorn1_1_p1,sDipole1_1_p1"`
	`]`
	`])`

SetTerminalReferenceImpedances

Sets the reference impedance for all terminals within a specified port.

<table>
<tr><th>UI Access</th><th>HFSS &gt; Excitations > Set Terminal Renormalizing Impedances...</th></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td>&lt;<i>RefImpValue</i>&gt;</td><td>String</td><td>Impedance value with unit.</td></tr>
<tr><td>&lt;<i>PortName</i>&gt;</td><td>String</td><td>Optional. Name of the port.</td></tr>
<tr><td>&lt;<i>RenormalizeTerminals</i>&gt;</td><td>Boolean</td><td>Optional.
• <b>True</b> - renormalize terminals.
• <b>False</b> - do not renormalize terminals.</td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	SetTerminalReferenceImpedances(<*RefImpValue*>, <*PortName*>, <*RenormalizeTerminals*>)
Python Example	`oModule.SetTerminalReferenceImpedances("50ohm", "", False)`

SetTotalFieldFormulation

Sets incident wave to total field formulation. **Note**: Incident field formulation is only applicable in a driven terminal/modal design with incident wave.

UI Access	Right-click on **Excitations**, then select **Set Incident Field Formulation** > **Total**.
Parameters	None.
Return Value	None.

Python Syntax	SetTotalFieldFormulation()
Python Example	`oModule.SetTotalFieldFormulation()`

SwapCircuitPortDirection

Swaps the direction of a circuit port.

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><i>&lt;PortName&gt;</i></td><td>String</td><td>Name of specified port.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	SwapCircuitPortDirection(<*PortName*>)

Python Example	`oModule.SwapCircuitPortDirection("1")`

SwapLatticePair

Swaps the faces defined in a lattice pair.

UI Access	Right-click on a lattice pair in the project tree, then select **Swap Faces**.
Parameters	Name Type Description *<LatticePairName>* String Name of specified lattice pair boundary to be edited.
Return Value	None.

Python Syntax	SwapLatticePair(<*LatticePairName*>)
Python Example	`oModule.SwapLatticePair("LatticePair1")`

Mechanical Boundary, Contact, & Excitation Commands

Boundary and excitation commands for Mechanical designs are dependent upon the solution type (Modal, Thermal, or Structural). Mechanical boundary and excitation commands should be executed by the oModule object. For example, use `oModule.AssignFrictionlessSupport` to assign a frictionless support boundary to a model for a Mechanical – Modal solution.

See the following sections to view the parameters used in the assignment and editing of each boundary and excitation:

Modal Boundary Commands:

- AssignFixedSupport and EditFixedSupport
- AssignFrictionlessSupporty and EditFrictionlessSupporty
- AssignCylindricalSupport and EditCylindricalSupport

Thermal Boundary Commands:

- AssignConvection and EditConvection
- AssignTemperature and EditTemperature
- AssignRotatingFluid and EditRotatingFluid

Thermal Contact Commands:

- AssignContact and EditContact

Thermal Excitation Commands:

- AssignEMLoss and EditEMLoss
- AssignHeatFlux and EditHeatFlux
- AssignHeatGeneration and EditHeatGeneration

Thermal Initial Condition Commands:

- AssignInitialTemperature and EditInitialTemperature (Transient Thermal solutions only)

Structural Boundary Commands:

- AssignFixedSupport and EditFixedSupport
- AssignFrictionlessSupporty and EditFrictionlessSupporty
- AssignCylindricalSupport and EditCylindricalSupport

Structural Excitation Commands:

- AssignDisplacement and EditDisplacement
- AssignForce and EditForce
- AssignPressure and EditPressure
- AssignThermalCondition and EditThermalCondition

AssignContact and *EditContact*

AssignContact

This command creates contact in a Mechanical–Thermal design for the purpose of defining thermal resistance between adjacent objects.

<table>
<tr><td><b>UI Access</b></td><td><b>Mechanical</b> &gt; <b>Contacts</b> &gt; <b>Assign</b> &gt; <b>Contact</b></td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td><i>&lt;NAME&gt;</i></td><td>string</td><td>Contact name</td></tr>
<tr><td><i>&lt;Faces&gt;</i></td><td>list</td><td>Faces included in the boundary condition assignment (integer list)</td></tr>
<tr><td><i>&lt;Objects&gt;</i></td><td>list</td><td>Shell objects included in the boundary condition assignment (string list)</td></tr>
<tr><td></td><td></td><td></td></tr>
<tr><td><i>&lt;Resistance Type&gt;</i></td><td>string</td><td>Specifies which one of the following five resistance parameters (*) to use:</td></tr>
<tr><td></td><td></td><td></td></tr>
<tr><td>* <i>&lt;Thermal Conductance&gt;</i></td><td>string</td><td>Total thermal contact conductance for selected assignment faces/objects (with units)</td></tr>
<tr><td>* <i>&lt;Thermal Conductance per Area&gt;</i></td><td>string</td><td>Thermal contact conductance per unit area (with units)</td></tr>
<tr><td>* <i>&lt;Thermal Resistance&gt;</i></td><td>string</td><td>Total thermal contact resistance for selected assignment faces/objects (with units)</td></tr>
<tr><td>* <i>&lt;Thermal Impedance&gt;</i></td><td>string</td><td>Total area-based thermal contact conductance for selected assignment faces/objects (with units)</td></tr>
<tr><td>* <i>&lt;Thickness&gt;</i></td><td>string</td><td>Thickness of the <i>&lt;Material&gt;</i> used to determine thermal contact conductivity (with unit)</td></tr>
<tr><td></td><td></td><td></td></tr>
<tr><td><i>&lt;Material&gt;</i></td><td>string</td><td>Name of material (additional parameter for <i>&lt;ResistanceType&gt;</i> = "Thickness" only)</td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>None</td></tr>
</table>

Python Syntax	AssignContact([<*NAME*>, <*Faces*>, <*Objects*>, <*Resistance Type*>, <*Thermal Conductance* | *Thermal Conductance per Area* | *Thermal Resistance* | *Thermal Impedance* | *Thickness*>, <*Material*>])

Python Example
Thermal Impedance
	`oModule.AssignConvection(`
	`[`
	`"NAME:Contact1"       ,`
	`"Objects:="           , ["Shell1"],`
	`"Faces:="             , [40,63],`
	`"Resistance Type:="   , "Thermal Impedance",`
	`"Thermal Impedance:=" , "4000cel_mm2_per_w"`
	`])`

Python Example
Thickness
	`oModule.AssignConvection(`
	`[`
	`"NAME:Contact1"     ,`
	`"Objects:="         , ["Shell1"],`
	`"Faces:="           , [40,63],`
	`"Resistance Type:=" , "Thickness",`
	`"Thickness:="       , "3mil",`
	`"Material:="        , "Mica-Typical"`
	`])`

EditContact

> **Note:**
>
> In the above script examples, you can substitute "*Edit*" for "*Assign*" to modify a pre-existing contact instead of creating a new one. The script must identify the **NAME** of the existing contact within the *EditContact* command line. Then, you can change any of the setup parameters with the exception of the assignment *Faces* or *Objects*.
>
> To change the assignment *Faces* or *Objects*, you must use the *ReassignBoundary* command, provide the excitation NAME, and specify the new Faces and/or Objects.
>
> Script examples for editing a contact are given below. In these examples, the name, resistance type, and associated parameter value are being modified.

- **UI Access:** Double-click the contact in the Project Manager or right-click it and choose **Properties** from the shortcut menu.
- **Python Example:**

```
oModule.EditContact("Contact1",
       [
               "NAME:Contact_Top",
               "Resistance Type:=" , "Thickness"
               "Thickness:="       , "2.5mil",
               "Material:="        , "polyehtylene"
       ])
```

AssignConvection and *EditConvection*

AssignConvection

This command creates a convection boundary in a Mechanical–Thermal design and optionally sets up a link to import heat transfer coefficients from an Icepak design.

<table>
<tr><td><b>UI Access</b></td><td><b>Mechanical</b> &gt; <b>Boundaries</b> &gt; <b>Assign</b> &gt; <b>Convection</b></td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td><i>&lt;NAME&gt;</i></td><td>string</td><td>Boundary condition name</td></tr>
<tr><td><i>&lt;Objects&gt;</i></td><td>list</td><td>Objects included in the boundary condition assignment</td></tr>
<tr><td><i>&lt;Faces&gt;</i></td><td>list</td><td>Faces included in the boundary condition assignment</td></tr>
<tr><td><i>&lt;Temperature&gt;</i></td><td>string</td><td>Temperature of the ambient environment</td></tr>
<tr><td><i>&lt;Uniform&gt;</i></td><td>bool</td><td>True or False: Type of film coefficient – Uniform when True, Non-Uniform (imported from Icepak) when False</td></tr>
<tr><td><i>&lt;FilmCoeff&gt;</i></td><td>string</td><td>Uniform convective heat transfer coefficient at surfaces (applicable only when <i>&lt;Uniform&gt;</i> is True)</td></tr>
<tr><td></td><td></td><td></td></tr>
<tr><td colspan="3">The following eight parameters (<b>*</b>) are applicable only when <i>&lt;Uniform&gt;</i> is False:</td></tr>
<tr><td>* <i>&lt;Project&gt;</i></td><td>string</td><td>Source project</td></tr>
<tr><td>* <i>&lt;Product&gt;</i></td><td>string</td><td>Source design product ("ElectronicsDesktop")</td></tr>
<tr><td>* <i>&lt;Design&gt;</i></td><td>string</td><td>Source design name</td></tr>
<tr><td>* <i>&lt;Soln&gt;</i></td><td>string</td><td>Source solution name</td></tr>
<tr><td>* <i>&lt;NAME:Params&gt;</i></td><td>string</td><td>Parameters array identifier (mapped variables, if any, and their values are included in this array)</td></tr>
<tr><td>* <i>&lt;ForceSourceToSolve&gt;</i></td><td>bool</td><td>True or False – simulate source design as needed</td></tr>
<tr><td>* <i>&lt;PreservePartnerSoln&gt;</i></td><td>bool</td><td>True or False – preserve source design solution</td></tr>
<tr><td>* <i>&lt;PathRelativeTo&gt;</i></td><td>string</td><td>Source path location relative to ("SourceProject" or "TargetProject")</td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>None</td></tr>
</table>

<table>
<tr><td><b>Python Syntax</b></td><td>AssignConvection ([&lt;<i>NAME</i>&gt;, &lt;<i>Objects</i>&gt;, &lt;<i>Faces</i>&gt;, &lt;<i>Temperature</i>&gt;, &lt;<i>Uniform</i>&gt;, &lt;<i>FilmCoeff</i>&gt;])</td></tr>
<tr><td><b>Python Example</b></td><td><pre>oModule.AssignConvection(
        [</pre></td></tr>
</table>

(**Uniform** Film Coefficient)

```
                                "NAME:Convection1",
                                "Objects:="         , ["Coil","HeatSink"],
                                "Faces:="           , [104,121],
                                "Temperature:="     , "AmbientTemp",
                                "Uniform:="         , True,
                                "FilmCoeff:="       , "2.5w_per_m2kel"
                ])
```

Python Syntax	AssignConvection ([<*NAME*>, <*Objects*>, <*Faces*>, <*Temperature*>, <*Uniform*>, <*Project*>, <*Product*>, <*Design*>, <*Soln*>, [<*NAME:Params*>], <*ForceSourceToSolve*>, <*PreservePartnerSoln*>, <*PathRelativeTo*>])

Python Example

(**Non-Uniform** Film Coefficient)

```
oModule.AssignConvection(
        [
                "NAME:Convection1"          ,
                "Objects:="                 , ["Coil","HeatSink"],
                "Faces:="                   , [104,121],
                "Temperature:="             , "AmbientTemp",
                "Uniform:="                 , False,
                "Project:="                 , "This Project*",
                "Product:="                 , "ElectronicsDesktop",
                "Design:="                  , "IcepakDesign1",
                "Soln:="                    , "Setup1 : SteadyState",
                [
                        "NAME:Params"        ,          # NOTE:  The highlighted line is a mapped va
name from the
                        "Pwr:="              , "0.625W"  # <--    source design and its associated va
mapped to the
                ],                                      #        source design (only present when
mapped variables exist).
                "ForceSourceToSolve:="   , True,
                "PreservePartnerSoln:="  , True,
                "PathRelativeTo:="       , "TargetProject"
```

	])

EditConvection

> **Note:**
>
> In the above script examples, you can substitute "*Edit*" for "*Assign*" to modify a pre-existing convection boundary instead of creating a new one. The script must identify the **NAME** of the existing boundary within the *EditConvection* command line. Then, you can change any of the setup parameters with the exception of the assignment *Faces* or *Objects*.
>
> To change the assignment *Faces* or *Objects*, you must use the *ReassignBoundary* command, provide the excitation NAME, and specify the new Faces and/or Objects.
>
> Script examples for editing a convection boundary with a uniform film coefficient are given below. In these examples, the name, ambient temperature, and film coefficient values are being modified.

- **UI Access:** Double-click the boundary in the Project Manager or right-click it and choose **Properties** from the shortcut menu.
- **Python Example:**

```
oModule.EditConvection("Convection1",
        [
                "NAME:Convection_Enclosure",
                "Temperature:="           , "75fah"
                "FilmCoeff:="             , "5w_per_m2kel",
        ])
```

AssignCylindricalSupport and *EditCylindricalSupport*

AssignCylindricalSupport

This command creates a Cylindrical Support boundary in a Mechanical–Modal or Mechanical-Structural design.

<table>
<tr><th>UI Access</th><th><b>Mechanical</b> &gt; <b>Boundaries</b> &gt; <b>Assign</b> &gt; <b>Cylindrical Support</b></th></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td>&lt;<i>NAME</i>&gt;</td><td>string</td><td>Boundary condition name</td></tr>
<tr><td>&lt;<i>Faces</i>&gt;</td><td>list</td><td>Faces included in the boundary condition assignment</td></tr>
<tr><td>&lt;<i>Axial</i>&gt;</td><td>string</td><td>State of axial direction constraint ("Fixed" or "Free")</td></tr>
<tr><td>&lt;<i>Radial</i>&gt;</td><td>string</td><td>State of radial direction constraint ("Fixed" or "Free")</td></tr>
<tr><td>&lt;<i>Tangential</i>&gt;</td><td>string</td><td>State of tangential direction constraint ("Fixed" or "Free")</td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>None</td></tr>
</table>

<table>
<tr><th>Python Syntax</th><th>AssignCylindricalSupport (&lt;<i>NAME</i>&gt;, &lt;<i>Faces</i>&gt;, &lt;<i>Axial</i>&gt;, &lt;<i>Radial</i>&gt;, &lt;<i>Tangential</i>&gt;)</th></tr>
<tr><td><b>Python Example</b></td><td><pre>
oModule.AssignCylindricalSupport(
        [
                "NAME:CylindricalSupport1",
                "Faces:="   , [27],
                "Axial:="   , "Free",
                "Radial:="   , "Fixed",
                "Tangential:="   , "Free",
        ])
</pre></td></tr>
</table>

EditCylindricalSupport

> **Note:**
>
> In the above script examples, you can substitute "*Edit*" for "*Assign*" to modify a pre-existing cylindrical support boundary instead of creating a new one. The script must identify the **Name** of the existing support within the *Edit* command line. Then, you can change the **Name** and the status of the **Axial, Radial,** and/or **Tangential** constraint directions (*Free* or *Fixed*). You cannot change the assignment *Faces* in this manner.
>
> Script examples for editing a cylindrical support boundary are given below.

- **UI Access:** Double-click the boundary in the Project Manager or right-click it and choose **Properties** from the shortcut menu.
- **Python Example:**

```
oModule.EditCylindricalSupport("CylindricalSupport1",
        [
                "NAME:CylindricalSupport_Left",
                "Axial:="                       , "Fixed",
                "Radial:="                      , "Fixed",
                "Tangential:="                  , "Free",
        ])
```

AssignDisplacement and *EditDisplacement*

AssignDisplacement

This command creates a Displacement excitation in a Mechanical–Structural design.

UI Access	**Mechanical** > **Excitations** > **Assign** > **Displacement**
Parameters	**Name** | **Type** | **Description**

<NAME>	string	Excitation name
<Faces>	list	Faces included in the excitation assignment
<Edges>	list	Edges included in the excitation assignment
<Vertices>	list	Vertices included in the excitation assignment (beta feature)
<DefinedBy>	string	Method of defining displacement direction (for faces only): "NormalTo" or "Component" – Omit this parameter when the assignment selection includes edges or vertices (always defined by components)
The following parameter (*) is only applicable when "DefinedBy" = "NormalTo":		
* *<Displacement>*	string	Magnitude of displacement normal to face (positive value pulls outward on face, negative pushes inward)
The following seven parameters (!) are only applicable when "DefinedBy" = "Component":		
! *<Coordinate System>*	string	Coordinate system name for displacement components
! *<FreeX>*	bool	Free in X direction (True or False)
! *<FreeY>*	bool	Free in Y direction (True or False)
! *<FreeZ>*	list	Free in Z direction (True or False)
! *<DisplacementX>*	string	Displacement in X direction (omit when "FreeX" = True)
! *<DisplacementY>*	list	Displacement in Y direction (omit when "FreeY" = True)
! *<DisplacementZ>*	list	Displacement in Z direction (omit when "FreeZ" = True)

Return Value	None

Python Syntax	For displacements normal to faces: AssignDisplacement ([<*NAME*>, <*Faces*>, <*DefinedBy*>, <*Displacement*>]) For component-based displacement of faces, edges, and/or vertices: AssignDisplacement ([<*NAME*>, <*Faces*>, <*Edges*>, <*Vertices*>, <*DefinedBy*>, *Coordinate System*>, <*FreeX*>, <*FreeY*>, <*FreeZ*>, *DisplacementX*>, <*DisplacementY*>, <*DisplacementZ*>])

Python Example

```
oModule.AssignDisplacement(
        [
                "NAME:Displacement1" ,
                "Faces:="            , ["11"],
                "DefinedBy:="        , "Component",
                "Coordinate System:=", "Global",
                "FreeX:="            , False,
                "FreeY:="            , True,
                "FreeZ:="            , False,
                "DisplacementX:="    , "0mm",
                "DisplacementZ:="    , "-0.05mm",
        ])
```

EditDisplacement

> **Note:**
>
> In the above script examples, you can substitute "*Edit*" for "*Assign*" to modify a pre-existing Displacement excitation instead of creating a new one. The script must identify the **Name** of the existing excitation within the *Edit* command line. Then, you can change any of the parameters except for the assignment *Objects*, which cannot be changed in this manner.
>
> Script examples for editing a Displacement excitation are given below.

- **UI Access:** Double-click the excitation in the Project Manager or right-click it and choose **Properties** from the shortcut menu.
- **Python Example:**

```
oModule.EditDisplacement("Displacement1",
       [
               "NAME:Displacement_Top",
               "DefinedBy:="          , "Component",
               "Coordinate System:="  , "RelCS2",
               "FreeX:="              , True,
               "FreeY:="              , False,
               "FreeZ:="              , False,
               "DisplacementY:="      , "0mm",
               "DisplacementZ:="      , "-0.05mm",
       ])
```

AssignEMLoss and *EditEMLoss*

AssignEMLoss

This command creates an EM Loss excitation in a Mechanical–Thermal design.

<table>
<tr><td><b>UI Access</b></td><td><b>Mechanical</b> &gt; <b>Excitations</b> &gt; <b>Assign</b> &gt; <b>EM Loss</b></td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;NAME&gt;</i></td><td>string</td><td>Boundary condition name</td></tr>
<tr><td><i>&lt;Objects&gt;</i></td><td>list</td><td>Objects included in the boundary condition assignment</td></tr>
<tr><td><i>&lt;Project&gt;</i></td><td>string</td><td>Source design's project name</td></tr>
<tr><td><i>&lt;Product&gt;</i></td><td>string</td><td>Source design's design type</td></tr>
<tr><td><i>&lt;Design&gt;</i></td><td>int</td><td>Source design's name</td></tr>
<tr><td><i>&lt;Soln&gt;</i></td><td>bool</td><td>Source design's solution name</td></tr>
<tr><td><i>&lt;NAME:Params&gt;</i></td><td>string</td><td>Parameters array identifier. Mapped variables, if any, and their values are included in this array</td></tr>
<tr><td><i>&lt;ForceSourceToSolve&gt;</i></td><td>bool</td><td>True or False</td></tr>
</table>
</td></tr>
</table>

<PreservePartnerSoln>	bool	True or False
<PathRelativeTo>	string	TargetProject or SourceProduct
<Intrinsics>	list	List of frequencies
<Q3DEMLossType>	string	Q3D only: DCVolOrACSurfLoss for DC volume or AC surface coupling or ContactResistanceLoss for DC contact resistance loss coupling
<SurfaceOnly>	list	List of surfaces

Return Value	None

Python Syntax	AssignEMLoss ([<*NAME*>, <*Objects*>, <*Project*>, <*Product*>, <*Design*>, <*Soln*>, [<*NAME:Params*>], <*ForceSourceToSolve*>, <*PreservePartnerSoln*>, <*PathRelativeTo*>, <*Intrinsics*>, <*SurfaceOnly*>])

Python Example

```
oModule.AssignEMLoss(
       [
               "NAME:EMLoss1",
               "Objects:="            , ["Bus3","Bus2","Bus1"],
               "Project:="            , "This Project*",
               "Product:="            , "ElectronicsDesktop",
               "Design:="             , "Bus_AC",
               "Soln:="               , "Setup1 : LastAdaptive",
               [
                       "NAME:Params",                   # NOTE:  The following two lines are mapped
ables from the
                       "appv:="           , "0",        # <-- source design and their associated val
mapped to the
                       "current:="        , "current"   # <-- source design (only present when mappe
ables exist).
               ],
               "ForceSourceToSolve:=" , True,
               "PreservePartnerSoln:=", True,
                "PathRelativeTo:="     , "TargetProject",
                "Intrinsics:="         , ["60Hz"],
                "Q3DEMLossType:="       , "ContactResistanceLoss",

                "SurfaceOnly:="        , ["Bus1"]
        ])
```

EditEMLoss

> **Note:**
>
> In the above script examples, you can substitute "*Edit*" for "*Assign*" to modify a pre-existing EM loss excitation instead of creating a new one. The script must identify the **NAME** of the existing excitation within the *Edit* command line. Then, you can change any of the parameters except for the assignment *Objects*.
>
> To change the assignment *Objects*, you must use the ReassignBoundary command, provide the excitation NAME, and specify the new Objects.
>
> Script examples for editing an EM loss excitation are given below.

- **UI Access:** Double-click the excitation in the Project Manager or right-click it and choose **Properties** from the shortcut menu.
- **Python Example:**

```
oModule.EditEMLoss("EMLoss1",
       [
               "NAME:EMLoss_250A",
               "Project:="            , "This Project*",
               "Product:="            , "ElectronicsDesktop",
               "Design:="             , "Bus_AC2",
                "Soln:="               , "Setup1 : LastAdaptive",
                [
                        "NAME:Params",
                        "appv:="           , "0",
                        "current:="        , "250A"
                ],
                "ForceSourceToSolve:=" , True,
                "PreservePartnerSoln:=", True,
                "PathRelativeTo:="     , "TargetProject",
                "Intrinsics:="         , ["50Hz"],
                "Q3DEMLossType:="       , "ContactResistanceLoss",
                "SurfaceOnly:="        , ["Bus1"]
        ])
```

AssignFixedSupport and *EditFixedSupport*

AssignFixedSupport

This command creates a Fixed Support boundary in a Mechanical–Modal or Mechanical-Structural design.

<table>
<tr><td><b>UI Access</b></td><td><b>Mechanical</b> &gt; <b>Boundaries</b> &gt; <b>Assign</b> &gt; <b>Fixed Support</b></td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td><i>&lt;NAME&gt;</i></td><td>string</td><td>Boundary condition name</td></tr>
<tr><td><i>&lt;Faces&gt;</i></td><td>list</td><td>Faces included in the boundary condition assignment</td></tr>
<tr><td><i>&lt;Edges&gt;</i></td><td>list</td><td>Edges included in the boundary condition assignment (Structural solutions only)</td></tr>
<tr><td><i>&lt;Vertices&gt;</i></td><td>list</td><td>Vertices included in the boundary condition assignment (beta feature, Structural solutions only)</td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>None</td></tr>
</table>

<table>
<tr><td><b>Python Syntax</b></td><td>AssignFixedSupport (&lt;<i>NAME</i>&gt;, &lt;<i>Faces</i>, &lt;<i>Edges</i>&gt;, &lt;<i>Vertices</i>&gt;)</td></tr>
<tr><td><b>Python Example</b></td><td><pre>oModule.AssignFixedSupport(
   [
      "NAME:FixedSupport1",
      "Faces:="   , [273],
      "Edges:="   , [45],
      "Vertices:="   , [25,28,59]
   ])</pre></td></tr>
</table>

EditFixedSupport

> **Note:**
>
> In the above script examples, you can substitute "*Edit*" for "*Assign*" to modify a pre-existing fixed support boundary instead of creating a new one. The script must identify the **Name** of the existing support within the *Edit* command line. Then, you can change the **Name** of the boundary. You cannot change the assignment entities in this manner.
>
> Script examples for editing a fixed support boundary are given below.

- **UI Access:** Double-click the boundary in the Project Manager or right-click it and choose **Properties** from the shortcut menu.
- **Python Example:**

```
oModule.EditFixedSupport("FixedSupport1",
   [
      "NAME:FixedSupport_Left"
   ])
```

AssignForce and *EditForce*

AssignForce

This command creates a Force excitation in a Mechanical–Structural design.

<table>
<tr><th>UI Access</th><th><b>Mechanical</b> > <b>Excitations</b> > <b>Assign</b> > <b>Force</b></th></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td><i><NAME></i></td><td>string</td><td>Excitation name</td></tr>
<tr><td><i><Faces></i></td><td>list</td><td>Faces included in the excitation assignment</td></tr>
<tr><td><i><Objects></i></td><td>list</td><td>Objects included in the excitation assignment (only for imported non-uniform forces from Maxwell 3D source designs; Uniform parameter = False)</td></tr>
<tr><td><i><Vertices></i></td><td>list</td><td>Vertices included in the excitation assignment (beta feature, only supported for uniform forces; Uniform parameter = True)</td></tr>
<tr><td><i><Uniform></i></td><td>bool</td><td>True or False
<b>Note:</b>
Mixed selection sets (including Faces and Vertices) are supported only when Uniform = True. When Uniform = False, and forces are being imported from a Maxwell 3D design, all assignement entities must be of the same type. Therefore, in this case you must assign forces to Faces and Objects using two separate Force excitations.</td></tr>
<tr><td colspan="3">The following four parameters (*) are applicable only when <i><Uniform></i> is True:</td></tr>
</table>
</td></tr>
</table>

* *<Coordinate System>*	string	Name of the coordinate system on which force components are based
* *<ForceX>*	string	Force component in X-direction (with units) assigned to each specified face
* *<ForceY>*	string	Force component in Y-direction (with units) assigned to each specified face
* *<ForceZ>*	string	Force component in Z-direction (with units) assigned to each specified face
The following nine parameters (**!**) are applicable only when *<Uniform>* is False:		
! *<Project>*	string	Source design's project name
! *<Product>*	string	Source design's design type
! *<Design>*	int	Source design's name
! *<Soln>*	string	Source design's solution name
! *<NAME:Params>*	string	Parameters array identifier. Mapped variables, if any, and their values are included in this array
! *<ForceSourceToSolve>*	bool	True or False
! *<PreservePartnerSoln>*	bool	True or False
! *<PathRelativeTo>*	string	TargetProject or SourceProduct
! *<Intrinsics>*	list	List of frequencies

Return Value	None

Python Syntax	AssignForce (<*NAME*>, <*Faces*>, <*Vertices*>, <*Uniform*>, <*Coordinate System*>, <*ForceX*>, <*ForceY*>, <*ForceZ*>)

Python Example

(**Uniform** Force)

```
oModule.AssignForce(
   [
      "NAME:Force1"          ,
      "Faces:="              , [12,19],
      "Vertices:="           , [45,57],
      "Uniform:="            , True,
      "Coordinate System:=", "Global",
```

	"ForceX:=" , "0.5kNewton", "ForceX:=" , "0newton", "ForceX:=" , "-5kNewton"])

EditForce

> **Note:**
>
> In the above script examples, you can substitute "*Edit*" for "*Assign*" to modify a pre-existing force excitation instead of creating a new one. The script must identify the **NAME** of the existing excitation within the *Edit* command line. Then, you can change any of the setup parameters with the exception of the assigned *Faces* or *Objects*.
>
> To change the assignment *Faces* or *Objects*, you must use the ReassignBoundary command, provide the excitation NAME, and specify the new Faces and/or Objects.
>
> Script examples for editing a Force excitation are given below. Both editing examples are for a uniform force excitation.

- **UI Access:** Double-click the excitation in the Project Manager or right-click it and choose **Properties** from the shortcut menu.
- **Python Example:**

```
oModule.EditForce("Force1",
   [
      "NAME:Force_Clamp"
      "Uniform:="          , True,
            "Coordinate System:=", "RelativeCS1",
            "ForceX:="           , "0.6kNewton",
            "ForceY:="           , "0newton",
            "ForceZ:="           , "-6kNewton"
        ])
```

AssignFrictionlessSupport and *EditFrictionlessSupport*

AssignFrictionlessSupport

This command creates a Frictionless Support boundary in a Mechanical–Modal or Mechanical-Structural design.

<table>
<tr><td><b>UI Access</b></td><td><b>Mechanical</b> > <b>Boundaries</b> > <b>Assign</b> > <b>Frictionless Support</b></td></tr>
<tr><td><b>Parameters</b></td><td><table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;NAME&gt;</i></td><td>string</td><td>Boundary condition name</td></tr>
<tr><td><i>&lt;Faces&gt;</i></td><td>list</td><td>Faces included in the boundary condition assignment</td></tr>
</table></td></tr>
<tr><td><b>Return Value</b></td><td>None</td></tr>
</table>

Python Syntax	AssignFrictionlessSupport(<*NAME*>, <*Faces*>)

Python Example	`oModule.AssignFrictionlessSupport(`
	`        [`
	`                "NAME:FrictionlessSupport1",`
	`                "Faces:="                  , [34,56]`
	`        ])`

EditFrictionlessSupport

> **Note:**
>
> In the above script examples, you can substitute "*Edit*" for "*Assign*" to modify a pre-existing frictionless support boundary instead of creating a new one. The script must identify the **Name** of the existing support within the *Edit* command line. Then, you can change the **Name** of the boundary. You cannot change the assignment *Faces* in this manner.
>
> Script examples for editing a frictionless support boundary are given below.

- **UI Access:** Double-click the boundary in the Project Manager or right-click it and choose **Properties** from the shortcut menu.
- **Python Example:**

```
oModule.EditFrictionlessSupport("FrictionlessSupport1",
        [
                "NAME:FrictionlessSupports_Front"
        ])
```

AssignHeatFlux and *EditHeatFlux*

AssignHeatFlux

This command creates a Heat Flux excitation in a Mechanical–Thermal design.

<table>
<tr><td><b>UI Access</b></td><td><b>Mechanical</b> > <b>Excitations</b> > <b>Assign</b> > <b>Heat Flux</b></td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;NAME&gt;</i></td><td>string</td><td>Excitation name</td></tr>
<tr><td><i>&lt;Faces&gt;</i></td><td>list</td><td>Faces included in the excitation assignment</td></tr>
<tr><td><i>&lt;Objects&gt;</i></td><td>list</td><td>Objects included in the excitation assignment. Assigning heat flux to an object is equivalent to assigning it to each of the object's faces.</td></tr>
</table>
</td></tr>
</table>

	<SourceType>	string	Choices are "Total Power" and "Surface Flux" with the latter representing uniform distributed power per unit surface area. This parameter only applies to the GetProperty, SetProperty, and ChangeProperty commands. When creating a heat flux excitation, you do not have to set this parameter. You only have to define either a TotalPower or SurfaceFlux value (below).
	<TotalPower>	string	Total thermal power applied to each of the specified faces (for SourceType = "Total Power").
	<SurfaceFlux>	string	Distributed thermal power per unit surface area (for SourceType="Surface Flux").
Return Value	None		

Python Syntax

AssignHeatFlux (<*NAME*>, <*Faces*>, <*Objects*>, <*TotalPower*>)

or

AssignHeatFlux (<*NAME*>, <*Faces*>, <*Objects*>, <*SurfaceFlux*>)

Python Example

(Total Power)

```
oModule.AssignHeatFlux(
       [
               "NAME:HeatFlux1",
               "Faces:="          , [92,51,8],
               "Objects:="        , ["Box1"],
               "TotalPower:="     , "1.5W"
       ])
```

Python Example

(Surface Flux)

```
oModule.AssignHeatFlux(
       [
               "NAME:HeatFlux1",
               "Faces:="          , [92,51,8],
               "Objects:="        , ["Box1"],
```

	`"SurfaceFlux:="    , "50KW_per_m2"`
	`])`

EditHeatFlux

> **Note:**
>
> In the above script examples, you can substitute "*Edit*" for "*Assign*" to modify a pre-existing heat flux excitation instead of creating a new one. The script must identify the **Name** of the existing excitation within the *Edit* command line. Then, you can change the **Name**, **TotalPower**, and/or **Surface Flux**. If you specify a *Surface Flux* value when editing an existing heat flux excitation of the Total Power type, the excitation is changed to the Surface Flux type. Conversely, specifing a *Total Power* value will change an excitation of the Surface Flux type to the Total Power type. You do not have to use ChangeProperty to reset the *SourceType* value.
>
> You cannot change the assignment *Faces* in this manner. Instead, you must use the *ReassignBoundary* command, provide the excitation *NAME*, and specify the new *Objects* or *Faces*.
>
> Script examples for editing a heat flux excitation are given below.

- **UI Access:** Double-click the excitation in the Project Manager or right-click it and choose **Properties** from the shortcut menu.
- **Python Example:**

```
oModule.EditHeatFlux("HeatFlux1",
        [
                "NAME:HeatFlux_ChipFaces",
                "TotalPower:="            , "12W"
        ])
```

AssignHeatGeneration and *EditHeatGeneration*

AssignHeatGeneration

This command creates a Heat Generation excitation in a Mechanical–Thermal design.

<table>
<tr><th>UI Access</th><th><b>Mechanical</b> &gt; <b>Excitations</b> &gt; <b>Assign</b> &gt; <b>Heat Generation</b></th></tr>
<tr><td><b>Parameters</b></td><td><table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;NAME&gt;</i></td><td>string</td><td>Excitation name</td></tr>
<tr><td><i>&lt;Objects&gt;</i></td><td>list</td><td>Objects included in the excitation assignment</td></tr>
<tr><td><i>&lt;TotalPower&gt;</i></td><td>string</td><td>Total thermal power distributed among the specified objects</td></tr>
</table></td></tr>
<tr><td><b>Return Value</b></td><td>None</td></tr>
</table>

Python Syntax	AssignHeatGeneration (<*NAME*>, <*Objects*>, <*TotalPower*>)

Python Example

```
oModule.AssignHeatGeneration(
       [
               "NAME:HeatGeneration1",
               "Objects:="           , "Ring",
               "TotalPower:="        , "75W"
       ])
```

EditHeatGeneration

> **Note:**
>
> In the above script examples, you can substitute "*Edit*" for "*Assign*" to modify a pre-existing heat generation excitation instead of creating a new one. The script must identify the **Name** of the existing excitation within the *Edit* command line. Then, you can change the **Name** and/or **TotalPower**. You cannot change the assignment *Objects* in this manner.
>
> Script examples for editing a heat generation excitation are given below.

- **UI Access:** Double-click the excitation in the Project Manager or right-click it and choose **Properties** from the shortcut menu.
- **Python Example:**

```
oModule.EditHeatGeneration("HeatGeneration1",
       [
               "NAME:HeatGeneration_Ring",
               "TotalPower:="            , "0.125HP"
       ])
```

AssignInitialTemperature and *EditInitialTemperature*

AssignInitialTemperature

Assigns an initial temperature patch to a object (for Transient Thermal solutions only).

<table>
<tr><td><b>UI Access</b></td><td><b>Mechanical</b> &gt; <b>Initial Temperature</b> &gt; <b>Assign</b></td></tr>
<tr><td><b>Parameters</b></td><td><table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;NAME&gt;</i></td><td>string</td><td>Temperature patch name</td></tr>
<tr><td><i>&lt;Objects&gt;</i></td><td>list</td><td>Objects included in the temperature patch assignment</td></tr>
<tr><td><i>&lt;Initial Temperature&gt;</i></td><td>string</td><td>"AmbientTemp" or initial temperature value and unit</td></tr>
</table></td></tr>
</table>

Return Value	None

Python Syntax	AssignInitialTemperature (<*NAME*>, <*Objects*>, <*Initial Temperature*>)

Python Example	`oModule.AssignInitialTemperature(`
	`        [`
	`                "NAME:InitTemp1",`
	`                "Objects:="             , ["Box1"],`
	`                "Initial Temperature:=" , "AmbientTemp"`
	`        ])`

EditInitialTemperature

> **Note:**
>
> In the above script examples, you can substitute "*Edit*" for "*Assign*" to modify a pre-existing initial temperature assignment instead of creating a new one. The script must identify the **NAME** of the existing initial temperature within the *EditInitialTemperature* command line. Then, you can change any of the setup parameters with the exception of the assignment *Objects*.
>
> To change the assignment *Objects*, you must use the *ReassignInitialTemperature, AddAssignmentToInitialTemperature,* or *RemoveAssignmentFromInitialTemperature* command. Provide the excitation NAME, and specify the new Object or Objects.
>
> Additionally, the *DeleteInitialTemperature, DeleteAllInitialTemperatures,* and *RenameInitialTemperature* commands are available to delete or rename previously assigned initial temperatures.
>
> Script examples for editing an initial temperature are given below. In these examples, the name and initial temperature parameters are both being modified.

- **UI Access:** Double-click the initial temperature in the Project Manager or right-click it and choose **Properties** from the shortcut menu.
- **Python Example:**

```
oModule.EditInitialTemperature("InitTemp1",
        [
                "NAME:InitTemp_Rod",
                "Initial Temperature:=" , "185cel"
        ])
```

AssignPressure and *EditPressure*

AssignPressure

This command creates a Pressure excitation in a Mechanical–Structural design.

<table>
<tr><th>UI Access</th><th><b>Mechanical</b> > <b>Excitations</b> > <b>Assign</b> > <b>Pressure</b></th></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td><i><NAME></i></td><td>string</td><td>Excitation name</td></tr>
<tr><td><i><Faces></i></td><td>list</td><td>Faces included in the excitation assignment</td></tr>
<tr><td><i><Faces></i></td><td>list</td><td>Faces included in the excitation assignment</td></tr>
<tr><td><i><DefinedBy></i></td><td>list</td><td>Specifies the type of pressure definition: "NormalTo" or "Component"</td></tr>
<tr><td></td><td></td><td>The following parameter is applicable when "DefinedBy" = "NormalTo":</td></tr>
<tr><td><i><Pressure></i></td><td>string</td><td>Normal pressure magnitude (with units) assigned to each specified face</td></tr>
<tr><td></td><td></td><td>The following four parameters are applicable when "DefinedBy" = "Component":</td></tr>
<tr><td><i><Coordinate System></i></td><td>string</td><td>Name of the coordinate system on which the pressure components are based</td></tr>
<tr><td><i><PressureX></i></td><td>string</td><td>Pressure component in X-direction (with units) assigned to each specified face</td></tr>
<tr><td><i><PressureY></i></td><td>string</td><td>Pressure component in Y-direction (with units) assigned to each specified face</td></tr>
<tr><td><i><PressureZ></i></td><td>string</td><td>Pressure component in Z-direction (with units) assigned to each specified face</td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>None</td></tr>
</table>

<table>
<tr>
<td><strong>Python Syntax</strong></td>
<td>AssignPressure (<*NAME*>, <*Faces*>, <*DefinedBy*>, <*Pressure*>)
or
AssignPressure (<*NAME*>, <*Faces*>, <*DefinedBy*>, <*Coordinate System*>, <*PressureX*>, <*PressureY*>, <*PressureZ*>)</td>
</tr>
<tr>
<td><strong>Python Example</strong>
(NormalTo)</td>
<td>

```
oModule.AssignPressure(
       [
               "NAME:Pressure1",
               "Faces:="       , [12,19],
               "DefinedBy:="   , "NormalTo",
               "Pressure:="    , "60psi"
       ])
```

</td>
</tr>
<tr>
<td><strong>Python Example</strong>
(Component)</td>
<td>

```
oModule.AssignPressure(
       [
               "NAME:Pressure1"     ,
               "Faces:="            , [12,19],
               "DefinedBy:="        , "Component",
               "Coordinate System:=", "Global",
               "PressureX:="        , "10kPascal",
               "PressureX:="        , "0pascal",
               "PressureX:="        , "-50kPascal"
       ])
```

</td>
</tr>
</table>

EditPressure

> **Note:**
>
> In the above script examples, you can substitute "*Edit*" for "*Assign*" to modify a pre-existing pressure excitation instead of creating a new one. The script must identify the **Name** of the existing excitation within the *Edit* command line. Then, you can change the **Name, DefinedBy, Coordinate System, Pressure, PressureX, PressureY,** and/or **PressureZ** parameters.
>
> You cannot change the assignment *Faces* in this manner. Instead, you must use the ReassignBoundary command, provide the excitation NAME, and specify the new Faces.
>
> Script examples for editing a Pressure excitation are given below.

- **UI Access:** Double-click the excitation in the Project Manager or right-click it and choose **Properties** from the shortcut menu.
- **Python Example:**

```
oModule.EditPressure("Pressure1",
        [
                "NAME:Pressure_Clamp",
                "DefinedBy:="        , "Component",
                "Coordinate System:=", "RelativeCS1",
                "PressureX:="        , "0.6kNewton"
                "PressureY:="        , "0newton"
                "PressureZ:="        , "-6kNewton"
        ])
```

AssignRotatingFluid and *EditRotatingFluid*

AssignRotatingFluid

This command creates a Rotating Fluid boundary in a Mechanical–Thermal design.

UI Access	**Mechanical** > **Boundaries** > **Assign** > **RotatingFluid**
Parameters	
Return Value	None

Name	Type	Description
<NAME>	string	Boundary name
<Objects>	list	Objects included in the boundary assignment
<BandObject>	list	Object indicating rotating portion of machine, extending from mid gap and encompassing the rotor
<GapThickness>	string	Thickness of the air gap between the rotor and stator
<Speed>	string	Rotation rate of the rotor
<AxisDirection>	string	Orientation of the axis of rotation
<RotorPosition>	string	Indicates the design of the rotating machine ("Inner" or "Outer" rotor)

Python Syntax	AssignRotatingFluid (<*NAME*>, <*Objects*>, <*BandObject*>, <*GapThickness*>, <*Speed*>, <*AxisDirection*>, <*RotorPosition*>)
Python Example	

```
oModule.AssignRotatingFluid(
       [
               "NAME:RotatingFluid1",
               "Objects:="          , ["Band","Cylinder1","InnerRegion"],
               "BandObject:="       , "Band"
               "GapThickness:="     , "1.5mm"
               "Speed:="            , "720rpm"
               "AxisDirection:="    , "Global::Z"
               "RotorPosition:="    , "Inner"
       ])
```

EditRotatingFluid

> **Note:**
>
> In the above script examples, you can substitute "*Edit*" for "*Assign*" to modify a pre-existing rotating fluid boundary instead of creating a new one. The script must identify the **Name** of the existing boundary within the *Edit* command line. Then, you can change the **Name, BandOjbect, GapThickness, Speed, AxisDirection,** and/or **RotorPosition**. You cannot change the assignment *Objects* in this manner.
>
> Script examples for editing a rotating fluid boundary are given below.

- **UI Access:** Double-click the boundary in the Project Manager or right-click it and choose **Properties** from the shortcut menu.
- **Python Example:**

```
oModule.EditRotatingFluid("RotatingFluid1",
        [
                "NAME:RotatingFluid_8PercentSlip",
                "BandObject:="                   , "Band_1"
                "GapThickness:=",                , "1.75mm"
                "Speed:="                        , "662rpm"
                "AxisDirection:="                , "Global::X"
                "RotorPosition:="                , "Outer"
        ])
```

AssignTemperature and *EditTemperature*

AssignTemperature

This command creates a Temperature boundary in a Mechanical–Thermal design.

UI Access	**Mechanical** > **Boundaries** > **Assign** > **Temperature**

	Name	Type	Description
Parameters	*<NAME>*	string	Boundary condition name
	<Faces>	list	Faces included in the boundary condition assignment
	<Temperature>	string	Temperature assigned to specified faces
Return Value	None		

Python Syntax	AssignTemperature (<*NAME*>, <*Faces*>, <*Temperature*>)
Python Example	

```
oModule.AssignTemperature(
        [
                "NAME:Temperature1",
                "Faces:="           , [8,17],
                "Temperature:="     , "90cel"
        ])
```

EditTemperature

> **Note:**
>
> In the above script examples, you can substitute "*Edit*" for "*Assign*" to modify a pre-existing temperature boundary instead of creating a new one. The script must identify the **Name** of the existing boundary within the *Edit* command line. Then, you can change the **Name** and/or **Temperature**. You cannot change the assignment *Faces* in this manner.
>
> Script examples for editing a temperature boundary are given below.

- **UI Access:** Double-click the boundary in the Project Manager or right-click it and choose **Properties** from the shortcut menu.
- **Python Example:**

```
oModule.EditTemperature("Temperature1",
        [
                "NAME:Temperature_ChipFace",
                "Temperature:="             , "180fah"
        ])
```

AssignThermalCondition and *EditThermalCondition*

AssignThermalCondition

This command creates a Thermal Condition excitation in a Mechanical–Structural design.

<table>
<tr><th>UI Access</th><th><b>Mechanical</b> &gt; <b>Excitations</b> &gt; <b>Assign</b> &gt; <b>Thermal Condition</b></th></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;NAME&gt;</i></td><td>string</td><td>Boundary condition name</td></tr>
<tr><td><i>&lt;Objects&gt;</i></td><td>list</td><td>Objects included in the boundary condition assignment</td></tr>
<tr><td><i>&lt;Uniform&gt;</i></td><td>bool</td><td>True or False</td></tr>
<tr><td></td><td></td><td></td></tr>
<tr><td colspan="3">The following parameter (^{1}) is only applicable when <i>&lt;Uniform&gt;</i> is True:</td></tr>
<tr><td>^{1} <i>&lt;ThermalCondition&gt;</i></td><td>string</td><td>User-specified uniform temperature (variable name, expression, or numerical value with units)</td></tr>
<tr><td></td><td></td><td></td></tr>
<tr><td colspan="3">The following eight parameters (^{2}) are only applicable when <i>&lt;Uniform&gt;</i> is False:</td></tr>
<tr><td>^{2} <i>&lt;Project&gt;</i></td><td>string</td><td>Source project</td></tr>
<tr><td>^{2} <i>&lt;Product&gt;</i></td><td>string</td><td>Source design product ("ElectronicsDesktop")</td></tr>
<tr><td>^{2} <i>&lt;Design&gt;</i></td><td>string</td><td>Source design name ("IcepakDesign<i>x</i>")</td></tr>
<tr><td>^{2} <i>&lt;Soln&gt;</i></td><td>string</td><td>Source solution name</td></tr>
<tr><td>^{2} <i>&lt;NAME:Params&gt;</i></td><td>string</td><td>Parameters array identifier (mapped variables, if any, and their values are included in this array)</td></tr>
</table>
</td></tr>
</table>

	[2] *<ForceSourceToSolve>*	bool	True or False (simulate source design as needed)
	[2] *<PreservePartnerSoln>*	bool	True or False (preserve source design solution)
	[2] *<PathRelativeTo>*	string	Source path location relative to ("SourceProject" or "TargetProject")
Return Value	None		

Python Syntax	AssignThermalCondition ([<*NAME*>, <*Objects*>, <*Uniform*>, <*ThermalCondition*>]) or AssignThermalCondition ([<*NAME*>, <*Objects*>, <*Uniform*>, <*Project*>, <*Product*>, <*Design*>, <*Soln*>, [<*NAME:Params*>], <*ForceSourceToSolve*>, <*PreservePartnerSoln*>, <*PathRelativeTo*>])

Python Example

[Uniform]

```
oModule.AssignThermalCondition(
    [
        "NAME:ThermalCondition1",
        "Objects:="             , ["Q1","Q2","U1"],
        "Uniform:="             , True,
        "ThermalCondition:="    , "EnvTemp"
    ])
```

Python Example

[Non-Uniform, with mapped variables]

```
oModule.AssignThermalCondition(
    [
        "NAME:ThermalCondition1",
        "Objects:="             , ["Q1","Q2","U1"],
        "Uniform:="             , False,
        "Project:="             , "This Project*"
        "Product:="             , "ElectronicsDesktop",
        "Design:="              , "IcepakDesign1",
        "Soln:="                , "Setup1 : SteadyState",
        [
            "NAME:Params",
            "Q1Pwr:="           , "50mW",  # <-- NOTE:  These three lines are variables
in the source
            "RPwr:="            , "25mW",  # <-- design and their associated values
mapped to the source
            "U1Pwr:="           , "40mW",  # <-- design (only present when mapped vari-
ables exist).
        ]
        "ForceSourceToSolve:="  , False,
        "PreservePartnerSoln:=" , False,
        "PathRelativeTo:="      , "TargetProject"
    ])
```

EditThermalCondition

> **Note:**
>
> In the above script examples, you can substitute "*Edit*" for "*Assign*" to modify a pre-existing Thermal Condition excitation instead of creating a new one. The script must identify the **Name** of the existing excitation within the *Edit* command line. Then, you can change any of the parameters except for the assignment *Objects*, which cannot be changed in this manner.
>
> Script examples for editing a Thermal Condition excitation are given below.

- **UI Access:** Double-click the excitation in the Project Manager or right-click it and choose **Properties** from the shortcut menu.
- **Python Example:**

```
oModule.EditThermalCondition("ThermalCondition1",
    [
        "NAME:ThermalCondition_95C",
        "Uniform:="                 , True,
        "ThermalCondition="         , "95cel"
    ])
```

12 - Thermal Monitor Script Commands

For *Mechanical* designs of the *Transient Thermal* solution type, a **Monitor** is available:

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

The thermal monitor is used for capturing and plotting temperatures at points assigned prior to solving the analysis setup. The following scripting commands are available:

AssignPointMonitor

ReassignPointMonitor

> **Note:**
>
> Displaying the *Thermal Monitor* tab of the *Solutions* window, where the monitor's results appear, is strictly controlled through the user interface. There are no script commands associated with viewing the thermal monitor, only for assigning or reassigning points that you want to monitor.

AssignPointMonitor

This command assigns one or more points at which temperature results will be monitored (for Mechanical–Transient Thermal designs only). Monitor points are defined according to the type of entity specified (at a vertex, midpoint of an edge, centroid of a flat face, or centroid of an object).

One point assignment is created for each specified entity, and the specified name is incremented automatically when more than one entity is specified. If the name you specify ends with a number, that number is incremented by 1 for subsequent points. If the name does not include a number, one is added to the second assignment point name, and that number is incremented for subsequent points.

<table>
<tr><td><b>UI Access</b></td><td><b>Mechanical</b> &gt; <b>Monitor</b> &gt; <b>Assign</b> &gt; <b>Point</b></td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><th>Name</th><th>Type</th><th>Description</th></tr><tr><td><i>&lt;NAME&gt;</i></td><td>string</td><td>Point monitor name</td></tr></table></td></tr>
</table>

	<Quantities>	string list	Result type to monitor – currently, only "Temperature" is supported
	<Objects>	string list	Solid, shell, or polyline objects used to define monitor points
	<Faces>	integer list	Faces (identified by number) used to define monitor points
	<Edges>	integer list	Edges (identified by number) used to define monitor points
	<Vertices>	integer list	Vertices (identified by number) used to define monitor points
Return Value	None		

Python Syntax	AssignPointMonitor([<*NAME*>, <*Quantities*>, <*Objects*>, <*Faces*>, <*Edges*>, <*Vertices*>])

Python Example

```
oModule.AssignPointMonitor(
       [
               "NAME:PointMonitor1",
               "Quantities:=" , ["Temperature"],
               "Objects:="    , ["Box1", "Cylinder1"],
               "Faces:="      , [15, 43],
               "Edges:="      , [28, 61],
               "Vertices:="   , [12, 20]
       ])
```

ReassignPointMonitor

This command reassigns a previously created thermal monitor point to a new entity and is applicable only to Mechanical–Transient Thermal designs. A monitor point is defined according to the type of entity specified (at a vertex, midpoint of an edge, centroid of a flat face, or centroid of an object).

You cannot reassign multiple points to an existing monitor point. Specify a single entity.

<table>
<tr><td><b>UI Access</b></td><td><b>Mechanical</b> &gt; <b>Monitor</b> &gt; <b>Reassign</b></td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td><i>&lt;NAME&gt;</i></td><td>string</td><td>Point monitor name</td></tr>
<tr><td><i>&lt;Objects&gt;</i></td><td>string list</td><td>Solid, shell, or polyline object used to define the reassigned monitor point *</td></tr>
<tr><td><i>&lt;Faces&gt;</i></td><td>integer list</td><td>Face (identified by number) used to define the reassigned monitor point *</td></tr>
<tr><td><i>&lt;Edges&gt;</i></td><td>integer list</td><td>Edge (identified by number) used to define the reassigned monitor point *</td></tr>
<tr><td><i>&lt;Vertices&gt;</i></td><td>integer list</td><td>Vertex (identified by number) used to define the reassigned monitor point *</td></tr>
<tr><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td>* <b>Note:</b> Despite <i>Type</i> = <b>list</b>, specify only a single entity from one category (object, face, edge, or vertex).</td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>None</td></tr>
</table>

<table>
<tr><td><b>Python Syntax</b></td><td>ReassignPointMonitor([&lt;<i>NAME</i>&gt;, &lt;<i>Objects</i>&gt; | &lt;<i>Faces</i>> | &lt;<i>Edges</i>> | &lt;<i>Vertices</i>&gt;])</td></tr>
<tr><td><b>Python Example</b></td><td><pre>
oModule.ReassignPointMonitor(
	[
		"NAME:PointMonitor1",
		"Faces:=" , [19]
	])
</pre></td></tr>
</table>

This page intentionally
left blank.

13 - Analysis Setup Module Script Commands

Mechanical analysis setup commands should be executed by the Analysis module, referred to in Mechanical scripts as the "AnalysisSetup" module.

Set oModule = oDesign.GetModule("AnalysisSetup")

AddTwoWayCoupling

Add a 2-Way Coupling setup to the specified analysis setup.

> **Note:**
>
> The only parameter you specify as part of adding a two-way coupling setup is the number of coupling iterations to run. The link setup (between source and target designs) is specified as part of an EM Loss excitation assignment in the target Mechanical design.

Access via Project Manager shortcut menu only.

<table>
<tr><td><b>UI Access</b></td><td>Under <i>Analysis</i>, right-click <b><i>&lt;SetupName&gt;</i></b> and choose <b>Add 2-Way Coupling</b> from the shortcut menu.</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>NumCouplingIters</td><td>string</td><td>Number of two-way coupling iterations to run. String represents an integer value (such as "3").</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>None</td></tr>
</table>

Python Syntax	AddTwoWayCoupling(<SetupName>, [<NAME>, <NumCouplingIters>])

Python Example

```
oModule.AddTwoWayCoupling("Setup1",
     [
          "NAME:Options",
          "NumCouplingIters:="     , 3
     ])
```

ClearLinkedData (Module)

Clear the linked data of the specified solution setups. (This command is similar to the ClearLinkedData command for the *design* level, which clears the linked data for all solution setups in the design.)

UI Access	**Project Manager** > ***{Design name}*** > **Analysis** > right-click ***{Setup name}*** > **Clear Linked Data** or, with a solution setup selected in the Project Manager: **Mechanical** > **Analysis** > **Clear Linked Data**
Parameters	<table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><i><SetupNameArray></i></td><td>Array</td><td>Specify the name of the setups whose linked data are to be cleaned</td></tr></table>
Return Value	None

Python Syntax	ClearLinkedData(<SetupNameArray>)
Python Example	`oModule.ClearLinkedData(["setup1"])`

CopySetup

Copy the specified Optimetrics setup.

UI Access	NA
Parameters	<table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><i><SetupName></i></td><td>String</td><td>Name of the setup.</td></tr></table>
Return Value	None.

Python Syntax	CopySetup (<*SetupName*>)
Python Example	`oModule.CopySetup ("OptimizationSetup1")`

CopyDrivenSetup

Copies a driven setup.

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;<i>SetupName</i>&gt;</td><td>String</td><td>Name of the setup.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	CopyDrivenSetup(<*SetupName*>)
Python Example	`oModule.CopyDrivenSetup("Setup1")`

CopyEigenSetup

Copies an eigen-analysis setup.

UI Access	N/A

<table>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;<i>SetupName</i>&gt;</td><td>String</td><td>Name of the setup.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	CopyEigenSetup(<*SetupName*>)
Python Example	`oModule.CopyEigenSetup("Setup1")`

DeleteSetups

Deletes one or more solution setups, which are specified by an array of solution setup names.

<table>
<tr><td><b>UI Access</b></td><td>Right-click a solution setup in the project tree and then click <b>Delete</b> on the shortcut menu, or delete selected solution setups in the <b>List</b> dialog box.</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;<i>SetupArray</i>&gt;</td><td>Array</td><td>Array of solution setup names.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	DeleteSetups (<*SetupArray*>)
Python Example	`oModule.DeleteSetups(["Setup1", "Setup2"])`

DeleteTwoWayCoupling

Delete the current 2-Way Coupling setup.

Access via Project Manager only.

<table>
<tr><td><b>UI Access</b></td><td>Under <i>Analysis</i>&gt; &lt;<i>SetupName</i>&gt; in the Project Manager, right-click <b>2-Way Coupling</b> and choose <b>Delete</b>from the shortcut menu.
Alternatively, under <i>Analysis</i>&gt; &lt;<i>SetupName</i>&gt; in the Project Manager, select <b>2-Way Coupling</b> and press <b>Delete</b>.</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>None</td><td></td><td></td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>None</td></tr>
</table>

Python Syntax	DeleteTwoWayCoupling()
Python Example	`oModule.DeleteTwoWayCoupling()`

EditSetup

Modifies an existing solution setup.

<table>
<tr><td><b>UI Access</b></td><td>Double-click a solution setup in the project tree to modify its settings.</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;<i>SetupName</i>&gt;</td><td>String</td><td>Name of the solve setup being edited.</td></tr></table></td></tr>
</table>

	<Attributes>	Array	Structured array.
			`Array("NAME:<NewSetupName>", <NamedParameters>)`
			`See the InsertSetup command for details and examples.`

Return Value	None.		

Python Syntax	EditSetup (<*SetupName*>, <*Attributes*>)

Python Example	`oModule.EditSetup("Setup1",`
	`[`
	`"NAME:NewSetup",`
	`"AdaptiveFreq:=", "1GHz",`
	`"EnableDistribProbTypeOption:=", false,`
	`"SaveFields:=", "true",`
	`"Enabled:=", true,`
	`["NAME:Cap",`
	`        "MaxPass:=", 10,`
	`        "MinPass:=", 1,`
	`        "MinConvPass:=", 2,`
	`        "PerError:=", 1,`
	`        "PerRefine:=", 30,`
	`        "AutoIncreaseSolutionOrder:=", false,`

```
        "SolutionOrder:=", "Normal"],
["NAME:DC",
        "Residual:=", 1E-005,
        "SolveResOnly:=", false,
        ["NAME:Cond",
                "MaxPass:=", 10,
                "MinPass:=", 1,
                "MinConvPass:=", 1,
                "PerError:=", 1,
                "PerRefine:=", 30),
        ["NAME:Mult",
                "MaxPass:=", 1,
                "MinPass:=", 1,
                "MinConvPass:=", 1,
                "PerError:=", 1,
                "PerRefine:=", 30]],
["NAME:AC",
        "MaxPass:=", 10,
        "MinPass:=", 1,
        "MinConvPass:=", 2,
        "PerError:=", 1,
        "PerRefine:=", 30]]
)
oModule.EditSetup("HfssDrivenAuto",
["NAME:Setup1",
  "IsEnabled:=", True,
  "AutoSolverSetting:=", "Balanced",
  ["NAME:Sweeps",
          ["NAME:Sweep",
            "RangeType:=", "LinearStep",
            "RangeStart:=", "1GHz",
            "RangeEnd:=", "10GHz",
            "RangeStep:=", "1GHz"
            ]
  ],
  "SaveRadFieldsOnly:=", False,
  "SaveAnyFields:=", True,
  "Type:=", "Discrete"
  ])

oModule.EditSetup("AC Magnetic",
[
  "NAME:AC Magnetic",
  "Enabled:="             , True,
  [
    "NAME:MeshLink",
    "ImportMesh:="          , False
  ],
  "MaximumPasses:="       , 4,
  "MinimumPasses:="       , 2,
  "MinimumConvergedPasses:=", 1,
  "PercentRefinement:="   , 30,
  "SolveFieldOnly:="      , False,
  "PercentError:="        , 0.1,
  "SolveMatrixAtLast:="   , True,
  "UseNonLinearIterNum:=" , False,
  [
    "NAME:ExpressionCache",
    [
      "NAME:CacheItem",
      "Title:="                , "eddy_loss1",
      "Expression:="           , "eddy_loss",
      "Intrinsics:="           , "Phase=\'0deg\'",
      "ReportType:="           , "Fields",
      [
        "NAME:ExpressionContext"
      ]
    ]
  ],
  "UseCacheFor:="          , ["Pass"],
  "UseIterativeSolver:="   , False,
  "RelativeResidual:="     , 0.0001,
  "NonLinearResidual:="    , 0.0001,
  "SmoothBHCurve:="        , False,
  "Frequency:="            , "200Hz",
  "HasSweepSetup:="        , False,
  "UseHighOrderShapeFunc:=", False,
  "UseMuLink:="            , False,
  "LossAdaptiveCtrl:="     , "0.3"
])
oModule.EditSetup("HfssDriven",
["NAME:Setup3",
        "AdaptMultipleFreqs:=", False,
        "Frequency:=", "5GHz",
        "MaxDeltaS:=", 0.02,
        "PortsOnly:=", False,
        "UseMatrixConv:=", False,
        "MaximumPasses:=", 6,
        "MinimumPasses:=", 1,
        "MinimumConvergedPasses:=", 1,
        "PercentRefinement:=", 30,
        "IsEnabled:=", True,
        "BasisOrder:=", 1,
        "DoLambdaRefine:=", True,
        "DoMaterialLambda:=", True,
        "SetLambdaTarget:=", False,
        "Target:=", 0.3333,
        "UseMaxTetIncrease:=", False,
        "PortAccuracy:=", 2,
        "UseABCOnPort:=", False,
        "SetPortMinMaxTri:=", False,
        "UseDomains:=", True,
        "UseIterativeSolver:=", False,
        "IterativeResidual:=", 1E-06,
        "DDMSolverResidual:=", 0.0001,
        "EnhancedLowFreqAccuracy:=", True,
        "SaveRadFieldsOnly:=", False,
        "SaveAnyFields:=", True,
        "IESolverType:=", "Auto",
        "LambdaTargetForIESolver:=", 0.15,
        "UseDefaultLambdaTgtForIESolver:=", True,
        "SkipIERegionSolveDuringAdaptivePasses:=", True
        "RayDensityPerWavelength:=", 4,
        "MaxNumberOfBounces:="  , 5,
        "InfiniteSphereSetup:=" , "Infinite Sphere1",
        "SkipSBRSolveDuringAdaptivePasses:=", True,
        "PTDUTDSimulationSettings:=", "PTD Correction + UTD Rays",
        "PTDEdgeDensity:="      , 20
])
```

Edit an SBR+ Setup with Fast Frequency Looping

```
oModule.EditSetup("HfssDriven",
        [
                "NAME:Setup1",
                "IsEnabled:="           , True,
[
        "NAME:MeshLink",
        "ImportMesh:="          , False
],
"IsSbrRangeDoppler:="   , False,
"RayDensityPerWavelength:=", 4,
"MaxNumberOfBounces:="  , 5,
"IsMonostaticRCS:="     , True,
"EnableCWRays:="        , False,
"RadiationSetup:="      , "",
"PTDUTDSimulationSettings:=", "None",
"FastFrequencyLooping:=", True,
[
        "NAME:Sweeps",
        [
                "NAME:Sweep",
                "RangeType:="           , "LinearStep",
                "RangeStart:="          , "1GHz",
                "RangeEnd:="            , "10GHz",
                "RangeStep:="           , "1GHz"
```

```
                        ]
                ],
                "ComputeFarFields:="    , True
                "UseSBREnhancedRadiatedPowerCalculation:=", True,
                "IsGOBlockageEnabled:=" , False,
                "GOBlockageSurfaceSelfBlock:=", False
        ])
```

Edit and RF Discharge Setup for HFSS

```
import ScriptEnv
ScriptEnv.Initialize("Ansoft.ElectronicsDesktop")
oDesktop.RestoreWindow()
oProject = oDesktop.SetActiveProject("coaxbend_discharge_r212")
oDesign = oProject.SetActiveDesign("HFSSDesign60degBendTeflon")
oModule = oDesign.GetModule("AnalysisSetup")
oModule.EditSetup("RFDischarge1",
  [
    "NAME:RFDischarge1",
    "Enabled:="             , True,
    [
      "NAME:MeshLink",
      "ImportMesh:="          , True,
          "Project:="             , "This Project*",
          "Product:="             , "HFSS",
          "Design:="              , "This Design*",
          "Soln:="                , "Setup1 : Sweep",
          [
            "NAME:Params",
            "bend_angle:="          , "bend_angle"
          ],
          "ForceSourceToSolve:="  , True,
          "PreservePartnerSoln:=" , False,
          "PathRelativeTo:="      , "SourceProduct",
          "ApplyMeshOp:="         , True
        ],
        [
          "NAME:Excitations",
          [
            "NAME:1:1",
            "Magnitude:="           , "1",
            "Phase:="               , "0deg"
          ],
    [
      "NAME:2:1",
      "Magnitude:="            , "0",
      "Phase:="                , "0deg"
    ]
  ],
  [
    "NAME:Frequencies",
    "10GHz"
  ],
  "Minimum Power:="       , "0.01",
  "Maximum Power:="       , "1000000",
  "Minimum Pressure:="    , "100pascal",
  "Maximum Pressure:="    , "101325pascal",
  "Postproc Sampling:="   , 500,
  "Temperature:="         , "0cel",
  "BuiltInGas:="          , "Helium"
])
```

EditTwoWayCoupling

Edit the properties of the current 2-Way Coupling setup.

Access via Project Manager only.

<table>
<tr><td><b>UI Access</b></td><td>Under <i>Analysis</i>&gt;, &lt;<i>SetupName</i>&gt; in the Project Manager, right-click <b>2-Way Coupling</b> and choose <b>Properties</b> from the shortcut menu.
Alternatively, under <i>Analysis</i>&gt;, &lt;<i>SetupName</i>&gt; in the Project Manager, select <b>2-Way Coupling</b> and edit settings in the <i>Mechanical</i> tab of the docked <i>Properties</i> window.</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>NumCouplingIters</td><td>string</td><td>Number of two-way coupling iterations to run. String represents an integer value (such as "5").</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>None</td></tr>
</table>

<table>
<tr><td><b>Python Syntax</b></td><td>EditTwoWayCoupling ([<NAME>, <NumCouplingIters&gt;])</td></tr>
<tr><td><b>Python Example</b></td><td><code>oModule.EditTwoWayCoupling("Setup1",</code>
<code>   [</code>
<code>   "NAME:Options",</code>
<code>   "NumCouplingIters:="   , 5</code>
<code>   ])</code></td></tr>
</table>

GetSetupCount

Gets the number of analysis setups in a design.

UI Access	N/A
Parameters	None.
Return Value	Integer containing number of setups.

Python Syntax	GetSetupCount ()
Python Example	`oModule.GetSetupCount()`

GetSetups

Gets the names of analysis setups in a design.

UI Access	N/A
Parameters	None.
Return Value	Array of analysis setup names

Python Syntax	GetSetups ()
Python Example	`oModule.GetSetups()`

GetSweeps

Gets the names of all sweeps in a given analysis setup.

UI Access	N/A
Parameters	<table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><i>&lt;SetupName&gt;</i></td><td>String</td><td>Name of specified setup.</td></tr></table>
Return Value	Array of sweep names.

Python Syntax	GetSweeps (<*SetupName*>)
Python Example	`oModule.GetSweeps("Setup1")`

InsertSetup [Mechanical]

This command adds a new solution setup. The information on this page is specific to the implementation of the InsertSetup command for Mechanical designs (all solution types: Modal, Steady-State Thermal, Transient Thermal, and Structural).

The parameters depend on the following model characteristics:

- whether you are setting up a *Modal, Steady-State Thermal, Transient Thermal,* or *Structural* solution (*SetupType* = **MechModal**, **MechSteadyStateThermal**, **MechTransientThermal**, or **MechStructural**, respectively)
- whether you are importing the mesh from a source design
- whether there are any variables mapped between the source and target designs (when importing a mesh)

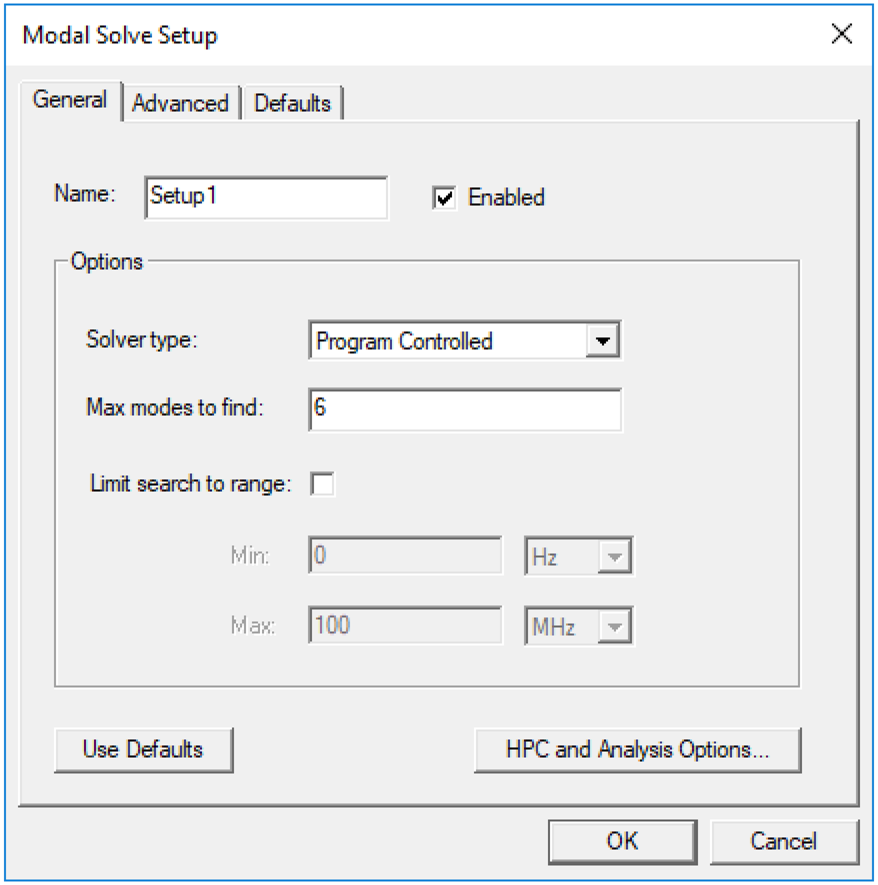
Certain options, when selected, activate additional parameters. For example, stepping parameters (**Initial**, **Min**, and **Max**) are only applicable if the *Stepping* option = **On**.

The following settings are common to all mechanical solution types and appear in the various tabs of the **Modal Solve Setup, Thermal Solve Setup**, or **Structural Solve Setup** dialog boxes or in the **Setup Link** dialog box. The *Setup Link* command is located under the *Advanced* tab of all of the mechanical **...Solve Setup** dialog boxes.

UI Access	**Mechanical** > **Analysis** > **Add Solution Setup**		
Parameters [Common]	**Name**	**Type**	**Description**
	<NAME>	string	Analysis setup name
	<Enabled>	bool	True | False (analysis setup enabled)
	<ImportMesh>	bool	True | False
	<NAME:MeshLink>	array name	Identifies the import mesh setup array, applicable when *<ImportMesh>* = True
	The following nine parameters (*) are only applicable when either *<ImportMesh>* = True (any solution type) or *<HasRestartLink>* = True (transient solutions only):		
	* *<Project>*	string	Source project
	* *<Product>*	string	Source design product ("ElectronicsDesktop")
	* *<Design>*	string	Source design name
	* *<Soln>*	string	Source solution name
	* *<NAME:Params>*	string	Parameters array identifier (mapped variables, if any, and their values are included in this array)
	* *<ForceSourceToSolve>*	bool	True | False – Simulate source design as needed
	* *<PreservePartnerSoln>*	bool	True | False – Preserve source design solution
	* *<PathRelativeTo>*	string	Source path location relative to ("SourceProject" | "TargetProject")
	* *<ApplyMeshOp>*	bool	True | False – Apply mesh operations in target design on the imported mesh (not applicable to transient restart analyses)
Return Value	None		

Parameters For Modal Solutions Only

The following image shows the default settings under the *General* tab of the **Modal Solve Setup** dialog box:


Modal Solve Setup

Name: Setup1

☑ Enabled

Options

Solver type: Program Controlled

Max modes to find: 6

Limit search to range: □

Min: 0 Hz

Max: 100 MHz

The following additional parameters correspond to Modal solutions:

UI Access	**Mechanical** > **Analysis** > **Add Solution Setup**

	Name	Type	Description
Parameters [Modal Solutions]	*<SetupType>*	string	"MechModal" for the Modal solution type
	<Max Modes>	integer	Number of requested modes for which to solve
	<Limit Search>	bool	True | False
	<Solver>	string	"Program Controlled" | "Direct" | "Iterative" | "Subspace" | "Supernode"
	The following two parameters (*) are only applicable when *<Limit Search>* is True:		
	* *<Range Max>*	string	Frequency upper limit of modes to solve
	* *<Range Min>*	string	Frequency lower limit of modes to solve
Return Value	None		

> **Note:**
>
> Only the settings specific to *Modal* solutions are listed in the preceding table. See the [Common] parameters section for other applicable settings.

Modal InsertSetup Script Examples

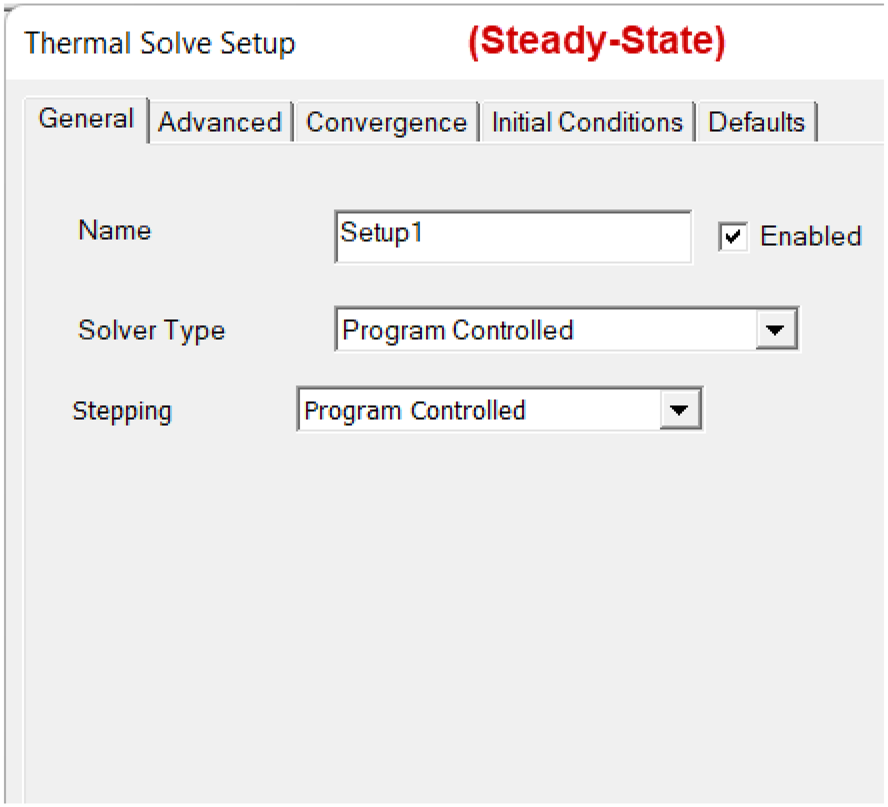
Python Syntax [Modal Solution]	InsertSetup (*<SetupType>*, [*<NAME>*, *<Enabled>*, [*<NAME:MeshLink>*, *<ImportMesh>*, *<Project>*, *<Product>*, *<Design>*, *<Soln>*, [*<NAME:Params>*], *<ForceSourceToSolve>*, *<PreservePartnerSoln>*, *<PathRelativeTo>*, *<ApplyMeshOps>*], *<Solver>*, *<Stepping>*, *<Initial>*, *<Min>*, *<Max>*])

Python Example
[Modal Solution]

```
oModule.InsertSetup("MechModal",
    [
       "NAME:Setup1",
       "Enabled:="                , True,
   [
      "ImportMesh:="         , True,
      "Project:="            , "This Project*",
      "Product:="            , "ElectronicsDesktop",
      "Design:="             , "HFSSDesign1",
      "Soln:="               , "Setup1 : LastAdaptive",
      [
         "NAME:Params",      # NOTE: The following two lines are variables
         "Width:=" , "6in",  # <---  in the source design and their associated
         "Height:=", "38in"  # <---  values mapped to the source design (only
      ],                     #       present when mapped variables exist).
      "ForceSourceToSolve:=" , True,
      "PreservePartnerSoln:=", True,
      "PathRelativeTo:="     , "TargetProject",
      "ApplyMeshOp:="        , True
   ]
   "Max Modes:="             , 6,
   "Limit Search:="          , True,
   "Range Max:="             , "20kHz",
   "Range Min:="             , "1Hz",
   "Solver:="                , "Program Controlled"
])
```

Parameters For Thermal Solutions Only

The following image shows the default settings under the *General* tab of the **Thermal Solve Setup** dialog box for **Steady-State** and **Transient** solutions, respectively:



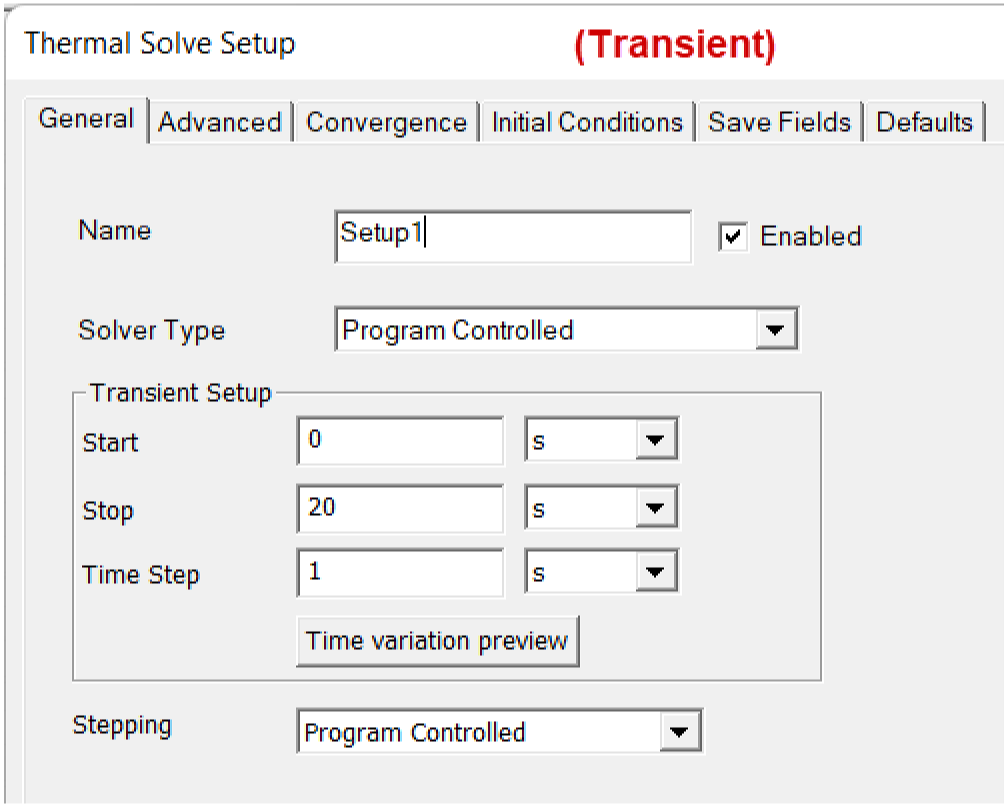
Thermal Solve Setup (Steady-State)

Name: Setup1

☑ Enabled

Solver Type: Program Controlled

Stepping: Program Controlled



Thermal Solve Setup **(Transient)**

Name: Setup1

☑ Enabled

Solver Type: Program Controlled

Transient Setup

Start: 0 s

Stop: 20 s

Time Step: 1 s

Stepping: Program Controlled

The following parameters correspond to *both* thermal solution types unless otherwise noted:

UI Access	**Mechanical > Analysis > Add Solution Setup**
Parameters	<table><tr><th>Name</th><th>Type</th><th>Description</th></tr></table>

[Steady-State and Transient Thermal Solutions]	*<SetupType>*	string	"MechThermal" for the Thermal solution type
	<Solver>	string	"Program Controlled" | "Direct" | "Iterative"
	<Stepping>	string	"Program Controlled" | "Off" | "On"
	<TemperatureConvergenceCondition>	string	"Program Controlled" | "On"
	<HeatConvergenceCondition>	string	"Program Controlled" | "Off" | "On"
	<Initial Temperature>	string	Global initial temperature assumed at start of solution (where not overridden by assigned temperature boundaries)
	<Start Time> (*)	string	Start time, with units (for transient solutions only)
	<Stop Time> (*)	string	Stop time, with units (for transient solutions only)
	<Time Step> (*)	string	Time step, with units (for transient solutions only)
	<SaveFieldsType> (*)	string	"None" | "Every N Steps" (for transient solutions only)
	<"N Steps"> (*)	string	How frequently to save fields (for transient solutions only when *<SaveFieldsType>* = "Every N Steps")
	<"HasRestartLink"> (*)	bool	True | False – Indicates if a link to a source design in specified for a restart analysis (for transient solutions only)
	<NAME:RestartSoln> (*)	array name	Identifies the restart solution setup array (for transient solutions only when *<HasRestartLink>* = True) (see "**Parameters** [Common]" in the first table above for the properties included in this array)
	The following three parameters (^{1}) are applicable when *<Stepping>* is "Off":		
	^{1} *<Stepping Define By>* (*)	string	"Substeps" | "Time"
	^{1} *<NumberOfSubSteps>*	string	Number of substeps to run
	^{1} *<TimeSubStep>* (*)	string	Substep time, with units (for transient solutions when *<Stepping Define By>* = "Time")
	The following three parameters (^{2}) are applicable when *<Stepping>* is "On":		

[2] *<Stepping Define By>* (*)	string	"Substeps" | "Time" (for transient solutions only)
[2] *<Initial>*	string	Initial number of substeps to run (or initial time for transient solutions when <*SteppingDefineBy*> = "Time")
[2] *<Min>*	string	Minimum number of substeps to solve (or minimum time for transient solutions when <*SteppingDefineBy*> = "Time"))
[2] *<Max>*	string	Maximum number of substeps to solve (or maximum time for transient solutions when <*SteppingDefineBy*> = "Time"))
The following two parameters ([3]) are applicable when <*TemperatureConvergenceCondition*> is "On":		
[3] *<TemperatureConvergenceTolerance>*	string	Percentage difference between adjacent iteration temperature results for the solution to be considered converged
[3] *<TemperatureConvergenceMinRef>*	string	Decimal numeric difference between adjacent iteration temperature results for the solution to be considered converged
The following two parameters ([4]) are applicable when <*HeatConvergenceCondition*> is "On":		
[4] *<HeatConvergenceTolerance>*	string	Percentage difference between adjacent iteration heat flux results for the solution to be considered converged
[4] *<HeatConvergenceMinRef>*	string	Decimal numeric difference between adjacent iteration heat flux results for the solution to be considered converged

Note:	An asterisk (*) after the **Name** and cyan-highlighted rows or text, indicate parameters applicable only to Transient Thermal solutions.
Return Value	None

> **Note:**
>
> Only the settings specific to thermal solutions are listed in the preceding table. See the common parameters section for other applicable settings.

Thermal InsertSetup Script Examples

Python Syntax [Steady-State Thermal Solution]	InsertSetup(<*SetupType*>, [<*NAME*>, <*Enabled*>, [<*NAME:MeshLink*>, <*ImportMesh*>, <*Project*>, <*Product*>, <*Design*>, <*Soln*>, [<*NAME:Params*>], <*ForceSourceToSolve*>, <*PreservePartnerSoln*>, <*PathRelativeTo*>, <*ApplyMeshOps*>], <*Solver*>, <*Stepping*>, <*Initial*>, <*Min*>, <*Max*>, <*TemperatureConvergenceCondition*>, <*TemperatureConvergenceTolerance*>, <*TemperatureConvergenceMinRef*>, <*HeatConvergenceCondition*>, <*HeatConvergenceTolerance*>, <*HeatConvergenceMinRef*>, <*Initial Temperature*>])

Python Example [Steady-State Thermal Solution]	`oModule.InsertSetup("MechSteadyStateThermal",`
	`[`
	`"NAME:Setup1",`
	`"Enabled:="                      , True,`
	`[`
	`"NAME:MeshLink",`
	`"ImportMesh:="                   , True,`
	`"Project:="                      , "This Project*",`
	`"Product:="                      , "ElectronicsDesktop",`
	`"Design:="                       , "HFSSDesign1",`
	`"Soln:="                         , "Setup1 : LastAdaptive",`
	`[`
	`"NAME:Params",                          # NOTE: The following two lines are`
	`variables`
	`"appv:="                         , "0",   # <---  in the source design and`
	`their associated`

```
                "current:="                       , "15A" # <---  values mapped to the source
design (only
         ],                                             #       present when mapped variables
exist).
         "ForceSourceToSolve:="          , True,
         "PreservePartnerSoln:="         , True,
         "PathRelativeTo:="              , "TargetProject",
         "ApplyMeshOp:="                 , True
      ]
      "Solver:="                         , "Program Controlled",
      "Stepping:="                       , "On",
      "Initial:="                        , "1",
      "Min:="                            , "1",
      "Max:="                            , "10",
      "TemperatureConvergenceCondition:=", "On",
      "TemperatureConvergenceTolerance:=", "0.25",
      "TemperatureConvergenceMinRef:="   , "0.01cel",
      "HeatConvergenceCondition:="       , "Program Controlled",
      "HeatConvergenceTolerance:="       , "0.5",
      "HeatConvergenceMinRef:="          , "1e-6W",
      "Initial Temperature:="            , "AmbientTemp"
   ])
```

Python Syntax [Transient Thermal Solution]	InsertSetup(<*SetupType*>, [<*NAME*>, <*Enabled*>, [<*NAME:MeshLink*>, <*ImportMesh*>, <*Project*>, <*Product*>, <*Design*>, <*Soln*>, [<*NAME:Params*>], <*ForceSourceToSolve*>, <*PreservePartnerSoln*>, <*PathRelativeTo*>, <*ApplyMeshOps*>], <*Solver*>, <*Stepping*>, <*Stepping Define By*>, <*Initial*>, <*Min*>, <*Max*>, <*TemperatureConvergenceCondition*>, <*TemperatureConvergenceTolerance*>, <*TemperatureConvergenceMinRef*>, <*HeatConvergenceCondition*>, <*HeatConvergenceTolerance*>, <*HeatConvergenceMinRef*>, <*Initial Temperature*>, <*Start Time*>, <*Stop Time*>, <*Time Step*>, <*SaveFieldsType*>, <*N Steps*>, <*HasRestartLink*>, [<*NAME:RestartSoln*>, <*Project*>, <*Product*>, <*Design*>, <*Soln*>, [<*NAME:Params*>], <*ForceSourceToSolve*>, <*PreservePartnerSoln*>, <*PathRelativeTo*>], <*Copy Fields from*

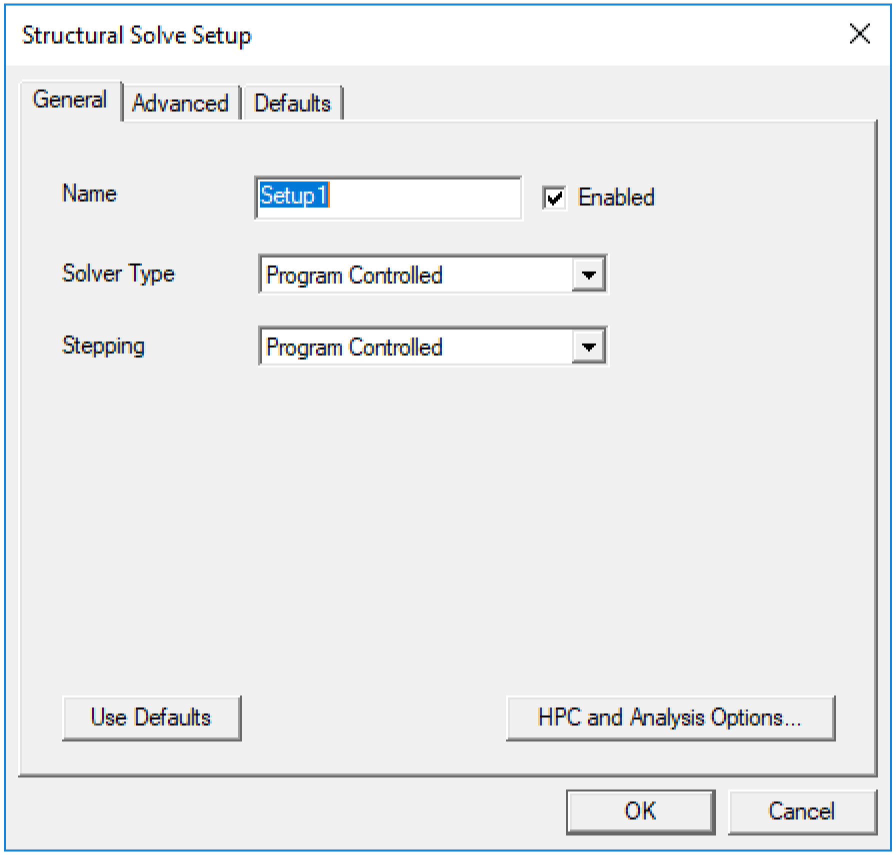
	Source>])

Python Example [Transient Thermal Solution]

```
oModule.InsertSetup("MechTransientThermal",
   [
      "NAME:Setup1",
      "Enabled:="                        , True,
      [
         "NAME:MeshLink",
         "ImportMesh:="                  , False,
      ]
      "Solver:="                         , "Program Controlled",
      "Stepping:="                       , "On",
      "Stepping Define By:="             , "Time",
      "Initial:="                        , "0.01s",
      "Min:="                            , "0.003s",
      "Max:="                            , "0.1s",
      "TemperatureConvergenceCondition:=", "On",
      "TemperatureConvergenceTolerance:=", "0.25",
      "TemperatureConvergenceMinRef:="   , "0.01cel",
      "HeatConvergenceCondition:="       , "Program Controlled",
      "HeatConvergenceTolerance:="       , "0.5",
      "HeatConvergenceMinRef:="          , "1e-6W",
      "Initial Temperature:="            , "AmbientTemp"
      "Start Time:="                     , "0s",
      "Stop Time:="                         , "20s",
      "Time Step:="                      , "1s",
      "SaveFieldsType:="                 , "Every N Steps",
      "N Steps:="                        , "2"
      "HasRestartLink:="                 , True,
      [
         "NAME:RestartSoln",
         "Project:="                     , "This Project*",
         "Product:="                     , "ElectronicsDesktop",
         "Design:="                      , "MechanicalDesign1",
         "Soln:="                          , "Setup1 : Solution",
         [
            "NAME:Params"
         ],
         "ForceSourceToSolve:="            , False,
         "PreservePartnerSoln:="           , True,
         "PathRelativeTo:="                , "TargetProject"
      ],
      "Copy Fields From Source:="          , True
   ])
```

Parameters For Structural Solutions Only

The following image shows the default settings under the *General* tab of the **Structural Solve Setup** dialog box:



Name: Setup1 ☑ Enabled

Solver Type: Program Controlled

Stepping: Program Controlled

The following parameters correspond to the fields contained under this tab:

<table>
<tr><td><b>UI Access</b></td><td><b>Mechanical</b> &gt; <b>Analysis</b> &gt; <b>Add Solution Setup</b></td></tr>
<tr><td><b>Parameters</b>
[Structural Solutions]</td><td><table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td><i>&lt;SetupType&gt;</i></td><td>string</td><td>"MechStructural" for the Structural solution type</td></tr>
</table></td></tr>
</table>

<Solver>	string	"Program Controlled" | "Direct" | "Iterative"
<Stepping>	string	"Program Controlled" | "Off" | "On"
The following parameter ([1]) is only applicable when *<Stepping>* is "Off":		
[1] *<NumSubSteps>*	string	Number of substeps to run
The following three parameters ([2]) are only applicable when *<Stepping>* is "On":		
[2] *<Initial>*	string	Initial number of substeps to run
[2] *<Min>*	string	Minimum number of substeps to solve
[2] *<Max>*	string	Maximum number of substeps to solve

Return Value	None

Note:

Only the settings specific to Structural solutions are listed in the preceding table. See the common parameters section for other applicable settings.

Structural InsertSetup Script Examples

Python Syntax [Structural Solution]	InsertSetup(<*SetupType*>, [<*NAME*>, <*Enabled*>, [<*NAME:MeshLink*>, <*ImportMesh*>, <*Project*>, <*Product*>, <*Design*>, <*Soln*>, [<*NAME:Params*>], <*ForceSourceToSolve*>, <*PreservePartnerSoln*>, <*PathRelativeTo*>, <*ApplyMeshOps*>], <*Solver*>, <*Stepping*>, <*Initial*>, <*Min*>, <*Max*>])
Python Example	`oModule.InsertSetup("MechStructural",`

[Structural Solution]

```
   [
      "NAME:Setup1",
      "Enabled:="               , True,
      [
         "NAME:MeshLink",
         "ImportMesh:="         , True,
         "Project:="            , "This Project*",
         "Product:="            , "ElectronicsDesktop",
         "Design:="             , "HFSSDesign1",
         "Soln:="               , "Setup1 : LastAdaptive",
         [
            "NAME:Params",              # NOTE: The following two lines are variables
            "appv:="            , "0",  # <---  in the source design and their asso-
ciated
            "current:="         , "15A" # <---  values mapped to the source design
(only
         ],                             #       present when mapped variables exist).
         "ForceSourceToSolve:=" , True,
         "PreservePartnerSoln:=", True,
         "PathRelativeTo:="     , "TargetProject",
         "ApplyMeshOp:="        , True
      ]
      "Solver:="                , "Program Controlled",
      "Stepping:="              , "On",
      "Initial:="               , "1",
      "Min:="                   , "1",
      "Max:="                   , "10"
   ])
```

PasteDrivenSetup

Pastes a driven setup.

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

Python Syntax	PasteDrivenSetup()
Python Example	`oModule.PasteDrivenSetup()`

PasteEigenSetup

Pastes an eigen-analysis setup.

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

Python Syntax	PasteEigenSetup ()
Python Example	`oModule.PasteEigenSetup()`

PasteSetup

Use: Paste a solve setup.

Syntax: PasteSetup

Return Value: None

RenameSetup

Renames an existing solution setup.

UI Access	Right-click a solution setup in the Project Manager and then click **Rename** on the shortcut menu.
Parameters	<table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;*OldSetupName*&gt;</td><td>String</td><td>Name of the solution setup being renamed.</td></tr><tr><td>&lt;*NewSetupName*&gt;</td><td>String</td><td>New name for the solution setup.</td></tr></table>
Return Value	None.

Python Syntax	RenameSetup (<*OldSetupName*>, <*NewSetupName*>)
Python Example	`oModule.RenameSetup("Setup1", "MySetup")`

ResetToTimeZero

Resets a simulation to time zero.

UI Access	**CleanStop** when running Electronics Desktop in Batchmode.

Parameters	Name	Type	Description
	<setupName>	String	Name of the simulation setup to be reset.
Return Value	None.		

Python Syntax	ResetToTimeZero(<*setupName*>)
Python Example	`oModule.ResetToTimeZero("Setup1")`

RevertAllToInitial

Marks the current mesh for all solution setups as invalid. This will force the next simulation to begin with the initial mesh.

UI Access	> **Analysis Setup** > **Revert to Initial Mesh**.
Parameters	None.
Return Value	None.

Python Syntax	RevertAllToInitial ()
Python Example	`oModule.RevertAllToInitial()`

RevertAllToZeroDisplacement

Reverts the displacement values of all objects to zero.

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

Python Syntax	RevertAllToZeroDisplacement()
Python Example	`oModule.RevertAllToZeroDisplacement()`

RevertCoupledSolution

Invalidate results of previously run iterations and begin at iteration 1 when solved again. This command is useful if you wish to reduce the number of iterations solved to fewer than the number of previously solved iterations.

Access via the menu bar or Project Manager.

<table>
<tr><td><b>UI Access</b></td><td>From menu bar: <b>Mechanical> Analysis> Revert Coupled Solution</b>
Right-click <b>Analysis</b> in the Project Manager and choose <b>Revert Coupled Solution</b>from the shortcut menu.
Alternatively, under <i>Analysis</i> in the Project Manager, right-click <b><i>&lt;SetupName&gt;</i></b> and choose <b>Revert Coupled Solution</b>.</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>None</td><td></td><td></td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>None</td></tr>
</table>

Python Syntax	RevertCoupledSolution()
Python Example	`oModule.RevertCoupledSolution()`

RevertSetupToInitial

Marks the current mesh for a solution setup as invalid. This will force the next simulation to begin with the initial mesh.

<table>
<tr><td><b>UI Access</b></td><td>Right-click a solution setup in the Project Manager and then click <b>Rename</b> on the shortcut menu.</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;<i>SetupName</i>&gt;</td><td>String</td><td>Name of specified setup.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	RevertSetupToInitial (<*SetupName*>)
Python Example	`oModule.RevertSetupToInitial("Setup1")`

RevertSetupToInitialCondition

Reverts one or more specified setups to their initial condition. Used for linked thermal designs to revert the target solution (clearing restart, thermal monitor, and field results).

UI Access	**Project Manager** > **Analysis** > right-click ***SetupName*** > **Revert to Initial Condition**. or, with a specific setup selected under **Analysis** in the Project Manager:

<table>
<tr><td></td><td><b>Mechanical</b> &gt; <b>Analysis</b> &gt; <b>Revert to Initial Condition</b> (from the menu bar)</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;<i>SetupName</i>&gt;</td><td>String</td><td>Name of solution setup.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>None</td></tr>
</table>

Python Syntax	RevertSetupToInitialCondition(<*setupName*>)
Python Example	`oModule.RevertSetupToInitialCondition('Setup1')`

RevertSetupToZeroDisplacement

Reverts the displacement values of objects for a specified setup to zero.

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;<i>SetupName</i>&gt;</td><td>String</td><td>Name of specified setup.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	RevertSetupToZeroDisplacement (<*SetupName*>)
Python Example	`oModule.RevertSetupToZeroDisplacement("Setup1")`

SolveSetup

Solves the specified setup.

UI Access	Right-click the setup in the project tree, and then click **Analyze** on the shortcut menu.
Parameters	<table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;*SetupName*&gt;</td><td>String</td><td>Name of the setup to be solved</td></tr></table>
Return Value	None

Python Syntax	SolveSetup (<*SetupName*>)
Python Example	`oModule.SolveSetup ("Setup1")`

14 - Optimetrics Module Script Commands

Optimetrics script commands should be executed by the `Optimetrics` module.

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

```
oModule.CommandName <args>
```

Conventions Used in this Chapter

`<VarName>`

Type: <string>

Name of a variable.

`<VarValue>`

Type: <string>

Value with unit (i.e., <value>, but cannot be an expression).

`<StartV>`

Type: <VarValue>

The starting value of a variable.

`<StopV>`

Type: <VarValue>

The stopping value of a variable.

`<MinV>`

Type: <VarValue>

The minimum value of a variable.

`<MaxV>`

Type: <VarValue>

The maximum value of a variable.

`<IncludeVar>`

Type: <bool>

Specifies whether the variable is included in the analysis.

`<StartingPoint>`

```
  Array("NAME:StartingPoint", "<VarName>:=",
    <VarValue>, .... "<VarName>:=", <VarValue>)
```

`<SaveField>`

Type: <bool>

Specifies whether HFSS will remove the non-nominal field solution.

`<MaxIter>`

Type: <int>

Maximum iteration allowed in an analysis.

`<PriorSetup>`

Type: <string>

The name of the embedded parametric setup.

`<Precede>`

Type: <bool>

If true, the embedded parametric setup will be solved before the analysis begins.

If false, the embedded parametric setup will be solved during each iteration of the analysis.

```
<Constraint>
  Array("NAME:LCS",
    "lc:=", Array("<VarName>:=",
      <Coeff>, …"<VarName>:=", <Coeff>, "rel:=", <Cond>, "rhs:=", <Rhs>), ...
    "lc:=", Array("<VarName>:=", <Coeff>, ..."
      <VarName>:=", <Coeff>, "rel:=", <Cond>, "rhs:=",
      <Rhs>))
```

`<Coeff>`

Type: <double>

Coefficient for a variable in the linear constraint.

`<Cond>`

Type: <string>

Inequality condition.

`<Rhs>`

Type: <double>

Inequality value.

`<OptiGoalSpec>`

```
  "Solution:=", <Soln>, "Calculation:=", <Calc>,
  "Context:=, <Geometry>
  Array("NAME:Ranges",
    "Range:", Array("Var:=",
      <VarName>, "Type:=", <RangeType>, "Start:=",
      <StartV>, "Stop:=", <StopV>), ...
    "Range:", Array("Var:=", <VarName>, "Type:=",
      <RangeType>, "Start:=", <StartV>, "Stop:=",
      <StopV>))
```

`<Soln>`

Type: <string>

Name of the solution.

`<Calc>`

Type: <string>

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

`<ContextName>`

Type: <string>

Name of context needed in the evaluation of <Calc>.

`<Geometry>`

Type: <string>

Name of geometry needed in the evaluation of <Calc>.

`<RangeType>`

Type: <string>

if "r", start and stop values specify a range for the variable.

if "s", start values specify the single value for the variable.

EditSetup

EditSetup [Optimization]

EditSetup [Sensitivity]

EditSetup [Statistical]

GetPropNames [Optimetrics]

GetPropValue [Optimetrics]

GetSetupNames [Optimetrics]

GetSetupNamesByType [Optimetrics]

InsertSetup [Parametric]

InsertSetup [Optimization]

InsertSetup [Sensitivity]

InsertSetup [Statistical]

PasteSetup [Optimetrics]

RenameSetup [Optimetrics]

SetPropValue [Optimetrics]

SolveSetup [Optimetrics]

The topics for this section include:

General Commands Recognized by the Optimetrics Module

Parametric Script Commands

Optimization Script Commands

Sensitivity Script Commands

Statistical Script Commands

CopySetup

Copy the specified Optimetrics setup.

<table>
<tr><td><b>UI Access</b></td><td>NA</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><i>&lt;SetupName&gt;</i></td><td>String</td><td>Name of the setup.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	CopySetup (<*SetupName*>)
Python Example	`oModule.CopySetup ("OptimizationSetup1")`

DeleteSetups [Optimetrics]

Deletes the specified Optimetrics setups.

<table>
<tr><td><b>UI Access</b></td><td>Right-click the setup in the project tree, and then click <b>Delete</b> on the shortcut menu</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;NameArray&gt;</td><td>Array of Strings</td><td>An Array of Setup Names</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>None</td></tr>
</table>

Python Syntax	DeleteSetups (<NameArray>)
Python Example	`oModule.DeleteSetups (["OptimizationSetup1"])`

DistributedAnalyzeSetup

Distributes all variable value instances within a parametric sweep to different machines already specified from within the user interface

<table>
<tr><td><b>UI Access</b></td><td>Right-click the parametric setup name in the project tree and select Distribute Analysis.</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;ParametricSetupName&gt;</td><td>String</td><td>Name of the Setup</td></tr><tr><td>isBlocking</td><td>Boolean</td><td>An optional arg that defaults to <code>true</code>. If it's <code>false</code>, the command immediately returns while the analysis or analyses run in the background. The status of the analyses can be checked with the <code>AreThereSimulationsRunning</code> command.</td></tr></table></td></tr>
</table>

Return Value	None

Python Syntax	DistributedAnalyzeSetup (<ParametricSetupName>)
Python Example	`oModule.DistributedAnalyzeSetup("ParametricSetup1")`

EditSetup

Modifies an existing solution setup.

<table>
<tr><td><b>UI Access</b></td><td>Double-click a solution setup in the project tree to modify its settings.</td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td>&lt;<i>SetupName</i>&gt;</td><td>String</td><td>Name of the solve setup being edited.</td></tr>
<tr><td>&lt;<i>Attributes</i>&gt;</td><td>Array</td><td>Structured array.
<code>Array("NAME:<NewSetupName>", <NamedParameters&gt;)</code>
<code>See the InsertSetup command for details and examples.</code></td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

<table>
<tr><td><b>Python Syntax</b></td><td>EditSetup (&lt;<i>SetupName</i>&gt;, &lt;<i>Attributes</i>&gt;)</td></tr>
<tr><td><b>Python Example</b></td><td><code>oModule.EditSetup("Setup1",</code>
<code>[</code></td></tr>
</table>

```
"NAME:NewSetup",
"AdaptiveFreq:=", "1GHz",
"EnableDistribProbTypeOption:=", false,
"SaveFields:=", "true",
"Enabled:=", true,
["NAME:Cap",
        "MaxPass:=", 10,
        "MinPass:=", 1,
        "MinConvPass:=", 2,
        "PerError:=", 1,
        "PerRefine:=", 30,
        "AutoIncreaseSolutionOrder:=", false,
        "SolutionOrder:=", "Normal"],
["NAME:DC",
        "Residual:=", 1E-005,
        "SolveResOnly:=", false,
        ["NAME:Cond",
                "MaxPass:=", 10,
                "MinPass:=", 1,
                "MinConvPass:=", 1,
                "PerError:=", 1,
                "PerRefine:=", 30),
        ["NAME:Mult",
                "MaxPass:=", 1,
                "MinPass:=", 1,
                "MinConvPass:=", 1,
                "PerError:=", 1,
                "PerRefine:=", 30]],
["NAME:AC",
        "MaxPass:=", 10,
        "MinPass:=", 1,
        "MinConvPass:=", 2,
        "PerError:=", 1,
        "PerRefine:=", 30]]
)
oModule.EditSetup("HfssDrivenAuto",
["NAME:Setup1",
  "IsEnabled:=", True,
  "AutoSolverSetting:=", "Balanced",
  ["NAME:Sweeps",
          ["NAME:Sweep",
            "RangeType:=", "LinearStep",
            "RangeStart:=", "1GHz",
            "RangeEnd:=", "10GHz",
            "RangeStep:=", "1GHz"
            ]
  ],
  "SaveRadFieldsOnly:=", False,
  "SaveAnyFields:=", True,
  "Type:=", "Discrete"
  ])

oModule.EditSetup("AC Magnetic",
[
  "NAME:AC Magnetic",
  "Enabled:="             , True,
  [
    "NAME:MeshLink",
    "ImportMesh:="           , False
  ],
  "MaximumPasses:="       , 4,
  "MinimumPasses:="       , 2,
"MinimumConvergedPasses:=", 1,
"PercentRefinement:="   , 30,
"SolveFieldOnly:="      , False,
"PercentError:="        , 0.1,
"SolveMatrixAtLast:="   , True,
"UseNonLinearIterNum:=" , False,
[
  "NAME:ExpressionCache",
  [
    "NAME:CacheItem",
    "Title:="               , "eddy_loss1",
    "Expression:="          , "eddy_loss",
    "Intrinsics:="          , "Phase=\'0deg\'",
    "ReportType:="          , "Fields",
    [
      "NAME:ExpressionContext"
    ]
  ]
],
"UseCacheFor:="         , ["Pass"],
  "UseIterativeSolver:="  , False,
  "RelativeResidual:="    , 0.0001,
  "NonLinearResidual:="   , 0.0001,
  "SmoothBHCurve:="       , False,
  "Frequency:="           , "200Hz",
  "HasSweepSetup:="       , False,
  "UseHighOrderShapeFunc:=", False,
  "UseMuLink:="           , False,
  "LossAdaptiveCtrl:="    , "0.3"
])
oModule.EditSetup("HfssDriven",
["NAME:Setup3",
        "AdaptMultipleFreqs:=", False,
        "Frequency:=", "5GHz",
        "MaxDeltaS:=", 0.02,
        "PortsOnly:=", False,
        "UseMatrixConv:=", False,
        "MaximumPasses:=", 6,
        "MinimumPasses:=", 1,
        "MinimumConvergedPasses:=", 1,
        "PercentRefinement:=", 30,
        "IsEnabled:=", True,
        "BasisOrder:=", 1,
        "DoLambdaRefine:=", True,
        "DoMaterialLambda:=", True,
        "SetLambdaTarget:=", False,
        "Target:=", 0.3333,
        "UseMaxTetIncrease:=", False,
        "PortAccuracy:=", 2,
        "UseABCOnPort:=", False,
        "SetPortMinMaxTri:=", False,
        "UseDomains:=", True,
        "UseIterativeSolver:=", False,
        "IterativeResidual:=", 1E-06,
        "DDMSolverResidual:=", 0.0001,
        "EnhancedLowFreqAccuracy:=", True,
        "SaveRadFieldsOnly:=", False,
        "SaveAnyFields:=", True,
        "IESolverType:=", "Auto",
        "LambdaTargetForIESolver:=", 0.15,
        "UseDefaultLambdaTgtForIESolver:=", True,
        "SkipIERegionSolveDuringAdaptivePasses:=", True
        "RayDensityPerWavelength:=", 4,
        "MaxNumberOfBounces:="   , 5,
        "InfiniteSphereSetup:="  , "Infinite Sphere1",
        "SkipSBRSolveDuringAdaptivePasses:=", True,
        "PTDUTDSimulationSettings:=", "PTD Correction + UTD Rays",
        "PTDEdgeDensity:="       , 20
])
```

Edit an SBR+ Setup with Fast Frequency Looping

```
oModule.EditSetup("HfssDriven",
        [
                "NAME:Setup1",
                "IsEnabled:="            , True,
                [
                        "NAME:MeshLink",
                        "ImportMesh:="           , False
                ],
                "IsSbrRangeDoppler:="    , False,
                "RayDensityPerWavelength:=", 4,
                "MaxNumberOfBounces:="   , 5,
                "IsMonostaticRCS:="      , True,
                "EnableCWRays:="         , False,
                "RadiationSetup:="      , "",
                "PTDUTDSimulationSettings:=", "None",
                "FastFrequencyLooping:=", True,
                [
                        "NAME:Sweeps",
                        [
                                "NAME:Sweep",
                                "RangeType:="           , "LinearStep",
                                "RangeStart:="          , "1GHz",
                                "RangeEnd:="            , "10GHz",
                                "RangeStep:="           , "1GHz"
                        ]
                ],
                "ComputeFarFields:="    , True
                "UseSBREnhancedRadiatedPowerCalculation:=", True,
                "IsGOBlockageEnabled:=" , False,
                "GOBlockageSurfaceSelfBlock:=", False
        ])
```

Edit and RF Discharge Setup for HFSS

```
import ScriptEnv
```

```
ScriptEnv.Initialize("Ansoft.ElectronicsDesktop")
oDesktop.RestoreWindow()
oProject = oDesktop.SetActiveProject("coaxbend_discharge_r212")
oDesign = oProject.SetActiveDesign("HFSSDesign60degBendTeflon")
oModule = oDesign.GetModule("AnalysisSetup")
oModule.EditSetup("RFDischarge1",
  [
    "NAME:RFDischarge1",
    "Enabled:="              , True,
    [
      "NAME:MeshLink",
      "ImportMesh:="           , True,
      "Project:="              , "This Project*",
      "Product:="              , "HFSS",
      "Design:="               , "This Design*",
      "Soln:="                 , "Setup1 : Sweep",
      [
        "NAME:Params",
        "bend_angle:="           , "bend_angle"
      ],
      "ForceSourceToSolve:="   , True,
  "PreservePartnerSoln:=" , False,
  "PathRelativeTo:="      , "SourceProduct",
  "ApplyMeshOp:="         , True
],
[
  "NAME:Excitations",
  [
    "NAME:1:1",
    "Magnitude:="          , "1",
    "Phase:="              , "0deg"
  ],
  [
    "NAME:2:1",
    "Magnitude:="          , "0",
    "Phase:="              , "0deg"
  ]
],
[
  "NAME:Frequencies",
  "10GHz"
```

```
    ],
    "Minimum Power:="       , "0.01",
    "Maximum Power:="       , "1000000",
    "Minimum Pressure:="    , "100pascal",
    "Maximum Pressure:="    , "101325pascal",
    "Postproc Sampling:="   , 500,
    "Temperature:="         , "0cel",
    "BuiltInGas:="          , "Helium"
  ])
```

EnableSetup

Enables and disables a defined optimetrics analysis setup.

<table>
<tr><td><b>UI Access</b></td><td>Right-click on a setup in the project tree, select <b>Enable Setup</b> or <b>Disable Setup</b></td></tr>
<tr><td><b>Parameters</b></td><td><table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;SetupName&gt;</i></td><td>String</td><td>Name of specified setup.</td></tr>
<tr><td><i>&lt;Enable&gt;</i></td><td>Boolean</td><td>Determines whether enable or disable a setup.
• <b>True</b> - enable setup.
• <b>False</b> - disable setup.</td></tr>
</table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	EnableSetup(<*SetupName*>, <*Enable*>)
Python Example	`oModule.EnableSetup("OptimizationSetup1", True)`

ExportDXConfigFile

Create an xml file with the setup information for Design Xplorer

UI Access	Right click on the Design Xplorer setup in the project tree and choose **Export External Connector Addin Configuration…**
Parameters	
Return Value	None

Name	Type	Description
<SetupName>	String	Must be one of existing DesignExplorer setup names
<FileName>	String	Must be a valid file path and name

Python Syntax	ExportDXConfigFile (<SetupName>, <FileName>)

Python Example	`oModule.ExportDXConfigFile ("DesignXplorerSetup1",`
	`"c:/exportdir/DXSetup1.xml")`

ExportOptimetricsProfile

Export Optimetrics profile data

<table>
<tr><td><b>UI Access</b></td><td>Right click on the Optimetrics setup in the project tree and choose <b>View Analysis Result…</b> On the <b>Post Analysis Display</b> dialog box, click the <b>Profile</b> tab and click on the <b>Export</b> button.</td></tr>
<tr><td><b>Parameters</b></td><td><table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td>&lt;SetupName&gt;</td><td>String</td><td>Must be one of the existing Parametric, Optimization, Sensitivity, Statistical or DesignXplorer setup names</td></tr>
<tr><td>&lt;FileName&gt;</td><td>String</td><td>Must be a valid file path and name with extension of csv, tab, dat, or txt</td></tr>
<tr><td>[profileNum]</td><td>String</td><td>Must be a numeric string. Optional: defaulted to last profile number. It should be a zero indexed profile number.</td></tr>
</table></td></tr>
<tr><td><b>Return Value</b></td><td>None</td></tr>
</table>

<table>
<tr><td><b>Python Syntax</b></td><td>ExportOptimetricsProfile (<SetupName>, <FileName>, [profileNum]</td></tr>
<tr><td><b>Python Example</b></td><td><pre>oModule.ExportOptimetricsProfile
                                                    ("StatisticalSetup1", "c:/exportdir/test.csv")</pre></td></tr>
</table>

ExportOptimetricsResult

Export an existing Optimization, Sensitivity, Statistical or DesignXplorer result. (Does not export Parametric results.)

<table>
<tr><td><b>UI Access</b></td><td>Right click on the desired Optimetrics setup in the project tree and choose <b>View Analysis Result…</b> On the <b>Post Analysis Display</b> dialog box, click the <b>Result</b> tab, then select <b>Table</b> view, and click on the <b>Export</b> button</td></tr>
<tr><td><b>Parameters</b></td><td><table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td>&lt;SetupName&gt;</td><td>String</td><td>Must be one of the existing Optimization, Sensitivity, Statistical, or DesignXplorer setup names</td></tr>
<tr><td>&lt;FileName&gt;</td><td>String</td><td>Must be a valid file path and name with extension of csv, tab, dat, or txt..</td></tr>
<tr><td>[useFullOutputName]</td><td>Boolean</td><td>Optional: defaulted to false. If set to true values will be printed with units.</td></tr>
</table></td></tr>
</table>

<table>
<tr><td></td><td><table><tr><td></td><td></td><td>This parameter is ignored for Optimization and Statistical results.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>None</td></tr>
</table>

Python Syntax	ExportOptimetricsResult (<SetupName>, <FileName>, [useFullOutputName])

Python Example	`oModule.ExportOptimetricsResult (`
	`"StatisticalSetup1", "c:/exportdir/test.csv",false)`

ExportParametricResults

Export existing Parametric results.

<table>
<tr><td><b>UI Access</b></td><td>Right click on the desired Parametric setup in the project tree and choose <b>View Analysis Result…</b> On the <b>Post Analysis Display</b> dialog box, click the <b>Result</b> tab, then select <b>Table</b> view, and click on the <b>Export</b> button.</td></tr>
<tr><td><b>Parameters</b></td><td><table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td>&lt;SetupName&gt;</td><td>String</td><td>Must be one of the existing Parametric setup names</td></tr>
<tr><td>&lt;FileName&gt;</td><td>String</td><td>Must be a valid file path and name with extension of csv, tab, dat or txt</td></tr>
<tr><td>&lt;bOutputUnits&gt;</td><td>Boolean</td><td>If set to true, values will be printed with units</td></tr>
</table></td></tr>
<tr><td><b>Return Value</b></td><td>None</td></tr>
</table>

Python Syntax	ExportParametricResults (<SetupName>, <FileName>, <bOutputUnits>)
Python Example	`oModule.ExportParametricResults (`

	`"ParametricSetup1", "c:/exportdir/test.csv", False)`

ExportParametricSetupTable

Exports the parametric setup table as a CSV file.

<table>
<tr><td><b>UI Access</b></td><td>Double-click parametric setup. Select <b>Table</b> tab. Click <b>Export</b>.</td></tr>
<tr><td><b>Parameters</b></td><td><table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><SetupName></td><td>String</td><td>Name of the setup.</td></tr>
<tr><td><filePath></td><td>String</td><td>Full path for file export.</td></tr>
</table></td></tr>
<tr><td><b>Return Value</b></td><td>None</td></tr>
</table>

Python Syntax	ExportParametricSetupTable (<SetupName>, <filePath>)
Python Example	`oModule.ExportParametricSetupTable('ParametricSetup1', 'E:/Files/ParametricSetup1_Table.csv')`

ExportRespSurfaceMinMaxTable

Exports min-max table from a response surface to a file

UI Access	Click on **Export...** From the **Response Surface** tab of the Design of Experiments **Post Analysis Display** dialog.

<table>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;DOEName&gt;</i></td><td>String</td><td>Name of the Design of Experiments (DOE) setup.</td></tr>
<tr><td><i>&lt;FileName&gt;</i></td><td>String</td><td>Output file name with path.</td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	ExportRespSurfaceMinMaxTable(<*DOEName*>, <*FileName*>)

Python Example	`oModule.ExportRespSurfaceMinMaxTable("DesignOfExperimentsSetup1",`
	`"C:/temp/DesignOfExperimentsSetup1_Min-Max_Search.csv")`

ExportRespSurfaceRefinePoints

Exports refinement points table to a file

<table>
<tr><td><b>UI Access</b></td><td>From the <b>Response Surface</b> tab of the Design of Experiments <b>Post Analysis Display</b> dialog box, select <b>Refinement Points</b> option under <b>View</b>, Click on <b>Export...</b>.</td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;DOEName&gt;</i></td><td>String</td><td>Name of the Design of Experiments (DOE) setup.</td></tr>
<tr><td><i>&lt;FileName&gt;</i></td><td>String</td><td>Output file name with path.</td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	ExportRespSurfaceRefinePoints(<*DOEName*>, <*FileName*>)

Python Example	`oModule.ExportRespSurfaceRefinePoints("DesignOfExperimentsSetup1", "C:/temp/DesignOfExperimentsSetup1_Refine_Points.csv")`

ExportRespSurfaceResponsePoints

Exports response points table to a file

<table>
<tr><td><b>UI Access</b></td><td>From the <b>Response Surface</b> tab of the Design of Experiments <b>Post Analysis Display</b> dialog box, select <b>Response Points</b> option under <b>View</b>, Click on <b>Export...</b>.</td></tr>
<tr><td><b>Parameters</b></td><td><table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;DOEName&gt;</i></td><td>String</td><td>Name of the Design of Experiments (DOE) setup.</td></tr>
<tr><td><i>&lt;FileName&gt;</i></td><td>String</td><td>Output file name with path.</td></tr>
</table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	ExportRespSurfaceResponsePoints (<*DOEName*>, <*FileName*>)
Python Example	`oModule.ExportRespSurfaceResponsePoints("DesignOfExperimentsSetup1", "C:/temp/DesignOfExperimentsSetup1_Response_Points.csv")`

ExportRespSurfaceVerificationPoints

Exports verification points table to a file

UI Access	From the **Response Surface** tab of the Design of Experiments **Post Analysis Display** dialog box, select **Verification Points** option under **View**, Click on **Export...**.

Parameters	Name	Type	Description
	<DOEName>	String	Name of the Design of Experiments (DOE) setup.
	<FileName>	String	Output file name with path.
Return Value	None.		

Python Syntax	ExportRespSurfaceVerificationPoints (<*DOEName*>, <*FileName*>)

Python Example	`oModule.ExportRespSurfaceVerificationPoints("DesignOfExperimentsSetup1",`
	`"C:/temp/DesignOfExperimentsSetup1_Veri_Points.csv")`

GenerateVariationData [Parametric]

Generate variation data before parametric solve for CAD integrated project

Command: Right click on the parametric setup in the project tree and choose "Generate Variation Data"

Syntax: GenerateVariationData <SetupName>

Return Value: None

`Parameters: <SetupName>`

Name of the setup.

GetChildNames [Optimetrics]

If used without a specific optimization setup name, gets a list of all setups for all types. If a with a specific setup name, returns names for that optimization setup.

<table>
<tr><td><b>UI Access</b></td><td>NA</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>typeName</td><td>text string</td><td>Optional, default to get all types of setup names. Or one of type name return in GetChildTypes(). Also, the type name can be used without the prefix “Opti”.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>Array of setup names.</td></tr>
</table>

<table>
<tr><td><b>Python Syntax</b></td><td>GetChildNames()</td></tr>
<tr><td><b>Python Example</b></td><td><code>oOptimModule = oDesign.GetChildObject(“Optimetrics”)</code>
<code>arrAllSetup = oOptimModule.GetChildNames()</code>
<code>arrParmSetup = oOptimModule.GetChildNames(“'OptiParametric'”)</code>
<code>arrOptimizeSetup = oOptimModule.GetChildNames(“'Optimization'”)</code></td></tr>
</table>

GetChildObject [Optimetrics]

Gets a Setup Object of the Optimetrics module

<table>
<tr><td><b>UI Access</b></td><td>NA</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>Setup Name</td><td>text string</td><td>A optimetrics setup name, names returned by the GetChildNames().</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>A script object for the setup See discussion of Optimetrics Setup Objects in Object Script Property Function Summary.</td></tr>
</table>

Python Syntax	GetChildObject()

Python Example	`oParamSetup = oOptModule.GetChildObject('ParametricSetup1')`
	`oOptSetup = oOptModule.GetChildObject('OptimizationSetup1')`

GetChildTypes [Optimetrics]

Use: Gets child types of queried Optimetrics module.

Syntax: GetChildTypes()

Return Value: Array of text string, it can be an empty array if there is no setup is defined. There are six types of setup, they are ['OptiParametric', 'OptiOptimization', 'OptiSensitivity', 'OptiStatistical', 'OptiDesignExplorer', 'OptiDXDOE'].

Python Syntax	GetChildTypes ()

Python Example	`oOptimModule = oDesign.GetChildObject(“Optimetrics”)`
	`arrSetupTypes = oOptimModule.GetChildTypes()`

GetName

Returns the design name of the active design, in that order separated by a semicolon.

UI Access	N/A
Parameters	None.
Return Value	String indicating the name of the active design.

Python Syntax	GetName()
Python Example	design_name = oDesign.GetName()

GetObjPath [Design]

Obtains the path to the design.

UI Access	N/A
Parameters	None.
Return Value	String containing the path to the design.

Python Syntax	GetObjPath()
Python Example	`oDesign.GetObjPath()`

GetOptimetricResult

Returns an Optimetric calculation. The specific calculation is determined by the setup.

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;SetupName&gt;</i></td><td>String</td><td>Optimetrics setup name.</td></tr>
<tr><td><i>&lt;vars&gt;</i></td><td>Array</td><td>Array containing string variable names.
Use the <b>Sweep Definitions</b> tab in the UI or the <SweepDefs> parameter in the InsertSetup script to determine appropriate inputs.</td></tr>
</table>
</td></tr>
</table>

	<values>	Array	*Optional*. Array containing string values. When multiple variables and values are provided, the order must be the same in both the <vars> and <values> arrays. The first variable is paired with the first value, the second variable is paired with the second value, and so on.
Return Value	Calculation result. If the setup contains more than one calculation, the output will be an array of values.		

Python Syntax	GetOptimetricResult(<*SetupName*>, <*vars*>, <*values*>)
Python Example	`oModule.GetOptimetricResult('ParametricSetup1',['AR','Re'],['4.64','6e+04'])`

GetOptimetricsResult

Get an existing Optimization, Sensitivity, Statistical, Parametric or DesignXplorer result.

UI Access	Right click on the desired Optimetrics setup in the project tree and choose **View Analysis Result…** On the **Post Analysis Display** dialog box, click the **Result** tab, then select **Table** view, and click on the **Export** button		
Parameters	Name	Type	Description
	<SetupName>	String	Must be one of the existing Optimization, Sensitivity, Statistical, or DesignXplorer setup names
	<FileName>	String	Must be a valid file path and name with extension of csv, tab, dat, or txt..
	[useFullOutputName]	Boolean	Optional: defaulted to false. If set to true values will be printed with units. This parameter is ignored for Optimization and Statistical results.
Return Value	None		

Python Syntax	GetOptimetricsResult (<SetupName>, <FileName>, [useFullOutputName])

Python Example	`oModule.GetOptimetricsResult (`
	`"StatisticalSetup1", "c:/exportdir/test.csv",false)`

GetPropNames [Optimetrics]

Use: Always returns the empty set for Optimetrics objects since they do not have properties.

Syntax: GetPropNames(bIncludeReadOnly)

Return Value: Returns empty set.

Parameters: `bIncludeReadOnly—optional, default to True.`

Python Syntax	GetPropNames ()

Python Example	`oOptModule.GetPropNames()`
	`oOptModule.GetPropNames(True)`
	`oOptModule.GetPropNames(False)`

GetPropValue [Optimetrics]

Returns the property value for a setup property.

<table>
<tr><th>UI Access</th><th>NA</th></tr>
<tr><td>Parameters</td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>property-path</td><td></td><td>a child object's property path. See property path discussion here.</td></tr></table></td></tr>
<tr><td>Return Value</td><td>Returns the value of an setup property.</td></tr>
</table>

Python Syntax	GetPropValue(propPath)

Python Example	`oOptModule.GetPropValue("OptimizationSetup1\Optimizer") //get the optimizer name for OptimizationSetup1`
	`oOptModule.GetPropValue("OptimizationSetup1\Optimizer\Choices") //Get the menu property’s menu items. In this case all Optimizer names.`

GetSetupNames [Optimetrics]

Gets a list of Optimetrics setup names

<table>
<tr><td><b>UI Access</b></td><td>NA</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>None</td><td></td><td></td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>IAnsoftCollectionObj – a collection of Optimetrics setup names</td></tr>
</table>

Python Syntax	GetSetupNames()

Python Example	`oModule = oDesign.GetModule("Optimetrics")`
	`setupNames = oModule.GetSetupNames()`

GetSetupNamesByType [Optimetrics]

Gets a list of Optimetrics setup names by type.

<table>
<tr><td><b>UI Access</b></td><td>NA</td></tr>
<tr><td><b>Parameters</b></td><td><table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td>&lt;Optimetrics type&gt;</td><td>String</td><td>Examples: parametric, optimization, statistical, sensitivity</td></tr>
</table></td></tr>
<tr><td><b>Return Value</b></td><td>Array of Optimetrics setup names of the given type.</td></tr>
</table>

<table>
<tr><td><b>Python Syntax</b></td><td>GetSetupNamesByType (<Optimetrics type&gt;)</td></tr>
<tr><td><b>Python Example</b></td><td><pre>for name in oModule.GetSetupNamesByType("optimization")

       AddInfoMessage(str(name))</pre></td></tr>
</table>

ImportSetup

Import an Optimetric setup from a file.

<table>
<tr><td><b>UI Access</b></td><td>NA</td></tr>
<tr><td><b>Parameters</b></td><td><table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td>&lt;SetupTypeName&gt;</td><td>String</td><td>Must be one of "OptiParametric" , "OptiOptimization", "OptiSensitivity", "OptiStatistical", or "OptiDesignExplorer".</td></tr>
<tr><td>&lt;SetupInfo&gt;</td><td>Array</td><td>Array("NAME:<SetupName>", "FilePath")
<SetupName&gt;</td></tr>
</table></td></tr>
</table>

<table>
<tr><td></td><td><table><tr><td></td><td></td><td>Type: <string>
Name of the setup.
<FilePath>
Type : <string: file path>
Must be a valid file path and name.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>None</td></tr>
</table>

Python Syntax	ImportSetup (<SetupTypeName>, <SetupInfo>)
Python Example	oModule.ImportSetup ("OptiStatistical", ["NAME:StatisticalSetup1", "c:/importdir/mySetupInfoFile"])

PasteSetup [Optimetrics]

Pastes the specified Optimetrics setup.

<table>
<tr><td><b>UI Access</b></td><td>NA</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><SetupName></td><td>String</td><td>Name of the Setup</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>None</td></tr>
</table>

Python Syntax	PasteSetup (<SetupName>)
Python Example	`oModule.PasteSetup ("OptimizationSetup1")`

RenameSetup [Optimetrics]

Renames the specified Optimetrics setup.

UI Access	**Right-click the setup in the project tree, and then click Rename** on the shortcut menu.
Parameters	Name | Type | Description <OldName> | String | The name that needs to be replaced <NewName> | String | Replacement name
Return Value	None

Python Syntax	RenameSetup (<OldName> <NewName>)
Python Example	`oModule.RenameSetup ("OptimizationSetup1" "MyOptimization")`

SetPropValue [Optimetrics]

Sets the property value for the active Optimetrics setup.

<table>
<tr><td><b>UI Access</b></td><td>Set Property value on Optimetrics objects</td></tr>
<tr><td><b>Parameters</b></td><td><table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td>Property path</td><td>text string</td><td>Setup property path. See discussion of Property Path</td></tr>
<tr><td>new Value</td><td>Text String, Number, or Boolean</td><td>New value data type is depending on the property type,</td></tr>
</table></td></tr>
<tr><td><b>Return Value</b></td><td>True if the property is found and the new value is valid. Otherwise return False.</td></tr>
</table>

<table>
<tr><td><b>Python Syntax</b></td><td>SetPropValue(propPath, newValue)</td></tr>
<tr><td><b>Python Example</b></td><td><code>oOptModule.SetPropValue("ParametricSetup1\Enabled", False) //disable ParametricSetup1</code>
<code>oOptModule.SetPropValue("OptimizationSetup1/Optimizer","Quasi Newton")</code></td></tr>
</table>

SolveAllSetup

Solves all Optimetrics setups

<table>
<tr><td><b>UI Access</b></td><td>Right-click on <b>Optimetrics</b> in Project Manager and select <b>Analyze&gt;All</b> from context menu</td></tr>
<tr><td><b>Parameters</b></td><td><table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td>isBlocking</td><td>Boolean</td><td>An optional arg that defaults to <code>true</code>. If it's <code>false</code>, the command immediately returns while the analysis or analyses run in the background. The status of the analyses can be checked with the <code>AreThereSimulationsRunning</code> command.</td></tr>
</table></td></tr>
</table>

Return Value	None

Python Syntax	SolveAllSetup()
Python Example	`oModule.SolveAllSetup()`

SolveSetup [Optimetrics]

Solves the specified Optimetrics setup.

<table>
<tr><td><b>UI Access</b></td><td>Right-click the setup in the project tree, and then click <b>Analyze</b> on the shortcut menu.</td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td>&lt;<i>SetupName</i>&gt;</td><td>String</td><td>Name of the setup to be solved</td></tr>
<tr><td>isBlocking</td><td>Boolean</td><td>An optional arg that defaults to <code>true</code>. If it's <code>false</code>, the command immediately returns while the analysis or analyses run in the background. The status of the analyses can be checked with the <code>AreThereSimulationsRunning</code> command.</td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>None</td></tr>
</table>

Python Syntax	SolveSetup (<*SetupName*>)
Python Example	`oModule.SolveSetup ("OptimizationSetup1")`

General Commands Recognized by the Optimetrics Module

Following are general script commands recognized by the **Optimetrics** module:

CopySetup

DistrbutedAnalyzeSetup

EditSetup

ExportDXConfigFile

ExportOptimetricsProfile

ExportOptimetricsResult

ExportParametricResults

GetOptimetricResult

GetPropNames [Optimetrics]

GetPropValue [Optimetrics]

GetSetupNames [Optimetrics]

GetSetupNamesByType [Optimetrics]

ImportSetup

PasteSetup [Optimetrics]

RenameSetup [Optimetrics]

SetPropValue [Optimetrics]

SolveSetup [Optimetrics]

SolveAllSetup

CopySetup

Copy the specified Optimetrics setup.

<table>
<tr><td><b>UI Access</b></td><td>NA</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;<i>SetupName</i>&gt;</td><td>String</td><td>Name of the setup.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	CopySetup (<*SetupName*>)
Python Example	`oModule.CopySetup ("OptimizationSetup1")`

DeleteSetups [Optimetrics]

Deletes the specified Optimetrics setups.

<table>
<tr><td><b>UI Access</b></td><td>Right-click the setup in the project tree, and then click <b>Delete</b> on the shortcut menu</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;NameArray&gt;</td><td>Array of Strings</td><td>An Array of Setup Names</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>None</td></tr>
</table>

Python Syntax	DeleteSetups (<NameArray>)
Python Example	`oModule.DeleteSetups (["OptimizationSetup1"])`

DistributedAnalyzeSetup

Distributes all variable value instances within a parametric sweep to different machines already specified from within the user interface

<table>
<tr><td><b>UI Access</b></td><td>Right-click the parametric setup name in the project tree and select Distribute Analysis.</td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td>&lt;ParametricSetupName&gt;</td><td>String</td><td>Name of the Setup</td></tr>
<tr><td>isBlocking</td><td>Boolean</td><td>An optional arg that defaults to <code>true</code>. If it's <code>false</code>, the command immediately returns while the analysis or analyses run in the background. The status of the analyses can be checked with the <code>AreThereSimulationsRunning</code> command.</td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>None</td></tr>
</table>

Python Syntax	DistributedAnalyzeSetup (<ParametricSetupName>)
Python Example	`oModule.DistributedAnalyzeSetup("ParametricSetup1")`

EditSetup

Modifies an existing solution setup.

UI Access	Double-click a solution setup in the project tree to modify its settings.

<table>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td>&lt;<i>SetupName</i>&gt;</td><td>String</td><td>Name of the solve setup being edited.</td></tr>
<tr><td>&lt;<i>Attributes</i>&gt;</td><td>Array</td><td>Structured array.
<code>Array("NAME:<NewSetupName>", <NamedParameters&gt;)</code>
<code>See the InsertSetup command for details and examples.</code></td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	EditSetup (<*SetupName*>, <*Attributes*>)

Python Example

```
oModule.EditSetup("Setup1",
[
"NAME:NewSetup",
"AdaptiveFreq:=", "1GHz",
"EnableDistribProbTypeOption:=", false,
"SaveFields:=", "true",
"Enabled:=", true,
["NAME:Cap",
        "MaxPass:=", 10,
        "MinPass:=", 1,
        "MinConvPass:=", 2,
        "PerError:=", 1,
        "PerRefine:=", 30,
        "AutoIncreaseSolutionOrder:=", false,
        "SolutionOrder:=", "Normal"],
["NAME:DC",
        "Residual:=", 1E-005,
        "SolveResOnly:=", false,
        ["NAME:Cond",
                "MaxPass:=", 10,
                "MinPass:=", 1,
                "MinConvPass:=", 1,
                "PerError:=", 1,
                "PerRefine:=", 30),
        ["NAME:Mult",
                "MaxPass:=", 1,
                "MinPass:=", 1,
                "MinConvPass:=", 1,
                "PerError:=", 1,
                "PerRefine:=", 30]],
["NAME:AC",
        "MaxPass:=", 10,
        "MinPass:=", 1,
        "MinConvPass:=", 2,
        "PerError:=", 1,
        "PerRefine:=", 30]]
)
oModule.EditSetup("HfssDrivenAuto",
["NAME:Setup1",
  "IsEnabled:=", True,
  "AutoSolverSetting:=", "Balanced",
  ["NAME:Sweeps",
          ["NAME:Sweep",
            "RangeType:=", "LinearStep",
            "RangeStart:=", "1GHz",
            "RangeEnd:=", "10GHz",
            "RangeStep:=", "1GHz"
            ]
  ],
  "SaveRadFieldsOnly:=", False,
  "SaveAnyFields:=", True,
  "Type:=", "Discrete"
  ])
```

```
oModule.EditSetup("AC Magnetic",
[
  "NAME:AC Magnetic",
  "Enabled:="             , True,
  [
    "NAME:MeshLink",
    "ImportMesh:="          , False
  ],
  "MaximumPasses:="       , 4,
  "MinimumPasses:="       , 2,
  "MinimumConvergedPasses:=", 1,
  "PercentRefinement:="   , 30,
  "SolveFieldOnly:="      , False,
  "PercentError:="        , 0.1,
  "SolveMatrixAtLast:="   , True,
  "UseNonLinearIterNum:=" , False,
  [
    "NAME:ExpressionCache",
    [
      "NAME:CacheItem",
       "Title:="               , "eddy_loss1",
       "Expression:="          , "eddy_loss",
       "Intrinsics:="          , "Phase=\'0deg\'",
       "ReportType:="          , "Fields",
       [
         "NAME:ExpressionContext"
       ]
     ]
  ],
  "UseCacheFor:="         , ["Pass"],
  "UseIterativeSolver:="  , False,
  "RelativeResidual:="    , 0.0001,
  "NonLinearResidual:="   , 0.0001,
  "SmoothBHCurve:="       , False,
  "Frequency:="           , "200Hz",
  "HasSweepSetup:="       , False,
  "UseHighOrderShapeFunc:=", False,
  "UseMuLink:="           , False,
  "LossAdaptiveCtrl:="    , "0.3"
])
oModule.EditSetup("HfssDriven",
["NAME:Setup3",
        "AdaptMultipleFreqs:=", False,
        "Frequency:=", "5GHz",
        "MaxDeltaS:=", 0.02,
        "PortsOnly:=", False,
        "UseMatrixConv:=", False,
        "MaximumPasses:=", 6,
        "MinimumPasses:=", 1,
        "MinimumConvergedPasses:=", 1,
        "PercentRefinement:=", 30,
        "IsEnabled:=", True,
        "BasisOrder:=", 1,
        "DoLambdaRefine:=", True,
        "DoMaterialLambda:=", True,
        "SetLambdaTarget:=", False,
        "Target:=", 0.3333,
        "UseMaxTetIncrease:=", False,
        "PortAccuracy:=", 2,
        "UseABCOnPort:=", False,
        "SetPortMinMaxTri:=", False,
        "UseDomains:=", True,
        "UseIterativeSolver:=", False,
        "IterativeResidual:=", 1E-06,
        "DDMSolverResidual:=", 0.0001,
        "EnhancedLowFreqAccuracy:=", True,
        "SaveRadFieldsOnly:=", False,
        "SaveAnyFields:=", True,
        "IESolverType:=", "Auto",
        "LambdaTargetForIESolver:=", 0.15,
        "UseDefaultLambdaTgtForIESolver:=", True,
        "SkipIERegionSolveDuringAdaptivePasses:=", True
        "RayDensityPerWavelength:=", 4,
        "MaxNumberOfBounces:="  , 5,
        "InfiniteSphereSetup:=" , "Infinite Sphere1",
        "SkipSBRSolveDuringAdaptivePasses:=", True,
        "PTDUTDSimulationSettings:=", "PTD Correction + UTD Rays",
        "PTDEdgeDensity:="      , 20
])
```

Edit an SBR+ Setup with Fast Frequency Looping

```
oModule.EditSetup("HfssDriven",
        [
                "NAME:Setup1",
                "IsEnabled:="           , True,
                [
                        "NAME:MeshLink",
                        "ImportMesh:="          , False
                ],
                "IsSbrRangeDoppler:="   , False,
                "RayDensityPerWavelength:=", 4,
                "MaxNumberOfBounces:="  , 5,
                "IsMonostaticRCS:="     , True,
                "EnableCWRays:="        , False,
                "RadiationSetup:="      , "",
                "PTDUTDSimulationSettings:=", "None",
                "FastFrequencyLooping:=", True,
                [
                        "NAME:Sweeps",
                        [
                                "NAME:Sweep",
                                "RangeType:="           , "LinearStep",
                                "RangeStart:="          , "1GHz",
                                "RangeEnd:="            , "10GHz",
                                "RangeStep:="           , "1GHz"
                        ]
                ],
                "ComputeFarFields:="    , True
                "UseSBREnhancedRadiatedPowerCalculation:=", True,
                "IsGOBlockageEnabled:=" , False,
                "GOBlockageSurfaceSelfBlock:=", False
        ])
```

Edit and RF Discharge Setup for HFSS

```
import ScriptEnv
ScriptEnv.Initialize("Ansoft.ElectronicsDesktop")
oDesktop.RestoreWindow()
oProject = oDesktop.SetActiveProject("coaxbend_discharge_r212")
oDesign = oProject.SetActiveDesign("HFSSDesign60degBendTeflon")
oModule = oDesign.GetModule("AnalysisSetup")
oModule.EditSetup("RFDischarge1",
  [
    "NAME:RFDischarge1",
    "Enabled:="             , True,
    [
      "NAME:MeshLink",
  "ImportMesh:="          , True,
  "Project:="             , "This Project*",
  "Product:="             , "HFSS",
  "Design:="              , "This Design*",
  "Soln:="                , "Setup1 : Sweep",
  [
    "NAME:Params",
    "bend_angle:="          , "bend_angle"
  ],
  "ForceSourceToSolve:="  , True,
  "PreservePartnerSoln:=" , False,
  "PathRelativeTo:="      , "SourceProduct",
  "ApplyMeshOp:="         , True
],
[
  "NAME:Excitations",
  [
    "NAME:1:1",
    "Magnitude:="           , "1",
    "Phase:="               , "0deg"
    ],
    [
      "NAME:2:1",
      "Magnitude:="           , "0",
      "Phase:="               , "0deg"
    ]
  ],
  [
    "NAME:Frequencies",
    "10GHz"
  ],
  "Minimum Power:="       , "0.01",
  "Maximum Power:="       , "1000000",
  "Minimum Pressure:="    , "100pascal",
  "Maximum Pressure:="    , "101325pascal",
  "Postproc Sampling:="   , 500,
  "Temperature:="         , "0cel",
  "BuiltInGas:="          , "Helium"
])
```

EnableSetup

Enables and disables a defined optimetrics analysis setup.

<table>
<tr><td><b>UI Access</b></td><td>Right-click on a setup in the project tree, select <b>Enable Setup</b> or <b>Disable Setup</b></td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;SetupName&gt;</i></td><td>String</td><td>Name of specified setup.</td></tr>
<tr><td><i>&lt;Enable&gt;</i></td><td>Boolean</td><td>Determines whether enable or disable a setup.
• <b>True -</b> enable setup.
• <b>False -</b> disable setup.</td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	EnableSetup(<*SetupName*>, <*Enable*>)
Python Example	`oModule.EnableSetup("OptimizationSetup1", True)`

ExportDXConfigFile

Create an xml file with the setup information for Design Xplorer

<table>
<tr><td><b>UI Access</b></td><td>Right click on the Design Xplorer setup in the project tree and choose <b>Export External Connector Addin Configuration…</b></td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td>&lt;SetupName&gt;</td><td>String</td><td>Must be one of existing DesignExplorer setup names</td></tr>
<tr><td>&lt;FileName&gt;</td><td>String</td><td>Must be a valid file path and name</td></tr>
</table>
</td></tr>
</table>

Return Value	None

Python Syntax	ExportDXConfigFile (<SetupName>, <FileName>)

Python Example	`oModule.ExportDXConfigFile ("DesignXplorerSetup1",`
	`"c:/exportdir/DXSetup1.xml")`

ExportOptimetricsProfile

Export Optimetrics profile data

<table>
<tr><td><b>UI Access</b></td><td>Right click on the Optimetrics setup in the project tree and choose <b>View Analysis Result…</b> On the <b>Post Analysis Display</b> dialog box, click the <b>Profile</b> tab and click on the <b>Export</b> button.</td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td>&lt;SetupName&gt;</td><td>String</td><td>Must be one of the existing Parametric, Optimization, Sensitivity, Statistical or DesignXplorer setup names</td></tr>
<tr><td>&lt;FileName&gt;</td><td>String</td><td>Must be a valid file path and name with extension of csv, tab, dat, or txt</td></tr>
<tr><td>[profileNum]</td><td>String</td><td>Must be a numeric string. Optional: defaulted to last profile number. It should be a zero indexed profile number.</td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>None</td></tr>
</table>

Python Syntax	ExportOptimetricsProfile (<SetupName>, <FileName>, [profileNum]

Python Example	`oModule.ExportOptimetricsProfile`
	`("StatisticalSetup1", "c:/exportdir/test.csv")`

ExportOptimetricsResult

Export an existing Optimization, Sensitivity, Statistical or DesignXplorer result. (Does not export Parametric results.)

<table>
<tr><td><b>UI Access</b></td><td>Right click on the desired Optimetrics setup in the project tree and choose <b>View Analysis Result…</b> On the <b>Post Analysis Display</b> dialog box, click the <b>Result</b> tab, then select <b>Table</b> view, and click on the <b>Export</b> button</td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td>&lt;SetupName&gt;</td><td>String</td><td>Must be one of the existing Optimization, Sensitivity, Statistical, or DesignXplorer setup names</td></tr>
<tr><td>&lt;FileName&gt;</td><td>String</td><td>Must be a valid file path and name with extension of csv, tab, dat, or txt..</td></tr>
<tr><td>[useFullOutputName]</td><td>Boolean</td><td>Optional: defaulted to false. If set to true values will be printed with units. This parameter is ignored for Optimization and Statistical results.</td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>None</td></tr>
</table>

Python Syntax	ExportOptimetricsResult (<SetupName>, <FileName>, [useFullOutputName])

Python Example	`oModule.ExportOptimetricsResult (`
	`"StatisticalSetup1", "c:/exportdir/test.csv",false)`

ExportParametricResults

Export existing Parametric results.

<table>
<tr><td><b>UI Access</b></td><td>Right click on the desired Parametric setup in the project tree and choose <b>View Analysis Result…</b> On the <b>Post Analysis Display</b> dialog box, click the <b>Result</b> tab, then select <b>Table</b> view, and click on the <b>Export</b> button.</td></tr>
<tr><td><b>Parameters</b></td><td><table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td>&lt;SetupName&gt;</td><td>String</td><td>Must be one of the existing Parametric setup names</td></tr>
<tr><td>&lt;FileName&gt;</td><td>String</td><td>Must be a valid file path and name with extension of csv, tab, dat or txt</td></tr>
<tr><td>&lt;bOutputUnits&gt;</td><td>Boolean</td><td>If set to true, values will be printed with units</td></tr>
</table></td></tr>
<tr><td><b>Return Value</b></td><td>None</td></tr>
</table>

Python Syntax	ExportParametricResults (<SetupName>, <FileName>, <bOutputUnits>)

Python Example	`oModule.ExportParametricResults (`
	`"ParametricSetup1", "c:/exportdir/test.csv", False)`

ExportParametricSetupTable

Exports the parametric setup table as a CSV file.

<table>
<tr><td><b>UI Access</b></td><td>Double-click parametric setup. Select <b>Table</b> tab. Click <b>Export</b>.</td></tr>
<tr><td><b>Parameters</b></td><td><table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td>&lt;SetupName&gt;</td><td>String</td><td>Name of the setup.</td></tr>
<tr><td>&lt;filePath&gt;</td><td>String</td><td>Full path for file export.</td></tr>
</table></td></tr>
<tr><td><b>Return Value</b></td><td>None</td></tr>
</table>

Python	ExportParametricSetupTable (<SetupName>, <filePath>)

Syntax	
Python Example	`oModule.ExportParametricSetupTable('ParametricSetup1', 'E:/Files/ParametricSetup1_Table.csv')`

ExportRespSurfaceMinMaxTable

Exports min-max table from a response surface to a file

<table>
<tr><td><b>UI Access</b></td><td>Click on <b>Export...</b> From the <b>Response Surface</b> tab of the Design of Experiments <b>Post Analysis Display</b> dialog.</td></tr>
<tr><td><b>Parameters</b></td><td><table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;DOEName&gt;</i></td><td>String</td><td>Name of the Design of Experiments (DOE) setup.</td></tr>
<tr><td><i>&lt;FileName&gt;</i></td><td>String</td><td>Output file name with path.</td></tr>
</table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	ExportRespSurfaceMinMaxTable(<*DOEName*>, <*FileName*>)

Python Example	`oModule.ExportRespSurfaceMinMaxTable("DesignOfExperimentsSetup1",`
	`"C:/temp/DesignOfExperimentsSetup1_Min-Max_Search.csv")`

ExportRespSurfaceRefinePoints

Exports refinement points table to a file

<table>
<tr><td><b>UI Access</b></td><td>From the <b>Response Surface</b> tab of the Design of Experiments <b>Post Analysis Display</b> dialog box, select <b>Refinement Points</b> option under <b>View</b>, Click on <b>Export...</b>.</td></tr>
<tr><td><b>Parameters</b></td><td><table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;DOEName&gt;</i></td><td>String</td><td>Name of the Design of Experiments (DOE) setup.</td></tr>
<tr><td><i>&lt;FileName&gt;</i></td><td>String</td><td>Output file name with path.</td></tr>
</table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	ExportRespSurfaceRefinePoints(<*DOEName*>, <*FileName*>)

Python Example	`oModule.ExportRespSurfaceRefinePoints("DesignOfExperimentsSetup1",`
	`"C:/temp/DesignOfExperimentsSetup1_Refine_Points.csv")`

ExportRespSurfaceResponsePoints

Exports response points table to a file

<table>
<tr><td><b>UI Access</b></td><td>From the <b>Response Surface</b> tab of the Design of Experiments <b>Post Analysis Display</b> dialog box, select <b>Response Points</b> option under <b>View</b>, Click on <b>Export...</b>.</td></tr>
<tr><td><b>Parameters</b></td><td><table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;DOEName&gt;</i></td><td>String</td><td>Name of the Design of Experiments (DOE) setup.</td></tr>
<tr><td><i>&lt;FileName&gt;</i></td><td>String</td><td>Output file name with path.</td></tr>
</table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	ExportRespSurfaceResponsePoints (<*DOEName*>, <*FileName*>)

Python Example	`oModule.ExportRespSurfaceResponsePoints("DesignOfExperimentsSetup1",`
	`"C:/temp/DesignOfExperimentsSetup1_Response_Points.csv")`

ExportRespSurfaceVerificationPoints

Exports verification points table to a file

<table>
<tr><td><b>UI Access</b></td><td>From the <b>Response Surface</b> tab of the Design of Experiments <b>Post Analysis Display</b> dialog box, select <b>Verification Points</b> option under <b>View</b>, Click on <b>Export...</b>.</td></tr>
<tr><td><b>Parameters</b></td><td><table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td>&lt;<i>DOEName</i>&gt;</td><td>String</td><td>Name of the Design of Experiments (DOE) setup.</td></tr>
<tr><td>&lt;<i>FileName</i>&gt;</td><td>String</td><td>Output file name with path.</td></tr>
</table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	ExportRespSurfaceVerificationPoints (<*DOEName*>, <*FileName*>)

Python Example	`oModule.ExportRespSurfaceVerificationPoints("DesignOfExperimentsSetup1",`
	`"C:/temp/DesignOfExperimentsSetup1_Veri_Points.csv")`

ExportDOEResponseCurve

Exports response curve from a response surface to a file.

UI Access	Click on **Export...** From the **Response Surface** tab of the Design of Experiments **Post Analysis Display** dialog.
Parameters	Name | Type | Description *<DOEName>* | String | Name of the Design of Experiments (DOE) setup. *<FileName>* | String | Output file name with path.
Return Value	None

Python Syntax	*DOEName* (*<FileName>*)

Python Example	`oModule.ExportDOEResponseCurve("DesignOfExperimentsSetup1",`
	`"C:/Users/AEDT/R24.1/D/DesignOfExperimentsSetup1_Response_Points.csv")`

ExportDOEResponseCurveSlices

Exports response curve slices from a response surface to a file.

UI Access	Click on **Export...** From the **Response Surface** tab of the Design of Experiments **Post Analysis Display** dialog.
Parameters	Name | Type | Description *<DOEName>* | String | Name of the Design of Experiments (DOE) setup. *<FileName>* | String | Output file name with path.
Return Value	None

Python Syntax	*DOEName* (<*FileName*>)

Python Example	`oModule.ExportDOEResponseCurveSlices("DesignOfExperimentsSetup1",`
	` "C:/Users/AEDT/R24.1/D/DesignOfExperimentsSetup1_Response_Points.csv")`

ExportDOEResponseSurface

Exports response surface to a file.

<table>
<tr><td><b>UI Access</b></td><td>Click on <b>Export...</b> From the <b>Response Surface</b> tab of the Design of Experiments <b>Post Analysis Display</b> dialog.</td></tr>
<tr><td><b>Parameters</b></td><td><table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td>&lt;<i>DOEName</i>&gt;</td><td>String</td><td>Name of the Design of Experiments (DOE) setup.</td></tr>
<tr><td>&lt;<i>FileName</i>&gt;</td><td>String</td><td>Output file name with path.</td></tr>
</table></td></tr>
<tr><td><b>Return Value</b></td><td>None</td></tr>
</table>

Python Syntax	*DOEName* (<*FileName*>)

Python Example	`oModule.ExportDOEResponseSurface("DesignOfExperimentsSetup1",`
	` "C:/Users/AEDT/R24.1/D/DesignOfExperimentsSetup1_Response_Points.csv")`

ExportDOELocalSensitivity

Exports local sensitivity to a file.

<table>
<tr><td><b>UI Access</b></td><td>Click on <b>Export...</b> From the <b>Response Surface</b> tab of the Design of Experiments <b>Post Analysis Display</b> dialog.</td></tr>
<tr><td><b>Parameters</b></td><td><table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;DOEName&gt;</i></td><td>String</td><td>Name of the Design of Experiments (DOE) setup.</td></tr>
<tr><td><i>&lt;FileName&gt;</i></td><td>String</td><td>Output file name with path.</td></tr>
</table></td></tr>
<tr><td><b>Return Value</b></td><td>None</td></tr>
</table>

<table>
<tr><td><b>Python Syntax</b></td><td><i>DOEName</i> (<i>&lt;FileName&gt;</i>)</td></tr>
<tr><td><b>Python Example</b></td><td><code>oModule.ExportDOELocalSensitivity("DesignOfExperimentsSetup1",</code>
<code>"C:/Users/AEDT/R24.1/D/DesignOfExperimentsSetup1_Response_Points.csv")</code></td></tr>
</table>

ExportDOELocalSensitivityCurve

Exports local sensitivity curves to a file.

<table>
<tr><td><b>UI Access</b></td><td>Click on <b>Export...</b> From the <b>Response Surface</b> tab of the Design of Experiments <b>Post Analysis Display</b> dialog.</td></tr>
<tr><td><b>Parameters</b></td><td><table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;DOEName&gt;</i></td><td>String</td><td>Name of the Design of Experiments (DOE) setup.</td></tr>
<tr><td><i>&lt;FileName&gt;</i></td><td>String</td><td>Output file name with path.</td></tr>
</table></td></tr>
<tr><td><b>Return Value</b></td><td>None</td></tr>
</table>

Python Syntax	*DOEName* (<*FileName*>)

Python Example	`oModule.ExportDOELocalSensitivityCurve("DesignOfExperimentsSetup1",`
	`"C:/Users/AEDT/R24.1/D/DesignOfExperimentsSetup1_Response_Points.csv")`

GenerateVariationData [Parametric]

Generate variation data before parametric solve for CAD integrated project

Command: Right click on the parametric setup in the project tree and choose "Generate Variation Data"

Syntax: GenerateVariationData <SetupName>

Return Value: None

Parameters: <SetupName>

Name of the setup.

GetChildNames [Optimetrics]

If used without a specific optimization setup name, gets a list of all setups for all types. If a with a specific setup name, returns names for that optimization setup.

UI Access	NA
Parameters	Name | Type | Description typeName | text string | Optional, default to get all types of setup names. Or one of type name return in GetChildTypes(). Also, the type name can be used without the prefix “Opti”.
Return Value	Array of setup names.

Python Syntax	GetChildNames()

Python Example	`oOptimModule = oDesign.GetChildObject(“Optimetrics”)`
	`arrAllSetup = oOptimModule.GetChildNames()`
	`arrParmSetup = oOptimModule.GetChildNames(“'OptiParametric'”)`
	`arrOptimizeSetup = oOptimModule.GetChildNames(“'Optimization'”)`

GetChildObject [Optimetrics]

Gets a Setup Object of the Optimetrics module

<table>
<tr><th>UI Access</th><th>NA</th></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>Setup Name</td><td>text string</td><td>A optimetrics setup name, names returned by the GetChildNames().</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>A script object for the setup See discussion of Optimetrics Setup Objects in Object Script Property Function Summary.</td></tr>
</table>

Python Syntax	GetChildObject()

Python Example	`oParamSetup = oOptModule.GetChildObject('ParametricSetup1')`
	`oOptSetup = oOptModule.GetChildObject('OptimizationSetup1')`

GetChildTypes [Optimetrics]

Use: Gets child types of queried Optimetrics module.

Syntax: GetChildTypes()

Return Value: Array of text string, it can be an empty array if there is no setup is defined. There are six types of setup, they are ['OptiParametric', 'OptiOptimization', 'OptiSensitivity', 'OptiStatistical', 'OptiDesignExplorer', 'OptiDXDOE'].

Python Syntax	GetChildTypes ()

Python Example	`oOptimModule = oDesign.GetChildObject("Optimetrics")`
	`arrSetupTypes = oOptimModule.GetChildTypes()`

GetName

Returns the design name of the active design, in that order separated by a semicolon.

UI Access	N/A
Parameters	None.
Return Value	String indicating the name of the active design.

Python Syntax	GetName()
Python Example	design_name = oDesign.GetName()

GetObjPath [Design]

Obtains the path to the design.

UI Access	N/A
Parameters	None.
Return Value	String containing the path to the design.

Python Syntax	GetObjPath()
Python Example	`oDesign.GetObjPath()`

GetOptimetricResult

Returns an Optimetric calculation. The specific calculation is determined by the setup.

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i><SetupName></i></td><td>String</td><td>Optimetrics setup name.</td></tr>
<tr><td><i><vars></i></td><td>Array</td><td>Array containing string variable names.
Use the <b>Sweep Definitions</b> tab in the UI or the <SweepDefs> parameter in the InsertSetup script to determine appropriate inputs.</td></tr>
<tr><td><i><values></i></td><td>Array</td><td><i>Optional</i>. Array containing string values.
When multiple variables and values are provided, the order must be the same in both the <vars> and <values> arrays. The first variable is paired with the first value, the second variable is paired with the second value, and so on.</td></tr>
</table>
</td></tr>
</table>

Return Value	Calculation result. If the setup contains more than one calculation, the output will be an array of values.

Python Syntax	GetOptimetricResult(<*SetupName*>, <*vars*>, <*values*>)
Python Example	oModule.GetOptimetricResult('ParametricSetup1',['AR','Re'],['4.64','6e+04'])

GetPropNames [Optimetrics]

Use: Always returns the empty set for Optimetrics objects since they do not have properties.

Syntax: GetPropNames(bIncludeReadOnly)

Return Value: Returns empty set.

Parameters: bIncludeReadOnly—optional, default to True.

Python Syntax	GetPropNames ()

Python Example	oOptModule.GetPropNames()
	oOptModule.GetPropNames(True)
	oOptModule.GetPropNames(False)

GetPropValue [Optimetrics]

Returns the property value for a setup property.

UI Access	NA

<table>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>property-path</td><td></td><td>a child object's property path. See property path discussion here.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>Returns the value of an setup property.</td></tr>
</table>

Python Syntax	GetPropValue(propPath)

Python Example	`oOptModule.GetPropValue("OptimizationSetup1\Optimizer") //get the optimizer name for OptimizationSetup1`
	`oOptModule.GetPropValue("OptimizationSetup1\Optimizer\Choices") //Get the menu property’s menu items. In this case all Optimizer names.`

GetSetupNames [Optimetrics]

Gets a list of Optimetrics setup names

<table>
<tr><td><b>UI Access</b></td><td>NA</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>None</td><td></td><td></td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>IAnsoftCollectionObj – a collection of Optimetrics setup names</td></tr>
</table>

Python Syntax	GetSetupNames()

Python Example	`oModule = oDesign.GetModule("Optimetrics")`
	`setupNames = oModule.GetSetupNames()`

GetSetupNamesByType [Optimetrics]

Gets a list of Optimetrics setup names by type.

<table>
<tr><td><b>UI Access</b></td><td>NA</td></tr>
<tr><td><b>Parameters</b></td><td><table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td>&lt;Optimetrics type&gt;</td><td>String</td><td>Examples: parametric, optimization, statistical, sensitivity</td></tr>
</table></td></tr>
<tr><td><b>Return Value</b></td><td>Array of Optimetrics setup names of the given type.</td></tr>
</table>

Python Syntax	GetSetupNamesByType (<Optimetrics type>)

Python Example

```
for name in oModule.GetSetupNamesByType("optimization")

       AddInfoMessage(str(name))
```

ImportSetup

Import an Optimetric setup from a file.

<table>
<tr><td><b>UI Access</b></td><td>NA</td></tr>
<tr><td><b>Parameters</b></td><td><table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td>&lt;SetupTypeName&gt;</td><td>String</td><td>Must be one of "OptiParametric" , "OptiOptimization", "OptiSensitivity", "OptiStatistical", or "OptiDesignExplorer".</td></tr>
<tr><td>&lt;SetupInfo&gt;</td><td>Array</td><td>Array("NAME:<SetupName>", "FilePath")</td></tr>
</table></td></tr>
</table>

<table>
<tr><td></td><td><table><tr><td></td><td></td><td>&lt;SetupName>
Type: <string>
Name of the setup.
<FilePath>
Type : <string: file path>
Must be a valid file path and name.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>None</td></tr>
</table>

Python Syntax	ImportSetup (<SetupTypeName>, <SetupInfo>)

Python Example	`oModule.ImportSetup ("OptiStatistical",`
	`["NAME:StatisticalSetup1",`
	`"c:/importdir/mySetupInfoFile"])`

PasteSetup [Optimetrics]

Pastes the specified Optimetrics setup.

<table>
<tr><td><b>UI Access</b></td><td>NA</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;SetupName&gt;</td><td>String</td><td>Name of the Setup</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>None</td></tr>
</table>

Python Syntax	PasteSetup (<SetupName>)
Python Example	`oModule.PasteSetup ("OptimizationSetup1")`

RenameSetup [Optimetrics]

Renames the specified Optimetrics setup.

UI Access	**Right-click the setup in the project tree, and then click Rename** on the shortcut menu.
Parameters	
Return Value	None

Name	Type	Description
<OldName>	String	The name that needs to be replaced
<NewName>	String	Replacement name

Python Syntax	RenameSetup (<OldName> <NewName>)
Python Example	`oModule.RenameSetup ("OptimizationSetup1" "MyOptimization")`

SetPropValue [Optimetrics]

Sets the property value for the active Optimetrics setup.

<table>
<tr><td><b>UI Access</b></td><td>Set Property value on Optimetrics objects</td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td>Property path</td><td>text string</td><td>Setup property path. See discussion of Property Path</td></tr>
<tr><td>new Value</td><td>Text String, Number, or Boolean</td><td>New value data type is depending on the property type,</td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>True if the property is found and the new value is valid. Otherwise return False.</td></tr>
</table>

<table>
<tr><td><b>Python Syntax</b></td><td>SetPropValue(propPath, newValue)</td></tr>
<tr><td><b>Python Example</b></td><td><code>oOptModule.SetPropValue("ParametricSetup1\Enabled", False) //disable ParametricSetup1</code>
<code>oOptModule.SetPropValue("OptimizationSetup1/Optimizer","Quasi Newton")</code></td></tr>
</table>

SolveAllSetup

Solves all Optimetrics setups

<table>
<tr><td><b>UI Access</b></td><td>Right-click on <b>Optimetrics</b> in Project Manager and select <b>Analyze>All</b> from context menu</td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td>isBlocking</td><td>Boolean</td><td>An optional arg that defaults to <code>true</code>. If it's <code>false</code>, the command immediately returns while the analysis or analyses run in the background. The status of the analyses can be checked with the <code>AreThereSimulationsRunning</code> command.</td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>None</td></tr>
</table>

Python Syntax	SolveAllSetup()
Python Example	`oModule.SolveAllSetup()`

SolveSetup [Optimetrics]

Solves the specified Optimetrics setup.

<table>
<tr><td><b>UI Access</b></td><td>Right-click the setup in the project tree, and then click <b>Analyze</b> on the shortcut menu.</td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td>&lt;<i>SetupName</i>&gt;</td><td>String</td><td>Name of the setup to be solved</td></tr>
<tr><td>isBlocking</td><td>Boolean</td><td>An optional arg that defaults to <code>true</code>. If it's <code>false</code>, the command immediately returns while the analysis or analyses run in the background. The status of the analyses can be checked with the <code>AreThereSimulationsRunning</code> command.</td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>None</td></tr>
</table>

Python Syntax	SolveSetup (<*SetupName*>)
Python Example	`oModule.SolveSetup ("OptimizationSetup1")`

Parametric Script Commands

EditSetup [Parametric]

ExportParametricSetupTable

GenerateVariationData [Parametric]

InsertSetup [Parametric]

EditSetup [Parametric]

Modifies an existing parametric setup

<table>
<tr><td><b>UI Access</b></td><td>Right-click the setup in the project tree, and then click <b>Properties</b> on the shortcut menu.</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;SetupName&gt;</td><td>String</td><td>Name of the Setup</td></tr><tr><td>&lt;ParametricParams&gt;</td><td>List</td><td>List that defines the parameters of the parametric setup; examples are listed below.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>None</td></tr>
</table>

Python Syntax	EditSetup (<SetupName>, <ParametricParams>)
Python Example	`See EditSetup [Optimization]`

ExportParametricSetupTable

Exports the parametric setup table as a CSV file.

<table>
<tr><td><b>UI Access</b></td><td>Double-click parametric setup. Select <b>Table</b> tab. Click <b>Export</b>.</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr></table></td></tr>
</table>

	<SetupName>	String	Name of the setup.
	<filePath>	String	Full path for file export.
Return Value	None		

Python Syntax	ExportParametricSetupTable (<SetupName>, <filePath>)
Python Example	`oModule.ExportParametricSetupTable('ParametricSetup1', 'E:/Files/ParametricSetup1_Table.csv')`

GenerateVariationData [Parametric]

Generate variation data before parametric solve for CAD integrated project

Command: Right click on the parametric setup in the project tree and choose "Generate Variation Data"

Syntax: GenerateVariationData <SetupName>

Return Value: None

Parameters: <SetupName>

Name of the setup.

InsertSetup [Parametric]

Inserts a new parametric setup.

UI Access	Right-click the **Optimetrics** folder in the project tree, and then click **Add> Parametric** on the shortcut menu.

Parameters

Name	Type	Description
<Parametric Params>	Array	Array("NAME:<SetupName>", "SaveFields:=", <SaveField>, <StartingPoint>, "Sim. Setups:=", <SimSetups>, <SweepDefs>, <SweepOps>, Array("NAME:Goals", Array("NAME:Goal", <OptiGoalSpec>), ... Array("NAME:Goal", <OptiGoalSpec>))
<SetupName>	String	Name of the parametric setup.
<SimSetups>	Array of Strings	An array of Twin Builder solution setup names.
<SweepDefs>	Array	Array("NAME:Sweeps", Array("NAME:SweepDefinition", "Variable:=", <VarName>, "Data:=", <SweepData>, "Synchronize:=", <SyncNum>), ... Array("NAME:SweepDefinition", "Variable:=", <VarName>, "Data:=", <SweepData>, "Synchronize:=", <SyncNum>))
<SweepData>	String	"<SweepType>, <StartV>, <StopV>, <StepV>"
<SweepType>	String	The type of sweep data.
<SyncNum>	Integer	`SweepDatas` with the same value are synchronized.
<SweepOps>		Array("NAME:Sweep Operations", "<OpType>:=, Array(<VarValue>, …, <VarValue>), ... <OpType>:=, Array(<VarValue>, …, <VarValue>))

	<OpType>	String	The sweep operation type.
Return Value	None		

Python Syntax	InsertSetup ("OptiParametric", <ParametricParams>)
Python Example	

```
oModule.InsertSetup("OptiParametric",
[
  "NAME:ParametricSetup1",
  "SaveFields:=", true,
  [
    "NAME:StartingPoint"
  ],
  "Sim. Setups:=", ["Setup1"],
  [
    "NAME:Sweeps",
    [
      "NAME:SweepDefinition",
      "Variable:=", "$width",
      "Data:=", "LIN 12mm 17mm 2.5mm",
      "OffsetF1:=", false,
        "Synchronize:=", 0
      ],
      [
        "NAME:SweepDefinition",
        "Variable:=", "$length",
        "Data:=", "LIN 8mm 12mm 2mm",
        "OffsetF1:=", false,
        "Synchronize:=", 0
      ]
    ],
    [
      "NAME:Sweep Operations"
    ],
    [
      "NAME:Goals",
      [
        "NAME:Goal",
        "Solution:=", "Setup1 : LastAdaptive",
        "Calculation:=", "returnloss",
        "Context:=", ""
        [
    "NAME:Ranges",
    "Range:=",
    [
    "Var:=", "Freq", "Type:=","s"
    "Start:=", "8GHz",
    "Stop:=", "8GHz"
    ]
  ],
  [
    "NAME:Goal",
    "Solution:=", "Setup1 : LastAdaptive"
    "Calculation:=", "reflect",
    "Context:=", "",
    [
      "NAME:Ranges",
      "Range:=",
      [
        "Var:=", "Freq",
        "Type:=","s",
        "Start:=", "8GHz",
                  "Stop:=", "8GHz"
               ]
            ]
         ]
      ]
   ]
])
```

Optimization Script Commands

EditSetup [Optimization]

InsertSetup [Optimization]

EditSetup [Optimization]

Modifies an existing optimization setup.

UI Access	Right-click the setup in the Project Manager tree, and then click **Properties** from the shortcut menu
Parameters	<table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;SetupName&gt;</td><td>String</td><td>Name of the Setup</td></tr><tr><td>&lt;OptimizationParams&gt;</td><td>List</td><td>Parameters that define the setup; examples are listed below.</td></tr></table>
Return Value	None

Python Syntax	EditSetup (<SetupName>, <OptimizationParams>)

Python Example

```
oModule.EditSetup("OptimizationSetup1",
[
  "NAME:OptimizationSetup1",
  "UseFastCalculationUpdateAlgo:=", False,
  "FastCalcOptCtrledByUser:=", False,
  "IsEnabled:=", True,
  "SaveSolutions:=", False,
  [
    "NAME:StartingPoint"
  ],
  "Optimizer:=", "Quasi Newton",
  [
    "NAME:AnalysisStopOptions",
    "StopForNumIteration:=" , True,
    "StopForElapsTime:=", False,
    "StopForSlowImprovement:=", False,
    "StopForGrdTolerance:=", False,
    "MaxNumIteration:=",1000,
    "MaxSolTimeInSec:=",3600,
    "RelGradientTolerance:=", 0,
  "MinNumIteration:=", 10
],
"CostFuncNormType:=", "L2",
"PriorPSetup:=", "",
"PreSolvePSetup:=", True,
[
  "NAME:Variables"
],
[
  "NAME:LCS"
],
[
  "NAME:Goals",
  [
    "NAME:Goal",
    "ReportType:=", "Standard",
    "Solution:=", "TR",
    [
      "NAME:SimValueContext",
      "SimValueContext:=", [1,0,2,0,False,False,-1,1,0,1,1,"",0,0
    ]
```

```
      ],
      "Calculation:=", "acosh(Time)",
      "Name:=", "Time",
      [
        "NAME:Ranges",
        "Range:=", ["Var:=", "Time","Type:=", "a"]
      ],
      "Condition:=", "==",
      [
        "NAME:GoalValue",
        "GoalValueType:=", "Independent",
        "Format:=", "Real/Imag",
        "bG:=", ["v:=", "[1;]"]
      ],
      "Weight:=", "[1;]"
    ],
    "Acceptable_Cost:="    , 0,
    "Noise:=", 0.0001,
    "UpdateDesign:=", False,
```

	"UpdateIteration:=", 5, "KeepReportAxis:=", True, "UpdateDesignWhenDone:=", True])

InsertSetup [Optimization]

Use: Inserts a new optimization setup.

<table>
<tr><th>UI Access</th><th>Right-click the <b>Optimetrics</b> folder in the project tree, and then click <b>Add &gt; Optimization</b> on the shortcut menu.</th></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>&lt;OptimizationParams&gt;</td><td>Array</td><td><code>Array("NAME:<SetupName>", "SaveFields:=", <SaveField>, <StartingPoint>, "Optimizer:=", <Optimizer&gt;,</code>
<code>"MaxIterations:=", <MaxIter>, "PriorPSetup:=", <PriorSetup>, "PreSolvePSetup:=", <Preceed>, <OptimizationVars>, <Constraint&gt;,</code>
<code>Array("NAME:Goals", Array("NAME:Goal", <OptiGoalSpec>, <OptimizationGoalSpec&gt;), ...</code>
<code>Array("NAME:Goal", <OptiGoalSpec>, <OptimizationGoalSpec&gt;)),</code>
<code>"Acceptable_Cost:=", <AcceptableCost>, "Noise:=", <Noise>, "UpdateDesignWhenDone:=", <UpdateDesign&gt;</code></td></tr>
</table>
</td></tr>
</table>

<OptimizationVars>	Array	Array("NAME:Variables", "VarName:=", Array("i:=", <IncludeVar>, "Min:=", <MinV>, "Max:=", <MaxV>, "MinStep:=", <MinStepV>, "MaxStep:=", <MaxStepV>), ………………. "VarName:=", Array("i:=", <IncludeVar>, "Min:=", <MinV>, "Max:=", <MaxV>, "MinStep:=", <MinStepV>, "MaxStep:=", <MaxStepV>))
<MinStepV>	VarValue	The minimum step of the variable.
<MaxStepV>	VarValue	The maximum step of the variable.
<AcceptableCost>	Double	The acceptable cost value for the optimizer to stop.
<Noise>	Double	The noise of the design.
<UpdateDesign>	Boolean	Specifies whether or not to apply the optimal variation to the design after the optimization is done.
<OptimizationGoalSpec>	Array	"Condition:=", <OptimizationCond>, Array("NAME:GoalValue", "GoalValeType:=", <GoalValueType>, "Format:=", <GoalValueFormat>, "bG:=", Array("v:=", <GoalValue>)), "Weight:=", <Weight>)
<OptimizationCond>	String	Either "<=", "==", or ">="
<GoalValueType>	String	Either "Independent" or "Dependent"
<GoalValueFormat>	String	Either "Real/Imag" or "Mag/Ang".
<GoalValue>	String	Value in string. Value can be a real number, complex number, or expression.

Return Value	None

`<MinStepV>`

Type : <VarValue>

The minimum step of the variable.

`<MaxStepV>`

Type: <VarValue>

The maximum step of the variable.

`<AcceptableCost>`

Type: <double>

The acceptable cost value for the optimizer to stop.

`<Noise>`

Type: <double>

The noise of the design.

`<UpdateDesign>`

Type: <bool>

Specifies whether or not to apply the optimal variation to the design after the optimization is done.

```
<OptimizationGoalSpec>
  "Condition:=", <OptimizationCond>,
  Array("NAME:GoalValue", "GoalValueType:=",
  <GoalValueType>,
  "Format:=", <GoalValueFormat>, "bG:=",
  Array("v:=", <GoalValue>)), "Weight:=", <Weight>)
```

<OptimizationCond>

Type: <string>

Either "<=", "==", or ">="

<GoalValueType>

Type: <string>

Either "Independent" or "Dependent"

<GoalValueFormat>

Type:<string>

Either "Real/Imag" or "Mag/Ang".

<GoalValue>

Type: <string>

Value in string. Value can be a real number, complex number, or expression.

Python Syntax	InsertSetup ("OptiOptimization", <OptimizationParams>)
Python Example	oModule.InsertSetup("OptiOptimization", ["NAME:OptimizationSetup1", "SaveFields:=", false, _ ["NAME:StartingPoint", "$length:=", "8mm", "$width:=", "14.5mm"], "Optimizer:=", "Quasi Newton",

```
"MaxIterations:=", 100,
"PriorPSetup:=", "ParametricSetup1",
"PreSolvePSetup:=", true,
["NAME:Variables",
"$length:=", ["i:=", true, "Min:=", "6mm",
"Max:=", "18mm",
"MinStep:=", "0.001mm", "MaxStep:=",
"1.2mm"],
"$width:=", ["i:=", true, "Min:=",
"6.5mm", "Max:=", "19.5mm",
"MinStep:=", "0.001mm", "MaxStep:=",
"1.3mm"]],
["NAME:LCS"],
["NAME:Goals",
["NAME:Goal",
"Solution:=", "Setup1 : LastAdaptive",
"Calculation:=", "reflect",
"Context:=", "",
["NAME:Ranges",
"Range:=", ["Var:=", "Freq",
```

	"Type:=", "s", "Start:=", "8GHz", "Stop:=", "8GHz"]], "Condition:=", "&lt;=", ["NAME:GoalValue", "GoalValueType:=", "Independent", "Format:=", "Real/Imag", "bG:=", ["v:=", "[0.0001]"]], "Weight:=", "[1]"]], "Acceptable_Cost:=", 0.0002, "Noise:=", 0.0001, "UpdateDesign:=", true, "UpdateIteration:=", 5, "KeepReportAxis:=", true, "UpdateDesignWhenDone:=", true])

Sensitivity Script Commands

EditSetup [Sensitivity]

InsertSetup [Sensitivity]

EditSetup [Sensitivity]

Modifies an existing sensitivity setup.

<table>
<tr><td><b>UI Access</b></td><td>Right-click the setup in the project tree, and then click Properties on the shortcut menu</td></tr>
<tr><td><b>Parameters</b></td><td><table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td>&lt;SetupName&gt;</td><td>String</td><td>Name of the Setup</td></tr>
<tr><td></td><td></td><td></td></tr>
</table></td></tr>
<tr><td><b>Return Value</b></td><td>None</td></tr>
</table>

<table>
<tr><td><b>Python Syntax</b></td><td>EditSetup (<SetupName>, <SensitivityParams&gt;)</td></tr>
<tr><td><b>Python Example</b></td><td><pre>
oModule.EditSetup("OptimizationSetup1",
[
"NAME:OptimizationSetup1",
"UseFastCalculationUpdateAlgo:=", False,
"FastCalcOptCtrledByUser:=", False,
"IsEnabled:=", True,
"SaveSolutions:=", False,
[
"NAME:StartingPoint"
],
"Optimizer:=", "Quasi Newton",
</pre></td></tr>
</table>

```
[
"NAME:AnalysisStopOptions",
"StopForNumIteration:=", True,
"StopForElapsTime:=", False,
"StopForSlowImprovement:=", False,
"StopForGrdTolerance:="        , False,
"MaxNumIteration:=", 1001,
"MaxSolTimeInSec:=", 3600,
"RelGradientTolerance:=", 0,
"MinNumIteration:=", 10
],
"CostFuncNormType:=", "L2",
"PriorPSetup:=", "",
"PreSolvePSetup:=", True,
[
"NAME:Variables"
],
[
"NAME:LCS"
```

```
],
[
"NAME:Goals",
[
"NAME:Goal",
"ReportType:=", "Standard",
"Solution:=", "TR3",
[
"NAME:SimValueContext",
"SimValueContext:=", [1,0,2,0,False,False,-1,1,0,1,1,"",0,0]
],
"Calculation:=", "mag(DIFF1.VAL)",
"Name:=", "DIFF1.VAL",
[
"NAME:Ranges",
"Range:=", [
"Var:=", "Time","Type:=", "a"]
],
"Condition:=", "==",
[
"NAME:GoalValue",
```

	"GoalValueType:=", "Independent", "Format:=", "Real/Imag", "bG:=", ["v:=", "[1;]"]], "Weight:=", "[1;]"]], "Acceptable_Cost:=", 0, "Noise:=", 0.0001, "UpdateDesign:=", False, "UpdateIteration:=", 5, "KeepReportAxis:=", True, "UpdateDesignWhenDone:=", True])

InsertSetup [Sensitivity]

Inserts a new sensitivity setup.

<table>
<tr><td><b>UI Access</b></td><td>Right-click <b>Optimetrics</b> in the project tree, and then click <b>Add&gt;Sensitivity</b> on the shortcut menu.</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr></table></td></tr>
</table>

<SensitivityParams>	Array	Array("NAME:<SetupName>", "SaveFields:=", <SaveField>, <StartingPoint>, "MaxIterations:=", <MaxIter>, "PriorPSetup:=", <PriorSetup>, "PreSolvePSetup:=", <Preceed>, <SensitivityVars>, <Constraint>, Array("NAME:Goals", Array("NAME:Goal", <OptiGoalSpec>), ..., Array("NAME:Goal", <OptiGoalSpec>)), "Primary Goal:=". <PrimaryGoalID>, "PrimaryError:=", <PrimaryError>)
<SensitivityVars>	Array	Array("NAME:Variables", "VarName:=", Array("i:=", <IncludeVar>, "Min:=", <MinV>, "Max:=", <MaxV>, "IDisp:=", <InitialDisp>),... "VarName:=", Array("i:=", <IncludeVar>, "Min:=", <MinV>, "Max:=", <MaxV>, "IDisp:=", <InitialDisp>))
<InitialDisp>	VarValue	Index of the Primary goal. Index starts from zero.
<PrimaryError>	Double	Error associated with the Primary goal.

Return Value	None

Python Syntax	InsertSetup ("OptiSensitivity", <SensitivityParams>)

Python Example	`oModule.InsertSetup(`
	`"OptiSensitivity", _`
	`["NAME:SensitivitySetup1",_`
	`"SaveFields:=", true,_`
	`["NAME:StartingPoint"], _`
	`"MaxIterations:=", 20,_`
	`"PriorPSetup:=", "",_`
	`"PreSolvePSetup:=", true, _`
	`["NAME:Variables"], _`
	`["NAME:LCS"],_`
	`"NAME:Goals",_`
	`["NAME:Goal", _`
	`"Solution:=", "Setup1 : LastAdaptive",_`
	`"Calculation:=", "returnloss",_`
	`"Context:=", "",_`
	`["NAME:Ranges",_`
	`"Range:=", ["Var:=", "Freq", "_`
	`Type:=", "s",_`
	`"Start:=", "8GHz", "Stop:=", "8GHz"]]],_`

```
["NAME:Goal",_
"Solution:=", "Setup1 : LastAdaptive",_
"Calculation:=", "reflect",_
"Context:=", "",_
["NAME:Ranges",_
"Range:=", ["Var:=","Freq",_
"Type:=", "s",_
"Start:=", "8GHz", "Stop:=", "8GHz"]]]],_
 "Primary Goal:=", 1,_
"PrimaryError:=", 0.001])
```

Statistical Script Commands

EditSetup [Statistical]

InsertSetup Statistical

EditSetup [Statistical]

Modifies an existing statistical setup.

<table>
<tr><td><b>UI Access</b></td><td>Right-click the setup in the project tree, and clickProperties on the shortcut menu.</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;SetupName&gt;</td><td>String</td><td>Name of the Setup</td></tr><tr><td></td><td></td><td></td></tr></table></td></tr>
</table>

Return Value	None

Python Syntax	EditSetup (<SetupName>, <StatisticalParams>)
Python Example	`See EditSetup [Optimization]`

InsertSetup [Statistical]

Inserts a new statistical setup.

<table>
<tr><td><b>UI Access</b></td><td>Right-click <b>Optimetrics</b> in the project tree, and then click <b>Add>Statistical</b> on the shortcut menu.</td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td>&lt;StatisticalParams&gt;</td><td>Array</td><td><code>Array("NAME:<SetupName>", "SaveFields:=", <SaveField>, <StartingPoint>, "MaxIterations:=", <MaxIter>, "PriorPSetup:=", <PriorSetup>, "PreSolvePSetup:=", <Preceed>, <StatisticalVars>, Array("NAME:Goals", Array("NAME:Goal", <OptiGoalSpec>), …, Array("NAME:Goal", <OptiGoalSpec&gt;)))</code></td></tr>
<tr><td>&lt;StatisticalVars&gt;</td><td>Array</td><td><code>Array("NAME:Variables", "VarName:=", Array("i:=", <IncludeVar>, "Dist:=", <DistType>, "Tol:=", <Tolerance>, "StdD:=", <StdD>, "Min:=", <MinCutoff>, "Max:=",</code></td></tr>
</table>
</td></tr>
</table>

			<MaxCutoff>, ... "VarName:=", Array("i:=", <IncludeVar>, "Dist:=", <DistType>, "Tol:=", <Tolerance>, "StdD:=", <StdD>, "Min:=", <MinCutoff>, "Max:=", <MaxCutoff>))
	<DistType>	String	Distrbution can be "Gaussian" or "Uniform".
	<Tolerance>	VarValue	The tolerance for the variable when distribution is Uniform
	<StdD>	VarValue	The standard deviation for the variable when distribution is Gaussian.
	<MinCutoff>	Double	The minimum cut-off for the variable when distribution is Gaussian.
	<MaxCutoff>	Double	The maximum cut-off for the variable when distribution is Gaussian.
Return Value	None		

Python Syntax	InsertSetup ("OptiStatistical", <StatisticalParams>)
Python Example	oModule.InsertSetup("OptiStatistical", _ ["NAME:StatisticalSetup1", _ "SaveFields:=", true, _ ["NAME:StartingPoint"],_ "MaxIterations:=", 50,_ "PriorPSetup:=", "", _ ["NAME:Variables"], _ ["NAME:Goals", _

	["NAME:Goal", _ "Solution:=", "Setup1 : LastAdaptive", _ "Calculation:=", "returnloss", _ "Context:=", "", _ ["NAME:Ranges", _ "Range:=", ["Var:=", "Freq", _ "Type:=", "s",_ "Start:=", "8GHz", "Stop:=", "8GHz"]]],_ ["NAME:Goal",_ "Solution:=", "Setup1 : LastAdaptive",_ "Calculation:=", "reflect",_ "Context:=", "", _ ["NAME:Ranges",_ "Range:=", ["Var:=", "Freq", "Type:=",_ "s", "Start:=", "8GHz", "Stop:=", "8GHz"]]]])

For Q3D Extractor and Circuit the command details are as follows:

Inserts a new statistical setup.

Command: Right-click **Optimetrics** in the project tree, and then click **Add>Statistical** on the shortcut menu.

Syntax: InsertSetup "OptiStatistical", <StatisticalParams>

Return Value: None

Parameters: <StatisticalParams>

```
  Array("NAME:<SetupName>", "SaveFields:=",
  <SaveField>, <StartingPoint>, "MaxIterations:=",
  <MaxIter>, "PriorPSetup:=", <PriorSetup>,
  "PreSolvePSetup:=", <Preceed>, <StatisticalVars>,
  Array("NAME:Goals", Array("NAME:Goal",
  <OptiGoalSpec>), …, Array("NAME:Goal",
  <OptiGoalSpec>))),
<StatisticalVars>
  Array("NAME:Variables",
    "VarName:=", Array("i:=", <IncludeVar>, "Dist:=",
      <DistType>, "Tol:=", <Tolerance>,
      "StdD:=", <StdD>, "Min:=", <MinCutoff>, "Max:=",
      <MaxCutoff>, ...
    "VarName:=", Array("i:=", <IncludeVar>, "Dist:=",
      <DistType>, "Tol:=", <Tolerance>, "StdD:=",
      <StdD>, "Min:=", <MinCutoff>, "Max:=",
      <MaxCutoff>))
```

Parameters:

```
<DistType>
  Type : <string>
  Distrbution can be "Gaussian" or "Uniform".
<Tolerance>
  Type: <VarValue>
  The tolerance for the variable when distribution is Uniform.
<StdD>
  Type: <VarValue>
  The standard deviation for the variable when distribution is Gaussian.
<MinCutoff>
  Type: <double>
  The minimum cut-off for the variable when distribution is Gaussian.
<MaxCutoff>
  Type: <double>
  The maximum cut-off for the variable when distribution is Gaussian.
```

15 - Field Overlays Module Script Commands

Field overlay commands should be executed by the Field Overlays module, which is called "FieldsReporter" in scripts.

Set oModule = oDesign.GetModule("FieldsReporter")

oModule.CommandName <args>

AddMarker[Fields Reporter]

Adds a marker to the current fields plot.

<table>
<tr><td><b>UI Access</b></td><td>Right-click <b>Field Overlays</b> > <b>Plot Fields</b> > <b>Marker</b> > <b>Add Marker</b>.</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;<i>Position</i>&gt;</td><td>Array</td><td>Array containing X, Y and Z coordinates.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

<table>
<tr><td><b>Python Syntax</b></td><td>AddMarker(&lt;<i>Position</i>&gt;)</td></tr>
<tr><td><b>Python Example</b></td><td><code>oModule.AddMarker([</code>
<code>"-267.007756778494mil",</code>
<code>"-640.898759461403mil",</code>
<code>"9.09494701772928e-13mil"])</code></td></tr>
</table>

AddMarkerToPlot

Adds a marker to a trace on a named field plot.

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;<i>Location</i>&gt;</td><td>Array</td><td>Array of strings containing X.Y, and Z coordinates for the marker.</td></tr><tr><td>&lt;<i>PlotName</i>&gt;</td><td>String</td><td>Name of the field plot.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	AddMarkerToPlot(<*Location*>, <*PlotName*>)

Python Example	`oModule.AddMarkerToPlot`
	`(["0.290455877780914in", "-0.616900205612183in",`
	`"1.77635683940025e-015in"],`
	`"Mag_H1")`
	`oModule.AddMarkerToPlot`
	`(["-0.317279517650604in",`
	`"1.22481322288513in", "0in"],`
	`"Mag_H1")`

AddNamedExpression

Creates a named expression using the expression at the top of the stack.

<table>
<tr><th>UI Access</th><th>Click <b>Add</b> in the <b>Fields Calculator</b> window.</th></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td>&lt;<i>ExprName</i>&gt;</td><td>String</td><td>Name for the new named expression.</td></tr>
<tr><td>&lt;<i>FieldType</i>&gt;</td><td>String</td><td>Type of field.</td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>None</td></tr>
</table>

Python Syntax	AddNamedExpression(<*ExprName*>, <*FieldType*>)
Python Example	`oModule.AddNamedExpression("Mag_JxE", "Fields")`

CalcOp

Performs a calculator operation.

<table>
<tr><td><b>UI Access</b></td><td>Operation commands like <b>Mag</b>, <b>+</b>, etc.</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;<i>OpString</i>&gt;</td><td>String</td><td>The text on the corresponding calculator button.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	CalcOp(<*OpString*>)
Python Example	`oModule.CalcOp("+")`

CalcStack

Performs an operation on the stack.

<table>
<tr><td><b>UI Access</b></td><td>Stack operation buttons such as <b>Push</b> and <b>Pop</b>.</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;<i>OpString</i>&gt;</td><td>String</td><td>The text on the corresponding calculator button.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	CalcStack(<*OpString*>)
Python Example	`oModule.CalcStack("push")`

CalculatorRead

Gets a register file and applies it to the calculator stack.

UI Access	Click **Read...** in the **Fields Calculator** dialog.
Parameters	Name | Type | Description <*FileName*> | String | Path to and including name of input register file. <*SoluName*> | String | Specified solution to read in. <*FieldType*> | String | Type of specified field. <*VarArray*> | Array | Array of variable names, value pairs.
Return Value	None.

Name	Type	Description
<*FileName*>	String	Path to and including name of input register file.
<*SoluName*>	String	Specified solution to read in.
<*FieldType*>	String	Type of specified field.
<*VarArray*>	Array	Array of variable names, value pairs.

Python Syntax	CalculatorRead(<*FileName*>, <*SoluName*>, <*FieldType*>, <*VarArray*>)

Python Example	`oModule.CalculatorRead(`
	`"c:\example.reg",`
	`"Setup1: LastAdaptive", "Fields",`
	`["Freq:=", "10GHz", "Phase:=", "0deg"])`

CalculatorWrite

Writes contents of top register to file.

<table>
<tr><td><b>UI Access</b></td><td>Click <b>Write...</b> in the <b>Fields Calculator</b> window.</td></tr>
<tr><td><b>Parameters</b></td><td><table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td>&lt;<i>OutputFilePath</i>&gt;</td><td>String</td><td>Path to and including name of output register file.</td></tr>
<tr><td>&lt;<i>SoluNameArray</i>&gt;</td><td>Array</td><td>Array of strings containing solution names.</td></tr>
<tr><td>&lt;<i>VarArray</i>&gt;</td><td>Array</td><td>Array of variable names, value pairs.</td></tr>
</table></td></tr>
<tr><td><b>Return Value</b></td><td>None</td></tr>
</table>

Python Syntax	CalculatorWrite(<*OutputFilePath*>, <*SoluNameArray*>, <*VarArray*>)

Python Example	`oModule.CalculatorWrite("C:\test.reg",`
	`   ["Solution:=", "Setup1 : LastAdaptive"],`
	`   ["Freq:=", "1GHz", "Phase:=", "0deg"]`
	`)`

ChangeProperty

Changes the properties of a field plot marker.

UI Access	Select a field plot marker in the marker table.

<table>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><i>&lt;propertyArgs&gt;</i></td><td>Array</td><td>Structured array.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

<table>
<tr><td><b>Python Syntax</b></td><td>ChangeProperty(&lt;<i>propertyArgs</i>&gt;)</td></tr>
<tr><td><b>Python Example</b></td><td><pre>
oDesign.ChangeProperty(
        [
                "NAME:AllTabs",
                [
                        "NAME:FieldPlotMarkerDataTab",
                        [
                                "NAME:PropServers",
                                "FieldsReporter:FieldsPlotMarker:m1"
                        ],
                        [
                                "NAME:ChangedProps",
                                [
                                        "NAME:Name",
                                        "Value:="   , "Temp"
                                ]
                        ]
                ]
        ])
oDesign.ChangeProperty(
</pre></td></tr>
</table>

```
        [
                "NAME:AllTabs",
                [
                        "NAME:FieldPlotMarkerDataTab",
                        [
                                "NAME:PropServers",
                                "FieldsReporter:FieldsPlotMarker:Temp"
                        ],
                        [
                                "NAME:ChangedProps",
                                [
                                        "NAME:Display Value",
                                        "Value:="               , True
                                ]
                        ]
                ]
        ])
oDesign.ChangeProperty(
        [
                "NAME:AllTabs",
                [
                        "NAME:FieldPlotMarkerDataTab",
                        [
                                "NAME:PropServers",
                                "FieldsReporter:FieldsPlotMarker:Temp"
                        ],
                        [
                                "NAME:ChangedProps",
AME:Display Name",
                                        "Value:="               , True
                                ]
                        ]
                ]
        ])
oDesign.ChangeProperty(
        [
                "NAME:AllTabs",
                [
                        "NAME:FieldPlotMarkerDataTab",
                        [
                                "NAME:PropServers",
                                "FieldsReporter:FieldsPlotMarker:Temp"
                        ],
                        [
                                "NAME:ChangedProps",
                                [
                                        "NAME:Display Units",
                                        "Value:="               , True
                                ]
                        ]
                ]
        ])
oDesign.ChangeProperty(
        [
                "NAME:AllTabs",
                [
                        "NAME:FieldPlotMarkerDataTab",
                        [
                                "NAME:PropServers",
                                "FieldsReporter:FieldsPlotMarker:Temp"
                        ],
```


ChangeGeomSettings

Changes the line discretization setting.

<table>
<tr><td><b>UI Access</b></td><td>In the <b>Fields Calculator</b> dialog box, click on <b>Geom Settings...</b></td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><i>&lt;LineDiscr&gt;</i></td><td>Integer</td><td>Line discretization value.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	ChangeGeomSettings(<*LineDiscr*>)
Python Example	`oModule.ChangeGeomSettings(500)`

ClcEval

Evaluates the expression at the top of the stack using the provided solution name and variable values.

<table>
<tr><td><b>UI Access</b></td><td>Click <b>Eval</b> in the <b>Fields Calculator</b> dialog.</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><i>&lt;SoluName&gt;</i></td><td>String</td><td>Name of specified solution.</td></tr><tr><td><i>&lt;VarArray&gt;</i></td><td>Array</td><td>Array of variable name, value pairs.</td></tr></table></td></tr>
</table>

<table>
<tr><td></td><td><table><tr><td><i>&lt;FieldType&gt;</i></td><td>String</td><td>Optional. Type of specified field.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>None</td></tr>
</table>

Python Syntax	ClcEval(<*SoluName*>, <*VarArray*>, [Optional <*FieldType*>])

Python Example	`oModule.ClcEval("Setup1: LastAdaptive",`
	`["Freq:=", "10GHz",`
	`"Phase:=", "0deg"])`

ClcMaterial

Performs a material operation on the top stack element.

<table>
<tr><td><b>UI Access</b></td><td>Click <b>Matl...</b> in the <b>Fields Calculator</b> dialog.</td></tr>
<tr><td><b>Parameters</b></td><td><table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;MatString&gt;</i></td><td>String</td><td>The material property to apply.</td></tr>
<tr><td><i>&lt;OpString&gt;</i></td><td>String</td><td>Name of operation. Possible values are "mult", or "div".</td></tr>
</table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	ClcMaterial(<*MatString*>, <*OpString*>)
Python Example	`oModule.ClcMaterial("Permeability (mu)" "mult")`

ClcMaterialValue

Shows the value of the material property without performing any operation.

<table>
<tr><td><b>UI Access</b></td><td>Select <b>None</b> in the <b>Material Operation</b> dialog.</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;<i>MaterialName</i>&gt;</td><td>String</td><td>Name of specified material property.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	ClcMaterialValue(<*MaterialName*>)
Python Example	`oModule.ClcMaterialValue("Mass Density")`

ClearAllMarkers[Fields Reporter]

Clears all markers in the current fields overlay plot.

UI Access	Right-click **Field Overlays** > **Plot Fields** > **Marker** > **Clear All**.
Parameters	None.
Return Value	None.

Python Syntax	ClearAllMarkers()

Python Example	`oModule.ClearAllMarkers()`

ClearAllNamedExpr

Clears all user-defined named expressions from the list.

UI Access	Click **ClearAll** in the **Fields Calculator** dialog.
Parameters	None.
Return Value	None.

Python Syntax	ClearAllNamedExpr()
Python Example	`oModule.ClearAllNamedExpr()`

CopyNamedExprToStack

Copies the named expression selected to the calculator stack.

<table>
<tr><td><b>UI Access</b></td><td>Select a named expression and then click <b>Copy to stack</b>.</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><i>&lt;ExprName&gt;</i></td><td>String</td><td>Name of the expression to be copied to the top of the stack.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	CopyNamedExprToStack(<*ExprName*>)
Python Example	`oModule.CopyNamedExprToStack("Mag_JxE")`

CreateFieldPlot

> **Note:**
>
> Use in conjunction with GetGeometryIdsForNetLayerCombinations and GetGeometryIdsForAllNetLayerCombinations.
>
> - GetGeometryIdsForNetLayerCombinations returns ID numbers of all faces and edges of the active design related to the current target combination of nets and layers.
> - GetGeometryIdsForAllNetLayerCombinations returns ID numbers for all possible combinations of nets and layers it is possible to choose.

Use: Creates a field/mesh plot "Field" or a visual ray trace ("VRT") plot.

Command: **Mechanical** > **Fields** > **Plot Fields** > <*field_quantity*>

Syntax: `CreateFieldPlot <PlotParameterArray> ["Field" | "VRT"]`

Return Value: None

Parameters: `<PlotParameterArray> "Field"`

```
Array("NAME:<PlotName>",
  "SolutionName:=", <string>,
  "QuantityName:=", <string>,
  "PlotFolder:=", <string>,
  "UserSpecifyName:=", <int>,
"UserSpecifyFolder:=", <int>,
"IntrinsicVar:=", <string>,
"PlotGeomInfo:=", <PlotGeomArray>,
"FilterBoxes:=", <FilterBoxArray>,
<PlotOnPointsSettings>,
<PlotOnLineSettings>,
<PlotOnSurfaceSettings>,
<PlotOnVolumeSettings>)
```

SolutionName:

Name of the solution setup and solution formatted as

`"<SolveSetupName>: <WhichSolution>",`

where `<WhichSolution>` can be `"Adaptive_<n>"`, `"LastAdaptive"`, or `"PortOnly"`.

For example: `"Setup1 : Adaptive_2"`

A space is required on either side of the colon (**:**) character. If omitted, the plot will not be created.

QuantityName:

Type of plot to create. Possible values vary according to the solver.

Mesh plots: `"Mesh"`

Mechanical **Field plots include**
(dependent on the solution type and whether there is trace mapping):

```
"Mag_Displacement", "Displacement_Vector", "Temperature",
"HeatFlux", "Mag_HeatFlux", "Surface Loss Density",
"Volume Loss Density", "Linked Heat Transfer Coefficient",
"Equivalent Stress", "Equivalent Strain",
"Thermal Conductivity X", "Thermal Conductivity Y",
"Thermal Conductivity Z", "Youngs Modulus X",
"Youngs Modulus Y", "Youngs Modulus Z", "Poisson Ratio XY",
"Poisson Ratio YZ", "Poisson Ratio XZ", "Thermal Expansion X",
"Thermal Expansion Y", "Thermal Expansion Z",
"Shear Modulus X", "Shear Modulus Y", "Shear Modulus Z"
```

PlotFolder:

Name of the folder to which the plot should be added. (Values vary with the design type.) Some possible values include

```
"E Field", "H Field", "Jvol", "Jsurf", "SAR Field",
"Jc","Surface-Loss", "QSurf", "Temperature", "Energy",
"Average-Surface-Loss-Density, "Dielectric_Loss",
"MeshPlots", "Heat Flux", and "Displacement"
```

UserSpecifyName

0 if default name for plot is used, 1 otherwise.

Not needed. <PlotName> will be respected regardless of whether thisflag is set.

`UserSpecifyFolder`

0 if default folder for plot is used, 1 otherwise.

Not needed. The specified PlotFolder will be respected regardless of whether this flag is set.

IntrinsicVar:

Formatted string that specifies the frequency and phase at which to make the plot.

For example: `"Freq='1GHz' Phase='30deg'"`

PlotGeomInfo:

Creating field plot on selected layer-net pairs:

For example, Maxwell: `"PlotGeomInfo:=", [1,"Volume","ObjList", 2,"LC1_1:Top","LC1_1:Top#L1"]`

HFSS example with "Plot on surface" option is not checked:

```
"PlotGeomInfo:=", [1,"Volume","LayerNets",2, "Top",2,
"net1","net2","Ground",1,"GND"]
```

The command creates field plot on two layer-nets combinations. It starts with type "Volume", and subtype "LayerNets", followed by the number of layer-nets combinations which is 2 in this example.

The two layer-nets combinations are (1) "Top" layer, two nets, net names are "net1" and "net2", *i.e.*, [Top, net1] pair, and [Top, net2] pair. (2) "Ground" layer and one net, net name is "GND", *i.e.*, [Ground, GND] pair.

HFSS example with "Plot on surface" option checked:

```
"PlotGeomInfo:=", [1,"Surface","LayerNetsExtFace",2,
"Top",2,"net1","net2",_"Ground",1,"GND"]
```

The command creates field plot on two layer-nets combinations. It starts with type "Surface", and subtype "LayerNetsExtFace", followed by the number of layer-nets combinations which is 2 in this example.

The two layer-nets combinations are (1) "Top" layer, two nets, net names are "net1" and "net2", i.e., [Top, net1] pair, and [Top, net2] pair. (2) "Ground" layer and one net, net name is "GND", i.e., [Ground, GND] pair.

Limitations:

- No Layer/Nets name is supported for field plot command recording should be able to add it based on this change.
- No Layer/Nets name support for pure Layout design field plot command.

`<PlotGeomArray>`

```
Array(<NumGeomTypes>, <GeomTypeData>, <GeomTypeData>, ...)
```

For example: `Array(4, "Volume", "ObjList", 1, "Box1", "Surface", "FacesList", 1, "12", "Line", 1, "Polyline1", "Point", 2, "Point1", "Point2"`

`<NumGeomTypes>`

Type: <int>

Number of different geometry types (volume, surface, line, point) plotted on at the same time.

`<GeomTypeData>`

```
<GeomType>, <ListType>, <NumIDs>, <ID>, <ID>, ...)
```

`<GeomType>`

Type: <string>

Possible values are `"Volume"`, `"Surface"`, `"Line"`, `"Point"`.

`<ListType>`

Type: <string>

Possible values are `"ObjList"`, or `"FacesList"`.

These are used for the `GeomType` of `"Line"` or `"Point"`.

`<NumIDs>`

Type: <int>

Number of IDs or object names that will follow.

`<ID>`

Type: <int> or <string>

ID of a face or name of an object, line, or point on which to plot.

`<FilterBoxArray>`

Array of names of objects to use to restrict the plot range.

```
Array(<NumFilters>, <ObjName>, <ObjName>, ...)
```

```
Example:Array(1, "Box1")
Example: Array(0) no filtering
```

`<PlotOnPointSettings>`

```
Array("NAME:PlotOnPointSettings",
    "PlotMarker:=", <bool>,
    "PlotArrow:=", <bool>)
```

`<PlotOnLineSettings>`

```
  Array("NAME:PlotOnLineSettings",
    Array("NAME:LineSettingsID",
    "Width:=", <int>,
    "Style:=", <string>),
    "IsoValType:=", <string>,
    "ArrowUniform:=", <bool>,
    "NumofArrow:=", <int>)
```

`Style`

Possible values are: `"Cylinder", "Solid", "Dashdash", "Dotdot", "Dotdash"`

`IsoValType`

Possible values are `"Tone", "Fringe", "Gourard"`

`<PlotOnSurfaceSettings>`

```
  Array("NAME:PlotOnSurfaceSettings",
  "Filled:=", <bool>,
  "IsoValType:=", <string>,
  "SmoothShade:=", <bool>,
  "AddGrid:=", <bool>,
  "MapTransparency:=", <bool>,
  "Transparency:=", <doubl.e>,
  "ArrowUniform:=", <bool>
  "ArrowSpacing:=", <double>
  "GridColor:=", Array(<int>, <int>, <int>)
```

`IsoValType`

Possible values are `"Tone", "Line", "Fringe", "Gourard"`

`GridColor`

Array containing the R, G, B components of the color. Components should be in the range 0 to 255.

```
<PlotOnVolumeSettings>
```

```
Array("NAME:PlotOnVolumeSettings",
"PlotIsoSurface:=", <bool>,
"CloudDensity:=", <double>,
"PointSize:=", <int>,
"ArrowUniform:=", <bool>,
"ArrowSpacing:=", <double>)
```

Python Example – E Magnitude Field Plot:

```
oModule.CreateFieldPlot(
[
   "NAME:Mag_E1",
   "SolutionName:="      , "Setup1 : LastAdaptive",
   "QuantityName:="      , "Mag_E",
   "PlotFolder:="        , "E Field1",
   "UserSpecifyName:="   , 0,
   "UserSpecifyFolder:=" , 0,
   "IntrinsicVar:="      , "Freq='1GHz' Phase='0deg'",
   "PlotGeomInfo:=", [1  , "Surface", "FacesList", 1, "7"],
   "FilterBoxes:=", [0],
   [
      "NAME:PlotOnSurfaceSettings",
      "Filled:="         , False,
      "IsoValType:="     , "Fringe",
      "SmoothShade:="    , True,
      "AddGrid:="        , False,
      "MapTransparency:=", True,
      "Transparency:="   , 0,
      "ArrowUniform:="   , True,
      "ArrowSpacing:="   , 0.100000001490116,
      "GridColor:="      , [255, 255, 255]
   ]
], "Field")
```

Electron Density Field Plot:

```
oModule = oDesign.GetModule("FieldsReporter")
oModule.CreateFieldPlot(
   [
      "NAME:Electron_Density3",
      "SolutionName:="         , "AIR : RFDischarge",
      "UserSpecifyName:="      , 0,
      "UserSpecifyFolder:="    , 0,
      "QuantityName:="         , "Electron_Density",
      "PlotFolder:="           , "RF Discharge Fields",
      "StreamlinePlot:="       , False,
      "AdjacentSidePlot:="     , False,
      "FullModelPlot:="        , False,
      "IntrinsicVar:="         , "Freq=\'0.20000000000000001GHz\'
                                 GasPressure=\'0.02kPascal\'",
      "PlotGeomInfo:="         , [1,"Surface","FacesList",1,"48"],
      "FilterBoxes:="          , [0],
      [
         "NAME:PlotOnSurfaceSettings",
         "Filled:="            , False,
         "IsoValType:="        , "Tone",
         "AddGrid:="           , False,
         "MapTransparency:="   , True,
         "Refinement:="        , 0,
         "Transparency:="      , 0,
            "SmoothingLevel:="    , 0,
            "ShadingType:="       , 0,
            [
               "NAME:Arrow3DSpacingSettings",
               "ArrowUniform:="   , True,
               "ArrowSpacing:="   , 0,
               "MinArrowSpacing:=", 0,
               "MaxArrowSpacing:=", 0
               "GridColor:="      , [255,255,255]
            ],
         ],
      "EnableGaussianSmoothing:=" , False,
      "SurfaceOnly:="             , False
   ], "Field")
```

HFSS Multipaction SEY Plot

```
# ------------------------------------------

# Script Recorded by Ansys Electronics Desktop Version 2025.2.0

# 15:36:54 Jan 15, 2025

# ------------------------------------------

import ScriptEnv

ScriptEnv.Initialize("Ansoft.ElectronicsDesktop")

oDesktop.RestoreWindow()

oProject = oDesktop.SetActiveProject("JPL_coax_r251")

oDesign = oProject.SetActiveDesign("Coaxial_7-8")

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

```
oModule.CreateFieldPlot(
  [
    "NAME:SEY_Plot1",
    "SolutionName:=" , "300MHz_low : Power",
    "UserSpecifyName:=" , 0,
    "UserSpecifyFolder:=" , 0,
    "QuantityName:=" , "Unknown",
    "PlotFolder:=" , "SEY_Plot",
    "StreamlinePlot:=" , False,
    "AdjacentSidePlot:=" , False,
    "FullModelPlot:=" , False,
    "IntrinsicVar:=" , "PowerMultiplier=\'250\' Time=\'0s\'",
    "PlotGeomInfo:=" , [1,"Surface","FacesList",4,"16","17","18","24"],
    "FilterBoxes:=" , [0],
    [
      "NAME:PlotOnSurfaceSettings",
      "Filled:=" , False,
      "IsoValType:=" , "Tone",
      "AddGrid:=" , False,
      "MapTransparency:=" , True,
      "Refinement:=" , 0,
    "Transparency:=" , 0,
    "SmoothingLevel:=" , 0,
    "ShadingType:=" , 0,
    [
      "NAME:Arrow3DSpacingSettings",
      "ArrowUniform:=" , True,
      "ArrowSpacing:=" , 0,
      "MinArrowSpacing:=" , 0,
      "MaxArrowSpacing:=" , 0
    ],
  "GridColor:=" , [255,255,255]
],
"EnableGaussianSmoothing:=", False,
"SurfaceOnly:=" , False,
"QuantityName:=" , "QuantityName_SecondaryEmission",
"PlotFolder:=" , "SEY_Plot",
"IntrinsicVar:=" , "PowerMultiplier=\'250\' Time=\'0s\'"
], "SecondaryEmission")
```

DeleteFieldPlot

Deletes one or more field plots.

<table>
<tr><td><b>UI Access</b></td><td>Right-click on one filed plot, select <b>Delete</b>.</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;<i>NameArray</i>&gt;</td><td>Array</td><td>Array of strings containing the names of the plots to delete.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	DeleteFieldPlot(<*NameArray*>)
Python Example	`oModule.DeleteFieldPlot(["Mag_E1", "Vector_E1"])`

DeleteMarker[Fields Reporter]

Deletes one or more marker in the current field plot.

<table>
<tr><td><b>UI Access</b></td><td>Right-click <b>Field Overlays</b> &gt; <b>Plot Fields</b> &gt; <b>Marker</b> &gt; <b>Delete Marker</b>.</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;<i>MarkerNames</i>&gt;</td><td>Array</td><td>Array of strings containing marker names.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	DeleteMarker(<*MarkerNames*>)

Python Example	`oModule.DeleteMarker(["m1", "m2"])`

DeleteNamedExpr

Deletes the selected named expression from the list.

<table>
<tr><td><b>UI Access</b></td><td>Select a named expression and then click <b>Delete</b>.</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><i>&lt;ExprName&gt;</i></td><td>String</td><td>Name of specified named expression.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	DeleteNamedExpr(<*ExprName*>)
Python Example	`oModule.DeleteNamedExpr("Mag_JxE")`

DeleteUneditablePlot

Deletes one or more uneditable plots.

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><i>&lt;NameArray&gt;</i></td><td>Array</td><td>Array of the plot names to delete.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	DeleteUneditablePlot(<*NameArray*>)
Python Example	`oModule.DeleteUneditablePlot(["Mag_E1", "Vector_E1"])`

DoesNamedExpressionExists

Determines whether specified named expression exists.

UI Access	N/A
Parameters	Name | Type | Description <*ExpName*> | String | Name of the specified named expression.
Return Value	Boolean: • **1** - named expression exists. • **0** - named expression does not exist.

Python Syntax	DoesNamedExpressionExists(<*ExpName*>)
Python Example	`oModule.DoesNamedExpressionExists("Mag_JxE")`

EnterComplex

Enters a complex number onto the stack.

<table>
<tr><td><b>UI Access</b></td><td>Click <b>Number</b>, and then click <b>Scalar</b>. <b>Complex</b> option is selected.</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><i>&lt;ComplexNum&gt;</i></td><td>String</td><td>String of complex value. Ex. "1 + 2j".</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	EnterComplex(<*ComplexNum*>)
Python Example	`oModule.EnterComplex("1 + 2 j")`

EnterComplexVector

Enters a complex vector onto the stack.

<table>
<tr><td><b>UI Access</b></td><td>Click <b>Number</b>, and then click <b>Vector</b>. <b>Complex</b> option is selected.</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><i>&lt;ComplexVector&gt;</i></td><td>Array</td><td>Array of strings containing X, Y and Z complex values.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	EnterComplexVector(<*ComplexVector*>)

Python Example	`oModule.EnterComplexVector`
	`([`

	`"1 + 2 j",`
	`"1 + 2 j",`
	`"1 + 2 j"`
	`])`

EnterCoord

Enters a coordinate system defined in the 3D Modeler editor.

<table>
<tr><td><b>UI Access</b></td><td>Click <b>Geometry</b> and then select <b>Coord</b>.</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;<i>CoordName</i>&gt;</td><td>String</td><td>Name of a coordinate system defined in the 3D Modeler editor.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	EnterCoord(<*CoordName*>)
Python Example	`oModule.EnterCoord("Global")`

EnterEdge

Enters an edge defined in the 3D Modeler editor.

UI Access	N/A

<table>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><i>&lt;EdgeName&gt;</i></td><td>String</td><td>Name of an edge defined in the 3D Modeler editor.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	EnterEdge(<*EdgeName*>)
Python Example	`oModule.EnterPoint("Edge_1")`

EnterLine

Enters a line defined in the 3D Modeler editor.

<table>
<tr><td><b>UI Access</b></td><td>Click <b>Geometry</b> and then select <b>Line</b></td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><i>&lt;LineName&gt;</i></td><td>String</td><td>Name of a line defined in the 3D Modeler editor.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	EnterLine(<*LineName*>)
Python Example	`oModule.EnterLine("Line1")`

EnterOutputVar

Enters Output Vars, only valid for Eigenmode problems.

<table>
<tr><td><b>UI Access</b></td><td>Click <b>Geometry</b> and then select <b>Output Vars</b>.</td></tr>
<tr><td><b>Parameters</b></td><td><table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td>&lt;<i>VarName</i>&gt;</td><td>String</td><td>Name of the output vars. Only 'freq' is supported.</td></tr>
<tr><td>&lt;<i>VarType</i>&gt;</td><td>String</td><td>Type of the output vars. 'Complex' type.</td></tr>
</table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	EnterOutputVar(<*VarName*>, <*VarType*>)
Python Example	`oModule.EnterOutputVar("Freq", "Complex")`

EnterPoint

Enters a point defined in the 3D Modeler editor.

<table>
<tr><td><b>UI Access</b></td><td>Click <b>Geometry</b> and then select <b>Point</b>.</td></tr>
<tr><td><b>Parameters</b></td><td><table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td>&lt;<i>PointName</i>&gt;</td><td>String</td><td>Name of a point defined in the 3D Modeler editor.</td></tr>
</table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	EnterPoint(<*PointName*>)
Python Example	`oModule.EnterPoint("Point1")`

EnterQty

Enters a field quantity.

<table>
<tr><td><b>UI Access</b></td><td>Click <b>Quantity</b>, and then select from the list.</td></tr>
<tr><td><b>Parameters</b></td><td><table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;FieldQty&gt;</i></td><td>String</td><td>The field quantity to be entered onto the stack.</td></tr>
</table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	EnterQty(<*FieldQty*>)
Python Example	`oModule.EnterQty("E")`

EnterScalar

Enters a scalar onto the stack.

<table>
<tr><td><b>UI Access</b></td><td>Click <b>Number</b> and then click <b>Scalar</b>. <b>Complex</b> option not selected.</td></tr>
<tr><td><b>Parameters</b></td><td><table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;Scalar&gt;</i></td><td>Double</td><td>The real number to enter onto the stack.</td></tr>
</table></td></tr>
</table>

Return Value	None.

Python Syntax	EnterScalar(<*Scalar*>)
Python Example	`oModule.EnterScalar(3.14159265358979)`

EnterScalarFunc

Enters a scalar function.

UI Access	Click **Function** and then select **Scalar**.
Parameters	Name / Type / Description <*VarName*> / String / Name of a variable to enter as a scalar function onto the stack.
Return Value	None.

Python Syntax	EnterScalarFunc(<*VarName*>)
Python Example	`oModule.EnterScalarFunc("Phase")`

EnterSurf

Enters a surface defined in the 3D Modeler editor.

UI Access	Click **Geometry** and then select **Surface**.

<table>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;<i>SurfName</i>&gt;</td><td>String</td><td>Name of a surface defined in the 3D Modeler editor.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	EnterSurf(<*SurfName*>)
Python Example	`oModule.EnterSurf("Rectangle1")`

EnterVector

Enters a vector onto the stack.

<table>
<tr><td><b>UI Access</b></td><td>Click <b>Number</b>, and then click <b>Vector</b>. <b>Complex</b> option not selected.</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;<i>VectorArray</i>&gt;</td><td>Array</td><td>Array of strings containing X, Y and Z components of the vector.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	EnterVector(<*VectorArray*>)
Python Example	`oModule.EnterVector([1.0, 1.0, 1.0])`

EnterVectorFunc

Enters a vector function.

<table>
<tr><td><b>UI Access</b></td><td>Click <b>Function</b> and then select <b>Vector</b>.</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;<i>VarNames</i>&gt;</td><td>Array</td><td>Array of strings containing names of a variable for the X, Y, and Z coordinates, respectively, to enter as a vector function on the stack.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	EnterVectorFunc(<*VarNames*>)
Python Example	`oModuleEnterVectorFunc(["X", "Y", "Z"])`

EnterVol

Enters a volume defined in the 3D Modeler editor.

<table>
<tr><td><b>UI Access</b></td><td>Click <b>Geometry</b> and then select <b>Volume</b>.</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;<i>VolumeName</i>&gt;</td><td>String</td><td>Name of a volume defined in the 3D Modeler editor.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	EnterVol(<*VolumeName*>)

Python Example	`oModule.EnterVol("Box1")`

ExportFieldPlot

Exports a field plot to a file.

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td>&lt;<i>PlotName</i>&gt;</td><td>String</td><td>Name of the field plot to export.</td></tr>
<tr><td>&lt;<i>ShowHeader</i>&gt;</td><td>Boolean</td><td><b>True -</b> export field plot header to a file.
<b>False -</b> export field plot.</td></tr>
<tr><td>&lt;<i>FileName</i>&gt;</td><td>String</td><td>Name of file to save as, including the file path.</td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	ExportFieldPlot(<*PlotName*>, <*ShowHeader*>, <*FileName*>)

Python Example	`oModule.ExportFieldPlot("Mag_E1",`
	`True, "C:/field_report.dsp")`

ExportMarkerTable

Exports the marker table to a .csv or .tab file.

<table>
<tr><td><b>UI Access</b></td><td>[product] &gt; <b>Fields</b> &gt; <b>Plot Fields</b> &gt; <b>Marker</b> &gt; <b>Export Marker Table</b>.</td></tr>
<tr><td><b>Parameters</b></td><td><table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;FileName&gt;</i></td><td>String</td><td>Name of export file include path.</td></tr>
</table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	ExportMarkerTable(<*FileName*>)
Python Example	`oModule.ExportMarkerTable("C:/work/FieldMarkerTable.csv")`

ExportOnGrid [Field Overlays]

Evaluates the top stack element at a set of points specified by a grid and exports the data to a file.

<table>
<tr><td><b>UI Access</b></td><td>Click <b>Export</b>, and then click <b>On Grid</b>.</td></tr>
<tr><td><b>Parameters</b></td><td><table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;OutputFile&gt;</i></td><td>String</td><td>Name of the output file.</td></tr>
<tr><td><i>&lt;Min&gt;</i></td><td>Array</td><td>Minimum values for the coordinate components of the grid system</td></tr>
<tr><td><i>&lt;Max&gt;</i></td><td>Array</td><td>Maximum values for the coordinate components of the grid system</td></tr>
<tr><td><i>&lt;Spacing&gt;</i></td><td>Array</td><td>Spacing values for the coordinate components of the grid system</td></tr>
<tr><td><i>&lt;SolnName&gt;</i></td><td>String</td><td>Name of the simulation setup</td></tr>
<tr><td><i>&lt;VarVals&gt;</i></td><td>Array</td><td>Array of strings containing setup definitions.</td></tr>
<tr><td><i>&lt;IncludePoints&gt;</i></td><td>Boolean</td><td>Optional. Specifies whether include points in the output file.</td></tr>
<tr><td><i>&lt;CSType&gt;</i></td><td>String</td><td>Optional. Type of coordinate system. "Cartesian" (default) | "Cylindrical" | "Spherical"</td></tr>
<tr><td><i>&lt;Offsets&gt;</i></td><td>Array</td><td>Optional. Origin for the offset coordinate system. For Cartesian, x, y, z, for Cylindrical, R, Phi, Z, for Spherical, Rho, Theta, Phi.</td></tr>
<tr><td><i>&lt;ByCount&gt;</i></td><td>Boolean</td><td>Optional.</td></tr>
</table></td></tr>
</table>

Return Value	None.

Note:Regarding the **ExportOnGrid** legacy script which only has “IncludePtInOutput” argument (the last Boolean one), AEDT can still read it and assign other new arguments as default values. Those default values are RefCSName = “global”, PtInSI = “True”, FieldInRefCS = “False”

Python Syntax	ExportOnGrid(<*OutputFile*>, <*Min*>, <*Max*>, <*Spacing*>, <*SolnName*>, <*SolnParameters*>, [<ExportOption>, "IncludePointsInOutput:=", <*boolean*>], " RefCSName:=", <*CSName*>, "PtsInSI:=, <boolean>, "FieldRefCS:=", <boolean>], "<CSType>", [<Offsets>, <ByCount>)

Python Example	`oModule.ExportOnGrid("C:\offset_grid_model_unit_ref.fld",`
	`        ["-1mm", "16mm", "0mm"],`
	`        ["1mm", "18mm", "1mm"],`
	`        ["2mm", "2mm", "1mm"],`
	`        "4500MHz : LastAdaptive",`
	`        ["Freq:=", "4.5GHz","Phase:="   , "0deg"],`
	`        [`
	`                "NAME:ExportOption",`
	`                "IncludePtInOutput:="   , True,`
	`                "RefCSName:="           , "offset",`
	`                "PtInSI:="              , False,`
	`                "FieldInRefCS:="        , True`
	`        ],`
	`        "Cartesian", ["0mm", "0mm", "0mm"], False`

	)

ExportPlotImageToFile

Deprecated. Use ExportPlotImageWithViewToFile.

Creates field plot exports of existing field plots from a given view points, and with the model being auto-sized automatically for each view.

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;FileName&gt;</i></td><td>String</td><td>Full path plus file name.</td></tr>
<tr><td><i>&lt;FolderName&gt;</i></td><td>String</td><td>Plot folder name.</td></tr>
<tr><td><i>&lt;ItemName&gt;</i></td><td>String</td><td>Name of fields to plot.</td></tr>
<tr><td><i>&lt;SetViewTopDownDirectionByRCS&gt;</i></td><td>String</td><td>Optional. Name of relative coordinate system to use for the field plot.</td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	ExportPlotImageToFile(<*FileName*>, <*FolderName*>, <*ItemName*>, <*SetViewTopDownDirectionByRCS*>)

Python Example	`oModule.ExportPlotImageToFile(`
	`"C:\TestEPITF2.jpg", "",`
	`"Mag_E2", "RelativeCS1")`

ExportPlotImageWithViewToFile [Reporter]

Exports a field plot image to a specified file.

> **Note:**
>
> This script replaces ExportPlotImageToFile, which is deprecated.

UI Access	**N/A**
Parameters	<table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;*FileName*&gt;</td><td>String</td><td>Full path of file to export.</td></tr><tr><td>&lt;*PlotQuantityName*&gt;</td><td>String</td><td>Name of quantity to plot.</td></tr><tr><td>&lt;*PlotItemName*&gt;</td><td>String</td><td>Name of fields to plot.</td></tr><tr><td>&lt;*PixelSizeX*&gt;</td><td>Integer</td><td>Desired image width, in pixels.</td></tr><tr><td>&lt;*PixelSizeY*&gt;</td><td>Integer</td><td>Desired image height, in pixels.</td></tr><tr><td>&lt;*ViewOrientation*&gt;</td><td>String</td><td>*Optional*. Name of orientation to use for plot.</td></tr></table>
Return Value	Image file is exported to the specified path.

Python Syntax	ExportPlotImageWithViewToFile(<*FileName*>, <*PlotQuantityName*>, <*PlotItemName*>, <*PixelSizeX*>, <*PixelSizeY*>, <*ViewOrientation*>)
Python Example	`oModule.ExportPlotImageWithViewToFile("E:/MyDir/OptimToutput/magE2.gif", "E Field", "Mag_E2", 1920, 1080, "newot2")`

ExportToFile

> **Note:**
>
> The ExportToFile script command has replaced the script command ExportReport. ExportReport remains in order to retain backward compatibility for existing scripts, but it is strongly recommended that you now use ExportToFile.

From a data table or plot, generates text format, comma delimited, tab delimited, or .dat type output files.

<table>
<tr><td><b>UI Access</b></td><td>Right-click on report name in the project tree and select <b>Export Data</b>.</td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;ReportName&gt;</i></td><td>String</td><td>Name of report to be exported.</td></tr>
<tr><td><i>&lt;FileName&gt;</i></td><td>String</td><td>Full path of the exported image file name; with extension of
.txt - Post processor format file
.csv - Comma-delimited data file
.tab - Tab-separated file
.dat - Ansys plot data file</td></tr>
<tr><td>&lt;UnitSpec&gt;</td><td>String</td><td>For example, “kV, Mhz, yd”</td></tr>
<tr><td>&lt;UseTraceNumberFormat&gt;</td><td>Boolean</td><td>"True", "False"</td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	ExportToFile(<*ReportName*>, <*FileName*>)

Python Example	oModule.ExportToFile("rETotal", "C:/Users/Documents/rETotal.csv")
	oModule.ExportToFile("S Parameter Table 1", "D:/Users/Documents/cftt.csv", False, " kV, MHz, yd ", True)

GetFieldFolderNames

Gets the folder names of the field plots.

UI Access	N/A
Parameters	None.
Return Value	Array of folder names.

Python Syntax	GetFieldFolderNames()
Python Example	`oModule.GetFieldFolderNames()`

GetFieldPlotNames

Gets the names of field overlay plots defined in a design.

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

Python Syntax	GetFieldPlotNames()
Python Example	`oModule.GetFieldPlotNames()`

GetFieldPlotQuantityName

Gets the quantity name of a specified field plot.

UI Access	N/A
Parameters	Name, Type, Description *<PlotName>*, String, Name of specified plot.
Return Value	String plot quantity name.

Python Syntax	GetFieldPlotQuantityName(<*PlotName*>)
Python Example	`oModule.GetFieldPlotQuantityName("Mag_H1")`

GetMeshPlotNames

Gets the names of mesh plots defined in a design.

UI Access	N/A
Parameters	None..

Return Value	Array of plot name.

Python Syntax	GetMeshPlotNames
Python Example	`oModule.GetMeshPlotNames()`

GetTopEntryValue

Gets the value of the top entry of the calculator stack.

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td><table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td>&lt;<i>SolnName</i>&gt;</td><td>String</td><td>Name of specified solution.</td></tr>
<tr><td>&lt;<i>VarVals</i>&gt;</td><td>Array</td><td>Array of variable name, value pairs.</td></tr>
</table></td></tr>
<tr><td><b>Return Value</b></td><td>Array of strings containing top entry values.</td></tr>
</table>

Python Syntax	GetTopEntryValue(<*SolnName*>, <*VarVals*>)

Python Example	`oModule.GetTopEntryValue(`
	`"Setup1:LastAdaptive",`
	`["Freq:=", "1GHz", "Phase:=", "0deg", "x_size:=", "2mm"]`
	`)`

LoadNamedExpressions

Loads a named expression definition from a saved file.

<table>
<tr><td><b>UI Access</b></td><td>In the Fields Calculator, click <b>Load From...</b> in the Library area.</td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;FileName&gt;</i></td><td>String</td><td>Filename and full path to the file to hold the named expression definition.</td></tr>
<tr><td><i>&lt;FieldType&gt;</i></td><td>String</td><td>For products with just one filed type, it is set to "Fields".</td></tr>
<tr><td><i>&lt;NamedExpr&gt;</i></td><td>Array</td><td>rray of strings containing the names of expression definitions to load from the file.</td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

<table>
<tr><td><b>Python Syntax</b></td><td>LoadNamedExpressions(&lt;<i>FileName</i>&gt;, &lt;<i>FieldType</i>&gt;, &lt;<i>NamedExpr</i>&gt;)</td></tr>
<tr><td><b>Python Example</b></td><td><pre>oModule.LoadNamedExpressions(

"C:\Ansoft\PersonalLib\smth.clc",

"Fields", ["smoothedtemp"])</pre></td></tr>
</table>

ModifyFieldPlot

Modifies a plot definition.

<table>
<tr><td><b>UI Access</b></td><td>[product] &gt; <b>Fields</b> &gt; <b>Modify Plot</b></td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
</table>
</td></tr>
</table>

	<OriginalName>	String	Name of the plot to be modified.
	<PlotParams>	Array	Array containing modified settings.
Return Value	None.		

Python Syntax	ModifyFieldPlot(<*OriginalName*>, <*PlotParams*>)

Python Example

```
oModule.ModifyFieldPlot("Vector_E1" ,
  ["NAME:Vector_E2",
    "SolutionName:=", "Setup1 : LastAdaptive",
    "QuantityName:=", "Vector_E",
    "PlotFolder:=", "E Field1",
    "UserSpecifyName:=", 0,
    "UserSpecifyFolder:=", 0,
    "IntrinsicVar:=","Freq='1GHz' Phase='30deg'",
    "PlotGeomInfo:=", [1, "Surface","FacesList", 1, "7"],
    "FilterBoxes:=", [0],
    ["NAME:PlotOnSurfaceSettings",
    "Filled:=", False,
    "IsoValType:=", "Fringe",
```

	"SmoothShade:=", True, "AddGrid:=", False, "MapTransparency:=", True, "Transparency:=", 0, _ "ArrowUniform:=", True, "ArrowSpacing:=", 0.100000001490116, "GridColor:=", [255, 255, 255]]])

ReassignFieldPlot

Reassigns a field plot.

<table>
<tr><td><b>UI Access</b></td><td>Right-click on a field plot > <b>Reassign</b>.</td></tr>
<tr><td><b>Parameters</b></td><td><table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;PlotName&gt;</i></td><td>String</td><td>Name of specified plot to reassign.</td></tr>
<tr><td><i>&lt;PlotParameterArray&gt;</i></td><td>Array</td><td>See CreateFieldPlot</td></tr>
</table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	ReassignFieldPlot(<*PlotName*>, <*PlotParameterArray*>)
Python Example	oModule.ReassignFieldPlot "Mag_H1", Array("NAME:Mag_H1", _

```
"SolutionName:=", "Setup1 : LastAdaptive", _
"UserSpecifyName:=", 0, _
"UserSpecifyFolder:=", 0, _
"QuantityName:=", "Mag_H", _
"PlotFolder:=", "H Field", _
"StreamlinePlot:=", False,
"AdjacentSidePlot:=", False,
"FullModelPlot:=", False,
"IntrinsicVar:=", "Freq=\'20GHz\' Phase=\'0deg\'",
"PlotGeomInfo:=", [1,"Surface","FacesList",1,"117"],
"FilterBoxes:=", [0],
["NAME:PlotOnSurfaceSettings",
  "Filled:=", False,
  "IsoValType:=", "Fringe",
  "AddGrid:=", False,
  "MapTransparency:=", True,
  "Refinement:=", 0,
  "Transparency:=", 0,
  "SmoothingLevel:=", 0,
  "ShadingType:=", 0,
  ["NAME:Arrow3DSpacingSettings",
    "ArrowUniform:=", True,
    "ArrowSpacing:=", 0,
    "MinArrowSpacing:=", 0,
    "MaxArrowSpacing:=", 0
  ],
  "GridColor:=", [255,255,255]
],
"EnableGaussianSmoothing:=", False
])
```

RenameFieldPlot

Renames a plot.

<table>
<tr><td><b>UI Access</b></td><td>Right-click the plot you want to rename in the project tree, and then click <b>Rename</b> on the shortcut menu.</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;<i>OldName</i>&gt;</td><td>String</td><td>Original name of the plot.</td></tr><tr><td>&lt;<i>NewName</i>&gt;</td><td>String</td><td>New name of the plot.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	RenameFieldPlot(<*OldName*>, <*NewName*>)

Python Example	`oModule.RenameFieldPlot(`
	`"Vector_E1", "Vector_E2")`

RenamePlotFolder

Renames a plot folder.

<table>
<tr><td><b>UI Access</b></td><td>Right-click a plot folder in the project tree, and then click <b>Rename</b> on the shortcut menu.</td></tr>
<tr><td><b>Parameters</b></td><td><table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td>&lt;<i>OldName</i>&gt;</td><td>String</td><td>Original name of the folder.</td></tr>
<tr><td>&lt;<i>NewName</i>&gt;</td><td>String</td><td>New name of the folder.</td></tr>
</table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	RenamePlotFolder(<*OldName*>, <*NewName*>)

Python Example	`oModule.RenamePlotFolder(`
	`"E Field", "Surface Plots")`

SaveFieldsPlots

Saves field plot(s) to a file.

UI Access	**HFSS** > **Fields** > **Save As...**

<table>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td>*<PlotNames>*</td><td>Array</td><td>Array of plot names.</td></tr>
<tr><td>*<FileName>*</td><td>String</td><td>Name of the file to save as, including the file path.</td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	SaveFieldsPlots(<*PlotNames*>, <*FileName*>)

Python Example	`oModule.SaveFieldsPlots(["Mag_E1", "Mag_E2"],`
	`"C:/field_report.dsp")`

SaveNamedExpressions

Saves a named expression definition to a file.

<table>
<tr><td><b>UI Access</b></td><td>In the Fields Calculator, click <b>Save To...</b> in the Library area.</td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td>*<FileName>*</td><td>String</td><td>Filename and full path to the file to hold the named expression definition.</td></tr>
<tr><td><NamedExprs></td><td>Array</td><td>Array of strings containing the names of expression definitions to load from the file.</td></tr>
<tr><td>*<Overwrite>*</td><td>Boolean</td><td>Specifies whether to overwrite the file.</td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	SaveNamedExpressions(<*FileName*>, <*NamedExprs*>, <*Overwrite*>)
Python Example	`oModule.SaveNamedExpressions(`

	`"C:\Ansoft\PersonalLib\smth.clc",`
	`["smoothedtemp"], True)`

SetFieldPlotSettings

Sets plot attributes.

<table>
<tr><td><b>UI Access</b></td><td>[product] > <b>Fields</b> > <b>Modify Plot Attributes</b></td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;PlotName&gt;</i></td><td>String</td><td>Name of the plot to modify.</td></tr>
<tr><td><i>&lt;PlotItemAttributes&gt;</i></td><td>Array</td><td>Structured array.
<code>Array("NAME:FieldsPlotItemSettings",</code>
<code>&lt;PlotOnPointsSettings&gt;,</code>
<code>&lt;PlotOnLineSettings&gt;,</code>
<code>&lt;PlotOnSurfaceSettings&gt;,</code>
<code>&lt;PlotOnVolumeSettings&gt;)</code>
<i>See description of</i> CreateFieldPlot command for details.</td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	SetFieldPlotSettings(<*PlotName*> <*PlotItemAttributes*>)
Python Example	`oModule.SetFieldPlotSettings("Mag_E2",`

	["NAME:FieldsPlotItemSettings", ["NAME:PlotOnLineSettings", ["NAME:LineSettingsID", "Width:=", 4, "Style:=", "Cylinder"), "IsoValType:=", "Tone", "ArrowUniform:=", True, "NumofArrow:=", 100), ["NAME:PlotOnSurfaceSettings", "Filled:=", False, "IsoValType:=", "Tone", "SmoothShade:=", True, "AddGrid:=", False, "MapTransparency:=", True, "Transparency:=", 0, "ArrowUniform:=", True, "ArrowSpacing:=", 0.100000001490116, "GridColor:=", [255, 255, 255]]])

SetPlotFolderSettings

Sets the attributes of all plots in the specified folder.

<table>
<tr><td><b>UI Access</b></td><td>[product] &gt; <b>Fields</b> &gt; <b>Modify Plot Attributes</b></td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td><i>&lt;PlotFolderName&gt;</i></td><td>String</td><td>Name of the folder with the attributes to modify.</td></tr>
<tr><td><i>&lt;PlotFolderAttributes&gt;</i></td><td>Array</td><td>Structured array.
Array("NAME:FieldsPlotSettings",
"Real time mode:=", <bool>,
<ColorMapSettings>,
<Scale3DSettings>,
<Marker3DSettings>,
<Arrow3DSettings&gt;)</td></tr>
<tr><td><i>&lt;ColorMapSettings&gt;</i></td><td>Array</td><td>Structured array.
Array("NAME:ColorMapSettings",
"ColorMapType:=", <string Possible values are "Uniform", "Ramp", "Spectrum">,
"SpectrumType:=", <string Possible values are "Rainbow", "Temperature", "Magenta", "Gray">,
"UniformColor:=", <array containing the R, G, B components of the color. Components should be in the range 0 to 255.>,
"RampColor:=", <array containing the R, G, B com-</td></tr>
</table>
</td></tr>
</table>

		ponents of the color. Components should be in the range 0 to 255.>)
<Scale3DSettings>	Array	Structured array. Array("NAME:Scale3DSettings", "m_nLevels:=", <int>, "m_autoScale:=", <bool>, "minvalue:=", <double>, "maxvalue:=", <double>, "log:=", <bool>, "IntrinsicMin:=", <double>, "IntrinsicMax:=", <double>)
<Marker3DSettings>	Array	Structured array. Array("NAME:Marker3DSettings", "MarkerType:=", <integer 9:Sphere 10: Box 11: Tetrahedron 12: Octahedron default: Sphere>, "MarkerMapSize:=", <bool>, "MarkerMapColor:=", <bool>, "MarkerSize:=", <double>)
<Arrow3DSettings>	Array	Structured array.

			Array("NAME:Arrow3DSettings", "ArrowType:=", <integer 0: Line 1: Cylinder 2: Umbrella default: Line>, "ArrowMapSize:=", <bool>, "ArrowMapColor:=", <bool>, "ShowArrowTail:=", <bool>, "ArrowSize:=", <double>)
Return Value	None.		

Python Syntax	SetPlotFolderSettings(<*PlotFolderName*>, <*PlotFolderAttributes*>)
Python Example	oModule.SetPlotFolderSettings("E Field1", ["NAME:FieldsPlotSettings", "Real time mode:=", True, ["NAME:ColorMapSettings", "ColorMapType:=", "Spectrum", "SpectrumType:=", "Rainbow",

```
"UniformColor:=", [127, 255, 255],
"RampColor:=", [255, 127, 127]],
["NAME:Scale3DSettings",
"m_nLevels:=", 27,
"m_autoScale:=", True,
"minvalue:=", 9.34379863739014,
"maxvalue:=", 13683.755859375,
"log:=", False,
"IntrinsicMin:=", 9.34379863739014,
"IntrinsicMax:=", 13683.755859375),
["NAME:Marker3DSettings",
"MarkerType:=", 0,
"MarkerMapSize:=", True,
"MarkerMapColor:=", False,
"MarkerSize:=", 0.25],
["NAME:Arrow3DSettings",
"ArrowType:=", 1,
"ArrowMapSize:=", True,
"ArrowMapColor:=", True,
"ShowArrowTail:=", True,
"ArrowSize:=", 0.25]]
```

	)

UpdateAllFieldsPlots

Updates the All Fields Plots.

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

Python Syntax	UpdateAllFieldsPlots()
Python Example	`oModule.UpdateAllFieldsPlots()`

16 - Fields Summary Script Commands

Fields Summary commands should be executed by the Field Overlays module, which is called "FieldsReporter" in scripts.

Set oModule = oDesign.GetModule("FieldsReporter")

oModule.CommandName <args>

EditFieldsSummarySetting

ExportFieldsSummary

EditFieldsSummarySetting

Creates a fields summary report in a Mechanical design (Thermal or Structural solutions).

<table>
<tr><td><b>UI Access</b></td><td><b>Mechanical</b> > <b>Fields</b> > <b>Create Fields Summary</b></td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;Calculation&gt;</i></td><td>list</td><td>"<i>&lt;Entity Type&gt;</i>", "<i>&lt;Geometry Type&gt;</i>", "<i>&lt;Selected Geometry&gt;</i>", "<i>&lt;Selected Quantity&gt;</i>", " <i>&lt;Normal&gt;</i>", "<i>&lt;Side&gt;</i>"</td></tr>
<tr><td><i>&lt;Entity Type&gt;</i></td><td>string</td><td>"Boundary" or "Object"</td></tr>
<tr><td><i>&lt;Geometry Type&gt;</i></td><td>string</td><td>"Surface" or "Volume"</td></tr>
<tr><td><i>&lt;Selected Geometry&gt;</i></td><td>string</td><td>Name of selected geometry</td></tr>
<tr><td><i>&lt;Selected Quantity&gt;</i></td><td>string</td><td>Name of selected quantity</td></tr>
<tr><td><i>&lt;Normal&gt;</i></td><td>string</td><td>Coordinate values of normal direction (for example, "0.00, 0.00, 1.00"), empty string ("") when not applicable</td></tr>
<tr><td><i>&lt;Side&gt;</i></td><td>string</td><td>"Default", "Adjacent", or "Combined"</td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>None</td></tr>
</table>

Python Syntax	EditFieldsSummarySetting(<*Calculation*>, <*Calculation*>, ... , <*Calculation*>)

Python Example (Structural)

```
oModule.EditFieldsSummarySetting(
   [
      "Calculation:=" , ["Object", "Volume", "EndCap", "Equivalent Stress", "", "Default"],
      "Calculation:=" , ["Object", "Volume", "EndCap_1", "Equivalent Stress", "", "Default"],
      "Calculation:=" , ["Object", "Volume", "ResistorBody", "Equivalent Stress", "", "Default"],
```

<table>
<tr><td></td><td><pre>
      "Calculation:=" , ["Object", "Volume", "Solder", "Equivalent Stress", "", "Default"],
      "Calculation:=" , ["Object", "Volume", "Solder_1", "Equivalent Stress", "", "Default"],
      "Calculation:=" , ["Object", "Volume", "Wire", "Equivalent Stress", "", "Default"],
      "Calculation:=" , ["Object", "Volume", "Wire_1", "Equivalent Stress", "", "Default"]
   ])
</pre></td></tr>
</table>

ExportFieldsSummary

Exports a fields summary report as a CSV file.

<table>
<tr><td><b>UI Access</b></td><td><b>Mechanical</b> &gt; <b>Fields</b> &gt; <b>Create Fields Summary</b> &gt; [<b>Apply and Export |</b> <b>Export</b>]</td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td>&lt;<i>SolutionName</i>&gt;</td><td>string</td><td>Setup name : SteadyState or Setup name : Transient</td></tr>
<tr><td>&lt;<i>DesignVariationKey</i>&gt;</td><td>string</td><td>Nominal</td></tr>
<tr><td>&lt;<i>ExportFileName</i>&gt;</td><td>string</td><td>File path to and name of CSV file</td></tr>
<tr><td>&lt;<i>IntrinsicValue</i>&gt;</td><td>string</td><td>Intrinsic variable</td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>None</td></tr>
</table>

<table>
<tr><td><b>Python Syntax</b></td><td>ExportFieldsSummary(&lt;<i>SolutionName</i>&gt;, &lt;<i>DesignVariationKey</i>&gt;, &lt;<i>ExportFileName</i>&gt;, &lt;<i>IntrinsicValue</i>&gt;)</td></tr>
<tr><td><b>Python Example</b></td><td><pre>
oModule.ExportFieldsSummary(
   [
      "SolutionName:="   , "Setup1 : Solution",
      "DesignVariationKey:=" , "Nominal",
      "ExportFileName:="   , "D:\\Documents\\Ansoft\\Structural\\CSV\\SummaryReport.csv",
      "IntrinsicValue:="   , ""
   ])
</pre></td></tr>
</table>

17 - User Defined Document Script Commands

The product has to implement the GetModule call to create the UserDefinedDocument scripting object (e.g., Check AltraSimDesign.cpp (function GetMgrIDispatch()). To access the UserDefinedDocuments scripting object, use:

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

Once you have the scripting object, you can use the following methods:

- AddDocument
- EditDocument
- RenameDocument
- DeleteDocument
- UpdateDocument
- ViewHtmlDocument
- ViewPdfDocument
- SaveHtmlDocumentAs
- SavePdfDocumentAs
- GetDocumentDefinitionNames
- DeleteAllDocuments
- UpdateAllDocuments

You can find examples of how to use these methods on each of the methods' pages. A complete example of a Python script is also available, as is an example with a line by line explanation.

AddDocument

Creates a new document based on provided data and traces. The document names come from UserDefinedDocument folder under syslib, userlib, and personallib. This creates a document and places it in the Project Manager under **Results** > **Documents**.

<table>
<tr><th>UI Access</th><th>Right click on <b>Results</b> > <b>Create Document</b>. Choose a document name.</th></tr>
<tr><td><b>Parameters</b></td><td><table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td>&lt;data&gt;</td><td>Array</td><td>Data the defines the document.</td></tr>
<tr><td>&lt;traces&gt;</td><td>Array</td><td>trace data for the inputs in the document.</td></tr>
</table></td></tr>
<tr><td><b>Return Value</b></td><td>None</td></tr>
</table>

Python Syntax	AddDocument (<*data*>, <*traces*>)
Python Example	oModule.AddDocument(["NAME:Design Summary", "", "SysLib", "DesignSummary", ["NAME:Inputs"]], ["NAME:DocTraces"])

DeleteAllDocuments

Deletes all documents in the object.

UI Access	Right click on **Documents** in the Project Manager and click **Delete All Documents**.
Parameters	None.
Return Value	None.

Python Syntax	DeleteAllDocuments()
Python Example	`oModule.DeleteAllDocuments()`

DeleteDocument

Deletes a specified document.

<table>
<tr><td><b>UI Access</b></td><td>Right click on the created document in the Project Manager under <b>Results</b> > <b>Documents</b> and click <b>Delete</b>.</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;name&gt;</td><td>String</td><td>Name of the document to be deleted.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	DeleteDocument(<*names*>)
Python Example	`oModule.DeleteDocument("Project Design Summary")`

EditDocument

Edits specified documents. If the document pops up a dialog box, the user can make a change the inputs for the document. The document is regenerated and updated. A new one is *not* created.

<table>
<tr><td><b>UI Access</b></td><td>Right click on the created document in the Project Manager under <b>Results</b> &gt; <b>Documents</b> and click <b>Modify document</b>.</td></tr>
<tr><td><b>Parameters</b></td><td><table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td>&lt;originalName&gt;</td><td>String</td><td>Name of the original document</td></tr>
<tr><td>&lt;modifiedData&gt;</td><td>Array</td><td>New data to add to or modify the document</td></tr>
<tr><td>&lt;modifiedraces&gt;</td><td>Array</td><td>Trace data for the inputs of the document</td></tr>
</table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	EditDocument(<*name*>,<*data*>,<*traces*>)

Python Example	`oModule.EditDocument("Design Summary",`
	`[`
	`   "NAME:Design Summary",`
	`   "",`
	`   "SysLib",`
	`   "DesignSummary",`
	`   ["NAME:Inputs"]`
	`],`

	`["NAME:DocTraces"]`
	`)`

GetDocumentDefinitionNames

Document definition names are the list of names that can be used to create a document. They appear when you click on **Create document**. This method returns the filenames of document definitions according to the files in the installation directories:

- syslib/UserDefinedDocuments
- userlib/UserDefinedDocuments
- personalllb/UserDefinedDocuments

UI Access	NA
Parameters	Name | Type | Description <separator> | String | Separator used to convey the directory "level"
Return Value	None.

Python Syntax	GetDocumentDefinitionNames(<*separator*>)
Python Example	`oModule.GetDocumentDefinitionNames("/")`

GetDocumentNames

Retrieves the names for all documents.

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

Python Syntax	GetDocumentNames()
Python Example	`oModule.GetDocumentNames()`

RenameDocument

Changes the name of a document.

<table>
<tr><td><b>UI Access</b></td><td>Right click on the created document in the Project Manager under <b>Results</b>&gt; <b>Documents</b> and click <b>Rename</b>.</td></tr>
<tr><td><b>Parameters</b></td><td><table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td>&lt;oldName&gt;</td><td>String</td><td>Current name of the document</td></tr>
<tr><td>&lt;newName&gt;</td><td>String</td><td>New name of the document</td></tr>
</table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	RenameDocument(<*oldName*>, <*newName*>)
Python Example	`oModule.RenameDocument("Design Summary", "Project Design Summary")`

SaveHtmlDocumentAs

Saves a pre-existing HTML file to a different name and/or location.

<table>
<tr><td><b>UI Access</b></td><td>Right click on the created document in the Project Manager under <b>Results</b> > <b>Documents</b> and click <b>Save As</b> > <b>HTML</b>.</td></tr>
<tr><td><b>Parameters</b></td><td><table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td>&lt;<i>name</i>&gt;</td><td>String</td><td>Name of the document to be saved.</td></tr>
<tr><td>&lt;<i>saveTo</i>&gt;</td><td>String</td><td>File path to save the document to.</td></tr>
</table></td></tr>
<tr><td><b>Return Value</b></td><td>none</td></tr>
</table>

Python Syntax	SaveHtmlDocumentAs(<*name*>, <*saveTo*>)
Python Example	`oModule.SaveHtmlDocumentAs("Design Summary 1", "DS1.html")`

SavePdfDocumentAs

Saves a pre-existing PDF file to a different name and/or location.

<table>
<tr><td><b>UI Access</b></td><td>Right click on the created document in the Project Manager under <b>Results</b> > <b>Documents</b> and click <b>Save As</b> > <b>PDF</b>.</td></tr>
<tr><td><b>Parameters</b></td><td><table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td>&lt;<i>name</i>&gt;</td><td>String</td><td>Name of the document to be saved.</td></tr>
<tr><td>&lt;<i>saveTo</i>&gt;</td><td>String</td><td>File path to save the document at.</td></tr>
</table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	SavePdfDocumentAs(<*name*>, <*saveTo*>)
Python Example	`oModule.SavePdfDocumentAs("Design Summary 1", "DS1.pdf")`

UpdateAllDocuments

Refreshes the contents of all created documents. This action is made on the folder rather than the individual document.

UI Access	Right click on **Results** > **Documents** in the Project Manager and click **Update All Documents**.
Parameters	None.
Return Value	None.

Python Syntax	UpdateAllDocuments()
Python Example	`oModule.UpdateAllDocuments()`

UpdateDocument

Refreshes the contents of the selected document.

UI Access	Right click on the created document in the Project Manager under **Results** > **Documents** and click **Update Document**.

Parameters	Name	Type	Description
	<name>	String	Name of the document to be updated
Return Value	None.		

Python Syntax	UpdateDocument(<*name*>)
Python Example	`oModule.UpdateDocument("Test UDD Report")`

ViewHtmlDocument

Displays a pre-existing document as HTML.

UI Access	Right click on the created document in the Project Manager under **Results** > **Documents** and click **View Xml Document**.		
Parameters	Name	Type	Description
	<name>	String	Name of the document to be viewed as a HTML
Return Value	None		

Python Syntax	ViewHtmlDocument(<*name*>)
Python Example	`oModule.ViewHtmlDocument("Design Summary 1")`

ViewPdfDocument

Displays a pre-existing document as a PDF file.

<table>
<tr><td><b>UI Access</b></td><td>Right click on the created document in the Project Manager under <b>Results</b> &gt; <b>Documents</b> and click <b>View PDF Document</b>.</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;<i>name</i>&gt;</td><td>String</td><td>Name of the document to be viewed as a PDF</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	ViewPdfDocument(<*name*>)
Python Example	`oModule.ViewPdfDocument("Design Summary 1")`

Example Python Script: Defining a Document

This script creates a user-defined solution and a document.

```
import ScriptEnv
ScriptEnv.Initialize("Ansoft.ElectronicsDesktop")
oDesktop.RestoreWindow()
oProject = oDesktop.SetActiveProject("BJT_Inverter")
oDesign = oProject.SetActiveDesign("Nexxim1")
oModule = oDesign.GetModule("UserDefinedSolutionModule")

oModule.CreateUserDefinedSolution("UDS Distance Trace Arithmetic Result1", "SysLib",
"TraceArithmetic/Distance Sweep Trace Arithmetic",
       [
              "Offset 1:=", "0",
              "Scale 1:=", "1",
              "Offset 2:=", "0",
              "Scale 2:=", "1",
              "Operation:=", "Add"
       ],
       [
              [
                     "Standard",
                     "probe1",
                     "Transient",
                     [
                            style="font-family: monospace;">"NAME:Context",
                            style="font-family: monospace;">"SimValueContext:=", [1,0,2,0,False,False,-
1,1,0,1,1,"",0,0,"DE",False,"0","DP",False,"500000000","DT",False,"0.001","NUMLEVELS",False,"0",-
"WE",False,"10us","WM",False,"10us","WN",False,"0ns","WS",False,"0ns"]
                     ],
                     [
                            "Time:=", ["All"]
                     ],
                     [
                            "Probe Component:=", ["V(Port1)"]
                     ],
                     []
              ],
              [
                     "Standard",
                     "probe2",
                     "Transient",
                     [
                            "NAME:Context",
                            "SimValueContext:=", [1,0,2,0,False,False,-
1,1,0,1,1,"",0,0,"DE",False,"0","DP",False,"500000000","DT",False,"0.001","NUMLEVELS",False,"0","-
WE",False,"10us","WM",False,"10us","WN",False,"0ns","WS",False,"0ns"]
                        ],
                        [
                                "Time:=", ["All"]
                        ],
                        [
                                "Probe Component:=", ["V(Port1)"]
                        ],
                        []
                ]
        ],
        [])
oModule = oDesign.GetModule("UserDefinedDocuments")
oModule.AddDocument(
        [
                "NAME:Test Report",
                "Test Report",
                "SysLib",
                "TestUDDInputs",
                [
                        "NAME:Inputs",
                        [
                                "NAME:UDS1",
                                "Solution",
                                "UDS Distance Trace Arithmetic Result1",
                                1000000,
                                0
                        ],
                        [
                                "NAME:UDS2",
                                "Solution",
                        "UDS Distance Trace Arithmetic Result2",
                        1000000,
                        2
                ]
        ]
],
[
        "NAME:DocTraces",
        [
                "NAME:UDS1",
                [
                        "User Defined",
                        "",
                        "UDS Distance Trace Arithmetic Result1",
                        [
                                "Context:=", ""
                        ],
                        [
                                "Distance:=", ["All"]
                        ],
                        [
                                "Probe Component:=", [""]
                        ],
                        []
                ]
        ],
        [
                "NAME:UDS2",
                [
                        "User Defined",
                        "",
                        "UDS Distance Trace Arithmetic Result1",
                        [
                                "Context:=", ""
                                ],
                                [
                                        "Distance:=", ["All"]
                                ],
                                [
                                        "Probe Component:=", [""]
                                ],
                        []
                ]
        ]
])
```

18 - User Defined Solutions Commands

User Defined Solution commands should be executed by the "UserDefinedSolutionModule" module.

```
oDesign = oProject.SetActiveDesign("TestDesign1")

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

The list of commands is as follows:

CreateUserDefinedSolution

Creates a new user defined solution.

<table>
<tr><td><b>UI Access</b></td><td>Right-click on <b>Results</b> > <b>Create User Defined Solution</b>.</td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;SoluName&gt;</i></td><td>String</td><td>Name of user defined solution.</td></tr>
<tr><td><i>&lt;LibType&gt;</i></td><td>String</td><td>Indicates the library where the UDS plugin file is located. This parameter must be one of the following values: "SysLib", "UserLib", "PersonalLib".</td></tr>
<tr><td><i>&lt;RelativePath&gt;</i></td><td>String</td><td>The path of the UDS plugin file relative to the "UserDefinedOutputs" subdirectory of the library specified by <i>&lt;LibType&gt;</i>.</td></tr>
<tr><td><i>&lt;PropList&gt;</i></td><td>Array</td><td>Strings specify name-value pairs corresponding to the UDS properties specified in the plugin file.
<code>For example:</code>
<code>Array("multiply_factor:=", "2.0", "component_name:=", "resistor1")</code></td></tr>
<tr><td><i>&lt;ProbeSelections&gt;</i></td><td>Array</td><td>Name of the probe being specified. Note: this must match a probe name spe-</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><td></td><td><table><tr><td></td><td></td><td>cified in the UDS plugin file.</td></tr><tr><td>&lt;DynamicProbes&gt;</td><td>Array</td><td>Array of <ProbeSelection>'s, representing the probes that are used by dynamic-probes.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>String name of created user defined solution.</td></tr>
</table>

Python Syntax	CreateUserDefinedSolution(<*SoluName*>, <*LibType*>, <*RelativePath*>, <*PropList*>, <*ProbeSelections*>, <*DynamicProbes*>)

Python Example	`oModule.CreateUserDefinedSolution(`
	`"ConstantTimestep1", "SysLib",`
	`"ConstantTimestep", [], [], [])`

DeleteUserDefinedSolutions

Deletes one or more user defined solutions.

<table>
<tr><td><b>UI Access</b></td><td><b>Delete</b> button from the <b>User Defined Solutions</b> dialog.</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;<i>SoluNames</i>&gt;</td><td>Array</td><td>Array of strings containing names of User Defined Solutions to be deleted.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	DeleteUserDefinedSolutions(<*SoluNames*>)
Python Example	`oModule.DeleteUserDefinedSolutions(["Solution1", "Solution2"])`

EditUserDefinedSolution

Modifies an existing user defined solution.

<table>
<tr><td><b>UI Access</b></td><td><b>Edit</b> button from the <b>User Defined Solutions</b> dialog box.</td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td>&lt;<i>SoluName</i>&gt;</td><td>String</td><td>Name of user defined solution to be edited.</td></tr>
<tr><td>&lt;<i>NewSoluName</i>&gt;</td><td>String</td><td>New name for the specified user defined solution.</td></tr>
<tr><td>&lt;<i>LibType</i>&gt;</td><td>String</td><td>Indicates the library where the UDS plugin file is located. This parameter must be one of the following values: "SysLib", "UserLib", "PersonalLib".</td></tr>
<tr><td>&lt;<i>RelativePath</i>&gt;</td><td>String</td><td>The path of the UDS plugin file relative to the "UserDefinedOutputs" sub-directory of the library specified by &lt;<i>LibType</i>&gt;.</td></tr>
<tr><td>&lt;<i>PropList</i>&gt;</td><td>Array</td><td>Strings specify name-value pairs corresponding to the UDS properties specified in the plugin file.
<code>For example:</code>
<code>Array("multiply_factor:=", "2.0", "component_name:=", "resistor1")</code></td></tr>
<tr><td>&lt;<i>ProbeSelections</i>&gt;</td><td>Array</td><td>Name of the probe being specified. Note: this must match a probe name specified in the UDS plugin file.</td></tr>
<tr><td>&lt;DynamicProbes&gt;</td><td>Array</td><td>Array of <ProbeSelection>'s, representing the probes that are used by dynamic-probes.</td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>String name of update user defined solution.</td></tr>
</table>

Python Syntax	EditUserDefinedSolution(<*SoluName*>, <*NewSoluName*>, <*LibType*>, <*RelativePath*>, <*PropList*>, <*ProbeSelections*>, <*DynamicProbes*>)

Python Example	`oModule.EditUserDefinedSolution("ConstantTimestep1"`
	`"ConstantTimestep1After", "SysLib",`
	`"ConstantTimestep", [], [], [])`

GetUserDefinedSolutionNames

Retrieves user defined solution names.

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

Python Syntax	GetUserDefinedSolutionNames()
Python Example	`oModule.GetUserDefinedSolutionNames()`

GetUserDefinedSolutionProperties

Obtains properties for a specified user defined solution.

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;<i>SoluName</i>&gt;</td><td>String</td><td>Name of a specified user defined solution.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>Array of strings containing properties values.</td></tr>
</table>

Python Syntax	GetUserDefinedSolutionProperties(<*SoluName*>)
Python Example	`oModule.GetUserDefinedSolutionProperties("ConstantTimestep1")`

This page intentionally
left blank.

19 - Network Data Explorer Script Commands

Network Data Explorer (NDE) scripting uses three objects, each with unique script commands:

- **Network Data Explorer** – top-level object obtained by calling oNDE=oDesktop.GetTool("ndExplorer"). Network Data Explorer commands are called using `oNDE`.
- **Network Data** – single set of S-parameters, corresponding to a single entry in the UI tree. Network Data commands are called using `oData`.
- **Post Process Settings** – settings that can be applied and removed from network data without making permanent changes to the underlying data. Post Process Settings commands are called using `oPostProc`.

Examples using the above objects:

```
oNDE = oDesktop.GetTool("ndExplorer")

oData = oNDE.Open("D:\folder\test.s2p")

success = oPostProc.AddDiffPair(2, 1, "Diff1", "Comm1", 100, 25)
```

The following commands are described in this section:

DisableDiffPairs

EnableDiffPairs

ExportCitiFile

ExportFullWaveSpice

ExportMatlab

ExportNMFData

ExportNetworkData

ExportSpreadsheet

ExportTouchstone

ExportTouchstone2

Extract (SPISim)

GetFrequencies

GetFrequencyCount

GetName

GetPortCount

GetPortNumber

GetPostProcSettings

GetSolutionVariation

GetVariation

HasSameData

> **Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.**

AddDiffPair

Specifies a differential pair from two terminal ports.

UI Access	From the **Network Data Explorer** tab, select **Differential Pairs** in the **NDE** ribbon to open the **Differential**

<table>
<tr><td></td><td><b>Pairs</b> window. Then select differential pairs from the <b>Pairs</b> group box and click <b>Add Pairs >></b>.</td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td><i>&lt;positiveTerminal&gt;</i></td><td>Integer</td><td>The portNumber of the positive terminal.</td></tr>
<tr><td><i>&lt;negativeTerminal&gt;</i></td><td>Integer</td><td>The portNumber of the negative terminal.</td></tr>
<tr><td><i>&lt;diffName&gt;</i></td><td>String</td><td>The new name of the differential pin (e.g., Diff1).</td></tr>
<tr><td><i>&lt;commName&gt;</i></td><td>String</td><td>The new name of the common pin (e.g., Comm1).</td></tr>
<tr><td><i>&lt;diffImpedance&gt;</i></td><td>Double</td><td>The differential pin characteristic impedance.</td></tr>
<tr><td><i>&lt;commImpedance&gt;</i></td><td>Double</td><td>The common pin characteristic impedance.</td></tr>
<tr><td><i>&lt;matched&gt;</i></td><td>Boolean</td><td>Specify whether to use matched pair.
<b>Note:</b> The default value is <b>False</b>.</td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>Boolean.</td></tr>
</table>

Python Syntax	AddDiffPair()
Python Example	`success = oPostProc.AddDiffPair(2, 1, "Diff1", "Comm1", 100, 25)`

Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.

Cascade (SPISim)

Creates new network data from a group of network data objects cascaded together. The network data must have an even number of terminal ports and must have the same number of ports.

> **Note:** Cascade opens and interacts with **SPISim** to complete.

UI Access	From the **Network Data Explorer** tab, select **Transforms** in the **NDE** ribbon. Then select **Cascade** from the drop-down menu.
Parameters	<table><tr><th>Name</th><th>Type</th><th>Description</th></tr><tr><td><em>&lt;dataArray&gt;</em></td><td>Array</td><td>These network data objects are cascaded together to create the new network data.</td></tr></table>
Return Value	Network IDispatch. **Note:** Cascade returns **None** if the specified network data does not have compatible terminal ports defined.

Python Syntax	Cascade()

Python Example	`oNDE = oDesktop.GetTool("ndExplorer")`
	`oData2 = oNDE.Cascade([oData1, oData2, oData3])`

> **Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.**

ClearDiffPairs

Clears all differential pair definitions. All other post-processing settings remain the same.

UI Access	From the **Network Data Explorer** tab, select **Differential Pairs** in the **NDE** ribbon to open the **Differential Pairs** window. Then select differential pairs from the rightmost box and click **<< Remove Pairs**.
Parameters	None.
Return Value	None.

Python Syntax	ClearDiffPairs()

Python Example	`oNDE = oDesktop.GetTool("ndExplorer")`
	`oPostProc.ClearDiffPairs()`

Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.

Clone

Creates a copy of the network data object.

UI Access	From the **Network Data Explorer** tab, select the network data object to clone. Then select **Clone** from the **NDE** ribbon.
Parameters	None.
Return Value	Network IDispatch.

Python Syntax	Clone()

Python Example	`oNDE = oDesktop.GetTool("ndExplorer")`
	`oData1 = oData.Clone()`

> **Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.**

Close

Closes the network data object. The object will no longer be accessible.

UI Access	From the **Network Data Explorer** tab, click **Close** in the **NDE** ribbon, or select **File** > **Close**.
Parameters	None.
Return Value	None.

Python Syntax	Close()

Python Example	`oNDE = oDesktop.GetTool("ndExplorer")`
	`oData.Close()`

> **Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.**

Combine (SPISim)

Combines frequencies from multiple network data objects to create a new network data with all of the frequencies.

> **Note:** Combine opens and interacts with **SPISim** to complete.

<table>
<tr><td><b>UI Access</b></td><td>From the <b>Network Data Explorer</b> tab, select <b>Transforms</b> in the <b>NDE</b> ribbon. Then select <b>Combine</b> from the drop-down menu.</td></tr>
<tr><td><b>Parameters</b></td><td><table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td><i>&lt;dataArray&gt;</i></td><td>Array</td><td>These network data objects are combined to create the new network data.</td></tr>
<tr><td><i>&lt;freqTol&gt;</i></td><td>Double</td><td>If abs(f1 – f2) < freqTol, f1 and f2 are considered the same frequency.
<b>Note:</b> The default value is <b>100</b>.</td></tr>
</table></td></tr>
<tr><td><b>Return Value</b></td><td>Network IDispatch.
<b>Note:</b> Combine returns <b>None</b> if the network data does not have the same number of ports.</td></tr>
</table>

<table>
<tr><td><b>Python Syntax</b></td><td>Combine()</td></tr>
<tr><td><b>Python Example</b></td><td><code>oNDE = oDesktop.GetTool("ndExplorer")</code>
<code>oData4 = oNDE.Combine([oData1, oData2, oData3], 1000)</code></td></tr>
</table>

> **Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.**

Deembed (SPISim)

Creates a new network data, dembedding the frontData and the backData from the front and back of the originalData.

> **Note:** Deembed opens and interacts with **SPISim** to complete.

<table>
<tr><td><b>UI Access</b></td><td>From the <b>Network Data Explorer</b> tab, select <b>Transforms</b> in the <b>NDE</b> ribbon. Then select <b>Deembed</b> from the drop-down menu to open the <b>Deembed...</b> window, and choose <b>Given 3 S-params in order of: Total, Front, Back</b> from the <b>Settings</b> group box.</td></tr>
<tr><td><b>Parameters</b></td><td><table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td><i>&lt;originalData&gt;</i></td><td>Object</td><td>Original network data that is deembeded.</td></tr>
<tr><td><i>&lt;frontData&gt;</i></td><td>Object</td><td>Network data that is deembeded from the front of the original.</td></tr>
<tr><td><i>&lt;backData&gt;</i></td><td>Object</td><td>Network data that is deembeded from the back of the original.</td></tr>
</table></td></tr>
<tr><td><b>Return Value</b></td><td>Network IDispatch.
<b>Note:</b> Deembed returns <b>None</b> if the specified network data does not have compatible terminal ports defined.</td></tr>
</table>

Python Syntax	Deembed()
Python Example	`oNDE = oDesktop.GetTool("ndExplorer")`

	`oData2 = oNDE.Deembed(oData1, oDataFront, oDataBack)`

> **Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.**

DeembedBack (SPISim)

Creates a new network data, dembedding the backData from the back of the originalData.

> **Note:** DeembedBack opens and interacts with **SPISim** to complete.

<table>
<tr><td><b>UI Access</b></td><td>From the <b>Network Data Explorer</b> tab, select <b>Transforms</b> in the <b>NDE</b> ribbon. Then select <b>Deembed</b> from the drop-down menu to open the <b>Deembed...</b> window, and choose <b>Given 3 S-params in order of: Total, Back</b> from the <b>Settings</b> group box.</td></tr>
<tr><td><b>Parameters</b></td><td><table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td><i>&lt;originalData&gt;</i></td><td>Object</td><td>Original network data that is deembeded.</td></tr>
<tr><td><i>&lt;backData&gt;</i></td><td>Object</td><td>Network data that is deembeded from the back of the original.</td></tr>
</table></td></tr>
<tr><td><b>Return Value</b></td><td>Network IDispatch.
<b>Note:</b> DeembedBack returns <b>None</b> if the specified network data does not have compatible terminal ports defined.</td></tr>
</table>

Python Syntax	DeembedBack()

Python Example	`oNDE = oDesktop.GetTool("ndExplorer")`
	`oData2 = oNDE.DeembedBack(oData1, oDataBack)`

> **Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.**

DeembedFront (SPISim)

Creates a new network data, dembedding the frontData from the front of the originalData.

> **Note:** DeembedFront opens and interacts with **SPISim** to complete.

UI Access	From the **Network Data Explorer** tab, select **Transforms** in the **NDE** ribbon. Then select **Deembed** from the drop-down menu to open the **Deembed...** window, and choose **Given 3 S-params in order of: Total, Front** from the **Settings** group box.
Parameters	
Return Value	Network IDispatch. **Note:** DeembedFront returns **None** if the specified network data does not have compatible terminal ports defined.

Name	Type	Description
<originalData>	Object	Original network data that is deembeded.
<frontData>	Object	Network Data that is deembeded from the front of the original.

Python Syntax	DeembedFront()

Python Example	`oNDE = oDesktop.GetTool("ndExplorer")`
	`oData2 = oNDE.DeembedFront(oData1, oDataFront)`

> **Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.**

DisableDiffPairs

Deactivates all differential pair definitions. This script does not remove them, so they can be reactivated.

UI Access	From the **Network Data Explorer** tab, select **Differential Pairs** in the **NDE** ribbon to open the **Differential Pairs** window. Once differential pairs have been added to the rightmost box (i.e., select differential pairs from the Pairs group box and click **Add Pairs >>**), remove check marks from the **Enabled** column to deactivate the corresponding differential pairs.
Parameters	None.
Return Value	None.

Python Syntax	DisableDiffPairs()

Python Example	`oNDE = oDesktop.GetTool("ndExplorer")`
	`oPostProc.DisableDiffPairs()`

> **Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.**

EnableDiffPairs

Enables all differential pair definitions.

UI Access	From the **Network Data Explorer** tab, select **Differential Pairs** in the **NDE** ribbon to open the **Differential Pairs** window. Once differential pairs have been added to the rightmost box (i.e., select differential pairs from the Pairs group box and click **Add Pairs >>**), add check marks to the **Enabled** column to activate the corresponding differential pairs.
Parameters	None.
Return Value	None.

Python Syntax	EnableDiffPairs()

Python Example	`oNDE = oDesktop.GetTool("ndExplorer")`
	`success = oPostProc.EnableDiffPairs()`

> **Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.**

ExportCitiFile

Writes the S-parameters to disk in Citifile format (*.cit text file).

UI Access	From the **Network Data Explorer** tab, select **File** > **Save As** to open a **Save As** explorer window. Then select **Citifile (*.cit)** from the **Save as type:** drop-down menu.

Parameters

Name	**Type**	**Description**
<filePath>	String	The full path to the file.
<matrixType>	Integer	One of the following: • 0 – S-Parameters • 1 – Y-Parameters • 2 – Z-Parameters • 3 – Gamma • 4 – Impedance **Note:** The default value is **0**.
<formalType>	Integer	One of the following: • 0 – Magnitude/Angle (degrees) • 1 – Real/Imaginary • 2 – dB/Angle (degrees) **Note:** The default value is **0**.
<precision>	Integer	The number of significant figures. **Note:** The default value is **10**.
<frequencies>	Array of Doubles	A subset of the calculated frequencies. Any array values that don’t have calculated data will be ignored. An empty array will cause all frequencies to be written.

	Note: The default value is an empty array.
Return Value	Boolean True if file was successfully written; else False.

Python Syntax	ExportCitiFile()

Python Example	`oNDE = oDesktop.GetTool("ndExplorer")`
	`success = oData.ExportCitiFile("D:\folder\ne133.cit", 0, 1, 12, [])`

ExportFullWaveSpice

Use: Export FullWaveSpice data in a format of your choice.

Command: File > Export MacroModel > Broadband (SYZ, FWS….)

Syntax: ExportFullWaveSpice

```
    "DesignName", // Design name. Can be left blank, if loading solution from a file.
    true/false, // true - solution loaded from file, false- loaded from design
    "Name", // If loading from design this is the solution name, else this is the
            // full path of the file from which the solution is loaded
    "variation", // Pick a particular variation. Leave blank if no variation.
    Array("NAME:Frequencies"), // Optional; if none defined all frequencies are used
    Array("NAME:SpiceData", // Spice export options object
    "SpiceType:=", "SSS", // SpiceType can be "PSpice", "HSpice", "Spectre", "SSS",
                   // "Simplorer", "TouchStone1.0", "TouchStone2.0"
"EnforcePassivity:=", false, // Enforce Passivity true/false
"EnforceCausality:=", false, // Enforce Causality true/false
"UseCommonGround:=", false, // Use common ground true/false
"FittingError:=", 0.5, // Fitting error
"MaxPoles:=", 400, // Maximum Order
"PassivityType:=", "ConvexOptimization", // Passivity Type can be "ConvexOptimization",
                        // "PassivityByPerturbation", or "IteratedFittingOfPV"
"ColumnFittingType:=", "Column", // Column FittingType can be "Column", "Entry", "Matrix"
"SSFittingType:=", "TWA", // SS Fitting Type can be "TWA", "IterativeRational"
"RelativeErrorToleranc:=", false, // Relative error tolerance true/false
"TouchstoneFormat:=", "MA", // Touchstone Format "MA", "RI", "DB"
"TouchstoneUnits:=", "Hz", // Touchstone Units "Hz", "KHz", "MHz", "MHz"
"TouchStonePrecision:=", 8, // Touchstone precision
"ExportDirectory:=", "C:/Examples/LNA/", // Directory to export to
"ExportSpiceFileName:=", "Linckt_HBTest_2.sss", // Spice export file
"FullwaveSpiceFileName:=", "Linckt_HBTest.sss", // FWS file
"CreateNPortModel:=", true // Create a model based on the exported file true/false
)
```

> **Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.**

ExportMatlab

Writes the S-parameters to disk in Matlab format (*.m text file).

<table>
<tr><td><b>UI Access</b></td><td>From the <b>Network Data Explorer</b> tab, select <b>File</b> &gt; <b>Save As</b> to open a <b>Save As</b> explorer window. Then select <b>MATLAB (*.m)</b> from the <b>Save as type:</b> drop-down menu.</td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td><i>&lt;filePath&gt;</i></td><td>String</td><td>The full path to the file.</td></tr>
<tr><td><i>&lt;matrixType&gt;</i></td><td>Integer</td><td>One of the following:
• 0 – S-Parameters
• 1 – Y-Parameters
• 2 – Z-Parameters
• 3 – Gamma
• 4 – Impedance
<b>Note:</b> The default value is <b>0</b>.</td></tr>
<tr><td><i>&lt;formalType&gt;</i></td><td>Integer</td><td>One of the following:
• 0 – Magnitude/Angle (degrees)
• 1 – Real/Imaginary
• 2 – dB/Angle (degrees)
<b>Note:</b> The default value is <b>0</b>.</td></tr>
</table>
</td></tr>
</table>

	<precision>	Integer	The number of significant figures. **Note:** The default value is **10**.
	<frequencies>	Array of Doubles	A subset of the calculated frequencies. Any array values that don’t have calculated data will be ignored. An empty array will cause all frequencies to be written. **Note:** The default value is an empty array.
Return Value	Boolean True if file was successfully written; else False.		

Python Syntax	ExportMatlab()

Python Example	`oNDE = oDesktop.GetTool("ndExplorer")`
	`success = oData.ExportMatlab("D:\folder\ne133.m", 0, 1, 12, [])`

ExportNMFData

ExportNetworkData

Exports matrix solution data to a file.

UI Access	N/A		
Parameters	Name	Type	Description

<DesignVariationKey>	String	Design variation key. Pass empty string for the current nominal variation.
<SolnSelectionArray>	Array	Array of selected solutions. `Array(<SolnSelector>, <SolnSelector>, ...)` If more than one array entry, this indicates a combined Interpolating sweep.
<SolnSelector>	String	Solution setup name and solution name, separated by a colon.
<FileFormat>	Integer	File format value. 2 : Tab delimited spreadsheet format (.tab) 3 : Touchstone (.sNp) 4 : CitiFile (.cit) 7 : Matlab (.m) 8 : Terminal Z0 spreadsheet
<OutFile>	String	Full path to the file to write out.
<FreqsArray>	Array	The frequencies to export. The <FreqsArray> argument contains a vector (e.g. "1GHz", "2GHz", ...) to use, or "all". To export all frequencies, use Array("all"). If no frequencies are specified, all frequencies are used.
<DoRenorm>	Boolean	For Touchstone format only. Specifies whether to renormalize the data before export.
<RenormImped>	Double	For Touchstone format only. Real impedance value in ohms, for renormalization. Required in syntax, but ignored if DoRenorm is false.
<DataType>	Array	Optional. Type: "S", "Y", or "Z". The matrix to export.
<pass>	Integer	Optional. The pass to export. This is ignored if the sourceName is a frequency sweep. Leaving out this value or specifying -1 gets all passes.
<ComplexFormat>	Integer	Optional. Type: "0", "1", or "2"

		The format to use for the exported data. 0 = Magnitude/Phase. 1= Real/Imaginary. 2= db/Phase.
<*Precision*>	Integer	Optional. Touchstone number of digits precision. Default if not specified is 15.
<*UseExportFreqs*>	Boolean	Specifies whether to use export frequencies.
<*IncludeGammaComments*>	Boolean	Touchstone only. Specifies whether to include Gamma and Impedance comments.
<*SupportNonStdExport*>	Boolean	Specifies whether to support non-standard Touchstone extensions for mixed reference impedance.

Return Value	None.

Python Syntax	ExportNetworkData(<*DesignVariationKey*>, <*SolnSelectionArray*>, <*SolnSelector*>, <*FileFormat*>, <*OutFile*>, <*FreqsArray*>, <*DoRenorm*>, <*RenormImped*>, [Optional <*DataType*>], [Optional <*pass*>], [Optional <*ComplexFormat*>], [Optional <*Precision*>], [Optional <*UseExportFreqs*>], [Optional <*IncludeGammaComments*>], [Optional <*SupportNonStdExport*>])
Python Example	

Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.

ExportSpreadsheet

Writes the S-parameters to disk in spreadsheet format (*.tab text file).

<table>
<tr><td><b>UI Access</b></td><td>From the <b>Network Data Explorer</b> tab, select <b>File</b> &gt; <b>Save As</b> to open a <b>Save As</b> explorer window. Then select <b>Data Table (spreadsheet) (*.tab)</b> from the <b>Save as type:</b> drop-down menu.</td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td><i>&lt;filePath&gt;</i></td><td>String</td><td>The full path to the file.</td></tr>
<tr><td><i>&lt;matrixType&gt;</i></td><td>Integer</td><td>One of the following:
• 0 – S-Parameters
• 1 – Y-Parameters
• 2 – Z-Parameters
• 3 – Gamma
• 4 – Impedance
<b>Note:</b> The default value is <b>0</b>.</td></tr>
<tr><td><i>&lt;formalType&gt;</i></td><td>Integer</td><td>One of the following:
• 0 – Magnitude/Angle (degrees)
• 1 – Real/Imaginary
• 2 – dB/Angle (degrees)
<b>Note:</b> The default value is <b>0</b>.</td></tr>
<tr><td><i>&lt;precision&gt;</i></td><td>Integer</td><td>The number of significant figures.
<b>Note:</b> The default value is <b>10</b>.</td></tr>
</table>
</td></tr>
</table>

	<frequencies>	Array of Doubles	A subset of the calculated frequencies. Any array values that don’t have calculated data will be ignored. An empty array will cause all frequencies to be written. **Note:** The default value is an empty array.
	<renormalize>	Boolean	If **True**, will renormalize the data to the value of renormImpedance before writing. **Note:** The default value is **False**.
	<renormImpedance>	Double	Data will be renormalized to this impedance before writing only if renormalize is **True**. **Note:** The default value is **50 ohms**.
Return Value	Boolean True if file was successfully written; else False.		

Python Syntax	ExportSpreadsheet()

Python Example	`oNDE = oDesktop.GetTool("ndExplorer")`
	`oData = oNDE.Open(<model path>)`
	`success = oData.ExportSpreadsheet("D:\folder\ne133.tab", 0, 1, 12, [], True, 23)`

> **Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.**

ExportTouchstone

Writes the S-parameters to disk in Touchstone 1.0 format (*.s*n*p text file).

<table>
<tr><td><b>UI Access</b></td><td>From the <b>Network Data Explorer</b> tab, select <b>File</b> &gt; <b>Save As</b> to open a <b>Save As</b> explorer window. Then select <b>Touchstone Format 1.0 (*.s<i>n</i>p)</b> from the <b>Save as type:</b> drop-down menu.</td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td><i>&lt;filePath&gt;</i></td><td>String</td><td>The full path to the file.</td></tr>
<tr><td><i>&lt;matrixType&gt;</i></td><td>Integer</td><td>One of the following:
• 0 – S-Parameters
• 1 – Y-Parameters
• 2 – Z-Parameters
• 3 – Gamma
• 4 – Impedance
<b>Note:</b> The default value is <b>0</b>.</td></tr>
<tr><td><i>&lt;formalType&gt;</i></td><td>Integer</td><td>One of the following:
• 0 – Magnitude/Angle (degrees)
• 1 – Real/Imaginary
• 2 – dB/Angle (degrees)
<b>Note:</b> The default value is <b>0</b>.</td></tr>
</table>
</td></tr>
</table>

<precision>	Integer	The number of significant figures. **Note:** The default value is **10**.
<frequencies>	Array of Doubles	A subset of the calculated frequencies. Any array values that don’t have calculated data will be ignored. An empty array will cause all frequencies to be written. **Note:** The default value is an empty array.
<renormalize>	Boolean	If **True**, will renormalize the data to the value of renormImpedance before writing. **Note:** The default value is **False**.
<renormImpedance>	Double	Data will be renormalized to this impedance before writing only if renormalize is **True**. **Note:** The default value is **50 ohms**.
<freqUnits>	String	One of the following: "Hz", "KHz", "MHz", "GHz". **Note:** The default value is **"Hz"**.

Return Value	Boolean True if file was successfully written; else False.

Python Syntax	ExportTouchstone()

Python Example	`oNDE = oDesktop.GetTool("ndExplorer")`
	`success = oData.ExportTouchstone("D:\folder\ne133.s2p", 0, 1, 12, [], True, 23, "GHz")`

> **Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.**

ExportTouchstone2

Writes the S-parameters to disk in Touchstone 2 format (*.ts text file).

<table>
<tr><td><b>UI Access</b></td><td>From the <b>Network Data Explorer</b> tab, select <b>File</b> &gt; <b>Save As</b> to open a <b>Save As</b> explorer window. Then select <b>Touchstone 2 Format (*.ts)</b> from the <b>Save as type:</b> drop-down menu.</td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td><i>&lt;filePath&gt;</i></td><td>String</td><td>The full path to the file.</td></tr>
<tr><td><i>&lt;matrixType&gt;</i></td><td>Integer</td><td>One of the following:
• 0 – S-Parameters
• 1 – Y-Parameters
• 2 – Z-Parameters
• 3 – Gamma
• 4 – Impedance
<b>Note:</b> The default value is <b>0</b>.</td></tr>
<tr><td><i>&lt;formalType&gt;</i></td><td>Integer</td><td>One of the following:
• 0 – Magnitude/Angle (degrees)
• 1 – Real/Imaginary</td></tr>
</table>
</td></tr>
</table>

			- 2 – dB/Angle (degrees) **Note:** The default value is **0**.
	<precision>	Integer	The number of significant figures. **Note:** The default value is **10**.
	<frequencies>	Array of Doubles	A subset of the calculated frequencies. Any array values that don’t have calculated data will be ignored. An empty array will cause all frequencies to be written. **Note:** The default value is an empty array.
	<renormalize>	Boolean	If **True**, will renormalize the data to the value of renormImpedance before writing. **Note:** The default value is **False**.
	<renormImpedance>	Double	Data will be renormalized to this impedance before writing only if renormalize is **True**. **Note:** The default value is **50 ohms**.
	<freqUnits>	String	One of the following: "Hz", "KHz", "MHz", "GHz". **Note:** The default value is **"Hz"**.
Return Value	Boolean True if file was successfully written; else False.		

Python Syntax	ExportTouchstone2()

Python Example	`oNDE = oDesktop.GetTool("ndExplorer")`
	`success = oData.ExportTouchstone2("D:\folder\ne133.ts", 0, 1, 12, [], True, 23, "GHz")`

> **Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.**

Extract (SPISim)

Creates a new smaller network data after removing data for the specified terminal port numbers.

> **Note:** Extract opens and interacts with **SPISim** to complete.

UI Access	From the **Network Data Explorer** tab, select **Transforms** in the **NDE** ribbon. Then select **Extract** from the drop-down menu.
Parameters	
Return Value	Network IDispatch.

Name	Type	Description
<oData>	Object	Data from this object is copied to a new network data and the copy is transformed.
<portNumbers>	Array of Integers	The port numbers that will be retained (integers between 1 and *n*).

Python Syntax	Extract()

Python Example	`oNDE = oDesktop.GetTool("ndExplorer")`
	`oData2 = oNDE.Extract(oData1, [3, 2])`

Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.

GetFrequencies

Returns the frequency values with calculated data in the network data.

UI Access	None.
Parameters	None.
Return Value	Array of doubles.

Python Syntax	GetFrequencies()

Python Example	`oNDE = oDesktop.GetTool("ndExplorer")`
	`Freqs = oData.GetFrequencies()`

> **Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.**

GetFrequencyCount

Returns the number of frequencies with calculated data in the network data.

UI Access	None.
Parameters	None.
Return Value	Integer number of frequencies.

Python Syntax	GetFrequencyCount()

Python Example	`oData = oDesktop.GetTool("ndExplorer")`
	`numFreqs = oData.GetFrequencyCount()`

> **Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.**

GetName

Returns the name of the network data.

UI Access	None.

Parameters	None.
Return Value	String name.

Python Syntax	GetName()

Python Example	`oNDE = oDesktop.GetTool("ndExplorer")`
	`strName = oData.GetName()`

> **Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.**

GetPortCount

Returns the total number of ports.

UI Access	From the **Network Data Explorer** tab, select **Edit Ports** in the **NDE** ribbon to open the **Port properties** window.
Parameters	None.
Return Value	Integer number of ports.

Python Syntax	GetPortCount()

Python Example	`oNDE = oDesktop.GetTool("ndExplorer")`
	`num = oData.GetPortCount()`

> **Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.**

GetPortNumber

Returns the terminal port number for a given terminal port name.

> **Note:** GetPortNumber can be used in other commands that require a port number, if a port name is preferable.
>
> The PortNumber is the index of the name port in the original data and is not changed by any other settings applied to the oPostProc, including the Reorder settings. It is permanently linked to the port name and can be used interchangeably with the port name in many of the PostProcSettings script functions.

UI Access	From the **Network Data Explorer** tab, ensure the **Full Port Names** check box is activated. Port numbers will be represented in the object portNames.
Parameters	<table><tr><th>Name</th><th>Type</th><th>Description</th></tr><tr><td><i>&lt;portName&gt;</i></td><td>String</td><td>The designation of the port number.</td></tr></table>
Return Value	Integer.

Python Syntax	GetPortNumber()
Python Example	`oNDE = oDesktop.GetTool("ndExplorer")`

	`index = oPostProc.GetPortNumber("Input35")`

> **Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.**

GetPostProcSettings

Returns a copy of the IDispatch to the postprocessing settings for the network data (oPostProc).

> **Note:** Making changes to the PostProcSettings will not change the network data. To make changes, call oData.SetPostProcSettings to change the network data.

UI Access	None.
Parameters	None.
Return Value	PostProcSettings IDispatch

Python Syntax	GetPostProcSettings()

Python Example	`oNDE = oDesktop.GetTool("ndExplorer")`
	`oPostProc = oData.GetPostProcSettings()`

> **Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.**

GetSolutionVariation

Returns the network data object corresponding to a variation of the solution.

<table>
<tr><td><b>UI Access</b></td><td>None.</td></tr>
<tr><td><b>Parameters</b></td><td><table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td><i>&lt;solutionID&gt;</i></td><td>Integer</td><td>ID of the solution (obtained with LoadSolution).</td></tr>
<tr><td><i>&lt;variation&gt;</i></td><td>String</td><td>The variation key.</td></tr>
</table></td></tr>
<tr><td><b>Return Value</b></td><td>Network IDispatch.</td></tr>
</table>

Python Syntax	GetSolutionVariation()

Python Example	`oNDE = oDesktop.GetTool("ndExplorer")`
	`oData = oNDE.GetSolutionVariation("0, "offset='0.1'")`

> **Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.**

GetVariation

Returns the variation key for the network data.

UI Access	None.

Parameters	None.
Return Value	String variation key.

Python Syntax	GetVariation()

Python Example	`oNDE = oDesktop.GetTool("ndExplorer")`
	`variation = oData.GetVariation()`

> **Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.**

HasSameData

Compares one network data object with another network data. Actual calculated values will be compared and no interpolation will be done.

<table>
<tr><td><b>UI Access</b></td><td>None.</td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td><i>&lt;NetworkData&gt;</i></td><td>Object</td><td>The second network data to compare against.</td></tr>
<tr><td><i>&lt;matrixType&gt;</i></td><td>Integer</td><td>One of the following:
• 0 – S-Parameters
• 1 – Y-Parameters
• 2 – Z-Parameters</td></tr>
</table>
</td></tr>
</table>

			• 3 – Gamma • 4 – Impedance **Note:** The default value is **0**.
	<comparePortNames>	Boolean	If **True**, port names must be identical. **Note:** The default value is **True**.
	<compareNoise>	Boolean	If **True**, and both network data have noise data, the comparison will fail unless the noise values also match. **Note:** The default value is **10**.
	<relativeTolerance>	Double	If the absoluteTolerance test fails, then abs(value1 - value2)/max (abs(value1), abs(value2)) must be less than this to be considered equal. **Note:** The default value is **1e-14**.
	<absoluteTolerance>	Double	If **True**, then abs(value1 - value2) < absoluteTolerance. **Note:** The default value is **0**.
Return Value	Boolean True if the compared network data have the same frequency and matrix values; otherwise, False.		

Python Syntax	HasSameData()
Python Example	`oNDE = oDesktop.GetTool("ndExplorer")`

	`success = oData.HasSameData(oData2, 0, True, False, 1e-10, 0)`

> **Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.**

LoadSolution

Loads the specified design (all variations) and returns an integer ID.

<table>
<tr><td><b>UI Access</b></td><td>After analyzing a design, from the <b>Project Manager</b> window, expand the <b>Project Tree</b> and Analysis folders. Then right-click on the completed analysis icon and select <b>Network Data Explorer</b> to open the solution in the <b>Network Data Explorer</b> tab.</td></tr>
<tr><td><b>Parameters</b></td><td><table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td><i><projectName></i></td><td>String</td><td>Full path to the Electronics Desktop file.</td></tr>
<tr><td><i><designName></i></td><td>String</td><td>Name of the design.</td></tr>
<tr><td><i><solutionName></i></td><td>String</td><td>Name of the solution.</td></tr>
</table></td></tr>
<tr><td><b>Return Value</b></td><td>Integer ID (identifying the solution); < 0 if the solution is not loaded.</td></tr>
</table>

Python Syntax	LoadSolution()

Python Example	`oNDE = oDesktop.GetTool("ndExplorer")`
	`id = oNDE.LoadSolution("D:\folder\Tee.aedt", "HFSS_Test", "Setup1:Sweep1")`

> **Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.**

Open

Reads contents of file and returns the IDispatch for the new network data.

<table>
<tr><td><b>UI Access</b></td><td>From the <b>Network Data Explorer</b> tab, select <b>Open</b> in the <b>NDE</b> ribbon, or select <b>File</b> > <b>Open</b> to open an explorer window. Then navigate to the required S-parameter file.</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><th>Name</th><th>Type</th><th>Description</th></tr><tr><td><i><fileName></i></td><td>String</td><td>Full path to the file containing S-parameters.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>Network IDispatch.</td></tr>
</table>

Python Syntax	Open()

Python Example	`oNDE = oDesktop.GetTool("ndExplorer")`
	`oData = oNDE.Open("D:\folder\test.s2p")`

> **Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.**

Rename (SPISim)

Creates a new network data of the same size with the terminal ports renamed.

> **Note:** Rename opens and interacts with **SPISim** to complete.

UI Access	From the **Network Data Explorer** tab, select **Transforms** in the **NDE** ribbon. Then select **Rename** from the drop-down menu.
Parameters	<table><tr><th>Name</th><th>Type</th><th>Description</th></tr><tr><td><i>&lt;oData&gt;</i></td><td>Object</td><td>Data from this object is copied to a new network data and the copy is transformed.</td></tr><tr><td><i>&lt;portNames&gt;</i></td><td>Array of Strings</td><td>The new names (one for each port).</td></tr></table>
Return Value	Network IDispatch.

Python Syntax	Rename()

Python Example	`oNDE = oDesktop.GetTool("ndExplorer")`
	`oData2 = oNDE.Rename(oData1, ["input", "control", "output"])`

> **Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.**

Renormalize (SPISim)

Creates a new network data of the same size after renormalizing the terminal ports using the specified impedance values.

> **Note:** Rename opens and interacts with **SPISim** to complete.

UI Access	From the **Network Data Explorer** tab, select **Transforms** in the **NDE** ribbon. Then select **Renormalize** from the drop-down menu.
Parameters	<table><tr><th>Name</th><th>Type</th><th>Description</th></tr><tr><td><i>&lt;oData&gt;</i></td><td>Object</td><td>Data from this object is copied to a new network data and the copy is transformed.</td></tr><tr><td><i>&lt;portImpedances&gt;</i></td><td>Array of Doubles</td><td>The new impedance values to be applied to each port.</td></tr></table>
Return Value	Network IDispatch. **Note:** Renormalize returns **None** if there is not an impedance value for each port.

Python Syntax	Renormalize()

Python Example	`oNDE = oDesktop.GetTool("ndExplorer")`
	`oData2 = oNDE.Renormalize(oData1, [3, 2])`

> **Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.**

Reorder

Rearranges the terminal port entries but does not change the port names (i.e., switching first and third terminal port positions will give switched values for S(1,1) and S(3,3) but S(Port1,Port1) and S(Port3,Port3) do not change).

<table>
<tr><td><b>UI Access</b></td><td>From the <b>Network Data Explorer</b> tab, select <b>Edit Ports</b> in the <b>NDE</b> ribbon to open the <b>Port properties</b> window. <b>Click+drag</b> the terminal port rows to reorder them. Make changes, as necessary, then click <b>OK</b>.</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><th>Name</th><th>Type</th><th>Description</th></tr><tr><td><i>&lt;newOrder&gt;</i></td><td>Array of Integers</td><td>Must have one entry for each port; all numbers 1 to <i>n</i> must be present.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	Reorder()

Python Example	`oNDE = oDesktop.GetTool("ndExplorer")`
	`oPostProc.Reorder([3, 2, 1])`

Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.

Reorder (SPISim)

Creates a new network data with the same number of terminal ports and with the port names in the same order. The data is also reordered so it corresponds to the new order (e.g., the data for S(Port1, Port1) may correspond to S(Port3, Port3).

> **Note:** Reorder opens and interacts with **SPISim** to complete.

UI Access	From the **Network Data Explorer** tab, select **Transforms** in the **NDE** ribbon. Then select **Reorder** from the drop-down menu.
Parameters	<table><tr><th>Name</th><th>Type</th><th>Description</th></tr><tr><td>*<oData>*</td><td>Object</td><td>Data from this object is copied to a new network data and the copy is transformed.</td></tr><tr><td>*<portNumbers>*</td><td>Array of Integers</td><td>The port numbers in new order (integers between 1 and *n*).</td></tr></table>
Return Value	Network IDispatch. **Note:** Reorder returns **None** if the array argument does not contain all the terminal port numbers, in any order.

Python Syntax	Reorder()

Python Example	`oNDE = oDesktop.GetTool("ndExplorer")`
	`oData2 = oNDE.Reorder([3, 2, 1])`

> **Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.**

Reset

Resets the post-processing to what it was when the network data was created.

> **Note:** This is not equivalent to setting post-processing to an empty state. The network data may have been read from a file or created from solution data that already had post-processing settings.
>
> Resetting the oPostProcSetting object does not change any data; it only changes the settings object. If you want to change a data object to apply the changed settings, you must follow the Reset function call with oData.SetPostProcSettings (oPostProc)

UI Access	From the **Network Data Explorer** tab, select **Reset postprocessing** in the **NDE** ribbon.
Parameters	None.
Return Value	None.

Python Syntax	Reset()

Python Example	`oNDE = oDesktop.GetTool("ndExplorer")`
	`oPostProc.Reset()`

> **Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.**

SetAllPortImpedances

Sets all terminal port impedances in a single call. The impedances are in an array of real or complex values.

UI Access	Through the UI, the terminal ports can **only** be edited individually. From the **Network Data Explorer** tab, select **Edit Ports** in the **NDE** ribbon to open the **Port properties** window. Make changes, then click **OK**.
Parameters	<table><tr><th>Name</th><th>Type</th><th>Description</th></tr><tr><td><i>&lt;impedances&gt;</i></td><td>Array of Doubles or String</td><td>Must have one entry for each port or a single complex (or real) value which will be applied to all ports.</td></tr></table>
Return Value	None.

Python Syntax	SetAllPortImpedances()

Python Example	`oNDE = oDesktop.GetTool("ndExplorer")`
	`oPostProc.SetAllPortImpedances([23, "2+3i", 50])` **-or-** `oData.SetAllPortImpedances(50)`

> **Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.**

SetPortDeembedDistance

Sets terminal port impedance for a single port.

UI Access	From the **Network Data Explorer** tab, select **Edit Ports** in the **NDE** ribbon to open the **Port properties** win-

	dow. Make changes, then click **OK**.
Parameters	<table><tr><th>Name</th><th>Type</th><th>Description</th></tr><tr><td><em>&lt;portNumber&gt;</em></td><td>Integer</td><td>The port number of the changed terminal port (integers between 1 and <strong>n</strong>) .</td></tr><tr><td><em>&lt;distance&gt;</em></td><td>String</td><td>Impedance with units (e.g., “20mm”); if no units, meters are assumed</td></tr></table>
Return Value	None.

Python Syntax	SetPortDeembedDistance()

Python Example	`oNDE = oDesktop.GetTool("ndExplorer")`
	`oPostProc.SetPortDeembedDistance(2, "20mm")`

> **Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.**

SetPortImpedance

Sets port impedance for a single port.

UI Access	From the **Network Data Explorer** tab, select **Edit Ports** in the **NDE** ribbon to open the **Port properties** window. Make changes, then click **OK**.
Parameters	<table><tr><th>Name</th><th>Type</th><th>Description</th></tr></table>

<table>
<tr><td></td><td><table>
<tr><td><i>&lt;portNumber&gt;</i></td><td>Integer</td><td>The port number of the changed terminal port (integers between 1 and <i>n</i>).</td></tr>
<tr><td><i>&lt;impedance&gt;</i></td><td>Double or String</td><td>Double if impedance value is real; string format (e.g., 24+10i) for complex impedance.</td></tr>
</table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	SetPortImpedance()

Python Example	`oNDE = oDesktop.GetTool("ndExplorer")`
	`oPostProc.SetPortImpedance(2, "25+3i")`

> **Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.**

SetPostProcSettings

Applies postprocessing settings (oPostProc) to the specified network data.

> **Note:** Making changes to the PostProcSettings will not change the network data. To make changes, call oData.SetPostProcSettings to change the network data.

<table>
<tr><td><b>UI Access</b></td><td>None.</td></tr>
<tr><td><b>Parameters</b></td><td><table>
<tr><td><b>Name</b></td><td><b>Type</b></td><td><b>Description</b></td></tr>
<tr><td><i>&lt;PostProcSettings&gt;</i></td><td>Object</td><td>The settings that will be applied to the data.</td></tr>
</table></td></tr>
</table>

Return Value	None.

Python Syntax	SetPostProcSettings()

Python Example	`oNDE = oDesktop.GetTool("ndExplorer")`
	`oData.SetPostProcSettings(oPostProc)`

> **Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.**

Smooth

Creates a new network data with values smoothed. The number of adjacent points smoothed is user-specified.

<table>
<tr><td><b>UI Access</b></td><td>From the <b>Network Data Explorer</b> tab, select <b>Transforms</b> in the <b>NDE</b> ribbon. Then select <b>Smooth</b> from the drop-down menu.</td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td><i>&lt;oData&gt;</i></td><td>Object</td><td>Data from this object is copied to a new network data and the copy is transformed.</td></tr>
<tr><td><i>&lt;order&gt;</i></td><td>Integer</td><td>The number of adjacent points that are smoothed.
<b>Note:</b> The default value is <b>10</b>.</td></tr>
<tr><td><i>&lt;enforceCausality&gt;</i></td><td>Boolean</td><td>Casual data is calculated before smoothing.</td></tr>
</table>
</td></tr>
</table>

	<table><tr><td></td><td></td><td>**Note:** The default value is **False**.</td></tr></table>
Return Value	Network IDispatch **Note:** Network IDispatch is returned only if the Smooth function was successfully able to perform the smoothing operation on the supplied data and meet the maximum 0.025 error tolerance. Otherwise it will return a < null object >. The user may want to try changing other input parameters, for example, `enforceCausality.`

Python Syntax	Smooth()

Python Example	`oNDE = oDesktop.GetTool("ndExplorer")`
	`oData2 = oNDE.Smooth(oData1, 3, True)`

> **Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.**

Stretch (SPISim)

Creates a new network data representing a matrix of transmission lines with the length of those transmission lines multiplied by a factor of *n*.

> **Note:** Stretch opens and interacts with **SPISim** to complete.

UI Access	From the **Network Data Explorer** tab, select **Transforms** in the **NDE** ribbon. Then select **Stretch** from the drop-down menu.
Parameters	<table><tr><th>Name</th><th>Type</th><th>Description</th></tr><tr><td><i>&lt;oData&gt;</i></td><td>Object</td><td>Data from this object is copied to a new network data and the copy is transformed.</td></tr><tr><td><i>&lt;factor&gt;</i></td><td>Double</td><td>The transmission line lengths are multiplied by this number.</td></tr></table>
Return Value	Network IDispatch. **Note:** Stretch returns **None** if the network data argument does not represent a matrix of transmission lines.

Python Syntax	Stretch()

Python Example	`oNDE = oDesktop.GetTool("ndExplorer")`
	`oData2 = oNDE.Stretch(oData1, 3.1)`

> **Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.**

Terminate

Creates a new network data with specified ports terminated.

<table>
<tr><td><b>UI Access</b></td><td>From the <b>Network Data Explorer</b> tab, select <b>Transforms</b> in the <b>NDE</b> ribbon. Then select <b>Terminate</b> from the drop-down menu.</td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td><i>&lt;oData&gt;</i></td><td>Object</td><td>Data from this object is copied to a new network data and the copy is transformed.</td></tr>
<tr><td><i>&lt;portNumbers&gt;</i></td><td>Array of Integers</td><td>The port numbers that are terminated (integers between 1 and <i>n</i>).</td></tr>
<tr><td><i>&lt;termImpedances&gt;</i></td><td>Array of Complex Numbers</td><td>The impedance values used to terminate the specified ports.</td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>Network IDispatch.
<b>Note:</b> Terminate returns <b>None</b> if the port numbers are not valid or there are no impedance value for the port numbers.</td></tr>
</table>

Python Syntax	Terminate()

Python Example	`oNDE = oDesktop.GetTool("ndExplorer")`
	`oData2 = oNDE.Terminate(oData1, [2, 3], [23, "10+2i"])`

This page intentionally
left blank.

20 - CompInstance Script Commands

CompInstance commands should be executed by the **oDesign2** or **oPropHost2** object.

```
Set oDesign2 = CompInstance.GetParentDesign
```

```
Set oPropHost2 = CompInstance.GetPropHost
```

Callback Scripting Using CompInstance Object

Callback scripts are scripts that can be set in the Property Dialog for individual properties by clicking the button in the Callback column and choosing a script that is saved with the project. Callback scripts can contain any legal script commands including general Ansys script function calls (e.g., GetApplicationName()). In addition, Callback scripts can also call functions on a special object named CompInstance.

You can obtain an interface to a CompInstance in a schematic or layout by calling oEditor.GetCompInstanceFromRefDes(refDes). For more information see Layout Scripting and Schematic Scripting. This interface is also available as a CompInstance object in CompInstance event callbacks, such as placing a component in a layout or schematic.

Definitions

<**propName**> = text string

<**value**> = double

<**valueText**> = text string

<**fileName**> = full path file name

<**choices**> = string containing menu choices separated by commas

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

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

<**bool**> is 1 for true or 0 for false

<**editorName**> is either "Layout" or "SchematicEditor"

The topics for this section include:

CompInstance Functions

CompInstance Functions

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

The topics for this section include:

GetComponentName

GetEditor

GetInstanceID

GetInstanceName

GetParentDesign

GetPropHost

GetPropServerName

GetComponentName

Returns the name of the component corresponding to this CompInstance.

UI Access	NA
Parameters	Name | Type | Description

	None
Return Value	String name of component (e.g. MS_TRL) and stores it in "name"

Python Syntax	GetComponentName()
Python Example	`name = CompInstance.GetComponentName()`

GetInstanceID [Component Instance]

Use: Returns the instanceID of the CompInstance.

Command: None

Syntax: GetInstanceID()

Return Value: String

Python Syntax	GetInstanceID()
Python Example	`id = CompInstance.GetInstanceID()`

GetInstanceName [Component Instance]

Use: Returns the instance name of the component corresponding to this CompInstance.

Command: None

Syntax: GetInstanceName()

Return Value: String

Returns instanceName (e.g. A7) of compInstance and stores it in "name".

Note that the Instance Name is not the same as the RefDes.

GetParentDesign

Use: Returns an interface to the compInstance's parent design. This interface can be used to call Design functions. See: Design Object Script Commands.

Command: None

Syntax: GetParentDesign()

Return Value: Returns interface to design.

Python Syntax	GetParentDesign()
Python Example	`oDesign2 = CompInstance.GetParentDesign()`

GetPropServerName

Use: Returns the PropServerName of the Component corresponding to this CompInstance.

Command: None

Syntax: GetPropServerName()

Return Value: String

Python Syntax	GetPropServerName()

Python Example	`name = CompInstance.GetPropServerName()`

This page intentionally
left blank.

21 - Definition Manager Script Commands

The definition manager controls the use of materials and scripts in a project. It also provides access to the managers for symbols, footprints, padstacks, and components in a project.

Set oProject = oDesktop.SetActiveProject("Project1")

Set oDefinitionManager = oProject.GetDefinitionManager()

The topics for this section include:

AddMaterial

CloneMaterial

DoesMaterialExist

EditMaterial

ExportMaterial

RemoveMaterial

Related Topics:

Component Manager Script Commands

Material Manager Script Commands

Model Manager Script Commands

Network Data Explorer Manager Script Commands

Script and Library Scripts

Symbol Manager Script Commands

Add [component manager]

*Use:*Add a component

Command: Tools > Edit Configured Libraries > Components > Add Component

*Syntax:*Add Array("NAME:<ComponentName>",

```
"Info:=", <ComponentInfo>,
"RefBase:=", <string>, // reference designator
"NumParts:=", <int>, // parts per component
"OriginalComponent:=", <string>
"Terminal:=", <TerminalInfo>,
"Terminal:=", <TerminalInfo>, ...
// The remaining parameters are optional
Array("NAME:Parameters", // any combo of the following
"VariableProp:=", <VariableInfo>,
"CheckboxProp:=", <CheckBoxInfo>,
"ButtonProp:=", <ButtonInfo>,
"TextProp:=", <TextInfo>,
"NumberProp:=", <NumberInfo>,
"SeparatorProp:=", <SeparatorInfo>,
"ValueProp:=", <ValueInfo>,
"MenuProp:=", <MenuInfo>),
Array("NAME:Properties", // any combo of the following
"CheckboxProp:=", <CheckBoxInfo>,
"TextProp:=", <TextInfo>,
"NumberProp:=", <NumberInfo>,
"SeparatorProp:=", <SeparatorInfo>,
"ValueProp:=", <ValueInfo>,
"MenuProp:=", <MenuInfo>),
"VPointProp:=", <VPointInfo>,
"PointProp:=", <PointInfo>),
Array("Quantities",
"QuantityProp:=", <QuantityPropInfo>...),
Array("NAME:CosimDefinitions",
<CosimDefInfo>,
<CosimDefInfo>...)
```

Return Value:<string>

```
// composite name of the component.
// If the name requested conflicts with the name of an existing
// component, the requested name is altered to be unique.
// The name returned reflects any change made to be unique.
```

Parameters:<ComponentName>:

```
<string> // simple name of the component

<ComponentInfo>:
Array("Type:=", <TypeInfo>,
"NumTerminals:=", <int>,
"DataSource:=", <string>,
"ModifiedOn:=", <ModifiedOnInfo>,
"Manufacturer:=", "<string>,
"Symbol:=", <string>,
"Footprint:=", <string>,
"Description:=", <string>,
"InfoTopic:=", <string>,
"InfoHelpFile:=", <string>,
"IconFile:=", <string>,
"LibraryName:=", "",
"OriginalLocation:=", "Project", // Project Location
"Author:=", <string>,
"OriginalAuthor:=", <string>,
"CreationDate:= ", <int>)
```

<TypeInfo>:

An integer that is the or-ing of bits for each product listed below. The default setting is 0xffffffff (4294967295) which indicates valid for all products. In the component editing dialog, checking different boxes in the "Specify products for which this component is valid" grid control sets specific flags that correspond to the following hex/decimal settings:

Nexxim -- 100 binary, 4 decimal, 0x4

SIwaveDeNovo -- 1000 binary, 8 decimal, 0x8

Simplorer -- 10000 binary, 16 decimal, 0x10

MaxwellCircuit -- 100000 binary, 32 decimal, 0x20

<ModifiedOnInfo>:

An integer that corresponds to the number of seconds that have elapsed since 00:00 hours, Jan 1, 1970 UTC from the system clock.

<TerminalInfo>:

Array(<string>, // symbol pin

<string> // footprint pin

<string >, // gate name

<bool>, // shared

<int>, // equivalence number

<int>, // what to do if unconnected: flag as error:0, ignore:1

<string> // description

<Nature>)

<Nature>:

<string> // content varies as follows

Nexxim/Circuit:

"Electrical" // the only choice

Simplorer:

// several choices

"Electrical", "Magnetic", "Fluidic", "Translational",

"Translational_V", "Rotational", "Rotational_V",

""Radiant", "Thermal", or <VHDLPackageName>

<VHDLPackageName>:

<string> // in the form <Library>.<Package>

<Library>:

<string> // name of the VHDL library

<Package>:

<string> // name of the VHDL package

```
<VariableInfo>:
Array(<string>, // name
<FlagLetters>,
<string>, // description
"CB:=", <string>, // optional - script for call back
<string>) // value: number, variable, or expression

<FlagLetters>:
<string> // "D" - has description parameter,
// "RD" - readonly & has description parameter,
// or "RHD" - readonly, hidden, & has description parameter

<CheckBoxInfo>:
Array(<string>, // name
<FlagLetters>,
<string>, // description
"CB:=", <string>, // optional - script for call back
<bool>) // value: true or false

<ButtonInfo>:
```

```
Array(<string>, // name
<FlagLetters>,
<string>, // description
<string>, // button title
<string>, // extra text
<ClientID>,
"ButtonPropClientData:= ", <ClientDataArray>)

<ClientID>:
<int> // specifies Button Prop Client
// 0 - unknown, ButtonPropClientData
// array will be empty
// 1 - Netlist Prop Client
// 2 - not used
// 3 - File Name Prop Client

<ClientDataArray>:
varies with <ClientID>

<ClientId> is 0 or 1: empty array
```

Array()

<ClientID> is 3:

Array("InternalFormatText:=", "<prefix><RelativePath>")

<prefix>:

<string> // "<Project>", "<PersonalLib>", "<UserLib>", or "<SysLib>"

<RelativePath>:

<string> // relative path to file from <prefix>

<TextInfo>:

Array(<string>, // name

<FlagLetters>,

<string>, // description

"CB:=", <string>, // optional - script for call back

<string>) // value: a text string

<NumberInfo>:

Array(<string>, // name

<FlagLetters>,

```
<string>, // description
"CB:=", <string>, // optional - script for call back
<real>, // value: a number
<string>) // units

<SeparatorInfo>:
Array(<string>, // name
<FlagLetters>,
<string>, // description
"CB:=", <string>, // optional - script for call back
<string>) // value: a text string

<ValueInfo>:
Array(<string>, // name
<FlagLetters>,
<string>, // description
"CB:=", <string>, // optional - script for call back
<string>) // value: a number, variable or expression


<MenuPropInfo>:
Array(<string>, // name
<FlagLetters>,
<string>, // description
<string>, // menu choices - separated by commas
<int>) // 0 based index of current menu choice
```

```
<VPointInfo>:
Array(<string>, // name
<FlagLetters>,
<string>, // description
"CB:=", <string>, // optional - script for call back
<string>, // x value: number with length units
<string>) // y value: number with length units
```

```
<PointInfo>:
Array(<string>, // name
<FlagLetters>,
<string>, // description
"CB:=", <string>, // optional - script for call back
<real>, // x value
<real>) // y value
```

```
<QuantityPropInfo>:
```

```
Array(<string>, // name
<FlagLetters>,
<string>, // description
<string>, // value
<TypeString>,
<TypeStringDependentInfo>)

<TypeString>:
<string> // "Across", "Through", or "Free"

<TypeStringDependentInfo>:

<TypeString> is "Free" :
<string>, // direction: "In", "Out", "InOut", or "DontCare"
// Following <string> is not present if direction is "DontCare"
<string> // when to calculate: "BeforeAnalogSolver",
// "BeforeStateGraph", "AfterStateGraph", or "DontCareWhen"

<TypeString> is "Across" or "Through":
<int>, // terminal 1
<int> // terminal 2
```

```
<CosimDefInfo>:
Array("NAME:CosimDefinition",
"CosimulatorType:=", <int>,
"CosimDefName:=", <string> // "HFSS 3D Layout", "Circuit",
// "Custom", or "Netlist"
"IsDefinition:=", <bool>,
final array member(s) vary with CosimDefName)

final array members for HFSS 3D Layout:
"CosimStackup:=", <string>,
"CosimDmbedRatio:=", <int>

final array members for Circuit:
"ExportAsNport:=", <int>,
"UsePjt:=", <int>

final array member for Custom:
"DefinitionCompName:=", <string>

final array member for Netlist:
```

"NetlistString:=", <string>

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

	"CheckboxProp:=", <CheckBoxInfo>, "TextProp:=", <TextInfo>, "NumberProp:=", <NumberInfo>, "SeparatorProp:=", <SeparatorInfo>, "ValueProp:=", <ValueInfo>, "MenuProp:=", <MenuInfo>, "VPointProp:=", <VPointInfo>, "PointProp:=", <PointInfo>], ["Quantities", "QuantityProp:=", <QuantityPropInfo>...], ["NAME:CosimDefinitions", <CosimDefInfo>, <CosimDefInfo>...]]
Python Example	oComponentManager.Add(["NAME:Component", "Info:=", ["Type:=", 0, "NumTerminals:=", 0, "DataSource:=", "", "ModifiedOn:=", 1467910752,

	"Manufacturer:=", "", "Symbol:=", "Component", "ModelNames:=", "", "Footprint:=", "", "Description:=" , "", "InfoTopic:=", "", "InfoHelpFile:=", "", "IconFile:=", "", "Library:=", "", "OriginalLocation:=", "Project", "IEEE:=", "", "Author:=", "", "OriginalAuthor:=", "", "CreationDate:=", 1467910746, "ExampleFile:=", ""], "Refbase:=", "U", "NumParts:=", 1, "ModSinceLib:=", True, "CompExtID:=", 2])

AddDataset

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

<table>
<tr><th>UI Access</th><th><b>Project</b> > <b>Datasets</b> > <b>Add</b>.</th></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td>&lt;<i>DatasetDataArray</i>&gt;</td><td>Array</td><td>Array("NAME:<DatasetName>",
Array("NAME:Coordinates", <CoordinateArray>,
<CoordinateArray>, ...)</td></tr>
<tr><td>&lt;<i>DatasetName</i>&gt;</td><td>String</td><td>Name of the dataset.</td></tr>
<tr><td>&lt;<i>CoordinateArray</i>&gt;</td><td>Array</td><td>Array("NAME:Coordinate", "X:=", <double>, "Y:=",<double&gt;)</td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

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
        ]
    ]
]
)
```

AddDefinitionFromBlock

Adds a material definition from block text (same definition format as would be contained in the material library file) by library type (using definition folder name). This scripting command directly supports the .AMAT (or .ASURF) definition formats.

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td>&lt;<i>defBlock</i>&gt;</td><td>String</td><td>Text of the new material definition in block form.</td></tr>
<tr><td>&lt;<i>defFolderName</i>&gt;</td><td>String</td><td>Library type (by definition folder name)</td></tr>
<tr><td>&lt;<i>newTimeStamp</i>&gt;</td><td>String</td><td>New timestamp (time_t as integer number of seconds since 1/1/1970 12:00am, as string), default is current time</td></tr>
<tr><td>&lt;<i>replaceExisting</i>&gt;</td><td>Boolean</td><td>True to replace existing, False to choose a new unique name if an existing definition is found</td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>A property scripting object for the definition.</td></tr>
</table>

P-yt-	AddDefinitionFromBlock(<*defBlock*>,<*defFolderName*>, <*newTimeStamp*>, <*replaceExisting*>)

hon Syntax	
Python Example	oProject = oDesktop.NewProject() oProject.InsertDesign("HFSS", "HFSSDesign1", "DrivenModal", "") oDesign = oProject.SetActiveDesign("HFSSDesign1") oEditor = oDesign.SetActiveEditor("3D Modeler") oEditor.CreateBox(["NAME:BoxParameters", "XPosition:=" , "-0.4mm", "YPosition:=" , "-1mm", "ZPosition:=" , "0mm", "XSize:=" , "1.4mm", "YSize:=" , "1.6mm", "ZSize:=" , "0.6mm"

```
oProject = oDesktop.NewProject()

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

oDesign = oProject.SetActiveDesign("HFSSDesign1")

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

oEditor.CreateBox(

  [

    "NAME:BoxParameters",

    "XPosition:="           , "-0.4mm",
                    "YPosition:="           , "-1mm",
                    "ZPosition:="           , "0mm",
                    "XSize:="               , "1.4mm",
                    "YSize:="               , "1.6mm",
                    "ZSize:="               , "0.6mm"
```

```
        ],
[
  "NAME:Attributes",
                  "Name:="                , "Box1",
  "Flags:="               , "",
  "Color:="               , "(143 175 143)",
  "Transparency:="        , 0,
  "PartCoordinateSystem:=", "Global",
  "UDMId:="               , "",
  "MaterialValue:="       , "\"vacuum\"",
  "SurfaceMaterialValue:=", "\"\"",
  "SolveInside:="         , True,
          "ShellElement:="        , False,
  "ShellElementThickness:=", "0mm",
        "IsMaterialEditable:="  , True,

        "UseMaterialAppearance:=", False,

        "IsLightweight:="       , False

    ])
oDefinitionManager = oProject.GetDefinitionManager()
defBlock = "$begin 'vacuum2' $begin 'AttachedData' $begin 'MatAppearanceData' property_data-
='appearance_data' Red=230 Green=230 Blue=230 Transparency=0.95 $end 'MatAppearanceData'
$end 'AttachedData' simple('permittivity', 1) ModTime=1499970477 $end 'vacuum2'"
added = oDefinitionManager.AddDefinitionFromBlock(defBlock, "Materials", "10101010", True)
addedName = ''
  if isinstance(added, basestring):
    addedName = added
  elif isinstance(added, list):
    addedName = added[0]
else:

  addedName = added.GetName().replace("Materials:", "")
AddInfoMessage(os.path.basename(__file__) + " result: " + addedName)
```

```
materialNameInQuotes = "\"" + addedName + "\""
oEditor.ChangeProperty(
          [
     "NAME:AllTabs",
     [
       "NAME:Geometry3DAttributeTab",
       [
         "NAME:PropServers",
         "Box1"
       ],
       [
         "NAME:ChangedProps",
         [
         "NAME:Material",
            "Value:=", materialNameInQuotes

            ]
                                  ]
                          ]
             ])
```

AddMaterial

Adds a local material.

<table>
<tr><th>UI Access</th><th><b>Add Material</b> in the material editor.</th></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;MaterialParams&gt;</i></td><td>Array</td><td>[
"NAME: <name of the material to be added>",
<MatProperty>, <MatProperty>, ...
]</td></tr>
<tr><td><i>&lt;MatProperty&gt;</i></td><td>Array</td><td>For simple material:
"<PropertyName>:=", <value>

For anisotropic material:</td></tr>
</table>
</td></tr>
</table>

		["NAME:<PropertyName>", "property_type:=", "AnisoProperty", "unit:=", <Unit>", "component1:=", <value>, "component2:=", <value>, "component3:=", <value>))]
<PropertyName>	String	Should be one of the following (depending on the material, design, and solution types): Electromagnetic (Maxwell-exclusive material properties omitted, see Maxwell Scripting help): "permittivity", "permeability", "conductivity", "dielectric_loss_tangent", "magnetic_loss_tangent", "electric_coercivity", "magnetic_coercivity", "saturation_mag", "lande_g_factor", "delta_H", "delta_h_freq", "mass_density" Thermal (including solids, Icepak fluid flow, and Mechanical rotating fluid modeling): "thermal_conductivity", "mass_density", "specific_heat", "thermal_expansion_coefficient", "thermal_material_type", "viscosity", "diffusivity", "molecular_mass", "clarity_type" Structural:

			"mass_density", "youngs_modulus", "poissons_ratio", "thermal_expansion_coefficient"
	<*Unit*>	String	Possible values (Maxwell-exclusive properties omitted, see Maxwell Scripting Help; other missing entries are unitless): conductivity: "siemens/m" saturation_mag: "uTesla", "mTesla", "tesla", "kTesla", "uGauss", "mGauss", "gauss", "kGauss" delta_H: "A_per_meter", "kA_per_meter", "Oe", "kOe" delta_h_frequency: "Hz", "kHz", "MHz", "GHz", "THz", "rps", "per_sec" mass_density: "kg/m^3" thermal_conductivity: "W/m-C" specific_heat: "J/kg-C" youngs_modulus: "N/m^2" thermal_expansion_coefficient: "1/C"
Return Value	None		

Python Syntax	AddMaterial (["NAME:<MaterialName>", <MatProperty>, <MatProperty>, ...])

Python Example	oDefinitionManager.AddMaterial(
	["permittivity:=", "2.2", "0.002"])
	
	oDefinitionManager.AddMaterial [("NAME:Material2",_
	  "dielectric_loss_tangent:=", "44",
	  Array("NAME:saturation_mag",_
	    "property_type:=", "AnisoProperty",_
	    "unit:=", "Gauss",_
	    "component1:=", "11", _
	    "component2:=", "22", _
	    "component3:=", "33"), _
	    "delta_H:=", "44Oe")]

Add [padstack manager]

Use: Add a padstack

Command: Tools > Edit Configured Libraries > Padstacks > Add Padstack

Syntax: Add Array("NAME:<PadstackName>",

"ModTime:=", <ModifiedOnInfo>,

"Library:=", "", // name of the library

"LibLocation:=", "Project", // location of the named library

Array("NAME:psd",

```
"nam:= ", <PadstackName>,
"lib:=", "", // name of the library
"mat:=", "", // hole plating material
"plt:=", "0", // percent of hole's radius filled by plating
Array("NAME:pds",
<LayerGeometryArray>,
<LayerGeometryArray....),
"hle:=", <PadInfo>
"hRg:=", <HoleRange>,
"sbsh:=", <SolderballShape>,
"sbpl:=", <SolderballPlacement>,
"sbr:=", <string>, // solderball diameter, real with units
"sb2:=", <string>, // solderball mid diameter, real with units
"sbn:=", <string>), // name of solderball material
"ppl:=", <PadPortLayerArray>)
```

Return Value: simple name of the added padstack

```
// If the name requested conflicts with the name of an existing
// padstack, the requested name is altered to be unique.
// The name returned reflects any change made to be unique.
```

Parameters: <PadstackName>:

<string> // simple name of padstack to create

<ModifiedOnInfo>:

An integer that corresponds to the number of seconds that have elapsed

since 00:00 hours, Jan 1, 1970 UTC from the system clock.

<LayerGeometryArray>:

Array("Name:lgm",

"lay:=", <string>, // definition layer name

"id:=", <int>, // definition layer id

"pad:=", <PadInfo>, // pad

"ant:=", <PadInfo>, // antipad

"thm:=", <PadInfo>, // themal pad

"X:=", <string>, // pad x connection, real with units

"Y:=", <string>, // pad y connection, real with units

"dir:=", <DirectionString>) // pad connection direction

<PadInfo>:

Array("shp:=", <PadShape>,

"Szs:=", <DimensionArray>,

```
"X:=", <string>, // x offset, real with units
"Y:=", <string>, // y offset, real with units
"R:=", <string>) // rotation, real with units

<PadShape>:
<string> one of these choices
"No" // no pad
"Cir" // Circle
"Sq" // Square
"Rct" // Rectangle
"Ov" // Oval
"Blt" // Bullet
"Ply" // Polygons
"R45" // Round 45 thermal
"R90" // Round 90 thermal
"S45" // Square 45 thermal
"S90" // Square 90 thermal

<DimensionArray>:
Array(<string>, ...) // each string is a real with units for one of the dimensions of the shape
```

<DirectionString>:

<string> one of these choices

"No" // no direction

"Any" // any direction

"0" // 0 degrees

"45" // 45 degrees

"90" // 90 degrees

"135" // 135 degrees

"180" // 180 degrees

"225" // 225 degrees

"270" // 270 degrees

"315" // 315 degrees

<HoleRange>:

<string> one of these choices

"Thr" // through all layout layers

"Beg" // from upper pad layer to lowest layout layer

"End" // from upper layout layer to lowest pad layer

"UTL" // from upper pad layer to lowest pad layer

<SolderballShape>:

<string> one of these choices

"None" // no solderball

"Cyl" // cylinder solderball

"Sph" // spheroid solderball

<SolderballPlacement>:

<string> one of these choices

"abv" // above padstack

"blw" // below padstack

<PadPortLayerArray>:

Array(<int>, <int>,....) where each int is a layer id

Add [symbol manager]

Use: Add a symbol

Command: Tools > Edit Configured Libraries > Symbols > Add Symbol

Syntax: Add Array("NAME:<SymbolName>",

"ModTime:=", <ModifiedTimeInfo>,

"Library:=", "", // Library name

"LibLocation:=", "Project", // Project Location

<PinDefInfo>,

<PinDefInfo>,... // optional, to define pins

<GraphicsDataInfo>, // optional, to define graphics

<PropDisplayMapInfo>)) // optional, to define property displays

Return Value: <string>

// composite name of the symbol.

// If the name requested conflicts with the name of an existing

// symbol, the requested name is altered to be unique.

// The name returned reflects any change made to be unique.

Parameters: <SymbolName>:

<string> // simple name of the symbol being added

<ModifiedOnInfo>:

An integer that corresponds to the number of seconds that have elapsed

since 00:00 hours, Jan 1, 1970 UTC from the system clock.

<PinDefInfo>:

Array("NAME:PinDef",

"Pin:=", Array (<string>, // pin name

<real>, // x location

<real>, // y location

<real>, // angle in radians

```
<PinType>,
<real>, // line width
<real>, // line length
<bool>, // mirrored
<int>, // color
<bool>, // true if visible, false if not
<string>, // hidden net name
<OptionalPinInfo>, // optional info
<PropDisplayMapInfo>)) // optional

<PinType>:
<string> // "N" : normal pin
// "I" : input pin
// "O" : output pin

<OptionalPinInfo>:
// Specify both or neither
<bool>, // true if name is to be shown
<bool>, // true if number is to be shown
```

```
<PropDisplayMapInfo>:
Array("NAME:PropDisplayMap",
<PropDisplayInfo>,
<PropDisplayInfo>,...)

<PropDisplayInfo>:
<NameString>, Array(<DisplayTypeInfo>,
<DisplayLocationInfo>,
<int>, // optional, level number
<TextInfo>)
<NameString>:
<string> // PropertyName:=, where PropertyName is the name of
// the property to be displayed

<DisplayTypeInfo>:
<int> // 0 : No display
// 1 : Display name only
// 2 : Display value only
// 3 : Display both name and value
// 4: Display evaluated value only
// 5: Display both name and evaluated value

<DisplayLocationInfo>:
<int> // 0 : Left
// 1 : Top
// 2 : Right
// 3 : Bottom
// 4 : Center
// 5 : Custom placement

<GraphicsDataInfo>:
Array("NAME:Graphics",
// one or more of the following
<RectInfo>,
<CircleInfo>,
<ArcInfo>,
<LineInfo>,
<PolygonInfo>,
<TextInfo>,
<ImageInfo>)
```

```
<RectInfo>:
"Rect:=", Array(<real>, // line width
<int>, // fill pattern
<int>, // color
<real>, // angle, in radians
<real>, // x position of center
<real>, // y position of center
<real>, // width
< real>) // height
```

```
<CircleInfo>:
"Circle:=", Array(<real>, // line width
<int>, // fill pattern
<int>, // color
<real>, // x position of center
<real>, // y position of center
< real>) // radius
```

```
<ArcInfo>:
"Arc:=", Array(<real>, // line width
<int>, // line pattern
<int>, // color
<real>, // x position of center
<real>, // y position of center
< real>, // radius
<real>, // start angle, in radians
<end>) // end angle, in radians

<LineInfo>:
"Line:=", Array(<real>, // line width
<int>, // line pattern
<int>, // color
<PointInfo>, // must specify at least 2 points
<PointInfo>...)
<PointInfo>:
<real>, // x position
<real> // y position

<PolygonInfo>:
"Polygon:=", Array(<real>, // line width
<int>, // fill pattern
<int>, // color
<PointInfo>, // must specify at least 3 points
<PointInfo>...)

<TextInfo>:
"Text:=", Array(<real>, // x position
<real>, // y position
<real>, // angle, in radians
<Justification>,
<bool>, // is plotter font
<string>, // font name
<int>, // color
<string>) // text string

<Justification>:
<int> // 0 : left top
// 1 : left base
// 2 : left bottom
// 3 : center top
// 4 : center base
// 5 : center bottom
// 6 : right top
// 7 : right base
// 8 : right bottom

<ImageInfo>:
"Image:=", Array(<RectInfo>,
<ImageData>,
<bool>) // is mirrored

<ImageData>:
<string>, // file path
<int>, // 0 : use the file path and link to it
// 1 : ignore file path and use next parameter
<string> // text data, only present if preceding int is 1
```

AreMaterialPropertiesEqual

Checks whether named material compared to added material data have equivalent major properties.

UI Access	None.

<table>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>material_data</td><td><string></td><td>This will be the same format as the AddMaterial() input parameter, material_data. For example, material_data can be something like this.["NAME:Mold_Material", "CoordinateSystemType:=", "Cartesian", "BulkOrSurfaceType:=", 1, ["NAME:PhysicsTypes", "set:=", ["Thermal"]], "thermal_conductivity:=", "0.8", "mass_density:=", "1980", "specific_heat:=", "960"].</td></tr>
<tr><td><i><MaterialName></i></td><td><String></td><td>Name of the material to compare with material database</td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>Boolean:
• <b>True</b> – named materials have equivalent major properties.
• <b>False</b> – named materials do not have equivalent major properties.</td></tr>
</table>

Python Syntax	AreMaterialPropetiesEqual(<material_data>, '<*MaterialName*>')
Python Example	`oDefinitionManager.AreMaterialPropertiesEqual(material_data, 'Mold_Material')`

AreSurfaceMaterialPropertiesEqual

Checks whether named surface material compared to added surface material data have equivalent major properties.

<table>
<tr><td><b>UI Access</b></td><td>None.</td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>material_data</td><td><string></td><td>This will be the same format as the AddMaterial() input parameter, material_data. For example, material_data can be</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><td></td><td><table>
<tr><td></td><td></td><td>something like this.["NAME:Mold_Material", "CoordinateSystemType:=", "Cartesian", "BulkOrSurfaceType:=", 1, ["NAME:PhysicsTypes", "set:=", ["Thermal"]], "thermal_conductivity:=", "0.8", "mass_density:=", "1980", "specific_heat:=", "960"]</td></tr>
<tr><td>&lt;<i>MaterialName</i>&gt;</td><td>&lt;String&gt;</td><td>Name of the surface material to compare with material database</td></tr>
</table></td></tr>
<tr><td><b>Return Value</b></td><td>Boolean:<ul><li><b>True</b> – named surface materials have equivalent major properties.</li><li><b>False</b> – named surface materials do not have equivalent major properties.</li></ul></td></tr>
</table>

Python Syntax	AreSurfaceMaterialPropetiesEqual(<material_data>, '<*MaterialName*>')
Python Example	`oDefinitionManager.AreSurfaceMaterialPropertiesEqual(material_data, 'Mold_Material')`

CloneMaterial

Clones a local material.

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td><table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td>&lt;<i>matName</i>&gt;</td><td>String</td><td>Name of existing material.</td></tr>
<tr><td>&lt;<i>newName</i>&gt;</td><td>String</td><td>Name for newly cloned material.</td></tr>
</table></td></tr>
</table>

Return Value	Boolean: • 1 - Material is cloned. • 0 - Existing material not found or a conflict with the new material name.

Python Syntax	CloneMaterial (<*matName*>, <*newName*>)
Python Example	`oDefinitionManager.CloneMaterial("copper1", "copper3")`

DeleteDataset

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

<table>
<tr><td><b>UI Access</b></td><td><b>Project</b> &gt; <b>Datasets</b> &gt; <b>Remove</b>.</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;<i>DatasetName</i>&gt;</td><td>String</td><td>Name of the dataset found in the project.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	DeleteDataset (<*DatasetName*>)

Python Example	`oProject.DeleteDataset('$ds1')`
	`oDesign.DeleteDataset('$ds1')`

DoesMaterialExist

Checks for the presence of a material in the library by name

<table>
<tr><td><b>UI Access</b></td><td>None.</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;<i>MaterialName</i>&gt;</td><td>&lt;String&gt;</td><td>Name of the material to search for in the material database</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>Boolean:
• <b>True</b> – specified material exists.
• <b>False</b> – specified material does not exist.</td></tr>
</table>

Python Syntax	DoesMaterialExist(<*MaterialName*>)
Python Example	`oDefinitionManager.DoesMaterialExist("modified_epoxy")`

Edit [component manager]

Modifies an existing component

Command: Tools > Edit Configured Libraries > Components > Edit Component

Syntax: Edit <ComponentName>,

Array("NAME:<NewComponentName>",

"Info:=", <ComponentInfo>,

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

"MenuProp:=", <MenuInfo>),

"VPointProp:=", <VPointInfo>,

"PointProp:=", <PointInfo>),

Array("Quantities",

"QuantityProp:=", <QuantityPropInfo>...),

Array("NAME:CosimDefinitions",

<CosimDefInfo>,

<CosimDefInfo>...)

Return Value: <string>

// composite name of the component.

// If the name requested conflicts with the name of an existing

// component, the requested name is altered to be unique.

// The name returned reflects any change made to be unique.

Parameters: <ComponentName>:

<string> // composite name of the component to edit

<NewComponentName>:

<string> // new simple name for the component

<ComponentInfo>:

Array("Type:=", <TypeInfo>,

"NumTerminals:=", <int>,

"DataSource:=", <string>,

"ModifiedOn:=", <ModifiedOnInfo>,

"Manufacturer:=", "<string>,

"Symbol:=", <string>,

"Footprint:=", <string>,

"Description:=", <string>,

"InfoTopic:=", <string>,

"InfoHelpFile:=", <string>,

"IconFile:=", <string>,

"LibraryName:=", <string>,

"OriginalLocation:=", <string>, // Project Location

"Author:=", <string>,

"OriginalAuthor:=", <string>,

"CreationDate:= ", <int>)

<TypeInfo>:

An integer that is the or-ing of bits for each product listed below. The default setting is 0xffffffff (4294967295) which indicates valid for all products. In the component editing dialog, checking different boxes in the "Specify products for which this component is valid" grid control sets specific flags that correspond to the following hex/decimal settings:

Nexxim -- 100 binary, 4 decimal, 0x4

SIwaveDeNovo -- 1000 binary, 8 decimal, 0x8

Simplorer -- 10000 binary, 16 decimal, 0x10

MaxwellCircuit -- 100000 binary, 32 decimal, 0x20

<ModifiedOnInfo>:

An integer that corresponds to the number of seconds that have elapsed since 00:00 hours, Jan 1, 1970 UTC from the system clock.

<TerminalInfo>:

Array(<string>, // symbol pin

<string> // footprint pin

<string >, // gate name

<bool>, // shared

<int>, // equivalence number

<int>, // what to do if unconnected: flag as error:0, ignore:1

<string>, // description

<Nature>)

<Nature>:

<string> // content varies as follows

Nexxim/Circuit:

"Electrical" // the only choice

Simplorer:

// several choices

"Electrical", "Magnetic", "Fluidic", "Translational",

"Translational_V", "Rotational", "Rotational_V",

""Radiant", "Thermal", or <VHDLPackageName>

<VHDLPackageName>:

<string> // in the form <Library>.<Package>

<Library>:

<string> // name of the VHDL library

<Package>:

<string> // name of the VHDL package

<VariableInfo>:

```
Array(<string>, // name
<FlagLetters>,
<string>, // description
"CB:=", <string>, // optional - script for call back
<string>) // value: number, variable, or expression

<FlagLetters>:
<string> // "D" - has description parameter,
// "RD" - readonly & has description parameter,
// or "RHD" - readonly, hidden, & has description parameter

<CheckBoxInfo>:
Array(<string>, // name
<FlagLetters>,
<string>, // description
"CB:=", <string>, // optional - script for call back
<bool>) // value: true or false

<ButtonInfo>:
Array(<string>, // name
<FlagLetters>,
<string>, // description
<string>, // button title
<string>, // extra text
<ClientID>,
"ButtonPropClientData:= ", <ClientDataArray>)

<ClientID>:
<int> // specifies Button Prop Client
// 0 - unknown, "ButtonPropClientData
// array will be empty
// 1 - Netlist Prop Client
// 2 - not used
// 3 - File Name Prop Client

<ClientDataArray>:
varies with <ClientID>

<ClientID> is 0 or 1: empty array
Array()
```

<ClienID> is 3:

Array("InternalFormatText:=", "<prefix><RelativePath>")

<prefix>:

<string> // "<Project>", "<PersonalLib>", "<UserLib>", or "<SysLib>"

<RelativePath>:

<string> // relative path to file from <prefix>

<TextInfo>:

Array(<string>, // name

<FlagLetters>,

<string>, // description

"CB:=", <string>, // optional - script for call back

<string>) // value: a text string

<NumberInfo>:

Array(<string>, // name

<FlagLetters>,

<string>, // description

```
"CB:=", <string>, // optional - script for call back
<real>, // value: a number
<string>) // units

<SeparatorInfo>:
Array(<string>, // name
<FlagLetters>,
<string>, // description
"CB:=", <string>, // optional - script for call back
<string>) // value: a text string

<ValueInfo>:
Array(<string>, // name
<FlagLetters>,
<string>, // description
"CB:=", <string>, // optional - script for call back
<string>) // value: a number, variable or expression


<MenuPropInfo>:
Array(<string>, // name
<FlagLetters>,
<string>, // description
```

```
<string>, // menu choices - separated by commas
<int>) // 0 based index of current menu choice

<VPointInfo>:
Array(<string>, // name
<FlagLetters>,
<string>, // description
"CB:=", <string>, // optional - script for call back
<string>, // x value: number with length units
<string>) // y value: number with length units

<PointInfo>:
Array(<string>, // name
<FlagLetters>,
<string>, // description
"CB:=", <string>, // optional - script for call back
<real>, // x value
<real>) // y value

<QuantityPropInfo>:
Array(<string>, // name
<FlagLetters>,
<string>, // description
<string>, // value
<TypeString>,
<TypeStringDependentInfo>)

<TypeString>:
<string> // "Across", "Through", or "Free"

<TypeStringDependentInfo>:

"Free" :
<string>, // direction: "In", "Out", "InOut", or "DontCare"
// Following <string> is not present if direction is "DontCare"
<string> // when to calculate: "BeforeAnalogSolver",
// "BeforeStateGraph", "AfterStateGraph", or "DontCareWhen"

"Across" or "Through"
<int>, // terminal 1
<int> // terminal 2

<CosimDefInfo>:
```

```
Array("NAME:CosimDefinition",
"CosimulatorType:=", <int>,
"CosimDefName:=", <string> // "HFSS3D", "Circuit",
// "Custom", or "Netlist"
"IsDefinition:=", <bool>,
final array member(s) vary with CosimDefName)

final array members for HFSS 3D Layout:
"CosimStackup:=", <string>,
"CosimDmbedRatio:=", <int>

final array members for Circuit:
"ExportAsNport:=", <int>,
"UsePjt:=", <int>

final array member for Custom:
"DefinitionCompName:=", <string>

final array member for Netlist:
"NetlistString:=", <string>
```

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

	"NumberProp:=", <NumberInfo>, "SeparatorProp:=", <SeparatorInfo>, "ValueProp:=", <ValueInfo>, "MenuProp:=", <MenuInfo>), "VPointProp:=", <VPointInfo>, "PointProp:=", <PointInfo>), ["Quantities", "QuantityProp:=", <QuantityPropInfo>...], ["NAME:CosimDefinitions", <CosimDefInfo>, <CosimDefInfo>...])

Python Example	`name = oComponentManager.Edit ("Simplorer Circuit Elements\BJTs:Level01_NPN", _`
	`["NAME:Level01_NPN", "Info:=", ["Type:=", 4294901764,_`
	`"NumTerminals:=", 3, "DataSource:=", "Ansoft built-in component",_`
	`"ModifiedOn:=", 1152722112, "Manufacturer:=", "", _`
	`"Symbol:=", "nexx_bjt_npn", "Footprint:=", "", _`
	`"Description:=", "BJT, GP, NPN", "InfoTopic:=", "NXBJT1.htm", _`
	`"InfoHelpFile:=", "nexximcomponents.chm", "IconFile:=", "bjtsn.bmp", _`
	`"Library:=", "Nexxim Circuit Elements\BJTs",_`
	`"OriginalLocation:=", "SysLibrary ", "Author:=", "", _`

```
"OriginalAuthor:=", "", "CreationDate:=", 1152722102], _
"Refbase:=", "Q", "NumParts:=", 1, "Terminal:=", ["collector", _
"collector", "A", false, 6, 0, "", "Electrical"], _
"Terminal:=", ["base", "base", "A", false, _
7, 0, "", "Electrical"], "Terminal:=", ["emitter", _
"emitter", "A", false, 8, 0, "", "Electrical"], _
["NAME:Parameters", "TextProp:=", ["LabelID", _
"HD", "Property string for netlist ID", _
"Q@ID"], "TextProp:=", ["MOD", "D", _
"Name of model data reference", "required"], _
"VariableProp:=", ["AREA", "D", _
"Emitter area multiplying factor, which affects
currents, resistances, and capacitances", "1"], _
"VariableProp:=", ["AREAB", "D", "Base AREA", _
"1"], "VariableProp:=", ["AREAC", "D", "Collector AREA", _
"1"], "VariableProp:=", ["DTEMP", "D", _
"The difference between element and circuit temperature (deg Cel)", _
"0"], "VariableProp:=", ["M", "D", _
"Multiplier factor to simulate multiple BJTs in parallel", _
"1"], "ButtonProp:=", ["NexximNetlist", "HD", "", _
```

	"Q@ID %0 %1 %2 *MOD(@MOD) *AREA(AREA=@AREA)"& _ " *AREAB(AREAB=@AREAB) *AREAC(AREAC=@" &_ "AREAC) *DTEMP(DTEMP=@DTEMP) *M(M=@M)", _ "Q@ID %0 %1 %2 *MOD(@MOD) " & "*AREA(AREA=@AREA) *AREAB(AREAB=@AREAB) *AREAC(AREAC=@" & _ "AREAC) *DTEMP(DTEMP=@DTEMP) *M(M=@M)", 1, _ "ButtonPropClientData:=", [], "TextProp:=", ["ModelName", "HD", "", "Q"]]])
Python Example 2	name2 = oComponentManager.Edit ("MyComponent",_ (["NAME:MyOtherComponent","Info:=", ["Type:=", 4294901767, _ "NumTerminals:=", 2, "DataSource:=", "", _ "ModifiedOn:=", 1071096503, "Manufacturer:=", "Ansoft", _ "Symbol:=", "bendo", "Footprint:=", "BENDO", _

```
"Description:=", "", "InfoTopic:=", "", _

"InfoHelpFile:=", "", "IconFile:=", "", _

"LibraryName:=", "", "OriginalLocation:=", "Project", _

"Author:=", "", "OriginalAuthor:=", "", _

"CreationDate:= ", 1147460679], "Refbase:=", "U", _

"NumParts:=", 1, "OriginalComponent:=", "", _

"Terminal:=", ["n1", "n1", "A", false, 0, 0, "", _

"Electrical"], "Terminal:=", ["n2", "n2", "A", _

false, 1, 0, "", Electrical"], ["NAME:Parameters", _

"MenuProp:=", ["CoSimulator", "D", "", _
```

```
"Default,Custom,Netlist", 0], "ButtonProp:=", ["CosimDefinition", _

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

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

"CosimulatorType:=", 0, "CosimDefName:=", "HFSS3D", _

"IsDefinition:=", true, "CosimStackup:=", "Layout stackup", _

"CosimDmbedRatio:=", 3], ["NAME:CosimDefinition", _

"CosimulatorType:=", 1, "CosimDefName:=", "", _

"IsDefinition:=", true, "ExportAsNport:=", 0, _

"UsePjt:=", 0], ["NAME:CosimDefinition", _
```

```
"CosimulatorType:=", 2, "CosimDefName:=", "Custom", _

"IsDefinition:=", true, "DefinitionCompName:=", ""], _

["NAME:CosimDefinition", "CosimulatorType:=", 3, _

"CosimDefName:=", "Netlist", "IsDefinition:=", true, _

"NetlistString:=", ""]]])
```

EditDataset

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

<table>
<tr><td><b>UI Access</b></td><td><b>Project</b> &gt; <b>Datasets</b> &gt; <b>Edit</b>.</td></tr>
<tr><td><b>Parameters</b></td><td><table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td>&lt;<i>OriginalName</i>&gt;</td><td>String</td><td>Name of the original dataset.</td></tr>
<tr><td>&lt;<i>DatasetDataArray</i>&gt;</td><td>Array</td><td>Data for the modified dataset.</td></tr>
</table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	EditDataset (<*OriginalName*> <*DatasetDataArray*>)

Python Example

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

EditMaterial

Modifies an existing material.

UI Access	**View/Edit Materials** command in the material editor
Parameters	

Name	Type	Description
<OriginalName>	String	Name of the material before editing.
<MatProperties>	Array	Structured array containing material properties:

```
[
  "NAME:<New material name>",
  "CoordinateSystemType:=", <string>,
  "BulkOrSurfaceType:=" , <integer>,
  [
    "NAME:PhysicsTypes",
```

		"set:=" , <array containing string physics types>], <Optional ModifierDataArray>, "permeability:=" , <string containing value>, "conductivity:=" , <string containing value>, "thermal_conductivity:=", <string containing value>, "mass_density:=" , <string containing value>, "specific_heat:=" , <string containing value>, "youngs_modulus:=" , <string containing value>, "poissons_ratio:=" , <string containing value>, "thermal_expansion_coefficient:=", <string containing value>]
<ModifierDataArray>	Array	Optional structured array containing thermal or spatial modifiers: ["NAME:ModifierData", ["NAME:<ThermalModifierData or SpatialModifierData>", "modifier_data:=" , <"thermal_modifier_data" or "spatial_modifier_data">, [

			"NAME:<all_thermal_modifiers or all_spatial_modifiers>", ["NAME:<modifierName>", "Property::=" , <string property being modified>, "Index::=" , <integer>, "prop_modifier:=" , <"thermal_modifier" or "spatial_modifier">, "use_free_form:=" , <Boolean>, "free_form_value:=" , <string modifier value>,]]]]
Return Value	None.		

Python Syntax	EditMaterial (<*OriginalName*>, <*MatProperties*>)
Python Example	**Without Modifiers:** oDefinitionManager.EditMaterial("alumina_92pct", [

```
   "NAME:alumina_92pct",
   "CoordinateSystemType:=" , "Cartesian",
   "BulkOrSurfaceType:=" , 1,
   [
     "NAME:PhysicsTypes",
     "set:=" , ["Electromagnetic","Thermal","Structural"]
   ],
   "permittivity:=" , "9.3",
   "dielectric_loss_tangent:=" , "0.008",
   "thermal_conductivity:=" , "26",
   "mass_density:=" , "3720",
   "specific_heat:=" , "790",
   "youngs_modulus:=" , "267000000000",
   "poissons_ratio:=" , "0.26",
   "thermal_expansion_coeffcient:=", "7.2e-006"
]
)
```

With Thermal Modifier:

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

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

Transient Solve, Non-linear Drude Data Plasma

```
import ScriptEnv
ScriptEnv.Initialize("Ansoft.ElectronicsDesktop")
oDesktop.RestoreWindow()
oProject = oDesktop.SetActiveProject("Drude_plasma_parameters_r231")
oDefinitionManager = oProject.GetDefinitionManager()
oDefinitionManager.EditMaterial("Drude",
[
  "NAME:Drude",
  "CoordinateSystemType:=", "Cartesian",
  "BulkOrSurfaceType:="   , 1,
  [
    "NAME:PhysicsTypes",
    "set:="                  , ["Electromagnetic"]
  ],
  [
    "NAME:AttachedData",
    [
      "NAME:MatNonLinearDrudeFreqDepData",
      "property_data:="     , "nonlinear_drude_data",
      "EpsilonInfinity:="    , "1",
      "PlasmaFrequency:="    , "4.62348462366278GHz",
      "CollisionFrequency:=" , "0.00054491190162662GHz",
      "FieldBreakdown:="     , "10000V_per_meter",
      "PlasmaMaintainFrequency:=", "2.31174231183139GHz",
      "NeutralDensity:="     , 2.65164580488373E+20,
      "ElectronDensity:="    , 2.65164580488373E+17,
      "CollisionRateConstant:=", 2.05499505485618E-15
```

	]]])

Edit [padstack manager]

Use: Edit an existing padstack.

Command: Tools > Edit Configured Libraries > Padstacks > Edit Padstack

Syntax: Edit <PadstackName>,

Array("NAME:<NewPadstackName>",

"ModTime:=", <ModifiedOnInfo>,

"Library:=", "", // name of the library

"LibLocation:=", "Project", // location of the named library

Array("NAME:psd",

"nam:= ", <PadstackName>,

"lib:=", "", // name of the library

"mat:=", "", // hole plating material

"plt:=", "0", // percent of hole's radius filled by plating

Array("NAME:pds",

<LayerGeometryArray>,

<LayerGeometryArray....),

"hle:=", <PadInfo>

"hRg:=", <HoleRange>,

"sbsh:=", <SolderballShape>,

"sbpl:=", <SolderballPlacement>,

"sbr:=", <string>, // solderball diameter, real with units

"sb2:=", <string>, // solderball mid diameter, real with units

"sbn:=", <string>), // name of solderball material

"ppl:=", <PadPortLayerArray>)

Return Value: <string> // composite name of the padstack

// If the name requested conflicts with the name of an existing

// padstack, the requested name is altered to be unique.

// The name returned reflects any change made to be unique.

`Parameters: <PadstackName>:`

<string> // composite name of padstack to edit

<NewPadstackName>:

<string> // new simple name for padstack

<ModifiedOnInfo>:

An integer that corresponds to the number of seconds that have elapsed since 00:00 hours, Jan 1, 1970 UTC from the system clock.

<LayerGeometryArray>:

```
Array("Name:lgm",
"lay:=", <string>, // definition layer name
"id:=", <int>, // definition layer id
"pad:=", <PadInfo>, // pad
"ant:=", <PadInfo>, // antipad
"thm:=", <PadInfo>, // themal pad
"X:=", <string>, // pad x connection, real with units
"Y:=", <string>, // pad y connection, real with units
"dir:=", <DirectionString>) // pad connection direction
```

<PadInfo>:

```
Array("shp:=", <PadShape>,
"Szs:=", <DimensionArray>,
"X:=", <string>, // x offset, real with units
"Y:=", <string>, // y offset, real with units
"R:=", <string>) // rotation, real with units
```

<PadShape>:

<string> one of these choices

"No" // no pad

"Cir" // Circle

"Sq" // Square

"Rct" // Rectangle

"Ov" // Oval

"Blt" // Bullet

"Ply" // Polygons

"R45" // Round 45 thermal

"R90" // Round 90 thermal

"S45" // Square 45 thermal

"S90" // Square 90 thermal

<DimensionArray>:

Array(<string>, ...) // each string is a real with units for one of the

// dimensions of the shape

<DirectionString>:

<string> one of these choices

"No" // no direction

"Any" // any direction

"0" // 0 degrees

"45" // 45 degrees

"90" // 90 degrees

"135" // 135 degrees

"180" // 180 degrees

"225" // 225 degrees

"270" // 270 degrees

"315" // 315 degrees

<HoleRange>:

<string> one of these choices

"Thr" // through all layout layers

"Beg" // from upper pad layer to lowest layout layer

"End" // from upper layout layer to lowest pad layer

"UTL" // from upper pad layer to lowest pad layer

<SolderballShape>:

<string> one of these choices

"None" // no solderball

"Cyl" // cylinder solderball

"Sph" // spheroid solderball

<SolderballPlacement>:

<string> one of these choices

"abv" // above padstack

"blw" // below padstack

<PadPortLayerArray>:

Array(<int>, <int>,....) where each int is a layer id

ExportDataset

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

UI Access	**Project** > **Datasets** > **Export**.
Parameters	<table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>*<datasetFileFullPath>*</td><td>String</td><td>The full path to the file.</td></tr></table>
Return Value	None.

Python Syntax	ExportDataset (<*datasetFileFullPath*>)

Python Example	`oProject.ExportDataset('e:/tmp/dsdata.txt')`
	`oDesign.ExportDataset('e:/tmp/dsdata.txt')`

Export [footprint manager]

Use: Export a footprint to a library

Command: Tools > Edit Configured Libraries > Footprints > Export to Library

Syntax: Export Array("NAME:<LibraryName>",

<FootprintName>,

<FootprintName>...),

<LibraryLocation>

Return Value: None

`*Parameters:* <LibraryName>:`

<string> // name of the library

<FootprintName>:

<string> // composite name of footprint to export

<LibraryLocation>:

<string> // location of the library in <LibraryName>

// One of "Project", "PersonalLib", or "UserLib"

ExportMaterial

Exports a local material to a library.

<table>
<tr><td><b>UI Access</b></td><td><b>Export to Library</b> command in the material editor.</td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;ExportData&gt;</i></td><td>Array</td><td>["NAME:<LibraryName>",
<MaterialName>, <MaterialName>, ...]</td></tr>
<tr><td><i>&lt;LibraryName&gt;</i></td><td>String</td><td>Name of the exported library.</td></tr>
<tr><td><i>&lt;MaterialName&gt;</i></td><td>String</td><td>Name of the material to be exported.</td></tr>
<tr><td><i>&lt;LibraryLocation&gt;</i></td><td>String</td><td>Location to save the library. Only "PersonalLib" and "UserLib" are allowed.</td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	ExportMaterial (<*ExportData*>, <*LibraryLocation*>)

Python Example	`oDefinitionManager.ExportMaterial (["NAME:mylib",_`
	`"Material1", "Material2", "Material3"], "PersonalLib")`

Export [padstack manager]

Use: Export a padstack to a library

Command: Tools > Edit Configured Libraries > Padstacks > Export to Library

Syntax: Export Array("NAME:<LibraryName>",

<PadstackName>,

<PadstackName>...),

<LibraryLocation>

Return Value: None

Parameters: `<LibraryName>:`

<string> // name of the library

<PadstackName>:

<string> // simple name of padstack to export

<LibraryLocation>:

<string> // location of the library in <LibraryName>

// One of "Project", "PersonalLib", or "UserLib"

ExportScript

Use: Export to Library in the script definition manager

Command: None

Syntax: ExportScript <ExportData>,<Library location>

Return Value: None

Parameters: `<ExportData>`

```
Array("NAME:<LibraryName>",<ScriptName>,<ScriptName>,…)
```

Python Syntax	ExportScript(<ExportData>,<Library location>)
Python Example	`oProject.ExportComponent`

	`(["NAME:mylib", "myscript"], "PersonalLib")`

Export [symbol manager]

Use: Exports symbol(s) to a library

Command: Tools > Edit Configured Libraries > Symbols > Export to Library

Syntax: Export Array("NAME:<LibraryName>",

<SymbolName>,

<SymbolName>...),

<LibraryLocation>

Return Value: None

Parameters: <LibraryName>:

<string> // name of the library

<SymbolName>:

<string> // composite name of symbol to export

<LibraryLocation>:

<string> // location of the library in <LibraryName>

// One of "Project", "PersonalLib", or "UserLib"

GetProjextMaterialNames

Returns the material names belonging to an active Project.

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>None</td><td></td><td></td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>String names of the materials in the active project.</td></tr>
</table>

Python Syntax	GetProjectMaterialNames()

Python Example	`oProject = oDesktop.GetActiveProject()`
	`oDefinitionManager = oProject.GetDefinitionManager()`
	`materials = oDefinitionManager.GetProjectMaterialNames()`
	`AddWarningMessage(str(materials))`

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: • PassedParameterTab ("Parameter Values") • DefinitionParameterTab (Parameter Defaults") • LocalVariableTab ("Variables" or "Local Variables") • ProjectVariableTab ("Project variables") • ConstantsTab ("Constants") • BaseElementTab ("Symbol" or "Footprint") • ComponentTab ("General") • Component("Component") • CustomTab ("Intrinsic Variables") • Quantities ("Quantities") • Signals ("Signals")
	<PropServer>	String	An object identifier, generally returned from another script method, such as `CompInst@R;2;3`
	<PropName>	String	Name of the property.
Return Value	String value of the property.		

Python Syntax	GetPropertyValue (*<PropTab>*, *<PropServer>*, *<PropName>*)

Python Example	`selectionArray = oEditor.GetSelections()`
	`for k in selectionArray:`
	`val = oEditor.GetPropertyValue("PassedParameterTab", k, "R")`
	`...`

ImportDataset

Imports a dataset from a named file. This can be executed by the oProject, or oDesign variables. The name of the dataset is filename+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.

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

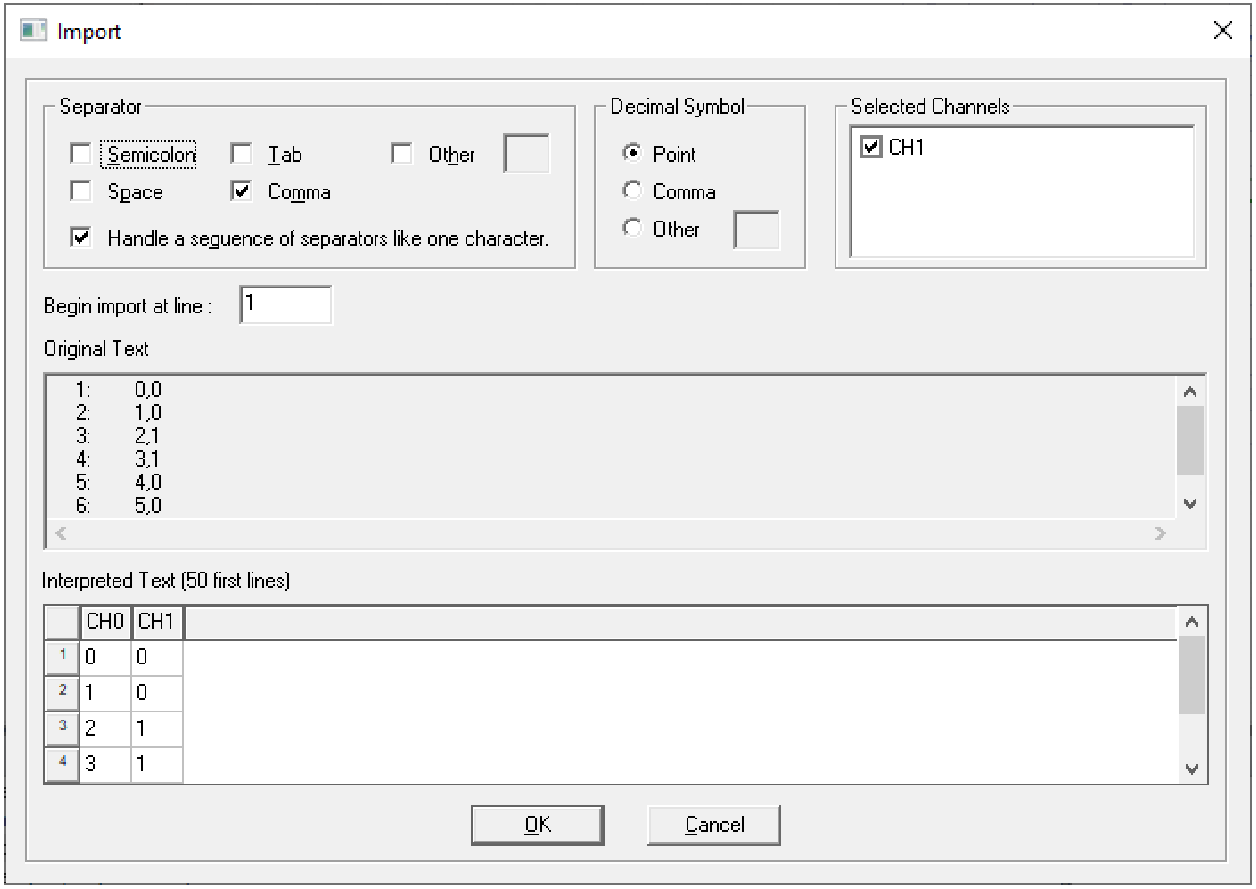
Python Syntax	ImportDataset (<*datasetFileFullPath*>,<*optionalDatasetName*>)

Python Example	`oProject.ImportDataset('e:\tmp\dsdata.tab')`
	`oDesign.ImportDataset('e:\tmp\dsdata.tab')`
	`oProject.ImportDataset('e:\tmp\dsdata.tab', 'MyDatasetName')`
	`oDesign.ImportDataset('e:\tmp\dsdata.tab', 'MyDatasetName')`

Note About File Types:

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

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



```
1:      0,0
2:      1,0
3:      2,1
4:      3,1
5:      4,0
6:      5,0
```

	CH0	CH1
1	0	0
2	1	0
3	2	1
4	3	1

Remove [component manager]

Remove a component from a library

Command: Tools > Edit Configured Libraries > Components > Remove Component

Syntax: Remove <ComponentName>,

<IsProjectComponent>,

<LibraryName>,

<LibraryLocation>

Return Value: None

Parameters: <ComponentName>:

<string> // composite name of the component to remove

<IsProjectComponent>:

<bool>

<LibraryName>:

<string> // name of the library

<LibraryLocation>:

<string> // location of the library in <LibraryName>

// One of "Project", "PersonalLib", or "UserLib"

Python Syntax	Remove (<ComponentName>, <IsProjectComponent>, <LibraryName>, <LibraryLocation>)
Python Example	`oComponentManager.Remove ("Simplorer Circuit`

	`Elements\BJTs:Level01_NPN", _ true, "Project")`

Remove [footprint manager]

Use: Removes a footprint from a library

Command: Tools > Edit Configured Libraries > Footprints > Remove Footprint

Syntax: Remove <FootprintName>,

<IsProjectFootprint>,

<LibraryName>,

<LibraryLocation>

Return Value: None

Parameters: <FootprintName>:

<string> // composite name of the footprint to remove

<IsProjectFootprint>:

<bool>

<LibraryName>:

<string> // name of the library

<LibraryLocation>:

<string> // location of the library in <LibraryName>

// One of "Project", "PersonalLib", or "UserLib"

RemoveMaterial

Removes a material from a library.

<table>
<tr><td><b>UI Access</b></td><td><b>Remove Material(s)</b> command in the material editor</td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td>&lt;<i>MaterialName</i>&gt;</td><td>String</td><td>Name of the material to be removed.</td></tr>
<tr><td>&lt;<i>IsProjectMaterial</i>&gt;</td><td>Boolean</td><td>If True, assumes the material is a project material. The last two parameters will be ignored.
If False, the material is not a project material.</td></tr>
<tr><td>&lt;<i>LibraryName</i>&gt;</td><td>String</td><td>Name of the user or personal library where the material resides.</td></tr>
<tr><td>&lt;<i>LibraryLocation</i>&gt;</td><td>String</td><td>Location of library. Valid options:"UserLib" or "PersonalLib".</td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	RemoveMaterial (<*MaterialName*>, <*IsProjectMaterial*>, <*LibraryName*>, <*LibraryLocation*>)

Python Example	`oDefinitionManager.RemoveMaterial ([`
	`"Material1", false, "mo0907","UserLib"])`

Remove [padstack manager]

Use: Removes a padstack from a library

Command: Tools > Edit Configured Libraries > Padstacks > Remove Padstacks

Syntax: Remove <PadstackName>,

<IsProjectPadstack>,

<LibraryName>,

<LibraryLocation>

Return Value: None

Parameters: <PadstackName>:

<string> // simple name of the padstack to remove

<IsProjectPadstack>:

<bool>

<LibraryName>:

<string> // name of the library

<LibraryLocation>:

<string> // location of the library in <LibraryName>

// One of "Project", "PersonalLib", or "UserLib"

RemoveScript

Use: Remove Script in the script definition manager

Command: None

Syntax: RemoveScript <ScriptName>,<IsProjectScript>, <LibraryName>,<LibraryLocation>

Return Value: None

```
Parameters:  <ScriptName>

Type: <string>

<IsProjectScript>

Type: <bool>

<LibraryName>

Type: <string>

<LibraryLocation>

Type: <string>
```

Python Syntax	RemoveScript (<ScriptName>,<IsProjectScript>, <LibraryName>,<LibraryLocation>)

Python Example	`oDefinitionManager.RemoveScript(`
	`"myscript", true, "Local", "Project")`

Remove [symbol manager]

Use: Removes a symbol from a library

Command: Tools > Edit Configured Libraries > Symbols > Remove Symbol

Syntax: Remove <SymbolName>,

<IsProjectSymbol>,

<LibraryName>,

<LibraryLocation>

Return Value: None

Parameters: <SymbolName>:

<string> // composite name of the symbol to remove

<IsProjectSymbol>:

<bool>

<LibraryName>:

<string> // name of the library

<LibraryLocation>:

<string> // location of the library in <LibraryName>

// One of "Project", "PersonalLib", or "UserLib"

Example:

```
oSymbolManager.Remove "Nexxim Circuit Elements\Distributed\Distributed:bendo", true, "Project"
```

RemoveUnusedDefinitions

Removes any unused project definitions.

<table>
<tr><td><b>UI Access</b></td><td><b>Tools</b> &gt; <b>Project Tools</b> &gt; <b>Remove Unused Definitions</b>.</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;<i>Definitions</i>&gt;</td><td>Array</td><td>Definitions to be removed, such as materials and surface materials.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	RemoveUnusedDefinitions(<*Definitions*>)
Python Example	

```
oProject.RemoveUnusedDefinitions(
  [
    [
      "NAME:Materials",
      "Al-Extruded"
    ],
    [
      "NAME:SurfaceMaterials",
      "Steel-oxidised-surface"
    ]
  ])
```

SetPropertyValue

Sets the value of a single propertybelonging 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.

<table>
<tr><th><b>UI Access</b></th><th>N/A</th></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;propTab&gt;</i></td><td>String</td><td>One of the following, where tab titles are shown in parentheses:
• PassedParameterTab ("Parameter Values")
• DefinitionParameterTab (Parameter Defaults")
• LocalVariableTab ("Variables" or "Local Variables")
• ProjectVariableTab ("Project variables")
• ConstantsTab ("Constants")
• BaseElementTab ("Symbol" or "Footprint")
• ComponentTab ("General")
• Component("Component")
• CustomTab ("Intrinsic Variables")
• Quantities ("Quantities")
• Signals ("Signals")</td></tr>
<tr><td><i>&lt;propServer&gt;</i></td><td>String</td><td>An object identifier, generally returned from another script method, such as CompInst@R;2;3</td></tr>
<tr><td><i>&lt;propName&gt;</i></td><td>String</td><td>Name of the property.</td></tr>
<tr><td><i>&lt;propValue&gt;</i></td><td>String</td><td>The value for the property</td></tr>
</table>
</td></tr>
</table>

Return Value	None.

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

UpdateDefFromBlock

Updates a material definition from block text (same definition format as would be contained in the material library file) by library type (using definition folder name). This scripting command directly supports the .AMAT (or .ASURF) definition formats.

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td>&lt;<i>targetDefName</i>&gt;</td><td>String</td><td>Name of the target definition, i.e. the name of the material to update.</td></tr>
<tr><td>&lt;<i>defBlock</i>&gt;</td><td>String</td><td>Text of the new material definition in block format (string); this block could use a new definition name, which will cause a rename as part of the update.</td></tr>
<tr><td>&lt;<i>defFolderName</i>&gt;</td><td>String</td><td>Library type (by definition, folder name).</td></tr>
<tr><td>&lt;<i>newTimeStamp</i>&gt;</td><td>String</td><td>New timestamp string (time_t as integer, number of seconds since 1/1/1970 12:00am), default is current time.</td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>A property scripting object for the definition.</td></tr>
</table>

Python	UpdateDefFromBlock(<*targetDefName*>, <*defBlock*>, <*defFolderName*>, <*newTimeStamp*>)

Syntax	

Python Example	oProject = oDesktop.NewProject()
	oProject.InsertDesign("HFSS", "HFSSDesign1", "DrivenModal", "")
	oDesign = oProject.SetActiveDesign("HFSSDesign1")
	oEditor = oDesign.SetActiveEditor("3D Modeler")
	oDefinitionManager = oProject.GetDefinitionManager()
	defBlock = "$begin 'vacuum2' $begin 'AttachedData' $begin 'MatAppearanceData' property_data='appearance_data' Red=230 Green=230 Blue=230 Transparency=0.95 $end 'MatAppearanceData' $end 'AttachedData' simple('permittivity', 1) ModTime=1499970477 $end 'vacuum2'"
	added = oDefinitionManager.AddDefinitionFromBlock(defBlock, "Materials", "10101010", True)
	addedName = ''
	  if isinstance(added, basestring):
	    addedName = added
	  elif isinstance(added, list):
	    addedName = added[0]
	else:
	  addedName = added.GetName().replace("Materials:", "")
	AddInfoMessage(os.path.basename(__file__) + " result: " + addedName)

```
materialNameInQuotes = "\"" + addedName + "\""

# rename vacuum2 to vacuum3

newDefBlock = "$begin 'vacuum3' $begin 'AttachedData' $begin 'MatAppearanceData' property_data='appearance_data' Red=230 Green=230 Blue=230 Transparency=0.95 $end 'MatAppearanceData' $end 'AttachedData' simple('permittivity', 1) ModTime=1499970477 $end 'vacuum3'"

updatedObj = UpdateDefFromBlock(addedName, newDefBlock, "Materials")
```

UpdateDefFromBlockEx

Updates a material definition from block text (same definition format as would be contained in the material library file) by library type (using definition folder name). This scripting command directly supports the .AMAT (or .ASURF) definition formats.This resembles the UpdateDefFromBlock command, except UpdateDefFromBlockEx also allows for:

- updating a definition using a block where the name has changed without triggering a rename of the definition being updated
- updating multiple definitions from the same block
- renaming a definition to something other than the name in the block

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i>&lt;targetDefName&gt;</i></td><td>String</td><td>Name of the target definition, i.e. the name of the material to update.
Name of the target definition, i.e. the name of the material to update (string).
Name of the target definition, i.e. the name of the material to update</td></tr>
</table>
</td></tr>
</table>

		(string).		
<desiredDefName>	String	New name to use for a rename. If this is the same as *targetDefName*, it means the definition is not being renamed. If this is blank, the block name will be used for this purpose instead.	New name to use for a rename. If this is the same as targetDefName, it means the definition is not being renamed. If this is blank, the block name will be used for this purpose instead.	New name to use for a rename. If this is the same as targetDefName, it means the definition is not being renamed. If this is blank, the block name will be used for this purpose instead.
<defBlock>	String	Text of the updated material definition in block form. (Same form as the defBlock input to the AddDefinitionFromBlock and UpdateDefFromBlock commands, for example.) Note that, unlike in the UpdateDefFromBlock command, the name of the definition in the block can be ignored, provided that desiredDefName is set.		
<defFolderName>	String	Library type (by definition, folder name).		
<newTimeStamp>	String	New timestamp string (time_t as integer, number of seconds since 1/1/1970 12:00am), default is current time.		
<continueIfConflict>	Boolean	A flag for whether to continue processing if there is some sort of conflict. A conflict only comes from the		

			target definition being in use, such as a material assigned to a particular part, and if trying to rename the target definition. If this argument is True, and there's a rename conflict, the tactic is to add/update the definition with the new name and leave the target definition untouched. If this argument is False, and there's a rename conflict, the command results in an exception.
Return Value	A property scripting object (extended definition object) for the definition. (In the event of a failure, the command throws an exception and there will be no result.)		

Python Syntax	UpdateDefFromBlockEx(<*targetDefName*>, <*desiredDefName*>,<*defBlock*>, <*defFolderName*>, <*newTimeStamp*>, <*continueIfConflict*>)

Python Example

```
# Note that the material block name will be ignored in this case
defBlock = "$begin 'unused_name' $begin 'AttachedData' $begin \
     'MatAppearanceData' property_data='appearance_data' \
     Red=255 Green=0 Blue=0 Transparency=0.19 $end \
     'MatAppearanceData' $end 'AttachedData' \
     simple('permittivity', 1.9) ModTime=1509090909 \
     $end 'unused_name'"
try:
  updated = oDefinitionManager.UpdateDefFromBlockEx("red_material",
         "red_material",
         defBlock,
         "Materials",
         "1609090909")
  updatedName = updated.GetName().replace("Materials:", "")
  AddInfoMessage("Result: " + updatedName)
except:
  AddErrorMessage("Unexpected error in UpdateDefFromBlockEx")
```

UpdateDefinitions

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

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

Python Syntax	UpdateDefinitions()
Python Example	`oProject.UpdateDefinitions()`

Component Manager Script Commands

The component manager provides access to components in a project. The manager object is accessed via the definition manager.

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

```
Set oComponentManager = oDefinitionManager.GetManager("Component")
```

The topics for this section include:

Add [component manager]

*Use:*Add a component

Command: Tools > Edit Configured Libraries > Components > Add Component

Syntax:

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

"NumberProp:=", <NumberInfo>,

"SeparatorProp:=", <SeparatorInfo>,

"ValueProp:=", <ValueInfo>,

"MenuProp:=", <MenuInfo>),

"VPointProp:=", <VPointInfo>,

"PointProp:=", <PointInfo>),

Array("Quantities",

"QuantityProp:=", <QuantityPropInfo>...),

Array("NAME:CosimDefinitions",

<CosimDefInfo>,

<CosimDefInfo>...)

Return Value:<string>

// composite name of the component.

// If the name requested conflicts with the name of an existing

// component, the requested name is altered to be unique.

// The name returned reflects any change made to be unique.

`*Parameters:*<ComponentName>:`

<string> // simple name of the component

<ComponentInfo>:

```
Array("Type:=", <TypeInfo>,
"NumTerminals:=", <int>,
"DataSource:=", <string>,
"ModifiedOn:=", <ModifiedOnInfo>,
"Manufacturer:=", "<string>,
"Symbol:=", <string>,
"Footprint:=", <string>,
"Description:=", <string>,
"InfoTopic:=", <string>,
"InfoHelpFile:=", <string>,
"IconFile:=", <string>,
"LibraryName:=", "",
"OriginalLocation:=", "Project", // Project Location
"Author:=", <string>,
"OriginalAuthor:=", <string>,
"CreationDate:= ", <int>)
```

<TypeInfo>:

An integer that is the or-ing of bits for each product listed below. The default setting is 0xffffffff (4294967295) which indicates valid for all products. In the component editing dialog, checking different boxes in the "Specify products for which this component is valid" grid control sets specific flags that correspond to the following hex/decimal settings:

Nexxim -- 100 binary, 4 decimal, 0x4

SIwaveDeNovo -- 1000 binary, 8 decimal, 0x8

Simplorer -- 10000 binary, 16 decimal, 0x10

MaxwellCircuit -- 100000 binary, 32 decimal, 0x20

<ModifiedOnInfo>:

An integer that corresponds to the number of seconds that have elapsed since 00:00 hours, Jan 1, 1970 UTC from the system clock.

<TerminalInfo>:

Array(<string>, // symbol pin

<string> // footprint pin

<string >, // gate name

<bool>, // shared

<int>, // equivalence number

<int>, // what to do if unconnected: flag as error:0, ignore:1

<string> // description

<Nature>)

<Nature>:

```
<string> // content varies as follows

Nexxim/Circuit:
"Electrical" // the only choice

Simplorer:
// several choices
"Electrical", "Magnetic", "Fluidic", "Translational",
"Translational_V", "Rotational", "Rotational_V",
""Radiant", "Thermal", or <VHDLPackageName>

<VHDLPackageName>:
<string> // in the form <Library>.<Package>

<Library>:
<string> // name of the VHDL library

<Package>:
<string> // name of the VHDL package
```

```
<VariableInfo>:
Array(<string>, // name
<FlagLetters>,
<string>, // description
"CB:=", <string>, // optional - script for call back
<string>) // value: number, variable, or expression

<FlagLetters>:
<string> // "D" - has description parameter,
// "RD" - readonly & has description parameter,
// or "RHD" - readonly, hidden, & has description parameter

<CheckBoxInfo>:
Array(<string>, // name
<FlagLetters>,
<string>, // description
"CB:=", <string>, // optional - script for call back
<bool>) // value: true or false

<ButtonInfo>:
Array(<string>, // name
<FlagLetters>,
<string>, // description
<string>, // button title
<string>, // extra text
<ClientID>,
"ButtonPropClientData:= ", <ClientDataArray>)

<ClientID>:
<int> // specifies Button Prop Client
// 0 - unknown, ButtonPropClientData
// array will be empty
// 1 - Netlist Prop Client
// 2 - not used
// 3 - File Name Prop Client

<ClientDataArray>:
varies with <ClientID>

<ClientId> is 0 or 1: empty array
Array()
```

<ClientID> is 3:

Array("InternalFormatText:=", "<prefix><RelativePath>")

<prefix>:

<string> // "<Project>", "<PersonalLib>", "<UserLib>", or "<SysLib>"

<RelativePath>:

<string> // relative path to file from <prefix>

<TextInfo>:

Array(<string>, // name

<FlagLetters>,

<string>, // description

"CB:=", <string>, // optional - script for call back

<string>) // value: a text string

<NumberInfo>:

Array(<string>, // name

<FlagLetters>,

<string>, // description

```
"CB:=", <string>, // optional - script for call back
<real>, // value: a number
<string>) // units

<SeparatorInfo>:
Array(<string>, // name
<FlagLetters>,
<string>, // description
"CB:=", <string>, // optional - script for call back
<string>) // value: a text string

<ValueInfo>:
Array(<string>, // name
<FlagLetters>,
<string>, // description
"CB:=", <string>, // optional - script for call back
<string>) // value: a number, variable or expression


<MenuPropInfo>:
Array(<string>, // name
<FlagLetters>,
<string>, // description
<string>, // menu choices - separated by commas
<int>) // 0 based index of current menu choice

<VPointInfo>:
Array(<string>, // name
<FlagLetters>,
<string>, // description
"CB:=", <string>, // optional - script for call back
<string>, // x value: number with length units
<string>) // y value: number with length units


<PointInfo>:
Array(<string>, // name
<FlagLetters>,
<string>, // description
"CB:=", <string>, // optional - script for call back
<real>, // x value
<real>) // y value


<QuantityPropInfo>:
Array(<string>, // name
<FlagLetters>,
<string>, // description
<string>, // value
<TypeString>,
<TypeStringDependentInfo>)

<TypeString>:
<string> // "Across", "Through", or "Free"

<TypeStringDependentInfo>:

<TypeString> is "Free" :
<string>, // direction: "In", "Out", "InOut", or "DontCare"
// Following <string> is not present if direction is "DontCare"
<string> // when to calculate: "BeforeAnalogSolver",
// "BeforeStateGraph", "AfterStateGraph", or "DontCareWhen"

<TypeString> is "Across" or "Through":
<int>, // terminal 1
<int> // terminal 2
```

```
<CosimDefInfo>:
Array("NAME:CosimDefinition",
"CosimulatorType:=", <int>,
"CosimDefName:=", <string> // "HFSS 3D Layout", "Circuit",
// "Custom", or "Netlist"
"IsDefinition:=", <bool>,
final array member(s) vary with CosimDefName)

final array members for HFSS 3D Layout:
"CosimStackup:=", <string>,
"CosimDmbedRatio:=", <int>

final array members for Circuit:
"ExportAsNport:=", <int>,
"UsePjt:=", <int>

final array member for Custom:
"DefinitionCompName:=", <string>

final array member for Netlist:
"NetlistString:=", <string>
```

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

	"TextProp:=", <TextInfo>, "NumberProp:=", <NumberInfo>, "SeparatorProp:=", <SeparatorInfo>, "ValueProp:=", <ValueInfo>, "MenuProp:=", <MenuInfo>, "VPointProp:=", <VPointInfo>, "PointProp:=", <PointInfo>], ["Quantities", "QuantityProp:=", <QuantityPropInfo>...], ["NAME:CosimDefinitions", <CosimDefInfo>, <CosimDefInfo>...]]
Python Example	oComponentManager.Add(["NAME:Component", "Info:=", ["Type:=", 0, "NumTerminals:=", 0, "DataSource:=", "", "ModifiedOn:=", 1467910752, "Manufacturer:=", "",

	"Symbol:=", "Component", "ModelNames:=", "", "Footprint:=", "", "Description:=" , "", "InfoTopic:=", "", "InfoHelpFile:=", "", "IconFile:=", "", "Library:=", "", "OriginalLocation:=", "Project", "IEEE:=", "", "Author:=", "", "OriginalAuthor:=", "", "CreationDate:=", 1467910746, "ExampleFile:=", ""], "Refbase:=", "U", "NumParts:=", 1, "ModSinceLib:=", True, "CompExtID:=", 2])

AddNPortData [component manager]

Use: Adds a component using the specified data

Command: Project Menu > Add Model > Add Nport Model

Syntax: AddNPortData Array("NAME:<ComponentDataName>",

```
        "ComponentDataType:=", "NportData",
        "name:=", <string>, // Name of the item
        "filename:=", <string>, // Path to the file to find the data
        "numberofports:=", <int>,
        "filelocation:=", <LocationType>,
        "domain:=", <string>, // "time" or "frequency"
        "datamode:=", <string> // "EnterData", "Import", or "Link"
        "devicename:=", <string>,
        "ImpedanceTab:=", <bool>,
        "NoiseDataTab:=", <bool>,
        "DCBehaviorTab:=", <bool>,
        "SolutionName:=", <string>,
        "displayformat:=", <DisplayInfo>,
        "datatype:=", <string>, // "SMatrix", "YMatrix", or "ZMatrix"
        "ShowRefPin:=", <bool>,
        "RefNodeCheckbox:=", <bool>, ...
        <ProductOptionsInfo>)
```

Return Value: <string>

// composite name of the component.

// If the name requested conflicts with the name of an existing

// component, the requested name is altered to be unique.

// The name returned reflects any change made to be unique.

`*Parameters:* <ComponentDataName>:`

<string> // simple name of the component

<LocationType>:

<string> // one of "UsePath", "PersonalLib", "UserLib", "SysLib",

// or "Project".

<dcInfo>:

<string> // one of "DCOpen", "DCShort", "DCShShort",

// "DCNone", or "DCEmpty".

<DisplayInfo>:

<string> // one of "MagnitudePhase", "RealImaginary",

// or "DbPhase".

```
<ProductOptionsInfo>:
// The remaining parameters differ by product

// HFSS 3D Layout - doesn't support interpolation/DC behavior
"DCOption:=", -1,
"InterpOption:=", -1,
"ExtrapOption:=",-1,
"DataType:=", 0

// Nexxim
"DCOption:=", <NexximDCOption>,
"InterpOption:=", <NexximInterpOption>,
"ExtrapOption:=", <NexximExtrapOption>,
"DataType:=", 2

<NexximDCOption>:
<int> // 0 : Zero Padding
// 1 : Same as last point
// 2 : Linear extrapolation from last 2 points
// 3 : Constant magnitude, linear phase extrapolation
```

// 4 : Leave all signal lines open circuited

// 5 : Short all signal lines together

// 6 : Short all signal lines to ground

<NexximInterpOption>:

<int> // 0 : Step

// 1 : Linear

<NexximExtrapOption>:

<int> // 0 : Zero padding

// 1 : Same as last point

// 2 : Linear extrapolation from last 2 points

// 3 : Constant magnitude, linear phase extrapolation

<CircuitDCOption>:

<int> // 0 : Leave all signal lines open circuited

// 1 : Short all signal lines together

// 2 : Short all signal lines to ground

// 3 : Extrapolate from data provided (not recommended)

<CircuitInterpOption>:

<int> // 0 : Linear

// 1 : Cubic spline

// 2 : Rational polynomial

<CircuitExtrapOption>:

<int> // 0 : Same as interpolation

// 1 : Zero padding

// 2 : Same as last point

AddSolverOnDemandModel

Use: This method looks for a local component of the name passed in, and to this component it adds an SOD model definition using the information passed in the VARIANT. It returns the name of the SOD model added.

Parameters: BSTR component name.

Parameters: VARIANT which is the SOD model data.

Return Value: Returns the name of the model added.

ClearSolutionCache [component manager]

Use: Clear the solution cache for dynamic link component.

*Command:*Each of the following commands will clear the solution cache:

— **Dynamic Link Item RCM > Clear Solution Cache**

— **Dynamic Link Component in schematic RCM > Clear Solution Cache**

Syntax: ClearSolutionCache <*Component Name* >

Return Value: None

Parameters: `<component name> is the name of the dynamic link component`

Edit [component manager]

Modifies an existing component

Command: Tools > Edit Configured Libraries > Components > Edit Component

Syntax: Edit <ComponentName>,

```
        Array("NAME:<NewComponentName>",
        "Info:=", <ComponentInfo>,
        "RefBase:=", <string>, // reference designator
        "NumParts:=", <int>, // parts per component
        "OriginalComponent:=", <string>
        "Terminal:=", <TerminalInfo>,
        "Terminal:=", <TerminalInfo>, ...
        // The remaining parameters are optional
        Array("NAME:Parameters", // any combo of the following
        "VariableProp:=", <VariableInfo>,
        "CheckboxProp:=", <CheckBoxInfo>,
        "ButtonProp:=", <ButtonInfo>,
        "TextProp:=", <TextInfo>,
        "NumberProp:=", <NumberInfo>,
        "SeparatorProp:=", <SeparatorInfo>,
"ValueProp:=", <ValueInfo>,
"MenuProp:=", <MenuInfo>),
Array("NAME:Properties", // any combo of the following
"CheckboxProp:=", <CheckBoxInfo>,
"TextProp:=", <TextInfo>,
"NumberProp:=", <NumberInfo>,
"SeparatorProp:=", <SeparatorInfo>,
"ValueProp:=", <ValueInfo>,
"MenuProp:=", <MenuInfo>),
"VPointProp:=", <VPointInfo>,
"PointProp:=", <PointInfo>),
Array("Quantities",
"QuantityProp:=", <QuantityPropInfo>...),
Array("NAME:CosimDefinitions",
<CosimDefInfo>,
<CosimDefInfo>...)
```

Return Value: <string>

```
// composite name of the component.
// If the name requested conflicts with the name of an existing
// component, the requested name is altered to be unique.
// The name returned reflects any change made to be unique.
```

Parameters: `<ComponentName>:`

<string> // composite name of the component to edit

<NewComponentName>:

<string> // new simple name for the component

<ComponentInfo>:

Array("Type:=", <TypeInfo>,

"NumTerminals:=", <int>,

"DataSource:=", <string>,

"ModifiedOn:=", <ModifiedOnInfo>,

"Manufacturer:=", "<string>,

"Symbol:=", <string>,

"Footprint:=", <string>,

"Description:=", <string>,

"InfoTopic:=", <string>,

"InfoHelpFile:=", <string>,

"IconFile:=", <string>,

"LibraryName:=", <string>,

"OriginalLocation:=", <string>, // Project Location

"Author:=", <string>,

"OriginalAuthor:=", <string>,

"CreationDate:= ", <int>)

<TypeInfo>:

An integer that is the or-ing of bits for each product listed below. The default setting is 0xffffffff (4294967295) which indicates valid for all products. In the component editing dialog, checking different boxes in the "Specify products for which this component is valid" grid control sets specific flags that correspond to the following hex/decimal settings:

Nexxim -- 100 binary, 4 decimal, 0x4

SIwaveDeNovo -- 1000 binary, 8 decimal, 0x8

Simplorer -- 10000 binary, 16 decimal, 0x10

MaxwellCircuit -- 100000 binary, 32 decimal, 0x20

<ModifiedOnInfo>:

An integer that corresponds to the number of seconds that have elapsed since 00:00 hours, Jan 1, 1970 UTC from the system clock.

<TerminalInfo>:

Array(<string>, // symbol pin

<string> // footprint pin

<string >, // gate name

<bool>, // shared

```
<int>, // equivalence number
<int>, // what to do if unconnected: flag as error:0, ignore:1
<string>, // description
<Nature>)

<Nature>:
<string> // content varies as follows

Nexxim/Circuit:
"Electrical" // the only choice

Simplorer:
// several choices
"Electrical", "Magnetic", "Fluidic", "Translational",
"Translational_V", "Rotational", "Rotational_V",
""Radiant", "Thermal", or <VHDLPackageName>

<VHDLPackageName>:
<string> // in the form <Library>.<Package>
```

<Library>:

<string> // name of the VHDL library

<Package>:

<string> // name of the VHDL package

<VariableInfo>:

Array(<string>, // name

<FlagLetters>,

<string>, // description

"CB:=", <string>, // optional - script for call back

<string>) // value: number, variable, or expression

<FlagLetters>:

<string> // "D" - has description parameter,

// "RD" - readonly & has description parameter,

// or "RHD" - readonly, hidden, & has description parameter

<CheckBoxInfo>:

Array(<string>, // name

```
<FlagLetters>,
<string>, // description
"CB:=", <string>, // optional - script for call back
<bool>) // value: true or false

<ButtonInfo>:
Array(<string>, // name
<FlagLetters>,
<string>, // description
<string>, // button title
<string>, // extra text
<ClientID>,
"ButtonPropClientData:= ", <ClientDataArray>)

<ClientID>:
<int> // specifies Button Prop Client
// 0 - unknown, "ButtonPropClientData
// array will be empty
// 1 - Netlist Prop Client
// 2 - not used
// 3 - File Name Prop Client

<ClientDataArray>:
varies with <ClientID>

<ClientID> is 0 or 1: empty array
Array()

<ClienID> is 3:
Array("InternalFormatText:=", "<prefix><RelativePath>")

<prefix>:
<string> // "<Project>", "<PersonalLib>", "<UserLib>", or "<SysLib>"

<RelativePath>:
<string> // relative path to file from <prefix>

<TextInfo>:
Array(<string>, // name
<FlagLetters>,
<string>, // description
```

```
"CB:=", <string>, // optional - script for call back
<string>) // value: a text string

<NumberInfo>:
Array(<string>, // name
<FlagLetters>,
<string>, // description
"CB:=", <string>, // optional - script for call back
<real>, // value: a number
<string>) // units

<SeparatorInfo>:
Array(<string>, // name
<FlagLetters>,
<string>, // description
"CB:=", <string>, // optional - script for call back
<string>) // value: a text string

<ValueInfo>:
Array(<string>, // name
<FlagLetters>,
<string>, // description
"CB:=", <string>, // optional - script for call back
<string>) // value: a number, variable or expression
```

```
<MenuPropInfo>:
Array(<string>, // name
<FlagLetters>,
<string>, // description
<string>, // menu choices - separated by commas
<int>) // 0 based index of current menu choice
```

```
<VPointInfo>:
Array(<string>, // name
<FlagLetters>,
<string>, // description
"CB:=", <string>, // optional - script for call back
<string>, // x value: number with length units
<string>) // y value: number with length units
```

```
<PointInfo>:
Array(<string>, // name
<FlagLetters>,
<string>, // description
"CB:=", <string>, // optional - script for call back
<real>, // x value
<real>) // y value

<QuantityPropInfo>:
Array(<string>, // name
<FlagLetters>,
<string>, // description
<string>, // value
<TypeString>,
<TypeStringDependentInfo>)

<TypeString>:
<string> // "Across", "Through", or "Free"

<TypeStringDependentInfo>:

"Free" :
<string>, // direction: "In", "Out", "InOut", or "DontCare"
```

```
// Following <string> is not present if direction is "DontCare"
<string> // when to calculate: "BeforeAnalogSolver",
// "BeforeStateGraph", "AfterStateGraph", or "DontCareWhen"

"Across" or "Through"
<int>, // terminal 1
<int> // terminal 2


<CosimDefInfo>:
Array("NAME:CosimDefinition",
"CosimulatorType:=", <int>,
"CosimDefName:=", <string> // "HFSS3D", "Circuit",
// "Custom", or "Netlist"
"IsDefinition:=", <bool>,
final array member(s) vary with CosimDefName)


final array members for HFSS 3D Layout:
"CosimStackup:=", <string>,
"CosimDmbedRatio:=", <int>


final array members for Circuit:
"ExportAsNport:=", <int>,
```

"UsePjt:=", <int>

final array member for Custom:

"DefinitionCompName:=", <string>

final array member for Netlist:

"NetlistString:=", <string>

Python Syntax	Edit <ComponentName>, ["NAME:<NewComponentName>", "Info:=", <ComponentInfo>, "RefBase:=", <string>, // reference designator "NumParts:=", <int>, // parts per component "OriginalComponent:=", <string> "Terminal:=", <TerminalInfo>, "Terminal:=", <TerminalInfo>, ... #The remaining parameters are optional ["NAME:Parameters", // any combo of the following "VariableProp:=", <VariableInfo>, "CheckboxProp:=", <CheckBoxInfo>, "ButtonProp:=", <ButtonInfo>,

	"TextProp:=", <TextInfo>, "NumberProp:=", <NumberInfo>, "SeparatorProp:=", <SeparatorInfo>, "ValueProp:=", <ValueInfo>, "MenuProp:=", <MenuInfo>], ["NAME:Properties", # any combo of the following "CheckboxProp:=", <CheckBoxInfo>, "TextProp:=", <TextInfo>, "NumberProp:=", <NumberInfo>, "SeparatorProp:=", <SeparatorInfo>, "ValueProp:=", <ValueInfo>, "MenuProp:=", <MenuInfo>), "VPointProp:=", <VPointInfo>, "PointProp:=", <PointInfo>), ["Quantities", "QuantityProp:=", <QuantityPropInfo>...], ["NAME:CosimDefinitions", <CosimDefInfo>, <CosimDefInfo>...])

Python Example	`name = oComponentManager.Edit ("Simplorer Circuit Elements\BJTs:Level01_NPN", _`
	`["NAME:Level01_NPN", "Info:=", ["Type:=", 4294901764,_`

```
"NumTerminals:=", 3, "DataSource:=", "Ansoft built-in component",_
"ModifiedOn:=", 1152722112, "Manufacturer:=", "", _
"Symbol:=", "nexx_bjt_npn", "Footprint:=", "", _
"Description:=", "BJT, GP, NPN", "InfoTopic:=", "NXBJT1.htm", _
"InfoHelpFile:=", "nexximcomponents.chm", "IconFile:=", "bjtsn.bmp", _
"Library:=", "Nexxim Circuit Elements\BJTs",_
"OriginalLocation:=", "SysLibrary ", "Author:=", "", _
"OriginalAuthor:=", "", "CreationDate:=", 1152722102], _
"Refbase:=", "Q", "NumParts:=", 1, "Terminal:=", ["collector", _
"collector", "A", false, 6, 0, "", "Electrical"], _
"Terminal:=", ["base", "base", "A", false, _
7, 0, "", "Electrical"], "Terminal:=", ["emitter", _
"emitter", "A", false, 8, 0, "", "Electrical"], _
["NAME:Parameters", "TextProp:=", ["LabelID", _
"HD", "Property string for netlist ID", _
"Q@ID"], "TextProp:=", ["MOD", "D", _
"Name of model data reference", "required"], _
"VariableProp:=", ["AREA", "D", _
"Emitter area multiplying factor, which affects
```

	currents, resistances, and capacitances", "1"], _ "VariableProp:=", ["AREAB", "D", "Base AREA", _ "1"], "VariableProp:=", ["AREAC", "D", "Collector AREA", _ "1"], "VariableProp:=", ["DTEMP", "D", _ "The difference between element and circuit temperature (deg Cel)", _ "0"], "VariableProp:=", ["M", "D", _ "Multiplier factor to simulate multiple BJTs in parallel", _ "1"], "ButtonProp:=", ["NexximNetlist", "HD", "", _ "Q@ID %0 %1 %2 *MOD(@MOD) *AREA(AREA=@AREA)"& _ " *AREAB(AREAB=@AREAB) *AREAC(AREAC=@" &_ "AREAC) *DTEMP(DTEMP=@DTEMP) *M(M=@M)", _ "Q@ID %0 %1 %2 *MOD(@MOD) " & "*AREA(AREA=@AREA) *AREAB(AREAB=@AREAB) *AREAC(AREAC=@" & _ "AREAC) *DTEMP(DTEMP=@DTEMP) *M(M=@M)", 1, _ "ButtonPropClientData:=", []], "TextProp:=", ["ModelName", "HD", "", "Q"]]])
Python Example 2	name2 = oComponentManager.Edit ("MyComponent",_ (["NAME:MyOtherComponent","Info:=", ["Type:=", 4294901767, _

```
"NumTerminals:=", 2, "DataSource:=", "", _

"ModifiedOn:=", 1071096503, "Manufacturer:=", "Ansoft", _

"Symbol:=", "bendo", "Footprint:=", "BENDO", _

"Description:=", "", "InfoTopic:=", "", _

"InfoHelpFile:=", "", "IconFile:=", "", _

"LibraryName:=", "", "OriginalLocation:=", "Project", _

"Author:=", "", "OriginalAuthor:=", "", _

"CreationDate:= ", 1147460679], "Refbase:=", "U", _

"NumParts:=", 1, "OriginalComponent:=", "", _
```

```
"Terminal:=", ["n1", "n1", "A", false, 0, 0, "", _

"Electrical"], "Terminal:=", ["n2", "n2", "A", _

false, 1, 0, "", Electrical"], ["NAME:Parameters", _

"MenuProp:=", ["CoSimulator", "D", "", _

"Default,Custom,Netlist", 0], "ButtonProp:=", ["CosimDefinition", _

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

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

"CosimulatorType:=", 0, "CosimDefName:=", "HFSS3D", _

"IsDefinition:=", true, "CosimStackup:=", "Layout stackup", _

"CosimDmbedRatio:=", 3], ["NAME:CosimDefinition", _
"CosimulatorType:=", 1, "CosimDefName:=", "", _
"IsDefinition:=", true, "ExportAsNport:=", 0, _
"UsePjt:=", 0], ["NAME:CosimDefinition", _
"CosimulatorType:=", 2, "CosimDefName:=", "Custom", _
"IsDefinition:=", true, "DefinitionCompName:=", ""], _
["NAME:CosimDefinition", "CosimulatorType:=", 3, _
"CosimDefName:=", "Netlist", "IsDefinition:=", true, _
"NetlistString:=", ""]]])
```

EditSolverOnDemandModel

Use: This method looks for a local component of the name passed in, and in this component it looks for an SOD model using the name passed in the second BSTR. It modifies the SOD model using the data in the VARIANT. It returns the name of the SOD model edited.

Return Value: Returns the name of the model edited.

Parameters: `BSTR component name.`

Parameters: `BSTR SOD model name.`

Parameters: `VARIANT which is the new SOD model data (can include changed name).`

EditWithComps [component manager]

Edit an existing component.

Command: None

Syntax: EditWithComps <ComponentName>,

```
        Array("NAME:<NewComponentName>",
        "ModTime:=", <ModifiedTimeInfo>,
        "Library:=", <string>, // Library name
        "LibLocation:=", <string>, // Project Location
        <PinDefInfo>,
        <PinDefInfo>,... // optional, to define pins
        <GraphicsDataInfo>, // optional, to define graphics
        <PropDisplayMapInfo>), // optional, to define property displays
        Array(<ListOfComponentNames>) // Component names
```

Return Value: <string>

// composite name of the component.

// If the name requested conflicts with the name of an existing

// component, the requested name is altered to be unique.

// The name returned reflects any change made to be unique.

Parameters: <ComponentName>:

<string> // composite name of the component being edited

<NewComponentName>:

<string> // new simple name for the component

<ModifiedOnInfo>:

An integer that corresponds to the number of seconds that have elapsed

since 00:00 hours, Jan 1, 1970 UTC from the system clock.

<PinDefInfo>:

Array("NAME:PinDef",

"Pin:=", Array (<string>, // pin name

<real>, // x location

<real>, // y location

```
<real>, // angle in radians
<PinType>,
<real>, // line width
<real>, // line length
<bool>, // mirrored
<int>, // color
<bool>, // true if visible, false if not
<string>, // hidden net name
<OptionalPinInfo>, // optional info
<PropDisplayMapInfo>)) // optional

<PinType>:
<string> // "N" : normal pin
// "I" : input pin
// "O" : output pin

<OptionalPinInfo>:
// Specify both or neither
<bool>, // true if name is to be shown
<bool>, // true if number is to be shown
```

```
<PropDisplayMapInfo>:
Array("NAME:PropDisplayMap",
<PropDisplayInfo>,
<PropDisplayInfo>,...)

<PropDisplayInfo>:
<NameString>, Array(<DisplayTypeInfo>,
<DisplayLocationInfo>,
<int>, // optional, level number
<TextInfo>)

<NameString>:
<string> // PropertyName:=, where PropertyName is the name of
// the property to be displayed

<DisplayTypeInfo>:
<int> // 0 : No display
// 1 : Display name only
// 2 : Display value only
// 3 : Display both name and value
// 4: Display evaluated value only
// 5: Display both name and evaluated value

<DisplayLocationInfo>:
<int> // 0 : Left
// 1 : Top
// 2 : Right
// 3 : Bottom
// 4 : Center
// 5 : Custom placement

<GraphicsDataInfo>:
Array("NAME:Graphics",
// one or more of the following
<RectInfo>,
<CircleInfo>,
<ArcInfo>,
<LineInfo>,
<PolygonInfo>,
<TextInfo>,
<ImageInfo>)
```

```
<RectInfo>:
"Rect:=", Array(<real>, // line width
<int>, // fill pattern
<int>, // color
<real>, // angle, in radians
<real>, // x position of center
<real>, // y position of center
<real>, // width
< real>) // height
```

```
<CircleInfo>:
"Circle:=", Array(<real>, // line width
<int>, // fill pattern
<int>, // color
<real>, // x position of center
<real>, // y position of center
< real>) // radius
```

```
<ArcInfo>:
```

```
"Arc:=", Array(<real>, // line width
<int>, // line pattern
<int>, // color
<real>, // x position of center
<real>, // y position of center
< real>, // radius
<real>, // start angle, in radians
<end>) // end angle, in radians

<LineInfo>:
"Line:=", Array(<real>, // line width
<int>, // line pattern
<int>, // color
<PointInfo>, // must specify at least 2 points
<PointInfo>...)
<PointInfo>:
<real>, // x position
<real> // y position

<PolygonInfo>:
"Polygon:=", Array(<real>, // line width
<int>, // fill pattern
<int>, // color
<PointInfo>, // must specify at least 3 points
<PointInfo>...)

<TextInfo>:
"Text:=", Array(<real>, // x position
<real>, // y position
<real>, // angle, in radians
<Justification>,
<bool>, // is plotter font
<string>, // font name
<int>, // color
<string>) // text string

<Justification>:
<int> // 0 : left top
// 1 : left base
// 2 : left bottom
// 3 : center top
// 4 : center base
// 5 : center bottom
// 6 : right top
// 7 : right base
// 8 : right bottom

<ImageInfo>:
"Image:=", Array(<RectInfo>,
<ImageData>,
<bool>) // is mirrored

<ImageData>:
<string>, // file path
<int>, // 0 : use the file path and link to it
// 1 : ignore file path and use next parameter
<string> // text data, only present if preceding int is 1

<ListOfComponentNames>:
<string>,<string> ...
// The list may be empty. When not empty, each string that is listed is a component
// that references the component to be edited. Prior to editing, a clone of the component is
```

// made, and the components that are listed are modified so that they now refer to

// the clone.

Export [component manager]

Export component(s) to a library

Command: Tools > Edit Configured Libraries > Components > Export to Library

Syntax: Export Array("NAME:<LibraryName>",

<ComponentName>,

<ComponentName>...),

<LibraryLocation>

Return Value: None

Parameters: <LibraryName>:

<string> // name of the library

<ComponentName>:

<string> // composite name of the component to export

<LibraryLocation>:

<string> // location of the library in <LibraryName>

// One of "Project", "PersonalLib", or "UserLib"

Python Syntax	Export(["NAME:<LibraryName>", <ComponentName>, <ComponentName>...], <LibraryLocation>)

Python Example	`oComponentManager.Export(["NAME:mylib",`
	`"Simplorer Circuit Elements\BJTs:Level01_NPN"],`
	`"PersonalLib")`

GetNames [component manager]

Returns the names of the components (used and unused) in a design. The following script command, **IsUsed,** can then be used to separate used and unused components.

Command: None

Syntax: GetNames()

Return Value: An array of strings

Parameters: None

Python Syntax	GetNames()
Python Example	`componentNames = oComponentManager.GetNames()`

GetNPortData [component manager]

Returns NPort data for the component with the specified name.

Command: None

Return Value: Variant array, whose contents depend on the type of component. The array will be empty if the component does not have NPort data. See the syntax for AddDynamicNPortData and AddNPortData for descriptions of the array contents for components with those types of NPort data.

GetSolverOnDemandData

Use: This method looks for a local component of the name passed in, and in this component it looks for an SOD model of the name passed in and returns the SOD data pertaining to that model.

Parameters: `BSTR component name.`

Parameters: `BSTR SOD model name`

Return Value: VARIANT which is the SOD data.

GetSolverOnDemandModelList

Use: This method looks for a local component of the name passed in, and returns a list of SOD model names defined in the component.

Parameters: `BSTR component name.`

Return Value: VARIANT which is a list of SOD model names.

IsUsed [component manager]

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

Command: None

Syntax: IsUsed(<ComponentName>)

Return Value: <Boolean> // true if the specified component is used in the design

Parameters: `<ComponentName>:`

<string>

Python Syntax	IsUsed(<ComponentName>)

Python Example	`IsUsed = oComponentManager.IsUsed("MyComponent")`

Remove [component manager]

Remove a component from a library

Command: Tools > Edit Configured Libraries > Components > Remove Component

Syntax: Remove <ComponentName>,

<IsProjectComponent>,

<LibraryName>,

<LibraryLocation>

Return Value: None

Parameters: <ComponentName>:

<string> // composite name of the component to remove

<IsProjectComponent>:

<bool>

<LibraryName>:

<string> // name of the library

<LibraryLocation>:

<string> // location of the library in <LibraryName>

// One of "Project", "PersonalLib", or "UserLib"

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

RemoveSolverOnDemandModel

Use: This method looks for a local component of the name passed in, and in this component it looks for an SOD model of the name passed in and deletes the SOD model definition from the component.

`*Parameters:* BSTR component name.`

`*Parameters:* BSTR SOD model name`

Return Value: None.

RemoveUnused [component manager]

Removes components that are not used in the design.

Command: **Project->Remove Unused Definitions** is similar but operates slightly different and does not record script commands.

Syntax: RemoveUnused()

Return Value: <bool> True if one or more components are removed.

`*Parameters:* None`

> **Note:**
>
> The order of calls to RemoveUnused is significant. As a result, removing definitions in an unordered fashion may cause other components in dependent definitions to be rendered unusable.
>
> Also, the symbol and footprint of an unused component are not unusable until after the component itself is removed using the Component Manager Remove script.

Python Syntax	RemoveUnused()
Python Example	`removedDefs = oComponentManager.RemoveUnused()`

Update Dynamic Link [component manager]

*Use:*Reads data from the linked design and updates the dynamic link component. This will update the following: properties, solutions, ports, geometry.

*Command:*Each of the following commands will record a non-undoable script command:

—**Dynamic Link Item RCM > Refresh Dynamic Link**

—**Dynamic Link Component in schematic RCM > Refresh Dynamic Link**

— Click the **Refresh Dynamic Link** button in the Footprint tab of the **Properties Window** for selected Dynamic Link components in the Layout Editor.

*Syntax:*UpdateDynamicLink(<component name>)

*Return Value:*None

Parameters:`<component name> is the name of the dynamic link component`

Example:

oComponentManager.UpdateDynamicLink("TeeModel_L1")

Add [footprint manager]

Use: Add a footprint

<table>
<tr><td><b>UI Access</b></td><td><b>Tools</b> &gt; <b>Edit Configured Libraries</b> &gt; <b>Footprints</b> &gt; <b>Add Footprint</b></td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>Name</td><td>Type</td><td>Description</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>Returns a string:<ul><li>Composite name of the footprint.</li><li>If the name requested conflicts with the name of an existing</li><li>Footprint, the requested name is altered to be unique.</li><li>The name returned reflects any change made to be unique.</li></ul></td></tr>
</table>

Syntax: Add Array("NAME:<FootprintName>,

"ModTime:=", <ModifiedOnInfo>,

"Library:=", "",

"LibLocation:=", "Project",

"OkayToMirror:=", <bool>,

"DefUnits:=", <UnitType>,

Array(NAME:Lyrs",

"Layer:=", <LayerArray>,

"Layer:=", <LayerArray>…,

"SLayer:=", <StackupLayerArray>,

"SLayer:=", <StackupLayerArray>...),

"ActLyr:=", <string>, // name of active layer

"Tol:=", <ToleranceArray> // optional

<PrimitivesInfo>, // optional

<PinsInfo>, // optional

<ViasInfo>, // optional

<EdgeportsInfo>, // optional

<ComponentPropertyInfo>,

<ScriptInfo>) // optional, specified for scripted footprints

Parameters: <FootprintName>:

<string> // simple name of footprint to create

<ModifiedOnInfo>:

An integer that corresponds to the number of seconds that have elapsed

since 00:00 hours, Jan 1, 1970 UTC from the system clock.

<UnitType>:

<string> // default length units to use if units are not specified in other

// parameters

```
<LayerArray>:
Array("N:=", <string>, // layer name
"ID:=", <int> ,
"T:=", <LayerTypeInfo>, // layer type
"TB:=", <TopBottomInfo>,
"Col:=", <int>, // optional - color
"Pat:=", <int>, // optional - fill pattern
"Vis:=", <bool>, // optional - are objects on layer visible
"Sel:=", <bool>, // optional - are objects on layer selectable
"L:=", <bool>) // optional
// are objects on layer locked (can't be edited)

<LayerTypeInfo>:
<string> // one of: signal, dielectric, metalized signal, assembly, silkscreen, soldermask, solderpaste, glue, or user

<TopBottomInfo>:
<string> // one of: top, neither, bottom, or template

<StackupLayerArray>:
Array(<LayerArray>,
"Elev:=", <ElevationInfo>,
"SubL:=", Array("Th:=", <Dimension>,
"LElev:=", <Dimension>,
"R:=", <Dimension>,
"M:=", <MaterialInfo>))

<ElevationInfo>:
<string> // "top" - snap to top
// "mid" - snap to middle
// "bot" - snap to bottom
// "edit" - manual edit
// "none"


<Dimension>:
<string> // real number, may include units


<MaterialInfo>:
<string> // name of the layer material


<ToleranceArray>:
Array(<real>, // distance tolerance
<real>, // angle tolerance (radians)
```

```
<real>) // dimensionless tolerance

<PrimitivesInfo>:
Array("NAME:Prims",
// one or more of the following
<RectInfo>,
<CircleInfo>,
<ArcInfo>,
<LineInfo>,
<PolygonInfo>,
<TextInfo>,
<ImageInfo>)

<RectInfo>:
"Rect:=", Array(<real>, // line width
<int>, // fill pattern
<int>, // color
<real>, // angle, in radians
<real>, // x position of center
<real>, // y position of center
```

```
<real>, // width
< real>) // height

<CircleInfo>:
"Circle:=", Array(<real>, // line width
<int>, // fill pattern
<int>, // color
<real>, // x position of center
<real>, // y position of center
< real>) // radius

<ArcInfo>:
"Arc:=", Array(<real>, // line width
<int>, // line pattern
<int>, // color
<real>, // x position of center
<real>, // y position of center
< real>, // radius
<real>, // start angle, in radians
<end>) // end angle, in radians
```

```
<LineInfo>:
"Line:=", Array(<real>, // line width
<int>, // line pattern
<int>, // color
<PointInfo>, // must specify at least 2 points
<PointInfo>...)
<PointInfo>:
<real>, // x position
<real> // y position

<PolygonInfo>:
"Polygon:=", Array(<real>, // line width
<int>, // fill pattern
<int>, // color
<PointInfo>, // must specify at least 3 points
<PointInfo>...)

<TextInfo>:
"Text:=", Array(<real>, // x position
<real>, // y position
<real>, // angle, in radians
<Justification>,
<bool>, // is plotter font
<string>, // font name
<int>, // color
<string>) // text string
<Justification>: <int>
// 0 : left top
// 1 : left base
// 2 : left bottom
// 3 : center top
// 4 : center base
// 5 : center bottom
// 6 : right top
// 7 : right base
// 8 : right bottom

<ImageInfo>:
"Image:=", Array(<RectInfo>,
<ImageData>,
<bool>) // is mirrored
```

```
<ImageData>:
<string>, // file path
<int>, // 0 : use the file path and link to it
// 1 : ignore file path and use next parameter
<string> // text data, only present if preceding int is 1
```

```
<PinsInfo>:
Array("NAME:Pins",
"P:=", <PinArray>,
"P:=", <PinArray>,...)
```

```
<PinArray>:
Array("Port:=", Array("Id:=", <int>,
"Clr:=", <real>, // optional - clearance
"N:=", <string>), // pin name
"Pos:=", Array("x:=", <Location>, // padstack (x,y) position
"y:=", <Location>),
"VRt:=", <Angle>, // optional - rotation
"HD:=", <Size>, // optional - hole diameter
```

<PadstackImplementationInfo>)

<Location>:

<string> specifying real number and units (may use variables)

<Angle>:

<string> specifying angle with a real number and units

<Size>:

<string> specifying size with a real number and units

<PadstackImplementationInfo>:

If another with the same implementation has already been specified:

"Ref:=", <int> // id of the other

If not:

"Ref:=", <string>, // name

"Frm:=", <int>, // id of highest layer

"To:=", <int>, // id of lowest layer

<LayerPlacementInfo>,

"Man:=", <int>, // optional, 1 if manually placed, 0 if not (default)

"Use:=", Array(<PadUseInfo>, <PadUseInfo>…) // array may be empty

```
<LayerPlacementInfo>:
"Lyr:=", Array("Mrg:=", <int>, // optional,
// 1 if all layers have been merged (default)
// 0 if layer are not merged
"Flp:=", <int>, // optional,
// 1 if placed bottom up
// 0 if not (default)
"Map:=", <ParentToLocalLayerInfo>)

<ParentToLocalLayerInfo>:
Array("U:=", <DirectionOfUniqueness>,
"F:=", Array(<int>, <int>, …), // forward mapping
// -1 is not mapped
"B:=", Array(<int>, <int>, …)) // backward mapping
// -1 is not mapped

<PadUseInfo>:
"Pad:=", Array("Lid:=", <int>, // layer id
"T:=", <string>, // type : "connected", "thermal",
// "no_pad, "not_connected",
// or "not_connected_thermal"
"Man:=", <int>) // optional, 1 if manually placed
// 0 if not (default)

<DirectionOfUniqueness>:
<string> // one of "forward", "backward", or "two ways"

<ViasInfo>:
Array("NAME:Vias", <ViaInfo>, <ViaInfo>…)

<ViaInfo>:
"V:=", Array("Id:=", <int>,
"N:=", <string>, // name
"Pos:=", Array("x:=", <Location>, // via (x,y) position
"y:=", <Location>),
"VRt:=", <Angle>, // optional - rotation
<ImplementationInfo>

<EdgeportsInfo>:
Array("NAME:EPorts", <EdgePortArray>, <EdgePortArray>…)
```

<EdgePortArray>:
Array("NAME:EP",
"LP:=", Array("Id:=", <int>, // port id
"N:=", <string>), // port name
"Eo:=", Array(<edge description>, <edge description>,...))

<edge description> for primitive edges

"et:=", "pe", "pr:=", <id>, "ei:=", <edge#>

<id>: integer that is the primitive id

<edge#>: integer that is the edge number on the primitive

<edge description> for via edges

"et:=", "pse", "layer:=", <layer id>, "se:=", <via id>,
"sx:=", <start X location>, "sy:=", <start Y location>, "ex:=", <end X location>,
"ey:=", <end Y location>, "h:=", <arc height>, "rad:=", <radians>

<via id>: an integer that is the id of the via to use

<layer id>: an integer that is the id of the layer of the pad of the via to use

<start X location>

<start Y Location>:

doubles that are the X, Y location of the start point of the edge arc <end X location>

<end Y Location>:

doubles that are the X, Y location of the end point of the edge arc

<arc height>: double giving the height of the edge arc (0 for a straight edge)

<radians>: double giving the arc size in radians (0 for a straight edge)

<ComponentPropertyInfo>:

Array("NAME:CProps",

"VariableProp:=", <VariableInfo>,

"VariableProp:=", <VariableInfo>,

…)

<VariableInfo>:

Array(<string>, // name

<FlagLetters>,

<string>, // description

"CB:=", <string>, // optional - script for call back

<string>) // value: number, variable, or expression

<ScriptInfo>:

Array("NAME:script",

"language:=", <string>, // script language

"UsesScript:=", true,

"script:=", <string>) // contents of script

Edit [footprint manager]

Deprecated command — please use EditWithComps.

EditWithComps [footprint manager]

Use: Edit an existing footprint.

Command: None

Syntax: EditWithComps <FootprintName>,

Array("NAME:<NewFootprintName>,

"ModTime:=", <ModifiedOnInfo>,

"Library:=", "",

```
"LibLocation:=", "Project",
"OkayToMirror:=", <bool>,
"DefUnits:=", <UnitType>,
Array(NAME:Lyrs",
"Layer:=", <LayerArray>,
"Layer:=", <LayerArray>…,
"SLayer:=", <StackupLayerArray>,
"SLayer:=", <StackupLayerArray>...),
"ActLyr:=", <string>, // name of active layer
"Tol:=", <ToleranceArray> // optional
<PrimitivesInfo>, // optional
<PinsInfo>, // optional
<ViasInfo>, // optional
<EdgeportsInfo>, // optional
<ComponentPropertyInfo>,
<ScriptInfo>, // optional, specified for scripted footprints
Array(<ListofComponentNames>) // Component names
```

Return Value: <string>

```
// composite name of the footprint.
// If the name requested conflicts with the name of an existing
// footprint, the requested name is altered to be unique.
```

// The name returned reflects any change made to be unique.

Parameters: `<FootprintName>:`

<string> // composite name of the footprint being edited

<NewFootprintName>:

<string> // new simple name for the footprint

<ModifiedOnInfo>:

An integer that corresponds to the number of seconds that have elapsed

since 00:00 hours, Jan 1, 1970 UTC from the system clock.

<**UnitType**>:

<string> // default length units to use if units are not specified in other

// parameters

<**LayerArray**>:

Array("N:=", <string>, // layer name

"ID:=", <int> ,

"T:=", <LayerTypeInfo>, // layer type

"TB:=", <TopBottomInfo>,

"Col:=", <int>, // optional - color

"Pat:=", <int>, // optional - fill pattern

"Vis:=", <bool>, // optional - are objects on layer visible

"Sel:=", <bool>, // optional - are objects on layer selectable

"L:=", <bool>) // optional

// are objects on layer locked (can't be edited)

<**LayerTypeInfo**>:

<string> // one of: signal, dielectric, metalized signal, assembly, silkscreen, soldermask, solderpaste, glue, or user

<**TopBottomInfo**>:

<string> // one of: top, neither, bottom, or template

<**StackupLayerArray**>:

Array(<LayerArray>,

"Elev:=", <ElevationInfo>,

"SubL:=", Array("Th:=", <Dimension>,

"LElev:=", <Dimension>,

"R:=", <Dimension>,

"M:=", <MaterialInfo>))

<**ElevationInfo**>:

<string> // "top" - snap to top

// "mid" - snap to middle

// "bot" - snap to bottom

// "edit" - manual edit

// "none"

<**Dimension**>:

<string> // real number, may include units

<**MaterialInfo**>:

<string> // name of the layer material

<**ToleranceArray**>:

Array(<real>, // distance tolerance

<real>, // angle tolerance (radians)

<real>) // dimensionless tolerance

<**PrimitivesInfo**>:

```
Array("NAME:Prims",
// one or more of the following
<RectInfo>,
<CircleInfo>,
<ArcInfo>,
<LineInfo>,
<PolygonInfo>,
<TextInfo>,
<ImageInfo>)
```

<**RectInfo**>:

```
"Rect:=", Array(<real>, // line width
<int>, // fill pattern
<int>, // color
<real>, // angle, in radians
<real>, // x position of center
<real>, // y position of center
<real>, // width
< real>) // height
```

<**CircleInfo**>:

"Circle:=", Array(<real>, // line width

<int>, // fill pattern

<int>, // color

<real>, // x position of center

<real>, // y position of center

< real>) // radius

<**ArcInfo**>:

"Arc:=", Array(<real>, // line width

<int>, // line pattern

<int>, // color

<real>, // x position of center

<real>, // y position of center

< real>, // radius

<real>, // start angle, in radians

<end>) // end angle, in radians

<**LineInfo**>:

"Line:=", Array(<real>, // line width

<int>, // line pattern

<int>, // color

<PointInfo>, // must specify at least 2 points

<PointInfo>...)

<PointInfo>:

<real>, // x position

<real> // y position

<**PolygonInfo**>:

"Polygon:=", Array(<real>, // line width

<int>, // fill pattern

<int>, // color

<PointInfo>, // must specify at least 3 points

<PointInfo>...)

<**TextInfo**>:

"Text:=", Array(<real>, // x position

<real>, // y position

<real>, // angle, in radians

<Justification>,

<bool>, // is plotter font

<string>, // font name

<int>, // color

<string>) // text string

<Justification>: <int>

// 0 : left top

// 1 : left base

// 2 : left bottom

// 3 : center top

// 4 : center base

// 5 : center bottom

// 6 : right top

// 7 : right base

// 8 : right bottom

<**ImageInfo**>:

"Image:=", Array(<RectInfo>,

<ImageData>,

<bool>) // is mirrored

<**ImageData**>:

<string>, // file path

<int>, // 0 : use the file path and link to it

// 1 : ignore file path and use next parameter

<string> // text data, only present if preceding int is 1

<**PinsInfo**>:

Array("NAME:Pins",

"P:=", <PinArray>,

"P:=", <PinArray>,...)

<**PinArray**>:

Array("Port:=", Array("Id:=", <int>,

"Clr:=", <real>, // optional - clearance

"N:=", <string>), // pin name

"Pos:=", Array("x:=", <Location>, // padstack (x,y) position

"y:=", <Location>),

"VRt:=", <Angle>, // optional - rotation

"HD:=", <Size>, // optional - hole diameter

<PadstackImplementationInfo>)

<**Location**>:

<string> specifying real number and units (may use variables)

<**Angle**>:

<string> specifying angle with a real number and units

<**Size**>:

<string> specifying size with a real number and units

<**PadstackImplementationInfo**>:

If another with the same implementation has already been specified:

"Ref:=", <int> // id of the other

If not:

"Ref:=", <string>, // name

"Frm:=", <int>, // id of highest layer

"To:=", <int>, // id of lowest layer

<LayerPlacementInfo>,

"Man:=", <int>, // optional, 1 if manually placed, 0 if not (default)

"Use:=", Array(<PadUseInfo>, <PadUseInfo>…) // array may be empty

<**LayerPlacementInfo**>:

"Lyr:=", Array("Mrg:=", <int>, // optional,

```
// 1 if all layers have been merged (default)
// 0 if layer are not merged
"Flp:=", <int>, // optional,
// 1 if placed bottom up
// 0 if not (default)
"Map:=", <ParentToLocalLayerInfo>)
```

<**ParentToLocalLayerInfo**>:

```
Array("U:=", <DirectionOfUniqueness>,
"F:=", Array(<int>, <int>, …), // forward mapping
// -1 is not mapped
"B:=", Array(<int>, <int>, …)) // backward mapping
// -1 is not mapped
```

<**PadUseInfo**>:

```
"Pad:=", Array("Lid:=", <int>, // layer id
"T:=", <string>, // type : "connected", "thermal",
// "no_pad, "not_connected",
// or "not_connected_thermal"
"Man:=", <int>) // optional, 1 if manually placed
// 0 if not (default)
```

<**DirectionOfUniqueness**>:

<string> // one of "forward", "backward", or "two ways"

<**ViasInfo**>:

Array("NAME:Vias", <ViaInfo>, <ViaInfo>…)

<**ViaInfo**>:

"V:=", Array("Id:=", <int>,

"N:=", <string>, // name

"Pos:=", Array("x:=", <Location>, // via (x,y) position

"y:=", <Location>),

"VRt:=", <Angle>, // optional - rotation

<ImplementationInfo>

<**EdgeportsInfo**>:

Array("NAME:EPorts", <EdgePortArray>, <EdgePortArray>…)

<**EdgePortArray**>

Array("NAME:EP",

"LP:=", Array("Id:=", <int>, // port id

"N:=", <string>), // port name

"Eo:=", Array(<edge description>, <edge description>,...))

<edgedescription> for primitive edges

"et:=", "pe", "pr:=", <id>, "ei:=", <edge#>

<id>: integer that is the primitive id

<edge#>: integer that is the edge number on the primitive

<edge description> for via edges

"et:=", "pse", "sel:=", <"via">, "layer:=", <layer id>,

"sx:=", <start X location>, "sy:=", <start Y location>, "ex:=", <end X location>, "ey:=", <end Y location>, "h:=", <arc height>, "rad:=", <radians>

<"via">: text that is the name of the via to use

<layer id>: an integer that is the id of the layer of the pad of the via to use

<start X location>, <start Y Location>:

doubles that are the X, Y location of the start point of the edge arc

<end X location>, <end Y Location>:

doubles that are the X, Y location of the end point of the edge arc

<arc height>: double giving the height of the edge arc (0 for a straight edge)

<radians>: double giving the arc size in radians (0 for a straight edge)

<**ComponentPropertyInfo**>:

```
Array("NAME:CProps",
"VariableProp:=", <VariableInfo>,
"VariableProp:=", <VariableInfo>,
…)
```

<**VariableInfo**>:

```
Array(<string>, // name
<FlagLetters>,
<string>, // description
"CB:=", <string>, // optional - script for call back
<string>) // value: number, variable, or expression
```

<**ScriptInfo**>:

Array("NAME:script",

"language:=", <string>, // script language

"UsesScript:=", true,

"script:=", <string>) // contents of script

<ListOfComponentNames>:

<string>,<string> ...

// The list may be empty. When not empty, each string that is listed is a component

// that references the footprint to be edited. Prior to editing, a clone of the footprint is

// made, and the components that are listed are modified so that they now refer to

// the clone.

Export [footprint manager]

Use: Export a footprint to a library

Command: Tools > Edit Configured Libraries > Footprints > Export to Library

Syntax: Export Array("NAME:<LibraryName>",

<FootprintName>,

<FootprintName>...),

<LibraryLocation>

Return Value: None

Parameters: <LibraryName>:

<string> // name of the library

<FootprintName>:

<string> // composite name of footprint to export

<LibraryLocation>:

<string> // location of the library in <LibraryName>

// One of "Project", "PersonalLib", or "UserLib"

GetNames [footprint manager]

Use: Returns the names of the footprints (used and unused) in a design. The following script command, **IsUsed,** can then be used to separate used and unused footprints.

Command: None

Syntax: GetNames()

Return Value: An array of strings

Parameters: None

IsUsed [footprint manager]

Use: Used to determine if a footprint is used in the design.

Command: None

Syntax: IsUsed(<FootprintName>)

Return Value: <Boolean> // true if the specified footprint is used in the design

Parameters: `<FootprintName>:`

<string>

Remove [footprint manager]

Use: Removes a footprint from a library

Command: Tools > Edit Configured Libraries > Footprints > Remove Footprint

Syntax: Remove <FootprintName>,

<IsProjectFootprint>,

<LibraryName>,

<LibraryLocation>

Return Value: None

Parameters: `<FootprintName>:`

<string> // composite name of the footprint to remove

<IsProjectFootprint>:

<bool>

<LibraryName>:

<string> // name of the library

<LibraryLocation>:

<string> // location of the library in <LibraryName>

// One of "Project", "PersonalLib", or "UserLib"

RemoveUnused [footprint manager]

Use: Removes footprints that are not used in the design.

Command: **Project->Remove Unused Definitions** is similar but operates slightly different and does not record script commands.

Syntax: RemoveUnused()

Return Value: <bool> True if one or more footprints are removed.

`Parameters: None`

> **Note:**
>
> The order of calls to RemoveUnused is significant. As a result, removing definitions in an unordered fashion may cause other footprints in dependent definitions to be rendered unusable.
>
> Also, the symbol and footprint of an unused component are not unusable until after the component itself is removed using the Component ManagerRemove script.

Material Manager Script Commands

The material manager provides access to materials in a project. The manager object is accessed via the definition manager.

```
Set oDefinitionManager = oProject.GetDefinitionManager()

Set oMaterialManager = oDefinitionManager.GetManager("Material")
```

The topics for this section include:

GetNames

GetProperties

IsUsed

RemoveUnused

GetNames [material manager]

Use: Get the names of the materials in a project

Command: None

Syntax: GetNames

Return Value: Names of the materials in a project

Parameters: None

Example:

```
materialNames = oMaterialMgr.GetNames()
```

Python Syntax	GetNames()
Python Example	`materialnames = oMaterialManager.GetNames()`

GetProperties [material manager]

Get material properties. Differs from GetData in that only material properties available to the user are returned by GetProperties.

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

	<material_name>	string	Name of the project material
		Boolean	True or False. If you use False or only a single argument, then GetProperties returns only the properties that are different from the defaults.
Return Value	Array of material data		

Python Syntax	GetProperties (<materialname>)

Python Example	`oMaterialManager.GetProperties("Gold", True)`
	`oMaterialManager.GetProperties("vacuum")`

IsUsed [material manager]

Use: Checks if a project material is in use

Command: None

Syntax: IsUsed <material_name>

Return Value: Returns 'True' if the material is in use.

Parameters: <material_name>

Type: string

Value: Name of the project material to check.

Python Syntax	IsUsed(<ComboName>)

Python Example	`isused = oMaterialManager.IsUsed("mylib:mymaterial")`

RemoveUnused [material manager]

Use: Remove all unused materials from the project.

Command: None

Syntax: RemoveUnused

Return Value: None

Parameters: None

Python Syntax	RemoveUnused()
Python Example	`thereWereExtras = oMaterialManager.RemoveUnused()`

Model Manager Script Commands

The model manager provides access to models in a project. The manager object is accessed via the definition manager.

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

```
Set oModelManager = oDefinitionManager.GetManager("Model")
```

The model manager script commands are listed below:

Add

ConvertToDynamic

ConvertToParametric

Edit

EditWithComps

Export

GetNames

IsUsed

Remove

RemoveUnused

Add [model manager]

Use: Add a model

Command: Tools > Edit Configured Libraries > Models > Add Model

Syntax: Add Array("NAME:<modelName>",

```
        "ModTime:=", <ModifiedTimeInfo>,
        "Library:=", "", // Library name
        "LibLocation:=", "Project", // Project Location
        <PinDefInfo>,
        <PinDefInfo>,... // optional, to define pins
        <GraphicsDataInfo>, // optional, to define graphics
        <PropDisplayMapInfo>)) // optional, to define property displays
```

Return Value: <string>

```
        // composite name of the model.
        // If the name requested conflicts with the name of an existing
```

// model, the requested name is altered to be unique.

// The name returned reflects any change made to be unique.

`Parameters: <modelName>:`

<string> // simple name of the model being added

<ModifiedOnInfo>:

An integer that corresponds to the number of seconds that have elapsed

since 00:00 hours, Jan 1, 1970 UTC from the system clock.

<PinDefInfo>:

Array("NAME:PinDef",

"Pin:=", Array (<string>, // pin name

<real>, // x location

<real>, // y location

<real>, // angle in radians

<PinType>,

<real>, // line width

<real>, // line length

<bool>, // mirrored

<int>, // color

```
<bool>, // true if visible, false if not
<string>, // hidden net name
<OptionalPinInfo>, // optional info
<PropDisplayMapInfo>)) // optional

<PinType>:
<string> // "N" : normal pin
// "I" : input pin
// "O" : output pin

<OptionalPinInfo>:
// Specify both or neither
<bool>, // true if name is to be shown
<bool>, // true if number is to be shown


<PropDisplayMapInfo>:
Array("NAME:PropDisplayMap",
<PropDisplayInfo>,
<PropDisplayInfo>,...)
```

```
<PropDisplayInfo>:
<NameString>, Array(<DisplayTypeInfo>,
<DisplayLocationInfo>,
<int>, // optional, level number
<TextInfo>)
<NameString>:
<string> // PropertyName:=, where PropertyName is the name of
// the property to be displayed

<DisplayTypeInfo>:
<int> // 0 : No display
// 1 : Display name only
// 2 : Display value only
// 3 : Display both name and value
// 4: Display evaluated value only
// 5: Display both name and evaluated value

<DisplayLocationInfo>:
<int> // 0 : Left
// 1 : Top
// 2 : Right
// 3 : Bottom
// 4 : Center
// 5 : Custom placement

<GraphicsDataInfo>:
Array("NAME:Graphics",
// one or more of the following
<RectInfo>,
<CircleInfo>,
<ArcInfo>,
<LineInfo>,
<PolygonInfo>,
<TextInfo>,
<ImageInfo>)

<RectInfo>:
"Rect:=", Array(<real>, // line width
<int>, // fill pattern
<int>, // color
```

```
<real>, // angle, in radians
<real>, // x position of center
<real>, // y position of center
<real>, // width
< real>) // height

<CircleInfo>:
"Circle:=", Array(<real>, // line width
<int>, // fill pattern
<int>, // color
<real>, // x position of center
<real>, // y position of center
< real>) // radius

<ArcInfo>:
"Arc:=", Array(<real>, // line width
<int>, // line pattern
<int>, // color
<real>, // x position of center
<real>, // y position of center
< real>, // radius
```

```
<real>, // start angle, in radians
<end>) // end angle, in radians

<LineInfo>:
"Line:=", Array(<real>, // line width
<int>, // line pattern
<int>, // color
<PointInfo>, // must specify at least 2 points
<PointInfo>...)
<PointInfo>:
<real>, // x position
<real> // y position

<PolygonInfo>:
"Polygon:=", Array(<real>, // line width
<int>, // fill pattern
<int>, // color
<PointInfo>, // must specify at least 3 points
<PointInfo>...)
```

```
<TextInfo>:
"Text:=", Array(<real>, // x position
<real>, // y position
<real>, // angle, in radians
<Justification>,
<bool>, // is plotter font
<string>, // font name
<int>, // color
<string>) // text string

<Justification>:
<int> // 0 : left top
// 1 : left base
// 2 : left bottom
// 3 : center top
// 4 : center base
// 5 : center bottom
// 6 : right top
// 7 : right base
// 8 : right bottom
```

```
<ImageInfo>:
"Image:=", Array(<RectInfo>,
<ImageData>,
<bool>) // is mirrored

<ImageData>:
<string>, // file path
<int>, // 0 : use the file path and link to it
// 1 : ignore file path and use next parameter
<string> // text data, only present if preceding int is 1
```

ConvertToDynamic

Use: Build a new dynamic model based on an existing parametric model.

Command: Right-click on a model under Definitions/Models in the Project Tree and choose ConvertToDynamic.

Syntax: ConvertToDynamic(defName, newname)

Return Value: <newname> // Name of the new model added

Parameters: <defName> // Model that is the base for the new conversion

ConvertToParametric

Use: Build a new parametric model based on an existing dynamic model.

Command: Right-click on a model under Definitions/Models in the Project Tree and choose ConvertToParametric.

Syntax: ConvertToParametric(defName, newname)

Return Value: <newname> // Name of the new model added

`*Parameters:* <defName> // Model that is the base for the new conversion`

Edit [deprecated]

Deprecated command — please use EditSymbolAndUpdateComps.

EditWithComps [model manager]

Use: Edit an existing model.

Command: None

Syntax: EditWithComps <ModelName>,

Array("NAME:<NewModelName>",

"ModTime:=", <ModifiedTimeInfo>,

"Library:=", <string>, // Library name

"LibLocation:=", <string>, // Project Location

<PinDefInfo>,

<PinDefInfo>,... // optional, to define pins

<GraphicsDataInfo>, // optional, to define graphics

<PropDisplayMapInfo>), // optional, to define property displays

Array(<ListOfComponentNames>) // Component names

Return Value: <string>

// composite name of the model.

// If the name requested conflicts with the name of an existing

// model, the requested name is altered to be unique.

// The name returned reflects any change made to be unique.

Parameters: `<ModelName>:`

<string> // composite name of the model being edited

<NewModelName>:

<string> // new simple name for the model

<ModifiedOnInfo>:

An integer that corresponds to the number of seconds that have elapsed

since 00:00 hours, Jan 1, 1970 UTC from the system clock.

<PinDefInfo>:

Array("NAME:PinDef",

"Pin:=", Array (<string>, // pin name

<real>, // x location

<real>, // y location

<real>, // angle in radians

<PinType>,

```
<real>, // line width
<real>, // line length
<bool>, // mirrored
<int>, // color
<bool>, // true if visible, false if not
<string>, // hidden net name
<OptionalPinInfo>, // optional info
<PropDisplayMapInfo>)) // optional

<PinType>:
<string> // "N" : normal pin
// "I" : input pin
// "O" : output pin

<OptionalPinInfo>:
// Specify both or neither
<bool>, // true if name is to be shown
<bool>, // true if number is to be shown

<PropDisplayMapInfo>:
Array("NAME:PropDisplayMap",
<PropDisplayInfo>,
<PropDisplayInfo>,...)

<PropDisplayInfo>:
<NameString>, Array(<DisplayTypeInfo>,
<DisplayLocationInfo>,
<int>, // optional, level number
<TextInfo>)

<NameString>:
<string> // PropertyName:=, where PropertyName is the name of
// the property to be displayed

<DisplayTypeInfo>:
<int> // 0 : No display
// 1 : Display name only
// 2 : Display value only
// 3 : Display both name and value
// 4: Display evaluated value only
// 5: Display both name and evaluated value
```

```
<DisplayLocationInfo>:
<int> // 0 : Left
// 1 : Top
// 2 : Right
// 3 : Bottom
// 4 : Center
// 5 : Custom placement

<GraphicsDataInfo>:
Array("NAME:Graphics",
// one or more of the following
<RectInfo>,
<CircleInfo>,
<ArcInfo>,
<LineInfo>,
<PolygonInfo>,
<TextInfo>,
<ImageInfo>)

<RectInfo>:
```

```
"Rect:=", Array(<real>, // line width
<int>, // fill pattern
<int>, // color
<real>, // angle, in radians
<real>, // x position of center
<real>, // y position of center
<real>, // width
< real>) // height

<CircleInfo>:
"Circle:=", Array(<real>, // line width
<int>, // fill pattern
<int>, // color
<real>, // x position of center
<real>, // y position of center
< real>) // radius

<ArcInfo>:
"Arc:=", Array(<real>, // line width
<int>, // line pattern
```

```
<int>, // color
<real>, // x position of center
<real>, // y position of center
< real>, // radius
<real>, // start angle, in radians
<end>) // end angle, in radians

<LineInfo>:
"Line:=", Array(<real>, // line width
<int>, // line pattern
<int>, // color
<PointInfo>, // must specify at least 2 points
<PointInfo>...)
<PointInfo>:
<real>, // x position
<real> // y position

<PolygonInfo>:
"Polygon:=", Array(<real>, // line width
<int>, // fill pattern
<int>, // color
```

```
<PointInfo>, // must specify at least 3 points
<PointInfo>...)

<TextInfo>:
"Text:=", Array(<real>, // x position
<real>, // y position
<real>, // angle, in radians
<Justification>,
<bool>, // is plotter font
<string>, // font name
<int>, // color
<string>) // text string

<Justification>:
<int> // 0 : left top
// 1 : left base
// 2 : left bottom
// 3 : center top
// 4 : center base
// 5 : center bottom
// 6 : right top
// 7 : right base
// 8 : right bottom

<ImageInfo>:
"Image:=", Array(<RectInfo>,
<ImageData>,
<bool>) // is mirrored

<ImageData>:
<string>, // file path
<int>, // 0 : use the file path and link to it
// 1 : ignore file path and use next parameter
<string> // text data, only present if preceding int is 1

<ListOfComponentNames>:
<string>,<string> ...
// The list may be empty. When not empty, each string that is listed is a component
// that references the model to be edited. Prior to editing, a clone of the model is
// made, and the components that are listed are modified so that they now refer to
// the clone.
```

Export [model manager]

Use: Exports model(s) to a library

Command: Tools > Edit Configured Libraries > Models > Export to Library

Syntax: Export Array("NAME:<LibraryName>",

<ModelName>,

<ModelName>...),

<LibraryLocation>

Return Value: None

Parameters: <LibraryName>:

<string> // name of the library

<ModelName>:

<string> // composite name of model to export

<LibraryLocation>:

<string> // location of the library in <LibraryName>

// One of "Project", "PersonalLib", or "UserLib"

Python Syntax	Export (["NAME:<LibraryName>", <ComboName>, <ComboName>...], <LibraryLocation>)

Python Example	`oModelManager.Export (["NAME:mylib", "model1", "model2"])`

GetNames [model manager]

Use: Returns the names of the models (used and unused) in a design. The following script command, **IsUsed,** can then be used to separate used and unused models.

Command: None

Syntax: GetNames()

Return Value: An array of strings

Parameters: None

Python Syntax	GetNames()
Python Example	`modelnames = oModelManager.GetNames()`

IsUsed [model manager]

Use: Used to determine if a model is used in the design.

Command: None

Syntax: IsUsed(<ModelName>)

Return Value: <Boolean> // true if the specified model is used in the design

Parameters: <ModelName>:

<string>

Python Syntax	IsUsed(<ComboName>)

Python Example	`isused = oModelManager.IsUsed ("mylib:mymodel")`

Remove [model manager]

Use: Removes a model from a library

Command: Tools > Edit Configured Libraries > Models > Remove Model

Syntax: Remove <ModelName>,

<IsProjectModel>,

<LibraryName>,

<LibraryLocation>

Return Value: None

Parameters: <ModelName>:

<string> // composite name of the model to remove

<IsProjectModel>:

<bool>

<LibraryName>:

<string> // name of the library

<LibraryLocation>:

<string> // location of the library in <LibraryName>

// One of "Project", "PersonalLib", or "UserLib"

Python Syntax	Remove (<ModelName>, <IsLocal>, <LibraryName>, <LibraryLocation>)

Python Example	`oModelManager.Export([`
	`"NAME:mylib", "model1", "model2"])`

RemoveUnused [model manager]

Use: Removes models that are not used in the design.

Command: **Project->Remove Unused Definitions** is similar but operates slightly different and does not record script commands.

Syntax: RemoveUnused()

Return Value: <bool> True if one or more models are removed.

Parameters: None

> **Note:**
>
> The order of calls to RemoveUnused is significant. As a result, removing definitions in an unordered fashion may cause other models in dependent definitions to be rendered unusable.
>
> Also, the model and footprint of an unused component are not unusable until after the component itself is removed using the Component Manager Remove script.

Python Syntax	RemoveUnused()
Python Example	`thereWereExtras = oModelManager.RemoveUnused()`

Network Data Explorer Manager Script Commands

The network data Explorer (NDE) Manager provides access to certain NDE data.

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

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

For NDE scripts accessed via the ndExplorer tool, see: Network Data Explorer Script Commands.

The topics for this section include:

ExportFullWaveSpice

ExportNetworkData

ExportNMFData

ExportFullWaveSpice

Use: Export FullWaveSpice data in a format of your choice.

Command: File > Export MacroModel > Broadband (SYZ, FWS….)

Syntax: ExportFullWaveSpice

```
        "DesignName", // Design name. Can be left blank, if loading solution from a file.
        true/false, // true - solution loaded from file, false- loaded from design
        "Name", // If loading from design this is the solution name, else this is the
              // full path of the file from which the solution is loaded
        "variation", // Pick a particular variation. Leave blank if no variation.
```

```
Array("NAME:Frequencies"), // Optional; if none defined all frequencies are used
Array("NAME:SpiceData", // Spice export options object
"SpiceType:=", "SSS", // SpiceType can be "PSpice", "HSpice", "Spectre", "SSS",
                    // "Simplorer", "TouchStone1.0", "TouchStone2.0"
"EnforcePassivity:=", false, // Enforce Passivity true/false
"EnforceCausality:=", false, // Enforce Causality true/false
"UseCommonGround:=", false, // Use common ground true/false
"FittingError:=", 0.5, // Fitting error
"MaxPoles:=", 400, // Maximum Order
"PassivityType:=", "ConvexOptimization", // Passivity Type can be "ConvexOptimization",
                         // "PassivityByPerturbation", or "IteratedFittingOfPV"
"ColumnFittingType:=", "Column", // Column FittingType can be "Column", "Entry", "Matrix"
"SSFittingType:=", "TWA", // SS Fitting Type can be "TWA", "IterativeRational"
"RelativeErrorToleranc:=", false, // Relative error tolerance true/false
"TouchstoneFormat:=", "MA", // Touchstone Format "MA", "RI", "DB"
"TouchstoneUnits:=", "Hz", // Touchstone Units "Hz", "KHz", "MHz", "MHz"
"TouchStonePrecision:=", 8, // Touchstone precision
"ExportDirectory:=", "C:/Examples/LNA/", // Directory to export to
"ExportSpiceFileName:=", "Linckt_HBTest_2.sss", // Spice export file
"FullwaveSpiceFileName:=", "Linckt_HBTest.sss", // FWS file
"CreateNPortModel:=", true // Create a model based on the exported file true/false
```

)

ExportNetworkData

Exports matrix solution data to a file.

<table>
<tr><th>UI Access</th><th>N/A</th></tr>
<tr><td>Parameters</td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><em>&lt;DesignVariationKey&gt;</em></td><td>String</td><td>Design variation key. Pass empty string for the current nominal variation.</td></tr>
<tr><td><em>&lt;SolnSelectionArray&gt;</em></td><td>Array</td><td>Array of selected solutions.
<code>Array(<SolnSelector>, <SolnSelector&gt;, ...)</code>
If more than one array entry, this indicates a combined Interpolating sweep.</td></tr>
<tr><td><em>&lt;SolnSelector&gt;</em></td><td>String</td><td>Solution setup name and solution name, separated by a colon.</td></tr>
<tr><td><em>&lt;FileFormat&gt;</em></td><td>Integer</td><td>File format value.
2 : Tab delimited spreadsheet format (.tab)
3 : Touchstone (.sNp)
4 : CitiFile (.cit)
7 : Matlab (.m)
8 : Terminal Z0 spreadsheet</td></tr>
<tr><td><em>&lt;OutFile&gt;</em></td><td>String</td><td>Full path to the file to write out.</td></tr>
<tr><td><em>&lt;FreqsArray&gt;</em></td><td>Array</td><td>The frequencies to export. The <FreqsArray> argument contains a vector (e.g. "1GHz", "2GHz", ...) to use, or "all". To export all frequencies, use Array("all"). If no frequencies are specified, all frequencies are used.</td></tr>
<tr><td><em>&lt;DoRenorm&gt;</em></td><td>Boolean</td><td>For Touchstone format only. Specifies whether to renormalize the</td></tr>
</table>
</td></tr>
</table>

		data before export.
<*RenormImped*>	Double	For Touchstone format only. Real impedance value in ohms, for renormalization. Required in syntax, but ignored if DoRenorm is false.
<*DataType*>	Array	Optional. Type: "S", "Y", or "Z". The matrix to export.
<*pass*>	Integer	Optional. The pass to export. This is ignored if the sourceName is a frequency sweep. Leaving out this value or specifying -1 gets all passes.
<*ComplexFormat*>	Integer	Optional. Type: "0", "1", or "2" The format to use for the exported data. 0 = Magnitude/Phase. 1= Real/Imaginary. 2= db/Phase.
<*Precision*>	Integer	Optional. Touchstone number of digits precision. Default if not specified is 15.
<*UseExportFreqs*>	Boolean	Specifies whether to use export frequencies.
<*IncludeGammaComments*>	Boolean	Touchstone only. Specifies whether to include Gamma and Impedance comments.
<*SupportNonStdExport*>	Boolean	Specifies whether to support non-standard Touchstone extensions for mixed reference impedance.

Return Value	None.

Python Syntax	ExportNetworkData(<*DesignVariationKey*>, <*SolnSelectionArray*>, <*SolnSelector*>, <*FileFormat*>, <*OutFile*>, <*FreqsArray*>, <*DoRenorm*>, <*RenormImped*>, [Optional <*DataType*>], [Optional <*pass*>], [Optional <*ComplexFormat*>], [Optional <*Precision*>], [Optional <*UseExportFreqs*>], [Optional <*IncludeGammaComments*>], [Optional <*SupportNonStdExport*>])
Python Example	

ExportNMFData

Add [padstack manager]

Use: Add a padstack

Command: Tools > Edit Configured Libraries > Padstacks > Add Padstack

Syntax: Add Array("NAME:<PadstackName>",

```
        "ModTime:=", <ModifiedOnInfo>,
        "Library:=", "", // name of the library
        "LibLocation:=", "Project", // location of the named library
        Array("NAME:psd",
        "nam:= ", <PadstackName>,
        "lib:=", "", // name of the library
        "mat:=", "", // hole plating material
        "plt:=", "0", // percent of hole's radius filled by plating
        Array("NAME:pds",
        <LayerGeometryArray>,
        <LayerGeometryArray....),
        "hle:=", <PadInfo>
        "hRg:=", <HoleRange>,
        "sbsh:=", <SolderballShape>,
"sbpl:=", <SolderballPlacement>,
"sbr:=", <string>, // solderball diameter, real with units
"sb2:=", <string>, // solderball mid diameter, real with units
"sbn:=", <string>), // name of solderball material
"ppl:=", <PadPortLayerArray>)
```

Return Value: simple name of the added padstack

```
// If the name requested conflicts with the name of an existing
// padstack, the requested name is altered to be unique.
// The name returned reflects any change made to be unique.
```

Parameters: `<PadstackName>:`

```
<string> // simple name of padstack to create
```

<ModifiedOnInfo>:

An integer that corresponds to the number of seconds that have elapsed since 00:00 hours, Jan 1, 1970 UTC from the system clock.

<LayerGeometryArray>:

```
Array("Name:lgm",
"lay:=", <string>, // definition layer name
"id:=", <int>, // definition layer id
"pad:=", <PadInfo>, // pad
"ant:=", <PadInfo>, // antipad
"thm:=", <PadInfo>, // themal pad
"X:=", <string>, // pad x connection, real with units
"Y:=", <string>, // pad y connection, real with units
"dir:=", <DirectionString>) // pad connection direction

<PadInfo>:
Array("shp:=", <PadShape>,
"Szs:=", <DimensionArray>,
"X:=", <string>, // x offset, real with units
"Y:=", <string>, // y offset, real with units
"R:=", <string>) // rotation, real with units

<PadShape>:
<string> one of these choices
"No" // no pad
"Cir" // Circle
"Sq" // Square
"Rct" // Rectangle
```

"Ov" // Oval

"Blt" // Bullet

"Ply" // Polygons

"R45" // Round 45 thermal

"R90" // Round 90 thermal

"S45" // Square 45 thermal

"S90" // Square 90 thermal

<DimensionArray>:

Array(<string>, ...) // each string is a real with units for one of the dimensions of the shape

<DirectionString>:

<string> one of these choices

"No" // no direction

"Any" // any direction

"0" // 0 degrees

"45" // 45 degrees

"90" // 90 degrees

"135" // 135 degrees

"180" // 180 degrees

"225" // 225 degrees

"270" // 270 degrees

"315" // 315 degrees

<HoleRange>:

<string> one of these choices

"Thr" // through all layout layers

"Beg" // from upper pad layer to lowest layout layer

"End" // from upper layout layer to lowest pad layer

"UTL" // from upper pad layer to lowest pad layer

<SolderballShape>:

<string> one of these choices

"None" // no solderball

"Cyl" // cylinder solderball

"Sph" // spheroid solderball

<SolderballPlacement>:

<string> one of these choices

"abv" // above padstack

"blw" // below padstack

<PadPortLayerArray>:

Array(<int>, <int>,....) where each int is a layer id

Edit [padstack manager]

Use: Edit an existing padstack.

Command: Tools > Edit Configured Libraries > Padstacks > Edit Padstack

Syntax: Edit <PadstackName>,

```
Array("NAME:<NewPadstackName>",
"ModTime:=", <ModifiedOnInfo>,
"Library:=", "", // name of the library
"LibLocation:=", "Project", // location of the named library
Array("NAME:psd",
"nam:= ", <PadstackName>,
"lib:=", "", // name of the library
"mat:=", "", // hole plating material
"plt:=", "0", // percent of hole's radius filled by plating
Array("NAME:pds",
<LayerGeometryArray>,
<LayerGeometryArray....),
"hle:=", <PadInfo>
"hRg:=", <HoleRange>,
"sbsh:=", <SolderballShape>,
"sbpl:=", <SolderballPlacement>,
"sbr:=", <string>, // solderball diameter, real with units
```

"sb2:=", <string>, // solderball mid diameter, real with units

"sbn:=", <string>), // name of solderball material

"ppl:=", <PadPortLayerArray>)

Return Value: <string> // composite name of the padstack

// If the name requested conflicts with the name of an existing

// padstack, the requested name is altered to be unique.

// The name returned reflects any change made to be unique.

`Parameters: <PadstackName>:`

<string> // composite name of padstack to edit

<NewPadstackName>:

<string> // new simple name for padstack

<ModifiedOnInfo>:

An integer that corresponds to the number of seconds that have elapsed

since 00:00 hours, Jan 1, 1970 UTC from the system clock.

<LayerGeometryArray>:

Array("Name:lgm",

```
"lay:=", <string>, // definition layer name
"id:=", <int>, // definition layer id
"pad:=", <PadInfo>, // pad
"ant:=", <PadInfo>, // antipad
"thm:=", <PadInfo>, // themal pad
"X:=", <string>, // pad x connection, real with units
"Y:=", <string>, // pad y connection, real with units
"dir:=", <DirectionString>) // pad connection direction

<PadInfo>:
Array("shp:=", <PadShape>,
"Szs:=", <DimensionArray>,
"X:=", <string>, // x offset, real with units
"Y:=", <string>, // y offset, real with units
"R:=", <string>) // rotation, real with units

<PadShape>:
<string> one of these choices
"No" // no pad
"Cir" // Circle
"Sq" // Square
```

"Rct" // Rectangle

"Ov" // Oval

"Blt" // Bullet

"Ply" // Polygons

"R45" // Round 45 thermal

"R90" // Round 90 thermal

"S45" // Square 45 thermal

"S90" // Square 90 thermal

<DimensionArray>:

Array(<string>, ...) // each string is a real with units for one of the

// dimensions of the shape

<DirectionString>:

<string> one of these choices

"No" // no direction

"Any" // any direction

"0" // 0 degrees

"45" // 45 degrees

"90" // 90 degrees

"135" // 135 degrees

"180" // 180 degrees

"225" // 225 degrees

"270" // 270 degrees

"315" // 315 degrees

<HoleRange>:

<string> one of these choices

"Thr" // through all layout layers

"Beg" // from upper pad layer to lowest layout layer

"End" // from upper layout layer to lowest pad layer

"UTL" // from upper pad layer to lowest pad layer

<SolderballShape>:

<string> one of these choices

"None" // no solderball

"Cyl" // cylinder solderball

"Sph" // spheroid solderball

<SolderballPlacement>:

<string> one of these choices

"abv" // above padstack

"blw" // below padstack

<PadPortLayerArray>:

Array(<int>, <int>,....) where each int is a layer id

EditWithComps [padstack manager]

Use: Edit an existing padstack.

Command: None

Syntax: EditWithComps <PadstackName>,

```
Array("NAME:<NewPadstackName>",
"ModTime:=", <ModifiedOnInfo>,
"Library:=", "", // name of the library
"LibLocation:=", "Project", // location of the named library
Array("NAME:psd",
"nam:= ", <PadstackName>,
"lib:=", "", // name of the library
"mat:=", "", // hole plating material
"plt:=", "0", // percent of hole's radius filled by plating
Array("NAME:pds",
<LayerGeometryArray>,
<LayerGeometryArray....),
```

"hle:=", <PadInfo>

"hRg:=", <HoleRange>,

"sbsh:=", <SolderballShape>,

"sbpl:=", <SolderballPlacement>,

"sbr:=", <string>, // solderball diameter, real with units

"sb2:=", <string>, // solderball mid diameter, real with units

"sbn:=", <string>), // name of solderball material

"ppl:=", <PadPortLayerArray>,

Array(<ListOfComponentNames>) // Component names

Return Value: <string>

// composite name of the padstack.

// If the name requested conflicts with the name of an existing

// padstack, the requested name is altered to be unique.

// The name returned reflects any change made to be unique.

Parameters: <PadstackName>:

<string> // composite name of the padstack being edited

<NewPadstackName>:

<string> // new simple name for the padstack

<ModifiedOnInfo>:

An integer that corresponds to the number of seconds that have elapsed

since 00:00 hours, Jan 1, 1970 UTC from the system clock.

<LayerGeometryArray>:

Array("Name:lgm",

"lay:=", <string>, // definition layer name

"id:=", <int>, // definition layer id

"pad:=", <PadInfo>, // pad

"ant:=", <PadInfo>, // antipad

"thm:=", <PadInfo>, // themal pad

"X:=", <string>, // pad x connection, real with units

"Y:=", <string>, // pad y connection, real with units

"dir:=", <DirectionString>) // pad connection direction

<PadInfo>:

Array("shp:=", <PadShape>,

"Szs:=", <DimensionArray>,

"X:=", <string>, // x offset, real with units

```
"Y:=", <string>, // y offset, real with units
"R:=", <string>) // rotation, real with units

<PadShape>:
<string> one of these choices
"No" // no pad
"Cir" // Circle
"Sq" // Square
"Rct" // Rectangle
"Ov" // Oval
"Blt" // Bullet
"Ply" // Polygons
"R45" // Round 45 thermal
"R90" // Round 90 thermal
"S45" // Square 45 thermal
"S90" // Square 90 thermal

<DimensionArray>:
Array(<string>, ...) // each string is a real with units for one of the
// dimensions of the shape
```

<DirectionString>:

<string> one of these choices

"No" // no direction

"Any" // any direction

"0" // 0 degrees

"45" // 45 degrees

"90" // 90 degrees

"135" // 135 degrees

"180" // 180 degrees

"225" // 225 degrees

"270" // 270 degrees

"315" // 315 degrees

<HoleRange>:

<string> one of these choices

"Thr" // through all layout layers

"Beg" // from upper pad layer to lowest layout layer

"End" // from upper layout layer to lowest pad layer

"UTL" // from upper pad layer to lowest pad layer

<SolderballShape>:

<string> one of these choices

"None" // no solderball

"Cyl" // cylinder solderball

"Sph" // spheroid solderball

<SolderballPlacement>:

<string> one of these choices

"abv" // above padstack

"blw" // below padstack

<PadPortLayerArray>:

Array(<int>, <int>,....) where each int is a layer id

<ListOfComponentNames>:

<string>,<string> ...

// The list may be empty. When not empty, each string that is listed is a component

// that references the padstack to be edited. Prior to editing, a clone of the padstack is

// made, and the components that are listed are modified so that they now refer to

// the clone.

Export [padstack manager]

Use: Export a padstack to a library

Command: Tools > Edit Configured Libraries > Padstacks > Export to Library

Syntax: Export Array("NAME:<LibraryName>",

<PadstackName>,

<PadstackName>...),

<LibraryLocation>

Return Value: None

`Parameters: <LibraryName>:`

<string> // name of the library

<PadstackName>:

<string> // simple name of padstack to export

<LibraryLocation>:

<string> // location of the library in <LibraryName>

// One of "Project", "PersonalLib", or "UserLib"

GetNames [padstack manager]

Use: Returns the names of the padstack (used and unused) in a design. The following script command, **IsUsed,** can then be used to separate used and unused padstacks.

Command: None

Syntax: GetNames()

Return Value: An array of strings

`Parameters: None`

IsUsed [padstack manager]

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

Command: None

Syntax: IsUsed(<PadstackName>)

Return Value: <Boolean> // true if the specified padstack is used in the design

`Parameters: <PadstackName>:`

<string>

Remove [padstack manager]

Use: Removes a padstack from a library

Command: Tools > Edit Configured Libraries > Padstacks > Remove Padstacks

Syntax: Remove <PadstackName>,

<IsProjectPadstack>,

<LibraryName>,

<LibraryLocation>

Return Value: None

`Parameters: <PadstackName>:`

<string> // simple name of the padstack to remove

<IsProjectPadstack>:

<bool>

<LibraryName>:

<string> // name of the library

<LibraryLocation>:

<string> // location of the library in <LibraryName>

// One of "Project", "PersonalLib", or "UserLib"

RemoveUnused [padstack manager]

Use: Removes padstacks that are not used in the design.

Command: **Project->Remove Unused Definitions** is similar but operates slightly different and does not record script commands.

Syntax: RemoveUnused()

Return Value: <bool> True if one or more padstacks are removed.

`Parameters: None`

> **Note:**
>
> The order of calls to RemoveUnused is significant. As a result, removing definitions in an unordered fashion may cause other padstacks in dependent definitions to be rendered unusable.

Script and Library Scripts

The definition manager provides access to materials in a project. The manager object is accessed via the definition manager.

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

The script and library script commands are listed below.

AddScript

EditScript

ExportScript

RemoveScript

ModifyLibraries

AddScript

Adds a script to the definition manager.

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><i><AddScriptArray></i></td><td>Array</td><td>Structured array.
Array("NAME:<string script name>",
  "ScriptLang:=", <string script language>
  "ScriptText:=, <string text of script>)</td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	AddScript(<*AddScriptArray*>)

Python Example	`oDefinitionManager.AddScript(`
	`  ["NAME:MyScript",`
	`    "ScriptLang:=", "language",`
	`    "ScriptText:=", "MsgBox(\"HelloWorld\")"`
	`  ]`
	`)`

EditScript

Edits a script in the definition manager.

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td>&lt;<i>OriginalName</i>&gt;</td><td>String</td><td>Name of the script to be edited.</td></tr>
<tr><td>&lt;<i>EditScriptArray</i>&gt;</td><td>Array</td><td>Structured array.
<code>Array("NAME:<string script name&gt;",</code>
<code> "ScriptLang:=", <string script language&gt;</code>
<code> "ScriptText:=, <string text of script&gt;)</code></td></tr>
</table>
</td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	EditScript(<*OriginalName*>, <*EditScriptArray*>)

Python Example	`oDefinitionManager.EditScript("MyScript",`
	`  ["NAME:MyNewScript",`
	`    "ScriptLang:=", "language",`
	`    "ScriptText:=", "MsgBox(\"HelloAgain\")"`
	`  ]`
	`)`

ExportScript

Use: Export to Library in the script definition manager

Command: None

Syntax: ExportScript <ExportData>,<Library location>

Return Value: None

Parameters: <ExportData>

`Array("NAME:<LibraryName>",<ScriptName>,<ScriptName>,…)`

Python Syntax	ExportScript(<ExportData>,<Library location>)

Python Example	`oProject.ExportComponent`
	`(["NAME:mylib", "myscript"], "PersonalLib")`

ModifyLibraries

Use: Configure Libraries on the Tools menu

Command: None

Syntax: ModifyLibraries <DesignName>,Array(<ConfigLibArray>)

Return Value: None

```
Parameters:  <DesignName>

Type: <string>

<ConfigLibArray>

Array("NAME:<LibraryType>,<ConfiguredLib>,<ConfiguredLib>,…),…

<ConfiguredLib> // blank to leave unchanged

<DefinitionType>

Array("<libraryname >","<libraryname>",…)
```

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

	"Vendor Elements\Nonlinear"]])

RemoveScript

Use: Remove Script in the script definition manager

Command: None

Syntax: RemoveScript <ScriptName>,<IsProjectScript>, <LibraryName>,<LibraryLocation>

Return Value: None

Parameters: <ScriptName>

Type: <string>

<IsProjectScript>

Type: <bool>

<LibraryName>

Type: <string>

<LibraryLocation>

Type: <string>

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

Symbol Manager Script Commands

The symbol manager provides access to symbols in a project. The manager object is accessed via the definition manager.

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

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

The symbol manager script commands are listed below.

Add

BringToFront

Edit

EditSymbolAndUpdateComps

Export

GetNames

IsUsed

Remove

RemoveUnused

Add [symbol manager]

Use: Add a symbol

Command: Tools > Edit Configured Libraries > Symbols > Add Symbol

Syntax: Add Array("NAME:<SymbolName>",

"ModTime:=", <ModifiedTimeInfo>,

"Library:=", "", // Library name

"LibLocation:=", "Project", // Project Location

<PinDefInfo>,

<PinDefInfo>,... // optional, to define pins

<GraphicsDataInfo>, // optional, to define graphics

<PropDisplayMapInfo>)) // optional, to define property displays

Return Value: <string>

// composite name of the symbol.

// If the name requested conflicts with the name of an existing

// symbol, the requested name is altered to be unique.

// The name returned reflects any change made to be unique.

Parameters: <SymbolName>:

<string> // simple name of the symbol being added

<ModifiedOnInfo>:

An integer that corresponds to the number of seconds that have elapsed

since 00:00 hours, Jan 1, 1970 UTC from the system clock.

<PinDefInfo>:

Array("NAME:PinDef",

"Pin:=", Array (<string>, // pin name

<real>, // x location

```
<real>, // y location
<real>, // angle in radians
<PinType>,
<real>, // line width
<real>, // line length
<bool>, // mirrored
<int>, // color
<bool>, // true if visible, false if not
<string>, // hidden net name
<OptionalPinInfo>, // optional info
<PropDisplayMapInfo>)) // optional

<PinType>:
<string> // "N" : normal pin
// "I" : input pin
// "O" : output pin

<OptionalPinInfo>:
// Specify both or neither
<bool>, // true if name is to be shown
```

```
<bool>, // true if number is to be shown



<PropDisplayMapInfo>:
Array("NAME:PropDisplayMap",
<PropDisplayInfo>,
<PropDisplayInfo>,...)

<PropDisplayInfo>:
<NameString>, Array(<DisplayTypeInfo>,
<DisplayLocationInfo>,
<int>, // optional, level number
<TextInfo>)
<NameString>:
<string> // PropertyName:=, where PropertyName is the name of
// the property to be displayed

<DisplayTypeInfo>:
<int> // 0 : No display
// 1 : Display name only
// 2 : Display value only
// 3 : Display both name and value
// 4: Display evaluated value only
// 5: Display both name and evaluated value

<DisplayLocationInfo>:
<int> // 0 : Left
// 1 : Top
// 2 : Right
// 3 : Bottom
// 4 : Center
// 5 : Custom placement

<GraphicsDataInfo>:
Array("NAME:Graphics",
// one or more of the following
<RectInfo>,
<CircleInfo>,
<ArcInfo>,
<LineInfo>,
<PolygonInfo>,
<TextInfo>,
<ImageInfo>)

<RectInfo>:
"Rect:=", Array(<real>, // line width
<int>, // fill pattern
<int>, // color
<real>, // angle, in radians
<real>, // x position of center
<real>, // y position of center
<real>, // width
< real>) // height

<CircleInfo>:
"Circle:=", Array(<real>, // line width
<int>, // fill pattern
<int>, // color
<real>, // x position of center
<real>, // y position of center
< real>) // radius
```

```
<ArcInfo>:
"Arc:=", Array(<real>, // line width
<int>, // line pattern
<int>, // color
<real>, // x position of center
<real>, // y position of center
< real>, // radius
<real>, // start angle, in radians
<end>) // end angle, in radians

<LineInfo>:
"Line:=", Array(<real>, // line width
<int>, // line pattern
<int>, // color
<PointInfo>, // must specify at least 2 points
<PointInfo>...)
<PointInfo>:
<real>, // x position
<real> // y position
```

```
<PolygonInfo>:
"Polygon:=", Array(<real>, // line width
<int>, // fill pattern
<int>, // color
<PointInfo>, // must specify at least 3 points
<PointInfo>...)

<TextInfo>:
"Text:=", Array(<real>, // x position
<real>, // y position
<real>, // angle, in radians
<Justification>,
<bool>, // is plotter font
<string>, // font name
<int>, // color
<string>) // text string

<Justification>:
<int> // 0 : left top
// 1 : left base
// 2 : left bottom
// 3 : center top
// 4 : center base
// 5 : center bottom
// 6 : right top
// 7 : right base
// 8 : right bottom

<ImageInfo>:
"Image:=", Array(<RectInfo>,
<ImageData>,
<bool>) // is mirrored

<ImageData>:
<string>, // file path
<int>, // 0 : use the file path and link to it
// 1 : ignore file path and use next parameter
<string> // text data, only present if preceding int is 1
```

BringToFront [symbol manager]

Use: Changes the drawing for the symbol so that the specified objects are drawn on top of other overlapping objects.

Command: Draw > Bring To Front

Syntax: BringToFront Array("NAME:Selections", "Selections:=", Array (<Object>, <Object>, ...))

Return Value: None

```
Parameters: <Object>
<string> // object to bring to the front
```

Edit [deprecated]

Deprecated command — please use EditSymbolAndUpdateComps.

EditSymbolAndUpdateComps [symbol manager]

Use: Edit an existing symbol.

Command: None

Syntax: EditSymbolAndUpdateComps <SymbolName>,

```
        Array("NAME:<NewSymbolName>",
            "ModTime:=", <ModifiedTimeInfo>,
            "Library:=", <string>, // Library name
            "LibLocation:=", <string>, // Project Location
            <PinDefInfo>,
            <PinDefInfo>,... // optional, to define pins
            <GraphicsDataInfo>, // optional, to define graphics
            <PropDisplayMapInfo>), // optional, to define property displays
        Array(<ListOfComponentNames>), // Component names
        <EditContext> //optional
```

Return Value: <string>

// composite name of the symbol.

// If the name requested conflicts with the name of an existing

// symbol, the requested name is altered to be unique.

// The name returned reflects any change made to be unique.

Parameters: <SymbolName>:

<string> // composite name of the symbol being edited

<NewSymbolName>:

<string> // new simple name for the symbol

<ModifiedOnInfo>:

An integer that corresponds to the number of seconds that have elapsed

since 00:00 hours, Jan 1, 1970 UTC from the system clock.

<PinDefInfo>:

Array("NAME:PinDef",

"Pin:=", Array (<string>, // pin name

<real>, // x location

```
<real>, // y location
<real>, // angle in radians
<PinType>,
<real>, // line width
<real>, // line length
<bool>, // mirrored
<int>, // color
<bool>, // true if visible, false if not
<string>, // hidden net name
<OptionalPinInfo>, // optional info
<PropDisplayMapInfo>)) // optional

<PinType>:
<string> // "N" : normal pin
// "I" : input pin
// "O" : output pin

<OptionalPinInfo>:
// Specify both or neither
<bool>, // true if name is to be shown
<bool>, // true if number is to be shown
```

```
<PropDisplayMapInfo>:
Array("NAME:PropDisplayMap",
<PropDisplayInfo>,
<PropDisplayInfo>,...)

<PropDisplayInfo>:
<NameString>, Array(<DisplayTypeInfo>,
<DisplayLocationInfo>,
<int>, // optional, level number
<TextInfo>)

<NameString>:
<string> // PropertyName:=, where PropertyName is the name of
// the property to be displayed

<DisplayTypeInfo>:
<int> // 0 : No display
// 1 : Display name only
// 2 : Display value only
// 3 : Display both name and value
// 4: Display evaluated value only
// 5: Display both name and evaluated value

<DisplayLocationInfo>:
<int> // 0 : Left
// 1 : Top
// 2 : Right
// 3 : Bottom
// 4 : Center
// 5 : Custom placement

<GraphicsDataInfo>:
Array("NAME:Graphics",
// one or more of the following
<RectInfo>,
<CircleInfo>,
<ArcInfo>,
<LineInfo>,
<PolygonInfo>,
<TextInfo>,
<ImageInfo>)

<RectInfo>:
"Rect:=", Array(<real>, // line width
<int>, // fill pattern
<int>, // color
<real>, // angle, in radians
<real>, // x position of center
<real>, // y position of center
<real>, // width
< real>) // height

<CircleInfo>:
"Circle:=", Array(<real>, // line width
<int>, // fill pattern
<int>, // color
<real>, // x position of center
<real>, // y position of center
< real>) // radius
```

```
<ArcInfo>:
"Arc:=", Array(<real>, // line width
<int>, // line pattern
<int>, // color
<real>, // x position of center
<real>, // y position of center
< real>, // radius
<real>, // start angle, in radians
<end>) // end angle, in radians

<LineInfo>:
"Line:=", Array(<real>, // line width
<int>, // line pattern
<int>, // color
<PointInfo>, // must specify at least 2 points
<PointInfo>...)
<PointInfo>:
<real>, // x position
<real> // y position

<PolygonInfo>:
```

```
"Polygon:=", Array(<real>, // line width
<int>, // fill pattern
<int>, // color
<PointInfo>, // must specify at least 3 points
<PointInfo>...)

<TextInfo>:
"Text:=", Array(<real>, // x position
<real>, // y position
<real>, // angle, in radians
<Justification>,
<bool>, // is plotter font
<string>, // font name
<int>, // color
<string>) // text string

<Justification>:
<int> // 0 : left top
// 1 : left base
// 2 : left bottom
// 3 : center top
// 4 : center base
// 5 : center bottom
// 6 : right top
// 7 : right base
// 8 : right bottom

<ImageInfo>:
"Image:=", Array(<RectInfo>,
<ImageData>,
<bool>) // is mirrored

<ImageData>:
<string>, // file path
<int>, // 0 : use the file path and link to it
// 1 : ignore file path and use next parameter
<string> // text data, only present if preceding int is 1

<ListOfComponentNames>:
<string>,<string> ...
// The list may be empty. When not empty, each string that is listed is a component
```

// that references the symbol to be edited. Prior to editing, a clone of the symbol is

// made, and the components that are listed are modified so that they now refer to

// the clone.

<EditContext>:

// Changes that will be made to the component in support of the symbol changes

<RefPinOption>:

// <int>

// 0 = implied reference to ground,

// 1 = single common reference port,

// 2 = individual hidden reference pins for each port,

// 3 = individual reference pin per port

<CompName> // <string>

<TermAttributes>:

//<array>

<Terminal Name> // <string>

<Symbol Pin Name> // <string>

<InOut>// <int>

// 0=in

// 1=out

// 2= inout

<Domain> // <array>

// kConservative=0

// kSignal

// kQuantity

// kParameter

// kState

// kFlexible

<ID> // <int>

//-1, not used.

<Description>

Export [symbol manager]

Use: Exports symbol(s) to a library

Command: Tools > Edit Configured Libraries > Symbols > Export to Library

Syntax: Export Array("NAME:<LibraryName>",

<SymbolName>,

<SymbolName>...),

<LibraryLocation>

Return Value: None

Parameters: <LibraryName>:

<string> // name of the library

<SymbolName>:

<string> // composite name of symbol to export

<LibraryLocation>:

<string> // location of the library in <LibraryName>

// One of "Project", "PersonalLib", or "UserLib"

GetNames [symbol manager]

Use: Returns the names of the symbols (used and unused) in a design. The following script command, **IsUsed,** can then be used to separate used and unused symbols.

Command: None

Syntax: GetNames()

Return Value: An array of strings

Parameters: None

IsUsed [symbol manager]

Use: Used to determine if a symbol is used in the design.

Command: None

Syntax: IsUsed(<SymbolName>)

Return Value: <Boolean> // true if the specified symbol is used in the design

Parameters: <SymbolName>:

<string>

Remove [symbol manager]

Use: Removes a symbol from a library

Command: Tools > Edit Configured Libraries > Symbols > Remove Symbol

Syntax: Remove <SymbolName>,

<IsProjectSymbol>,

<LibraryName>,

<LibraryLocation>

Return Value: None

Parameters: <SymbolName>:

<string> // composite name of the symbol to remove

<IsProjectSymbol>:

<bool>

<LibraryName>:

<string> // name of the library

<LibraryLocation>:

<string> // location of the library in <LibraryName>

// One of "Project", "PersonalLib", or "UserLib"

Example:

```
oSymbolManager.Remove "Nexxim Circuit Elements\Distributed\Distributed:bendo", true, "Project"
```

RemoveUnused [symbol manager]

Use: Removes symbols that are not used in the design.

Command: **Project->Remove Unused Definitions** is similar but operates slightly different and does not record script commands.

Syntax: RemoveUnused()

Return Value: <bool> True if one or more symbols are removed.

Parameters: None

Note:

The order of calls to RemoveUnused is significant. As a result, removing definitions in an unordered fashion may cause other symbols in dependent definitions to be rendered unusable.

Also, the symbol and footprint of an unused component are not unusable until after the component itself is removed using the Component Manager Remove script.

This page intentionally
left blank.

22 - Core Global Script Context Commands

To run these commands:

```
import CoreGlobalScriptContextFunctions

CoreGlobalScriptContextFunctions.[CommandName]
```

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

- AddErrorMessage
- AddFatalMessage
- AddInfoMessage
- AddWarningMessage
- LogDebug
- LogError

AddErrorMessage

Adds an error message to the **Message Manager** window. AddErrorMessageis a function of CoreGlobalScriptContextFunctions.

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><i>&lt;message&gt;</i></td><td>String</td><td>Error message.</td></tr></table></td></tr>
<tr><td><b>Return Value</b></td><td>None.</td></tr>
</table>

Python Syntax	AddErrorMessage(<*message*>)

Python Example	`import CoreGlobalScriptContextFunctions`
	`CoreGlobalScriptContextFunctions.AddErrorMessage('My error message.')`

AddFatalMessage

Adds a fatal error message to the **Message Manager** window. AddFatalMessageis a function of CoreGlobalScriptContextFunctions.

<table>
<tr><td><b>UI Access</b></td><td>N/A</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td>Name</td><td>Type</td><td>Description</td></tr></table></td></tr>
</table>

	<message>	String	Error message.
Return Value	None.		

Python Syntax	AddFatalMessage(<*message*>)

Python Exampl-e	`import CoreGlobalScriptContextFunctions`
	`CoreGlobalScriptContextFunctions.AddFatalMessage('My fatal error message.')`

AddInfoMessage

Adds an informational message to the **Message Manager** window. AddInfoMessageis a function of CoreGlobalScriptContextFunctions.

UI Access	N/A		
Parameters	Name	Type	Description
	<message>	String	Informational message.
Return Value	None.		

Python Syntax	AddInfoMessage(<*message*>)

Python Example	`import CoreGlobalScriptContextFunctions`
	`CoreGlobalScriptContextFunctions.AddInfoMessage('My info.')`

AddWarningMessage

Adds a warning message to the **Message Manager** window. AddWarningMessageis a function of CoreGlobalScriptContextFunctions.

UI Access	N/A		
Parameters	Name	Type	Description
	<message>	String	Warning message.
Return Value	None.		

Python Syntax	AddWarningMessge(<*message*>)

Python Example	`import CoreGlobalScriptContextFunctions`
	`CoreGlobalScriptContextFunctions.AddWarningMessage('My warning.')`

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	`import CoreGlobalScriptContextFunctions`
	`CoreGlobalScriptContextFunctions.LogDebug('My debug message.')`

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	`import CoreGlobalScriptContextFunctions`
	`CoreGlobalScriptContextFunctions.LogError('My error.')`

23 - Example Scripts

This section contains IronPython example scripts.

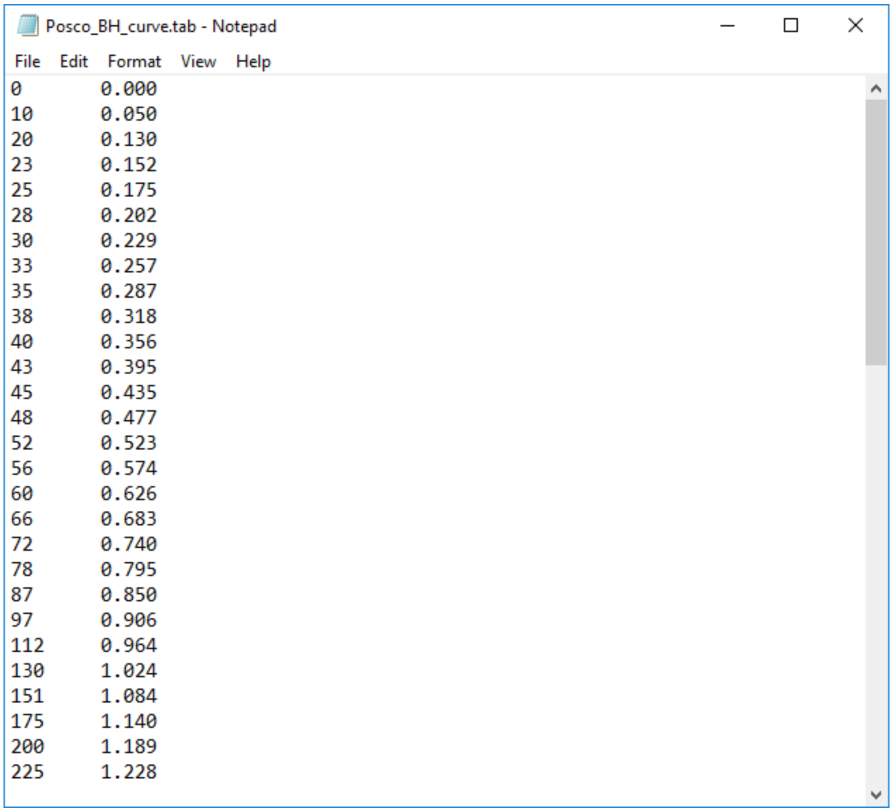
IronPython Example Scripts

IronPython Examples:

- BH Coordinates Python Script
- Equation Based Curve Python Script

BH Coordinates Python Script

This sample Python script adds a material ("Posco 35PN250") with BH coordinates from a specified file (Posco_BH_curve.tab):



```
0       0.000
10      0.050
20      0.130
23      0.152
25      0.175
28      0.202
30      0.229
33      0.257
35      0.287
38      0.318
40      0.356
43      0.395
45      0.435
48      0.477
52      0.523
56      0.574
60      0.626
66      0.683
72      0.740
78      0.795
87      0.850
97      0.906
112     0.964
130     1.024
151     1.084
175     1.140
200     1.189
225     1.228
```

To recreate this tab file, paste the text below the script into a text editor and save as Posco_BH_curve.tab.

The script itself includes comment lines, which are preceded by # and offer explanations for the subsequent line(s).

Script Contents

```
# specify path to the file with BH coordinates

path_to_file = r"D:\Posco_BH_curve.tab"

# specify name of the material

material_name = "Posco 35PN250"

oProject = oDesktop.GetActiveProject()

oDefinitionManager = oProject.GetDefinitionManager()

""" create list with B and H points to pass to AddMaterial command

bh_coordinates list is a three dimensional array: 1D: name, 2D:
Coordinate list, 3D: BH point"""

bh_coordinates = ["NAME:BHCoordinates", ["NAME:DimUnits", "", ""]]

with open(path_to_file) as input_file:

  for line in input_file:

    h, b = line.split()


    bh_coordinates.append(["NAME:Coordinate", [

                  "NAME:CoordPoint",

                  float(h),

                  float(b)

                ]

              ])

# create a new magnetic material with BH curve

oDefinitionManager.AddMaterial(

  [

    "NAME:" + material_name,

    "CoordinateSystemType:=", "Cartesian",

    "BulkOrSurfaceType:=" , 1,

    [

      "NAME:PhysicsTypes",

      "set:=" , ["Electromagnetic"]
    ],
    [
      "NAME:permeability",
      "property_type:=" , "nonlinear",
      "BTypeForSingleCurve:=" , "normal",
      "HUnit:=" , "A_per_meter", # unit can be specified as variable
      "BUnit:=" , "tesla", # unit can be specified as variable
      "IsTemperatureDependent:=", False,
      bh_coordinates,
      [
        "NAME:Temperatures"
      ]
    ],
    "conductivity:=" , "1818181.82",
    [
      "NAME:magnetic_coercivity",
      "property_type:=" , "VectorProperty",
      "Magnitude:=" , "0A_per_meter",
      "DirComp1:=" , "1",
      "DirComp2:=" , "0",
      "DirComp3:=" , "0"
    ]
  ])
```

Posco_BH_Curve.tab Contents

```
0       0.000
10      0.050
20      0.130
23      0.152
25      0.175
28      0.202
```

```
30      0.229
33      0.257
35      0.287
38      0.318
40      0.356
43      0.395
45      0.435
48      0.477
52      0.523
56      0.574
60      0.626
66      0.683
72      0.740
78      0.795
87      0.850
97      0.906
112     0.964
130     1.024
151     1.084
175     1.140
200     1.189
225     1.228
252     1.258
282     1.283
317     1.304
357     1.324
402     1.343
450     1.360
500     1.375
550     1.387
```

```
602      1.398
657      1.407
717      1.417
784      1.426
867      1.437
974      1.448
1117     1.461
1299     1.478
1515     1.495
1752     1.513
2000     1.529
2253     1.543
2521     1.557
2820     1.570
3167     1.584
3570     1.600
4021     1.617
4503     1.634
5000     1.652
5503     1.669
6021     1.685
6570     1.701
7167     1.717
7839     1.733
8667     1.749
9745     1.769
11167    1.793
12992    1.820
15146    1.851
17518    1.882
```

```
20000   1.909
22552   1.931
25417   1.948
28906   1.961
33333   1.971
38932   1.981
```

Equation Based Curve Python Script

This sample Python script creates an equation based curve that produces a helix.

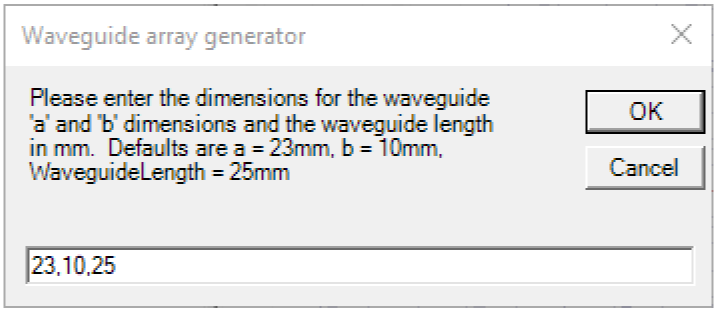
```
from math import pi, sin, cos

oProject = oDesktop.GetActiveProject()
oDesign = oProject.GetActiveDesign()
oEditor = oDesign.SetActiveEditor("3D Modeler")

Start_t = 0
End_t = pi*2

Npoint = 128
Nsection = Npoint-1

d_t = (End_t-Start_t)/Nsection

for n in range(1,Nsection):

  P1 = Start_t+d_t*(n-1)
  P2 = P1+d_t
  X_t1 = cos(P1*6)
  Y_t1 = sin(P1*6)
  Z_t1 = P1
  X_t2 = cos(P2*6)
  Y_t2 = sin(P2*6)
  Z_t2 = P2


oEditor.CreatePolyline(
  [
    "NAME:PolylineParameters",
    "IsPolylineCovered:=" , True,
    "IsPolylineClosed:=" , False,
    [
    "NAME:PolylinePoints",
    [
    "NAME:PLPoint",
    "X:=" , '1mm*' + str(X_t1),
    "Y:=" , '1mm*' + str(Y_t1),
    "Z:=" , '1mm*' + str(Z_t1)
    ],
    [
    "NAME:PLPoint",
    "X:=" , '1mm*' + str(X_t2),
    "Y:=" , '1mm*' + str(Y_t2),
    "Z:=" , '1mm*' + str(Z_t2)
    ]
    ],
    [
    "NAME:PolylineSegments",
    [
    "NAME:PLSegment",
    "SegmentType:=" , "Line",
    "StartIndex:=" , 0,
  "NoOfPoints:=" , 2
  ]
  ],
  [
  "NAME:PolylineXSection",
  "XSectionType:=" , "None",
  "XSectionOrient:=" , "Auto",
  "XSectionWidth:=" , "0mm",
  "XSectionTopWidth:=" , "0mm",
  "XSectionHeight:=" , "0mm",
  "XSectionNumSegments:=" , "0",
  "XSectionBendType:=" , "Corner"
  ]
],
[
  "NAME:Attributes",
  "Name:=" , "Polyline"+str(n),
  "Flags:=" , "",
  "Color:=" , "(132 132 193)",
  "Transparency:=" , 0,
  "PartCoordinateSystem:=", "Global",
  "UDMId:=" , "",
  "MaterialValue:=" , "\"vacuum\"",
  "SurfaceMaterialValue:=", "\"\"",
  "SolveInside:=" , True,
  "IsMaterialEditable:=" , True,
  "UseMaterialAppearance:=", False,
  "IsLightweight:=" , False
])
```

HFSS Waveguide Array Python Script

This sample Python script creates an HFSS Waveguide array. The script includes comment lines, which are preceded by an apostrophe (’), that offer explanations for each subsequent line or lines. The script includes examples of creating a 3D model, boundaries and excitation, solution setup and sweep, as well as reports.

You must insert an HFSS project before running the script. Running the script causes a series of input dialogs to appear that lets you set parameters.

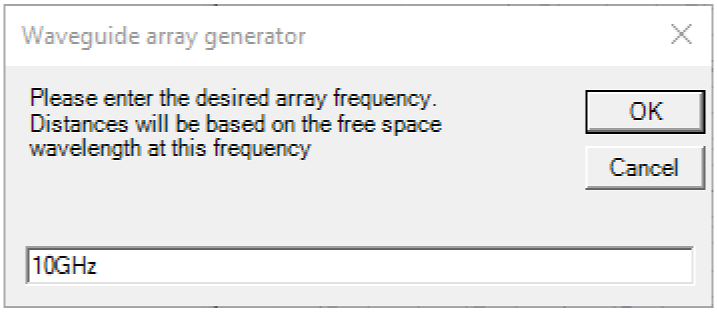
The first dialog asks for input for a and be dimensions and the waveguide length.



Please enter the dimensions for the waveguide 'a' and 'b' dimensions and the waveguide length in mm. Defaults are a = 23mm, b = 10mm, WaveguideLength = 25mm

23,10,25

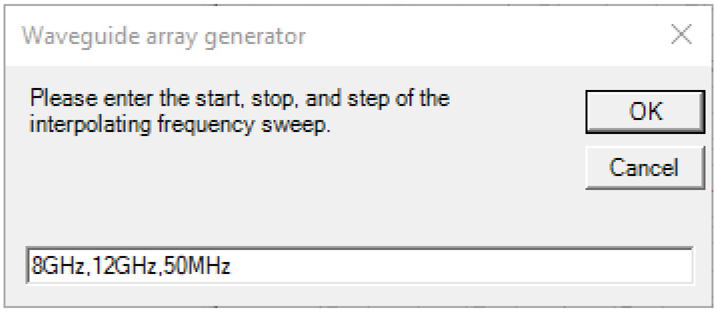
The next asks for the array frequency.



Please enter the desired array frequency. Distances will be based on the free space wavelength at this frequency

10GHz

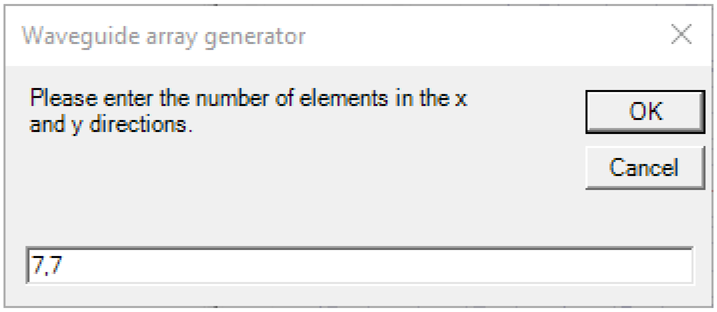
The next asks for start, stop and step values for in an interpolating frequency sweep.



Please enter the start, stop, and step of the interpolating frequency sweep.

8GHz,12GHz,50MHz

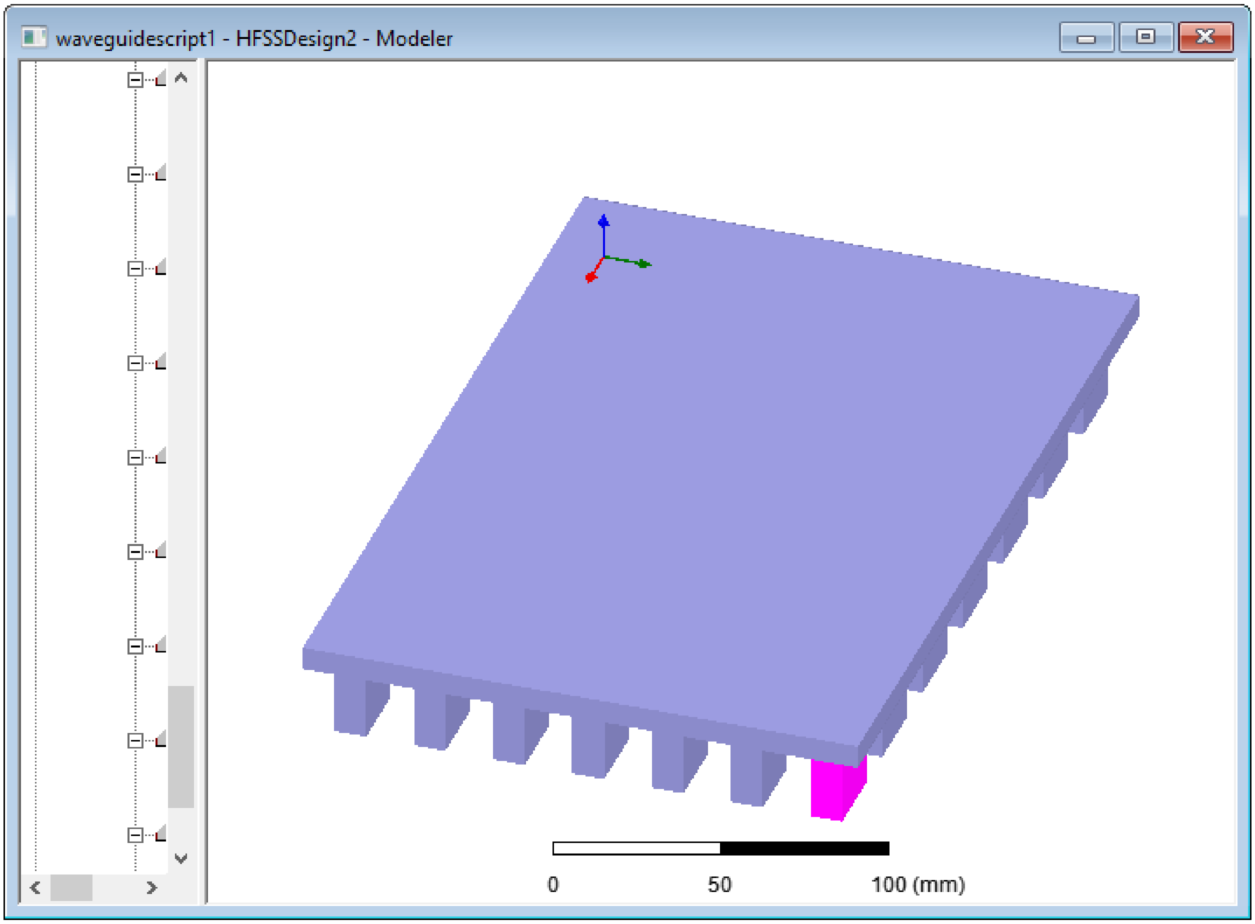
The last one asks for the array definition.



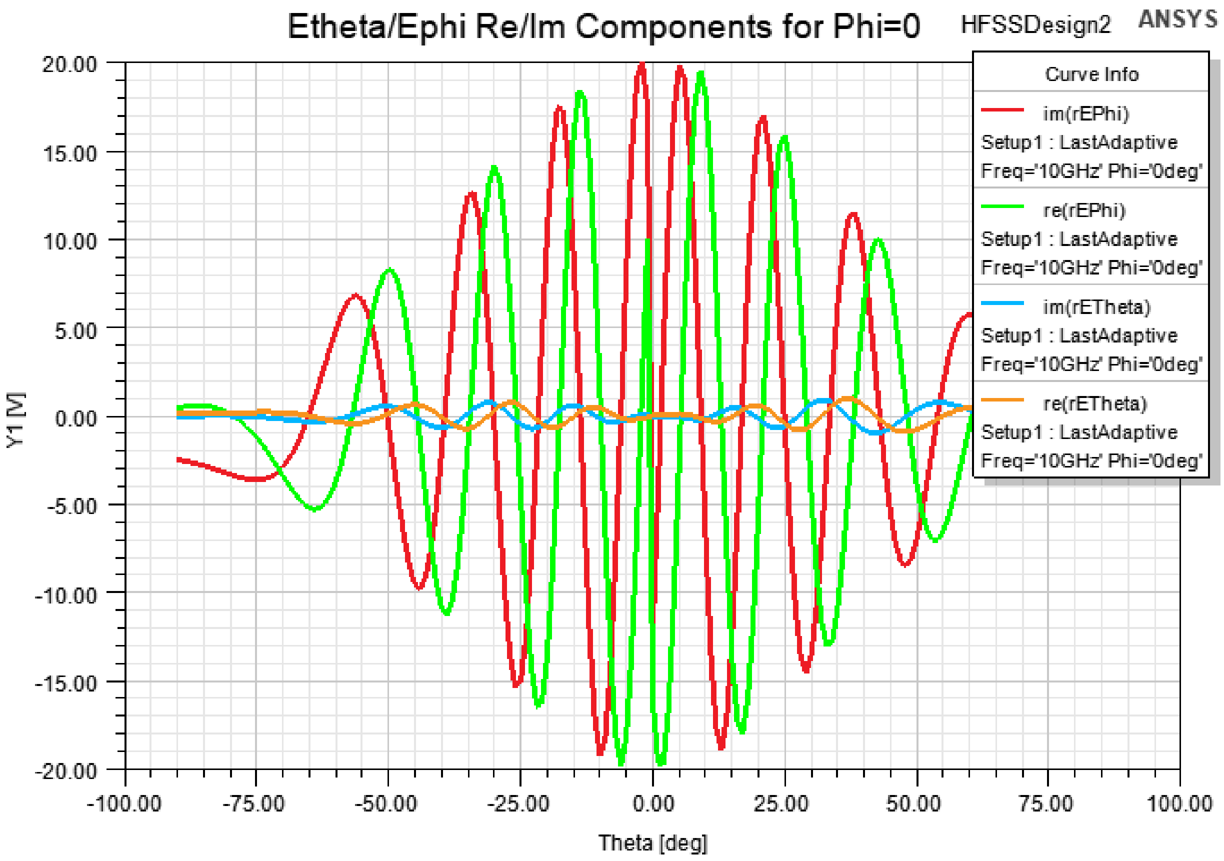
Please enter the number of elements in the x and y directions.

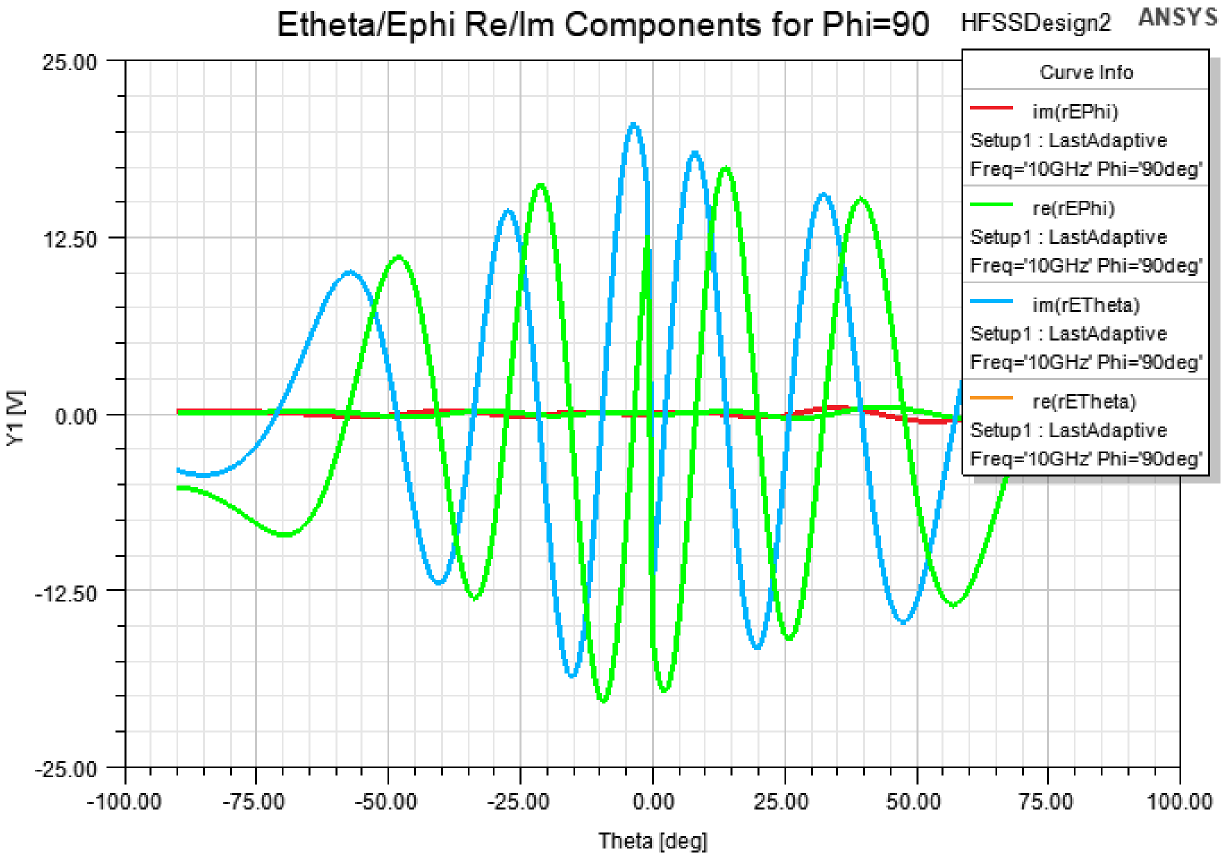
7,7

The following figure shows the waveguide array generated by the defaults.



After running the simulation, the plots defined by the script shows the following results.


Etheta/Ephi Re/Im Components for Phi=0
HFSSDesign2
ANSYS
Curve Info
im(rEPhi)
Setup1 : LastAdaptive
Freq='10GHz' Phi='0deg'
re(rEPhi)
Setup1 : LastAdaptive
Freq='10GHz' Phi='0deg'
im(rETheta)
Setup1 : LastAdaptive
Freq='10GHz' Phi='0deg'
re(rETheta)
Setup1 : LastAdaptive
Freq='10GHz' Phi='0deg'
20.00
15.00
10.00
5.00
0.00
-5.00
-10.00
-15.00
-20.00
Y1 [V]
-100.00
-75.00
-50.00
-25.00
0.00
25.00
50.00
75.00
100.00
Theta [deg]



The waveguide script in Python follows.

```
import clr

clr.AddReferenceByPartialName("Microsoft.VisualBasic")

from Microsoft.VisualBasic.Constants import
vbOKOnly,vbOKCancel,vbAbortRetryIgnore,vbYesNoCancel,vbYesNo,vbRetryC-
ancel

from Microsoft.VisualBasic.Constants import
vbOK,vbCancel,vbAbort,vbRetry,vbIgnore,vbYes,vbNo

from Microsoft.VisualBasic.Interaction import InputBox, MsgBox


oProject = oDesktop.GetActiveProject()

oDesign = oProject.GetActiveDesign()

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

```
# Ask for dimensions for waveguide dimensions

# ------------------------------------------------------------

dim = InputBox("Please enter the dimensions for the waveguide 'a' and
'b' dimensions "

  + "and the waveguide length in mm. Defaults are a = 23mm, b = 10mm,
  WaveguideLength = 25mm",

  "Waveguide array generator", "23,10,25")

Dimensions = dim.split(',')


a_str = Dimensions[0] + "mm"

b_str = Dimensions[1] + "mm"

b_over2 = float(Dimensions[1])/2


WaveguideLength_str = Dimensions[2] + "mm"


#Ask for the frequency of operation

#----------------------------------

Frequency = InputBox("Please enter the desired array frequency.
Distances will " +

  "be based on the free space wavelength at this frequency",

    "Waveguide array generator", "10GHz")


#Ask for the start, stop, and step of the interpolating frequency
sweep

#-------------------------------------------------------------------
--

inputText = InputBox("Please enter the start, stop, and step of the
interpolating " +

  "frequency sweep.", "Waveguide array generator",
  "8GHz,12GHz,50MHz")

  StartFrequency, StopFrequency, StepFrequency = inputText.split(',')
```

```
#Ask for the number of elements in the x and y directions
#-------------------------------------------------------
inputText = InputBox("Please enter the number of elements in the x
and y directions.",
  "Waveguide array generator", "7,7")
numX, numY = [float(elem) for elem in inputText.split(',')]
TotalElements = numX*numY

# Make variables and set them equal to the values from the user input
# ---------------------------------------------------------------------
oDesign.ChangeProperty(
  [
    "NAME:AllTabs",
    [
    "NAME:LocalVariableTab",
    [
    "NAME:PropServers",
    "LocalVariables"
    ],
    [
    "NAME:NewProps",
    [
    "NAME:a", "PropType:=", "VariableProp", "UserDef:=", True,
    "Value:=", a_str
  ],
  [
    "NAME:b", "PropType:=", "VariableProp", "UserDef:=", True,
    "Value:=", b_str
  ],
  [
    "NAME:NumX", "PropType:=", "VariableProp", "UserDef:=", True,
    "Value:=", numX
  ],
  [
    "NAME:NumY", "PropType:=", "VariableProp", "UserDef:=", True,
    "Value:=", numY
  ],
  [
    "NAME:WaveguideLength", "PropType:=", "VariableProp", "UserDef:=", True,
    "Value:=", WaveguideLength_str
  ],
  [
    "NAME:Frequency", "PropType:=", "VariableProp", "UserDef:=", True,
    "Value:=", Frequency
  ],
  [
    "NAME:Lambda", "PropType:=", "VariableProp", "UserDef:=", True,
    "Value:=", "c0/" + Frequency
  ],
  [
    "NAME:RadBoundDist", "PropType:=", "VariableProp", "UserDef:=", True,
    "Value:=", "Lambda/4"
  ]
  ]
  ]
  ])
#Create the radiation box
#------------------------
```

```
oEditor.CreateBox(
  [
    "NAME:BoxParameters",
    "XPosition:=" , "-a/2-RadBoundDist",
    "YPosition:=" , "-b/2-RadBoundDist",
    "ZPosition:=" , "0mm",
    "XSize:=" , "NumX*a+(NumX-1)*Lambda/2+2*RadBoundDist",
    "YSize:=" , "NumY*b+(NumY-1)*Lambda/2+2*RadBoundDist",
    "ZSize:=" , "RadBoundDist"
  ],
  [
    "NAME:Attributes",
    "Name:=" , "RadiationBox",
    "Flags:=" , "",
    "Color:=" , "(132 132 193)",
    "Transparency:=" , 0.8,
    "PartCoordinateSystem:=", "Global",
    "UDMId:=" , "",
    "MaterialValue:=" , "\"vacuum\"",
    "SurfaceMaterialValue:=", "\"\"",
    "SolveInside:=" , True,
    "IsMaterialEditable:=" , True,
    "UseMaterialAppearance:=", False,
    "IsLightweight:=" , False
  ])

oEditor.FitAll() # Zoom out
#Create first element
#--------------------
oEditor.CreateBox(
  [
    "NAME:BoxParameters",
    "XPosition:=" , "-a/2",
    "YPosition:=" , "-b/2",
    "ZPosition:=" , "0mm",
    "XSize:=" , "a",
    "YSize:=" , "b",
    "ZSize:=" , "-WaveguideLength"
  ],
  [
  "NAME:Attributes",
    "Name:=" , "Element1",
    "Flags:=" , "",
    "Color:=" , "(132 132 193)",
    "Transparency:=" , 0.8,
    "PartCoordinateSystem:=", "Global",
    "UDMId:=" , "",
    "MaterialValue:=" , "\"vacuum\"",
    "SurfaceMaterialValue:=", "\"\"",
    "SolveInside:=" , True,
    "IsMaterialEditable:=" , True,
    "UseMaterialAppearance:=", False,
    "IsLightweight:=" , False
  ])

# Define the port
# ---------------
# Get the numeric value of half of the b dimension of the port.
# The values fed in to the "Start" and "End" arrays cannot have mathematical operators. For example, if HFSS
# has a variable 'b', and a desired coordinate is 'b/2', that value
cannot be entered here as "b/2". The number

# has to be computed explicitly in script and entered as a string
such as "-200mm".

oModule = oDesign.GetModule("BoundarySetup")


# get bottom face ID

element1FaceID = oEditor.GetFaceByPosition(

  [

    "NAME:Parameters",

    "BodyName:=", "Element1",

    "XPosition:=", "-a/2",

    "YPosition:=", "-b/2",

    "ZPosition:=", "-WaveguideLength"

  ])

oModule.AssignWavePort(

  [

    "NAME:WavePort1",

    "Faces:=" , [element1FaceID],

    "NumModes:=" , 1,

    "RenormalizeAllTerminals:=", True,

    "UseLineModeAlignment:=", False,

    "DoDeembed:=" , False,

  [

    "NAME:Modes",

    [

    "NAME:Mode1",

    "ModeNum:=" , 1,

    "UseIntLine:=" , True,

    [

    "NAME:IntLine",
    "Start:=" , ["0mm", "-" +str(b_over2) + "mm", "-" + WaveguideLength_str],

    "End:=" , ["0mm", str(b_over2) + "mm", "-" + WaveguideLength_str]

    ],

    "AlignmentGroup:=" , 0,

    "CharImp:=" , "Zpi"

    ]

    ],

  "ShowReporterFilter:=" , False,

  "ReporterFilter:=" , [True],

  "UseAnalyticAlignment:=", False

  ])


#Set the radiation boundary

#--------------------------

# Get face IDs for further assignment

top_face_id = oEditor.GetFaceByPosition(["NAME:FaceParameters",

    "BodyName:=", "RadiationBox",

    "XPosition:=", "0mm",

    "YPosition:=", "NumY*b-b/2+(NumY-1)*Lambda/2+RadBoundDist",

    "ZPosition:=", "0mm"])


faceIDs = [int(elem) for elem in oEditor.GetFaceIDs("RadiationBox")
if elem != str(top_face_id)]

oModule.AssignRadiation(

  [

    "NAME:Radiation",

    "Faces:=" , faceIDs,

    "IsFssReference:=" , False,

    "IsForPML:=" , False

  ])
```

```
# Copy and paste elements/ports into a rectangular array.

# Duplicate boundaries with geometry" must be turned on under Tools-
>Options->HFSS Options

# -------------------------------------------------------------------
----------------------

ElementNum = 1

for i in range(1, int(numX)+1):

  for j in range(1, int(numY)+1):

    if ElementNum == 1:

  pass


elif ElementNum <= numY: #If in the first column, only

    oEditor.Copy(["NAME:Selections", "Selections:=", "Element1"])

    oEditor.Paste()

    oEditor.Move(["NAME:Selections", "Selections:=", "Element" + str
    (ElementNum)],

  ["NAME:TranslateParameters", "CoordinateSystemID:=", -1,

  "TranslateVectorX:=", "0mm",

  "TranslateVectorY:=", str(j-1) + "*(b+Lambda/2)",

  "TranslateVectorZ:=", "0mm"])


elif ElementNum > numY:

    oEditor.Copy(["NAME:Selections", "Selections:=", "Element1"])

    oEditor.Paste()

    oEditor.Move(["NAME:Selections", "Selections:=", "Element" + str
    (ElementNum)],

  ["NAME:TranslateParameters", "CoordinateSystemID:=", -1,

    "TranslateVectorX:=", str(i-1) + "*(a+Lambda/2)",

    "TranslateVectorY:=", str(j-1) + "*(b+Lambda/2)",

    "TranslateVectorZ:=", "0mm"])
```

```
ElementNum += 1

#Create the setup and interpolating sweep
#-----------------------------------------
oModule = oDesign.GetModule("AnalysisSetup")
oModule.InsertSetup("HfssDriven",
  [
    "NAME:Setup1",
    "AdaptMultipleFreqs:=" , False,
    "Frequency:=" , Frequency,
    "MaxDeltaS:=" , 0.02,
    "PortsOnly:=" , False,
    "UseMatrixConv:=" , False,
    "MaximumPasses:=" , 15,
    "MinimumPasses:=" , 1,
    "MinimumConvergedPasses:=", 1,
    "PercentRefinement:=" , 50,
    "IsEnabled:=" , True,
    "BasisOrder:=" , 1,
    "DoLambdaRefine:=" , True,
    "DoMaterialLambda:=" , True,
    "SetLambdaTarget:=" , False,
    "Target:=" , 0.3333,
    "UseMaxTetIncrease:=" , False,
    "PortAccuracy:=" , 2,
    "UseABCOnPort:=" , False,
    "SetPortMinMaxTri:=" , False,
    "UseDomains:=" , False,
    "UseIterativeSolver:=" , False,
    "SaveRadFieldsOnly:=" , False,
    "SaveAnyFields:=" , True,
    "IESolverType:=" , "Auto",
    "LambdaTargetForIESolver:=", 0.15,
    "UseDefaultLambdaTgtForIESolver:=", True
    ])

oModule.InsertFrequencySweep("Setup1",
  [
    "NAME:InterpolatingSweep",
    "IsEnabled:=" , True,
    "RangeType:=" , "LinearStep",
    "RangeStart:=" , StartFrequency,
    "RangeEnd:=" , StopFrequency,
    "RangeStep:=" , StepFrequency,
    "Type:=" , "Interpolating",
    "SaveFields:=" , False,
    "InterpTolerance:=" , 0.5,
    "InterpMaxSolns:=" , 50,
    "InterpMinSolns:=" , 0,
    "InterpMinSubranges:=" , 1,
    "ExtrapToDC:=" , False,
    "InterpUseS:=" , True,
    "InterpUsePortImped:=" , False,
    "InterpUsePropConst:=" , True,
    "UseDerivativeConvergence:=", False,
    "InterpDerivTolerance:=", 0.2,
    "UseFullBasis:=" , True,
    "EnforcePassivity:=" , True,
    "PassivityErrorTolerance:=", 0.0001
    ])
```

```
#Create a relative coordinate system centered on the array for radi-
ation pattern calculations
#--------------------------------------------------------------------
------------------------
oEditor.CreateRelativeCS(
  [
    "NAME:RelativeCSParameters",
    "Mode:=" , "Axis/Position",
    "OriginX:=" , "-a/2+NumX*a/2+(NumX-1)*Lambda/4",
    "OriginY:=" , "-b/2+NumY*b/2+(NumY-1)*Lambda/4",
    "OriginZ:=" , "0mm",
    "XAxisXvec:=" , "1mm",
    "XAxisYvec:=" , "0mm",
    "XAxisZvec:=" , "0mm",
    "YAxisXvec:=" , "0mm",
    "YAxisYvec:=" , "1mm",
    "YAxisZvec:=" , "0mm"
    ],
    [
    "NAME:Attributes",
    "Name:=" , "RelativeCS1"
  ])
#Create an infinite sphere with fine theta resolution and phi cuts at
0 and 90 degrees
#--------------------------------------------------------------------
-----------------
oModule = oDesign.GetModule("RadField")
oModule.InsertFarFieldSphereSetup(
  [
    "NAME:Infinite Sphere1",
    "UseCustomRadiationSurface:=", False,
    "ThetaStart:=" , "-90deg",
    "ThetaStop:=" , "90deg",
    "ThetaStep:=" , "1deg",
    "PhiStart:=" , "0deg",
    "PhiStop:=" , "90deg",
    "PhiStep:=" , "90deg",
    "UseLocalCS:=" , True,
    "CoordSystem:=" , "RelativeCS1"
  ])


#Create output plots for Ephi/Etheta real and imaginary components
for Phi = 0 and separately for Phi = 90
#---------------------------------------------------------------------
-------------------------------------
oModule = oDesign.GetModule("ReportSetup")
#For phi = 0
oModule.CreateReport("Etheta/Ephi Re/Im Components for Phi=0", "Far
Fields",
  "Rectangular Plot", "Setup1 : LastAdaptive",
    [
    "Context:=" , "Infinite Sphere1"
    ],
    [
    "Theta:=" , ["All"],
    "Phi:=" , ["0deg"],
    "Freq:=" , ["All"],
    "a:=" , ["Nominal"],
    "b:=" , ["Nominal"],
    "NumX:=" , ["Nominal"],
    "NumY:=" , ["Nominal"],
    "WaveguideLength:=" , ["Nominal"],
    "Frequency:=" , ["Nominal"]
    ],
    [
    "X Component:=" , "Theta",
    "Y Component:=" , ["im(rEPhi)","re(rEPhi)","im(rETheta)","re
    (rETheta)"]
  ], [])

#For phi = 90
#------------
oModule.CreateReport("Etheta/Ephi Re/Im Components for Phi=90", "Far
Fields",
  "Rectangular Plot", "Setup1 : LastAdaptive",
    [
    "Context:=" , "Infinite Sphere1"
    ],
    [
    "Theta:=" , ["All"],
    "Phi:=" , ["90deg"],
    "Freq:=" , ["All"],
    "a:=" , ["Nominal"],
    "b:=" , ["Nominal"],
    "NumX:=" , ["Nominal"],
    "NumY:=" , ["Nominal"],
    "WaveguideLength:=" , ["Nominal"],
    "Frequency:=" , ["Nominal"]
    ],
    [
    "X Component:=" , "Theta",
    "Y Component:=" , ["im(rEPhi)","re(rEPhi)","im(rETheta)","re
    (rETheta)"]
```

```
], [])
```

This page intentionally
left blank.

Index