Circuit Scripting Guide

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

Release 2024 R2
July 2024

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 VBScript or IronPython, and can run external scripts written in VBScript, IronPython, CPython, or JavaScript. Additionally, it contains an IronPython command shell for executing scripts.

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

The following sections contain more information about scripting:

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

Scripting Help Conventions

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

[ScriptName]

[Description of script use.]

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

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

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

Variable Types

The following data types are used throughout the help:

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

Introduction to VBScript

Ansys Electronics Desktop can use Microsoft® Visual Basic® Scripting (VBScript) to record macros. VBScript is based on the Microsoft Visual Basic programming language.

This chapter provides an overview of key VBScript components.

For more details about VBScript, please see the *Recommended VBScript References* section at the end of this chapter.

Simple and Composite Names

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

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

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

VBScript Variables

A VBScript variable is a placeholder representing information that may change during the time your script is running. Variables are useful because they let you assign a short and easy to remember name to each piece of data you plan to use. Use a variable name in a script to view or modify its value.

Declaring Variables

To declare variables explicitly in a script, use the Dim, Public, or Private statements. For example:

```
Dim box_xsize
```

After declaring a variable, you can assign information to it. For example:

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

You can declare multiple variables by separating each variable name with a comma. For example:

```
Dim Top, Bottom, Left, Right
```

You can also declare a variable implicitly by simply using its name in your script. Doing so is not generally a good practice because you could misspell the variable name in one or more places, causing unexpected results when your script is run. For that reason, the **Option Explicit** statement is available to require explicit declaration of all variables. The **Option Explicit** statement should be the first statement in your script.

Declaring Variables in Python

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

You can directly assign information to it. For example:

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

Variable Naming Conventions

You should use names that are short but intuitive and easy to remember. Use the following conventions for naming variables in VBScript:

- Begin with an alphabetic character.
- Cannot contain an embedded period.
- Must not exceed 255 characters.
- Must be unique in the scope in which it is declared.
- Do not use VBScript keywords.

Scope and Lifetime of Variables

Variables at the script level are available to all procedures within the script. At the procedure level, variables are available only within the procedure. It has local scope and is a procedure-level variable.

The lifetime of a variable depends on how long it exists. The script-level variables exist from declaration until the end of the script. A procedure-level variable exists only as long as you are in the procedure and is destroyed when the procedure exits.

Array Variables

Create an array variable when you want to assign more than one related value to a single variable. An array variable contains a series of values. For example:

```
Dim Primitives(2)
```

All arrays in VBScript are zero-based, so the array above actually contains 3 elements. You assign data to each of the array's elements using an index into the array. Data can be assigned to the elements of an array as follows:

```
Primitives(0) = "Box1"

Primitives(1) = "Cone1"
```

```
Primitives(2) = "Cylinder1"
```

Similarly, the data can be retrieved from any element using an index into a particular array element. For example:

```
one_prim = Primitives(1)
```

You can also use the Array function to assign an array of elements to a variable. For example:

```
Dim Primitives

Primitives = Array ("Box1", "cone1", "Cylinder1")
```

> **Note:**
>
> When using the Array function, do not use parentheses on the variable when it is declared. For example, use Dim myarray, not Dim myarray().

If you do not know the size of the array at declaration or the size changes during the time your script is running, you can use dynamic arrays. They are declared without size or number of dimensions inside the parentheses. For example:

```
Dim FirstArray()

ReDim SecondArray()
```

To use a dynamic array, you must subsequently use **ReDim** to determine the number of dimensions and the size of each dimension. You can also use the **Preserve** keyword to preserve the contents of the array as the resizing takes place.

```
ReDim FirstArray(25)

ReDim Preserve FirstArray(30)
```

VBScript Operators

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

Please see the online *VBScript User's Guide* for more details.

Operator Precedence

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

When an expression contains operators from more than one category, they are evaluated in the following order:

1. Arithmetic Operators
2. String Concatenation Operator (&)
3. Comparison Operators
4. Logical Operators

Arithmetic operators within a single expression are evaluated in the following order of precedence:

1. Exponentiation (^)
2. Multiplication and Division (*,/): These two operators are of equal precedence and are evaluated in the left-to-right order in which they appear within the expression.
3. Integer Division (\)
4. Modulus Arithmetic (Mod)
5. Addition and Subtraction (+,-): These two operators are of equal precedence and are evaluated in the order in which they appear within the expression.

If the same arithmetic operator appears multiple times within a single expression, they are evaluated in the left-to-right order in which they appear.

Comparison operators all have equal precedence and are evaluated in the left-to-right order in which they appear within the expression.

Logical operators all have equal precedence and are evaluated in the left-to-right order in which they appear within the expression.

Arithmetic Operators

Following is a list of VBScript's arithmetic operators:

Symbol	Description
^	Exponentiation
-	Unary negation
*	Multiplication
/	Division
\	Integer division
Mod	Modulus arithmetic
+	Addition
-	Subtraction

> **Note:**
>
> For the order of precedence for these operators, see Operator Precedence.

String Concatenation Operator

The ampersand symbol (**&**) is used to perform string concatenation. This operator appends the second string to the first string.

Example:

```
"Hello," & " isn't it a lovely day?"
```

produces the following resultant string:

```
"Hello, isn't it a lovely day?"
```

Comparison Operators

Following is a list of VBScript's comparison operators:

Symbol	Description
=	Equality
<>	Inequality
<	Less than
>	Greater than
<=	Less than or equal to
>=	Greater than or equal to
Is	Object equivalence

> **Note:**
>
> All comparison operators have the same precedence. When multiple comparisons exist in a single expression, evaluate them in the left-to-right order in which they appear.

Logical Operators

Following is a list of VBScript's logical operators:

Symbol	Description
Not	Logical negation
And	Logical conjunction
Or	Logical disjunction
Xor	Logical exclusion
Eqv	Logical equivalence
Imp	Logical implication

> **Note:**
>
> All logical operators have the same precedence. When multiple logical operators exist in a single expression, evaluate them in the left-to-right order in which they appear.

Controlling Program Execution

You can use conditional statements to control the flow of a script. There are two types of conditional statements in VBScript:

- If...Then...Else
- Select Case

Using If...Then...Else

Following is an example that demonstrates the If...Then...Else conditional statement:

```
If obj = "Box1" Then
  <statements to execute>
ElseIf obj = "Cylinder1" Then
  <statements to execute>
Else
  <statements to execute>
End If
```

Using Select Case

Following is an example that demonstrates the Select Case conditional statement:

```
Select Case primitive_name
  Case "Box1"
```

<statements to execute>

```
  Case "Cylinder1"
```

<statements to execute>

```
  Case Else
```

<statements to execute>

```
End Select
```

Looping Through Code

Looping allows you to run a group of statements repeatedly. There are two types of loops:

- For...Next: Uses a counter to run statements a specified number of times.
- Do...Loop: Loops while or until a condition is True.

When using conditional statements that test for zero voltage/current, it is important to note that a real voltage or current should not be trusted to be exactly zero, even when it should be. Typically, the voltage or current is often on the order of 'epsilon' (1e-16) or smaller; hence, it is nonzero in value.

Using a For...Next Loop

The For...Next type of loop allows you to run a group of statements repeatedly. It uses a counter to run statements a specified number of times. Following is an example that demonstrates the For...Next loop:

```
For variable = start To end
  <statements to execute>
Next
```

You can exit early from a For...Next loop with the Exit For statement.

Using a Do Loop

You can use **Do...Loop** statements to run a block of statements until (or while) a condition is true.

Repeating Statements While a Condition is True

Use the While keyword to check a condition in a Do...Loop statement. The syntax is as follows:

```
Do While condition
  <statements to execute>
Loop
```

Repeating a Statement Until a Condition Becomes True

Following is the syntax:

```
Do Until condition
  <statements to execute>
Loop
```

You can exit early from a loop by using the Exit For statement.

VBScript Procedures

In VBScript, there are two kinds of procedures, Sub and Function. These procedures are called by name, they can receive arguments, and each performs a specific task with a group of VBScript statements. If there is no argument, then the Sub or Function statement must include an empty set of parentheses.

Function Procedures

A Function returns a value by assigning a value to its name in one or more statements. Following is the syntax of a Function:

```
Function FunctionName([arguments])

  <Function statements>

End Function
```

Sub Procedures

A Sub procedure is like a function procedure, except that it does not return a value through its name. Following is the syntax of a Sub:

```
Sub ProcedureName([arguments])

  <Procedure statements>

  End Sub
```

Converting Between Data Types

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

CStr	Syntax: CStr(variablename). Converts variablename to a string. For example, it can be used to convert the number 2.5 to the string "2.5".
CBool	Syntax: CBool(variablename). Converts variablename to a boolean. If variablename is 0 or "0", CBool returns False. Otherwise it returns True.
CDbl	Syntax: CDbl(variablename). Converts variablename to a double precision number. For example, it can be used to convert the string "2.5" to the number 2.5.
CInt	Syntax: CInt(variablename). Converts variablename to an integer.

Including Scripts

You can include one script within another using the following command:

```
#include "<scriptfilename>"
```

Where scriptfilename is the full path name to a file that contains script text, or is the name of a script in the project library or script library (listed in the project window under the Definitions/Scripts directory).

The command works for VBScript, JScript, and for the following:

- Scripts in the project library that are run by right-clicking the script icon in the project window and choosing **Run Script**
- Scripts in files that are external are run by choosing **Tools> Run Script**
- Scripts that are specified as callbacks in the **Property** dialog box
- Scripts that are run to draw parameterized footprints in layout

An include command can be placed anywhere in a script, but for readability it is recommended that commands be placed at the beginning of a file. The same script can be included multiple times without error, and circular inclusions will be ignored.

Aborting Scripts

You can abort a script that is running in the desktop simply by pressing the ESC key. Terminating a script in this manner works for each of the following:

- Scripts in the project library that are run by right-clicking the script icon in the project window and choosing **Run Script**.
- Scripts in files that are external can be run by choosing **Tools** > **Run Script**.
- Scripts that are specified as callbacks in the **Property** dialog box.
- Scripts that are run to draw parameterized footprints in layout.

Interacting with a Script

VBScript provides two functions that enable you to interact with a script while it is running:

the InputBox function and the MsgBox function.

The InputBox function displays a dialog box with an input field. The value that is typed into the input field is returned. For example:

```
Dim users_string

users_string = InputBox ("text prompt", "title of the pop-up dialog _

  box", "default text for the input box")
```

The last two arguments to the function are optional.

The MsgBox function shows a message and returns a number based on the button the user presses. For example:

```
MsgBox ("message text")
```

Recommended VBScript References

Microsoft Corporation. *VBScript User's Guide.*

Available http://msdn.microsoft.com/library/default.asp?url=/library/en-us/script56/html/vbstutor.asp.

Childs, M., Lomax, P., and Petrusha, R. *VBScript in a Nutshell: A Desktop Quick Reference*.

May 2002. O'Reilly & Associates. ISBN: 1-56592-720-6.

Sample HFSS Script

Following is an example of an Ansys Electronics Desktop script. It includes comment lines, which are preceded by either an apostrophe (') or the word REM, that offer explanations for each preceding line or lines. VBScript keywords appear in bold font.

```
' ----------------------------------------------
' Script Recorded by Ansoft HFSS Version 10.0
' 11:03 AM May 3, 2005
' ----------------------------------------------
Dim oDesign
Dim oEditor
Dim oModule
```

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

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

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

oDesktop.NewProject
' In VBScript, a Sub procedure is a procedure that is called by name, can receive arguments,
' and can perform a specific task with a group of statements. Here the Sub procedure
```

```
' NewProject of the object oDesktop is called. This Sub does not take an input.

Set oProject = oDesktop.GetActiveProject
oProject.InsertDesign "Hfss", "HFSSDesign1", "DrivenModal", ""
' In a Sub or Function procedure call, you can group the input parameters inside
' parentheses or without parentheses. Here the four strings are the input parameters of
' the Sub procedure InsertDesign of the object oProject.

Set oDesign = oProject.SetActiveDesign("HFSSDesign1")
Set oEditor = oDesign.SetActiveEditor("3D Modeler")
oEditor.CreateBox Array("NAME:BoxParameters", "XPosition:=", _
  "0mm", "YPosition:=", "0mm", "ZPosition:=", "0mm", _
  "XSize:=", "1.6mm", "YSize:=", "1.2mm", "ZSize:=", _
  "0.8mm"), Array("NAME:Attributes", "Name:=", "Box1", "Flags:=", _
  "", "Color:=", "(132 132 193)", "Transparency:=", _
  0.400000005960464, "PartCoordinateSystem:=", _
  "Global", "MaterialName:=", "vacuum", "SolveInside:=", true)
' oEditor.CreateBox is a Sub procedure that takes two array variables as input. The
' first array is for the box's geometric parameters and the second array is for the box's
' attributes. You can modify the italicized entries to create a different box. In VBScript,
' Array is a function that returns a variant containing an array. The underscore
' character ( _ ) here indicates that the statement continues to the next line. The
' underscore character must be placed outside of string constants, or else VBScript will
' recognize the character as part of the string constant rather than an indication that the
' string continues on the next line. Following is an example of proper use of the underscore
' character:
' Msgbox("Please include units when creating variables " & _
' "that require dimensions."
' Following is an example of improper use of the underscore character:
' Msgbox("Please include units when creating variables _
```

```
' that require dimensions."
```

For additional Ansys Electronics Desktop script examples, see *Example Scripts*.

Sample Q3D Extractor Script

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

```
Dim oAnsoftApp

Dim oDesktop

Dim oProject

Dim oDesign

Dim oEditor

Dim oModule
```

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

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

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

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

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

```
oDesktop.NewProject
```

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

```
Set oProject = oDesktop.GetActiveProject

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

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

```
Set oDesign = oProject.SetActiveDesign("Q3DDesign1")
Set oEditor = oDesign.SetActiveEditor("3D Modeler")
oEditor.CreateBox Array("NAME:BoxParameters", "XPosition:=", _
  "0mm", "YPosition:=", "0mm", "ZPosition:=", "0mm", _
  "XSize:=", "1.6mm", "YSize:=", "1.2mm", "ZSize:=", _
  "0.8mm"), Array("NAME:Attributes", "Name:=", "Box1", "Flags:=", _
  "", "Color:=", "(132 132 193)", "Transparency:=", _
  0.400000005960464, "PartCoordinateSystem:=", _
  "Global", "MaterialName:=", "vacuum", "SolveInside:=", true)
```

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

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

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

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

For additional Ansys Electronics Desktop script examples, see *Example Scripts*.

Introduction to IronPython

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

Scope

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

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

Python compatibility

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

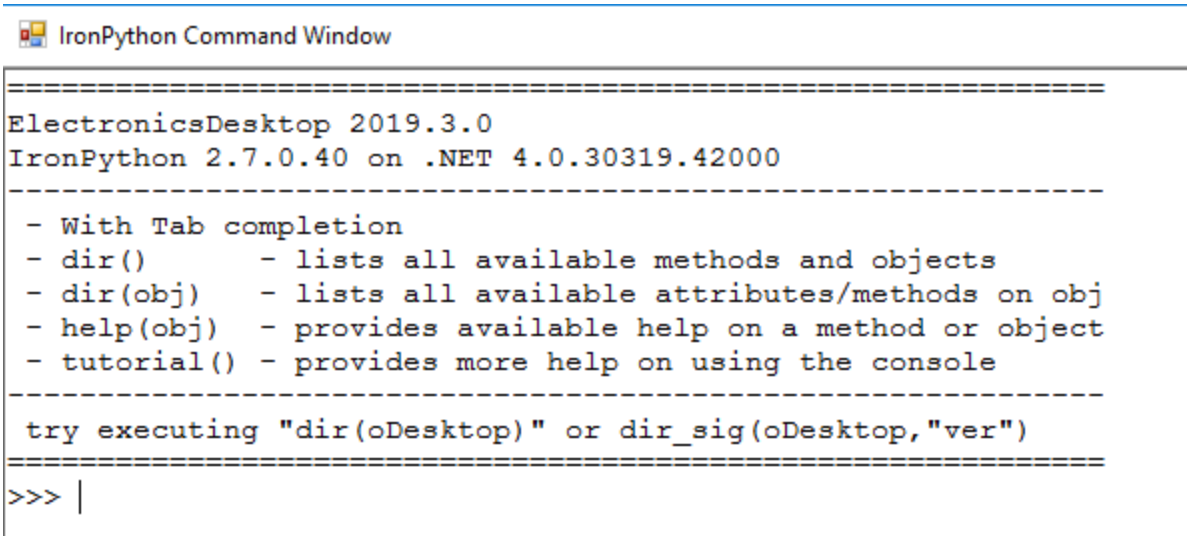
(http://code.google.com/p/ironclad/).

Advantages of IronPython

The advantages that IronPython use provides are significant:

- Python has a large eco-system with plenty of supporting libraries, Visual IDEs and debuggers. It is actively developed and enhanced.
- IronPython, in addition, has access to the entire .NET eco system. This allows us, for instance, to create a modern GUI using the **System.Windows.Forms** assembly from IronPython code and call any other .NET assembly for that matter.
- The use of IronPython's technologies enables the ability to interactively script Desktop (feature in development). This allows better discovery of the scripting APIs as well as directly programming to the scripting API in python, a language more tractable and platform independent compared with VBScript.
- The Python syntax of dictionaries is somewhat easier to read and write when supplying arguments to the scripting methods.

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



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

Introduction to IronPython

IronPython Mini Cookbook

Translating Script Commands from VBScript to IronPython

Scripting Using Iron Python

Standalone IronPython and Desktop IronPython

IronPython Examples

Creating User Defined Primitives and User Defined Models in Python Scripts

IronPython Mini Cookbook

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

Comments

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

Assigning/Creating Variables

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

Create Lists/Arrays

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

Create Dictionaries/Maps

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

Boolean Values

VBScript	IronPython
Boolean literals are in lower case: true false	The first letter is capitalized: True False

Converting Numbers to Strings and Vice Versa

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

String Formatting/Concatenation

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

Looping over Lists

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

Looping over a Range

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

Related Topics:

Indentation in IronPython

Methods in IronPython

Introduction to IronPython

Translating Script commands from VBScript to IronPython

Scripting Using Iron Python

IronPython Samples

Indentation in IronPython

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

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

> **Note:**
>
> Python recommends the use of spaces over tabs.

Indenting Functions

Define a function that starts at 0 indentation:

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

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

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

Indenting If Conditions

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

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

Methods in IronPython

Finding Methods

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

```
import string
```

Then get the directory listing:

```
dir(string)
```

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

Help

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

Related Topics

Introduction to IronPython

Translating Script commands from VBScript to IronPython

Scripting Using Iron Python: Putting it all Together

IronPython Samples

Translating Script Commands from VBScript to IronPython

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

Related Topics

Script Method Argument

VBscript Method Call Types

Converting VBScript Function calls to IronPython Syntax

Introduction to IronPython

IronPython Mini Cookbook

Scripting Using Iron Python

IronPython Samples

Script Method Argument

Script method calls in VBscript generally take the form:

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

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

- Primitive Types
- Named Arrays
- Named Functions

Primitive Types

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

Named Arrays

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

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

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

Named Functions

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

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

Related Topics

Translating Script commands from VBScript to IronPython

VBScript Method Call Types

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

VBScript Functions

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

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

```
Set oProject = oDesktop.NewProject
```

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

VBScript Sub-Routines

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

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

Related Topics

Translating Script commands from VBScript to IronPython

Converting VBScript Function calls to IronPython Syntax

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

So:

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

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

Return Values

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

Primitive Method Arguments

Replace each VBScript primitive with an equivalent IronPython primitive.

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

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

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

Named Array Arguments

Formatting helps readability immensely but is not required.

All that *must* be done is:

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

Named Array Values with All Key Value Pairs

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

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

Named Arrays with Nested Named Arrays

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

["NAME:name",

"key1:=" , 1,

"key2:=" , 2,

["NAME:name2", "R:=", 255]

]

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

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

{ "NAME" : "name",

"key1" : 1,

"key2" : 2,

"name2" : ["NAME:name2", "R:=", 255]

}

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

{ "NAME" : "name",

"key1" : 1,

"key2" : 2,

"name2" : {"R" : 255}

}

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

{ "NAME" : "name",

"key1" : 1,

"key2" : 2,

"name2" : {

"NAME" : "name2",

"R" : 255

}

}

Function Blocks

Function blocks in VBScript argument syntax are represented as arrays without the "NAME:.." field. However, functions are always introduced by a key in a parent structure. Function blocks

can therefore never exist as a top-level argument. They are only found as the value pairs inside a named array or inside another function block.

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

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

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

Scripting Using Iron Python

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

Translating a script in VBScript to IronPython

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

Writing an IronPython script from scratch

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

For python syntax and the differences, the mini-cookbook chapter will also be useful.

IronPython Script Execution Environment

Scripting using Embedded VBScript or JavaScript

Scripting with IronPython

Standalone IronPython and Desktop IronPython

Related Topics

Introduction to IronPython

IronPython Mini-cookbook

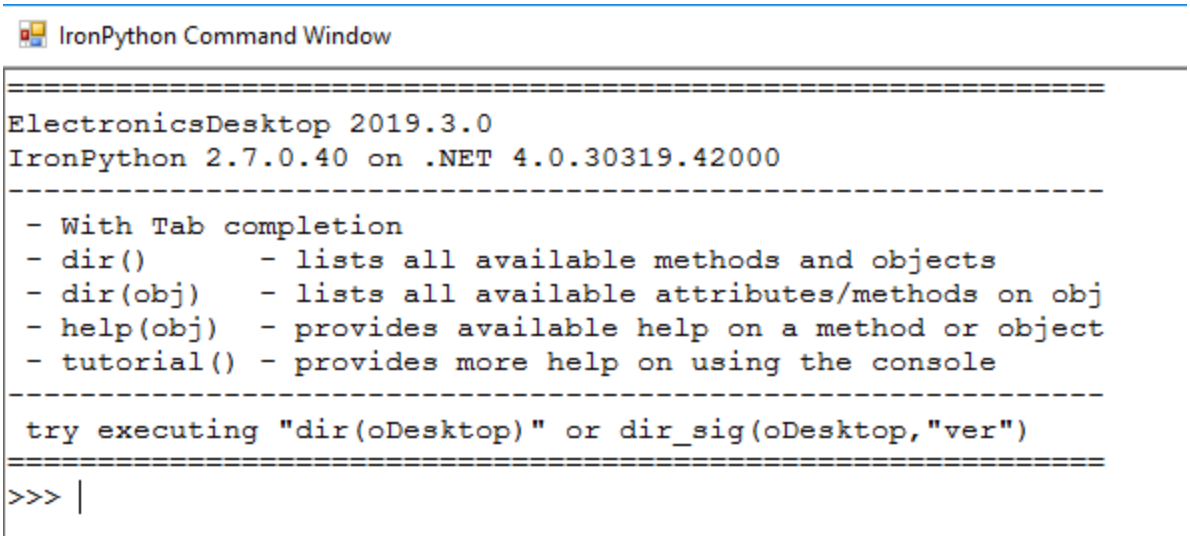
Translating Script commands from VBScript to IronPython

Appendix: IronPython Samples

IronPython Script Execution Environment

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

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



- **Tools** > **Run Script** menu item, select "IronPython" from the file type drop-down list.
- Launch the product with a script argument.
    - HFSS -runscript someScript.py to keep HFSS GUI open after completing script execution.
    - HFSS -features=beta -ng -runscriptandexit someScript.py to run HFSS in a non-graphical mode and exit after script completion. Note that this is a beta feature supported for the HFSS and 3D Layout design types.
- Register an IronPython script as an external tool using the **Tools** > **External Tools** menu item.

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

Script Argument for IronPython

Script Argument for IronPython

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

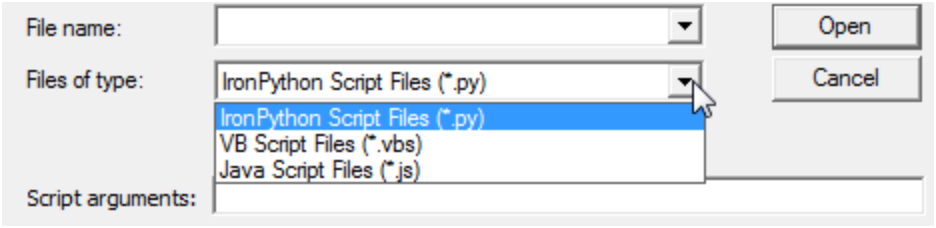


Figure 1: Run Script dialog and script arguments

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

Related Topics

IronPython Script Execution Environment

Scripting using Embedded VBScript or JavaScript

Since script recording is still done in VBScript and users are expected to have a significant collection of VBScript or JavaScript assets, it is useful to continue to use existing script files and snippets even when scripting in IronPython. The various **Run<*>Command** methods have been designed for this purpose.

For instance: one can create a parameterized cone in HFSS by executing the following IronPython script from the **Tools** > **Run Script** menu.

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

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

coneScript = """Dim oAnsoftApp

Dim oDesktop

Dim oProject

Dim oDesign

Dim oEditor

Dim oModule

Set oAnsoftApp = CreateObject("Ansoft.ElectronicsDesktop")

Set oDesktop = oAnsoftApp.GetAppDesktop()

oDesktop.RestoreWindow

Set oProject = oDesktop.GetActiveProject()

oProject.InsertDesign "HFSS", "HFSSPyTestDesign", "DrivenModal", ""
```

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

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

oEditor.CreateCone Array("NAME:ConeParameters", _

 "XCenter:=", "0mm", "YCenter:=", "0mm", "ZCenter:=", "0mm",_

 "WhichAxis:=", "Z", "Height:=", "2mm", _

 "BottomRadius:=", "3mm", _

 "TopRadius:=", "0mm"), Array("NAME:Attributes", "Name:=", _

 "Cone1", "Flags:=", "", "Color:=", "(132 132 193)", "Trans-
parency:=", 0, _

 "PartCoordinateSystem:=", "Global", "UDMId:=", "", "Mater-
ialValue:=", _

 "" & Chr(34) & "vacuum" & Chr(34) & "", "SolveInside:=", _

 true)

"""

SetScriptingLanguageToVBScript()

RunScriptCommand(coneScript)
```

Sample Script 11: Hybrid VBScript + IronPython scripting: parameterized Cone Creation

Even though recorded VBScript is used for scripting, the incremental functionality that is provided using IronPython is the ability to write a GUI using IronPython/.NET, collect information from the user and then modify or generate the VBScript commands to actually script the Ansys EM desktop. This GUI functionality is cross platform and a significant positive. The following example demonstrates a contrived use of a .NET window form to display the argument supplied to the IronPython script (via the ***ScriptArgument*** variable).

```
#import the CLR references

import clr

clr.AddReference("System.Windows.Forms")


from System.Windows.Forms import Application, Form, Label, Button,
DockStyle


# the GUI form to show some text

# the class below derives from From (System.Windows.Forms.Form)
```

```
# imported above from the .NET assembly.
class ShowPropertiesForm(Form):
  def __init__(self, name, text):
    self.Name = name
    self._label = Label()
    self._label.Text = text
    self._label.Dock = DockStyle.Fill

    _button = Button()
    _button.Text = "Close"
    _button.Dock = DockStyle.Bottom
    _button.Click += self._buttonPressed

    self.Controls.Add(self._label)
    self.Controls.Add(_button)

  def _buttonPressed(self,sender,args):
    self.Close()

#-----------------------------------------------------
# Main script code
#-----------------------------------------------------
#display the ScriptArgument variable as the text label
# in the form.
gui = ShowPropertiesForm("Sample Form", ScriptArgument)

# This makes it a modal dialog.
gui.ShowDialog()

# the following will make it a non-modal dialog
```

```
#Application.Run(gui)
```

Sample Script 12: Demonstrates the use of a .NET form from IronPython

While creating cross platform user interfaces from scripts is one of the main motivations driving the adoption of IronPython, any .NET assembly can be used with the caveat that Linux use requires Mono compatibility of any used assemblies.

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

Related Topics

IronPython Script Execution Environment

Scripting with IronPython

While this section talks about directly interacting with the script objects, note that you can execute VBScript or Javascript at any point using any of the available Run*Command functions. using your existing script assets in this fashion and mixing with IronPython code for new functionality as needed is a viable and option.

Access to the application scripting objects is provided via the predefined **oDesktop** object (as listed in Script Objects). Interacting with the script objects is very natural, method calls are made just like in VBScript except that the argument syntax is somewhat simplified to follow natural Python syntax. All primitive types (string, integer, double) map to the natural primitive types in python. The only differences from the VBScript syntax are seen when specifying array type arguments. The differences are described in earlier chapters.

> **Note:**
>
> The typical VBScript calls to obtain the registered COM scripting interface via CreateObject calls and then obtain the oDesktop object from it using the GetAppDesktop() is not needed (or even supported on all platforms). Since all scripting occurs in the context of a running workbench, the available Desktop object is always provided and expected to be used directly.

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

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

- Any compound/block arguments should be translated to the appropriate IronPython array or dictionary syntax.

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

Related Topics

IronPython Script Execution Environment

Standalone IronPython and Desktop IronPython

Standalone IronPython

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

Running Standalone IronPython

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

It is located in:

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

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

```
"C:\Program Files\AnsysEM\v242\Win64\common\IronPython\ipy64.exe"
"<filePath>\myScript.py"
```

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

Using a Recorded Script

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

Creating an External Script

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

- `import sys`

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

- `sys.path.append("<InstallationPath>")`

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

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

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

- `import ScriptEnv`

  # This imports ScriptEnv.py from the installation path specified above. ScriptEnv.py performs an operating system check and defines functions used in Electronics Desktop scripts. See the annotations in the ScriptEnv.py file for more information.

- `ScriptEnv.Initialize("Ansoft.ElectronicsDesktop")`

  or `ScriptEnv.InitializeNew(NonGraphical=True)`

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

You must end the script with:

- `ScriptEnv.Shutdown()`

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

Example Script

```
import sys

sys.path.append(r"C:\Program Files\AnsysEM\v242\Win64")

sys.path.append(r"C:\Program Files\An-
sysEM\v242\Win64\PythonFiles\DesktopPlugin")


import ScriptEnv

ScriptEnv.Initialize("Ansoft.ElectronicsDesktop")

oDesktop.RestoreWindow()

oProject = oDesktop.NewProject()

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

oDesign = oProject.SetActiveDesign("HFSSDesign1")

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

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

IronPython Samples

Change property

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

```
oEditor.ChangeProperty Array("NAME:AllTabs", Array("NAME:Geo-
metry3DAttributeTab",_
  Array("NAME:PropServers", "Cone1"), _
  Array("NAME:ChangedProps", _
  Array("NAME:Color", "R:=", 255, "G:=", 255, "B:=", 0))))
```

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

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

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

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

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

```
   }]]
])
```

Sample Script 15: ChangeProperty command to change the color of a cone using Python arrays and dictionaries

Create a Cone using IronPython

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

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

```
  "MaterialValue" : "\"vacuum\"",
  "SolveInside" : True
  }
  )
```

Sample Script 16: IronPython script to create a cone

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

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

```
oProject = oDesktop.GetActiveProject()
oDesign = oProject.InsertDesign("HFSS","Hersheys 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()
```

Sample Script 17: Sample script to create a cone and then use copy/paste/move to replicate it.

Related Topics

Introduction to IronPython

IronPython Mini-cookbook

Translating Script commands from VBScript to IronPython

Scripting Using Iron Python: Putting it all Together

Creating User Defined Primitives and User Defined Models in Python Scripts

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

Advantages Compared to C++

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

Changes compared to C

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

Structures

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

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

Return Values for UDM and UDP Functions

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

Constants

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

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

# with the type of the enum: UnitType

unitType = UnitType.LengthUnit
```

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

Methods

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

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

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

Output Parameters

The rule for the output parameters is as follows:

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

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

udmDefinition = udmFunctionLibrary.GetDefinition()


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

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

# Two output parameter; return value must be preserved

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

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

Comparison with C function:

C	Python
bool getDefinition(UDMDefinition* udmDefinition, void* callbackData); where udmDefinition is an output parameter	udmDefinition = udmFunctionLibrary.GetDefinition() (Note: callbackData is omitted in py interface)
long detachlFaces(int nFacesAndPartIds, long* faceIds, long* partIds, void* callbackData); where partIds is an output parameter	(bRet, partIds) = udmFunctionLibrary.DetachlFaces (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); where numObjects and objectIds are input parameters and numObjects is the size of objectIds.	lret = udpFunctionLibrary.Unite (objectIds) (ignore numObjects as redundant: folded into objectIds callbackData is omitted)
long detachFaces(long nSize, long* 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, UDPCoordinateSystemAxis axis, double angle, int nClones, long* nClones, void* callbackData);	(lret, cloneIds) = udmFunctionLibrary.DuplicateAroundAxis (partId, axis, angle, nClones, cloneIdsSize) (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 below.

IUDPExtension methods

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

Mandatory methods.

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

GetLengthParameterUnits()

- returns string.

GetPrimitiveTypeInfo()

- returns UDPPrimitiveTypeInfo.

GetPrimitiveParametersDefinition2()

- returns a list of UDPPrimitiveParameterDefinition2 or None on failure

AreParameterValuesValid2(errorMsg, udpParams)

- errorMsg is a c# list of strings
- udpParams is a c# list of 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 a long
- 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 a long
- This should release any resources assigned to this particular instance of UDM.
- returns True on success, False on failure.

GetAttribNameForEntityId()

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

GetAttribNameForPartId()

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

Optional methods

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

DialogForDefinitionOptionsAndParams(self, defData, optData, params):

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

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

DialogForDefinitionAndOptions(self, defData, optData) [Deprecated]

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

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

GetInstanceSourceInfo(instanceId)

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

ShouldAttachDefinitionFilesToProject()

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

Example UDM

```
class UDMExtension(IUDMExtension):


  def IsAttachedToExternalEditor(self):
    return False


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


    return udmInfo
  ...
  ...
```

UDMFunctionLibrary

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

- CreateInstance
- Refresh

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

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

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

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

```
(bret, udpPositionLow, udpPositionHigh) = udmFunc-
tionLibrary.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***: center-Point, ***UDPPosition***: startPosition, ***int***: numOfSides, ***double***: fHeight**)**
22. ***int***: ID = **CreateHelix(*UDPHelixDefinition***: helixDefinition**)**
23. ***bool***: SI = **Unite(*List_of_int***: pObjectIDArray**)**
24. ***bool***: SI = **Subtract(*List_of_int***: pBlankObjectIDArray, ***List_of_int***: pToolObjectIDArray**)**
25. ***bool***: SI = **Intersect(*List_of_int***: pObjectIDArray**)**
26. ***bool***: SI = **Imprint(*List_of_int***: pBlankObjectIDArray, ***List_of_int***: pToolObjectIDArray**)**
27. ***bool***: SI = **SweepAlongVector(*int***: profileID, ***UDPVector***: sweepVector, UDPSweep-Options: sweepOptions**)**
28. ***bool***: SI = **SweepAroundAxis(*int***: profileID, ***CoordinateSystemAxis***: whichAxis, ***double***: sweepAngle, ***UDPSweepOptions***: sweepOptions**)**
29. ***bool***: SI = **SweepAlongPath(*int***: profileID, ***int***: pathID, ***UDPSweepOptions***: sweep-Options**)**
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***: mir-rorPlaneNormalVector**)**
33. ***bool***: SI = **Transform(*int***: partID, ***List_of_double***: rotationMatrix, ***UDPVector***: trans-lateVector**)**
34. ***bool***: SI = **Scale(*int***: partID, ***double***: xScale, ***double***v: yScale, ***double***: zScale**)**
35. (***bool***: SI, ***List_of_int***: cloneIDs**)** = **DuplicateAlongLine(*int***: partID, ***UDPVector***: trans-lateVector, ***int***: numTotalObjs, ***int***: cloneIDsListSize**)**
36. (***bool***: SI, ***List_of_int***: cloneIDs) = **DuplicateAroundAxis(*int***: partID, ***Coordin-ateSystemAxis***: whichAxis, ***double***: rotateAngle, ***int***: numTotalObjs, ***int***: cloneIDsListS-ize**)**
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 Name string Purpose

Structure	Construction	Members
	string purpose, string company, string date, string version)	string Company string Date string Version
UDPPrim-itiveParameterDefinition	UDPPrim-itiveParameterDefinition(string name, string description, UnitType unitType, double defaultValue)	string Name string Description UnitType UnitType double DefaultValue
UDPParam	UDPParam() UDPParam(ParamDataType dataType, object data) object can be int, double, string, bool or UDPPosition **methods:** setInt(int val) setBool(bool val) setString(string val) setDouble(double val) setPosition(UDPPosition val)	ParamDataType DataType object Data object can be int, double , string, bool or UDPPosition
UDPPrim-itiveParameterDefinition2	UDPPrim-itiveParameterDefinition2(string name, string description, UnitType unitType,	string Name string Description UnitType UnitType ParamPropType PropType ParamPropFlag PropFlag

Structure	Construction	Members
	ParamPropType propType, ParamPropFlag propFlag, UDPParam defaultValue)	UDPParam DefaultValue
UDPPosition	UDPPosition(double x, double y, double z)	double X double Y double Z
UDPVector	UDPVector(double x, double y, double z)	double X double Y double Z
UDPSweepOptions	UDPSweepOptions(SweepDraftType draftType, double draftAngle, double twistAngle)	SweepDraftType DraftType double DraftAngle double TwistAngle
UDPPolylineSegmentDefinition	UDPPolylineSegmentDefinition(PolylineSegmentType segmentType, int segmentStartIndex, int numberOfPoints, double angle, UDPPosition centerPoint, CoordinateSystemPlane arcPlane)	PolylineSegmentType SegmentType int segmentStartIndex, int numberOfPoints, double angle, UDPPosition centerPoint, CoordinateSystemPlane arcPlane)
UDPPolylineDefinition	UDPPolylineDefinition() UDPPolylineDefinition(List_of_UDPPosition positions, List_of_UDPPolylineSegmentDefinition segDefs, int closed,	int IsClosed int IsCovered List_of_UDPPosition ArrayOfPosition List_of_UDPPolylineSegmentDefinition ArrayOfSegmentDefinition

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

Structure	Construction	Members
	double pitch)	
UDPSpiralDefinition	UDPSpiralDefinition(int profileID, UDPPosition ptOnAxis, UDPPosition axisDir, double noOfTurns, bool isRightHanded, double radi-usChangePerTurn)	int ProfileID UDPPosition PtOnAxis UDPPosition AxisDir double NoOfTurns bool IsRightHanded double RadiusChangePerTurn
UDPBLNDElements	UDPBLNDElements(int partID, int noOfEdges; int* listOfEdges;) UDPBLNDElements(int partID, int noOfVertices; int* listOfVertices;)	UDPBLNDElements can hold either edges or vertices, but not both at the same time. Edges should be applied to solids, and vertices should be applied to sheets. int PartID /* part to be blended i.e. filleted/chamfered */ int noOfEdges; int* listOfEdges; /* edges to be blended */ int noOfVertices; int* listOfVertices; /* vertices to be blended */
UDPBLNDFilletOptions	UDPBLNDFilletOptions(bool supressFillet, BLNDFilletRadiusLaw fil-letRadiusLaw, double filletStartRadius, double filletEndRadius, bool fol-lowSmoothEdgeSequence, BLNDFilletType filletType, double setbackDistance,	bool SupressFillet /* Reserved for future */ BLNDFilletRadiusLaw Fil-letRadiusLaw double FilletStartRadius double FilletEndRadius bool Fol-lowSmoothEdgeSequence /* Reserved for future */ BLNDFilletType FilletType double SetbackDistance

Structure	Construction	Members
	double bulgeFactor)	double BulgeFactor /* Reserved for future */
UDPBLNDChamferOptions	UDPBLNDChamferOptions(bool supressChamfer, BLNDChamferRangeLaw chamferRangeLaw, double chamferLeftRange, double chamferRightRange)	bool SupressChamfer BLNDChamferRangeLaw Cham-ferRangeLaw double ChamferLeftRange double ChamferRightRange
UDPProgress	UDPProgress(int prog, int subProg, string mesg, string subMesg)	int Prog int SubProg string Mesg string SubMesg
UDMInfo	UDMInfo(string name, string purpose, string company, string date, string version)	string Name string Purpose string Company string Date string Version
UDMDefinition	UDMDefinition() UDMDefinition(string name, UDParam value, ParamPropType propType, ParamPropFlag propFlag)	string DefName UDPParam DefValue ParamPropType PropType ParamPropFlag PropFlag
UDMOption	UDMOption() UDMOption(string name, UDParam value,	string OptName UDPParam OptValue ParamPropType PropType ParamPropFlag PropFlag

Structure	Construction	Members
	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

enum Constant	Parameters
	Value FileName Checkbox Position Unknown
ParamPropFlag	NoFlag ReadOnly MustBeInt MustBeReal Hidden Unknown
CoordinateSystemAxis	XAxis YAxis ZAxis
CoordinateSystemPlane	XYPlane YZPlane ZXPlane
SweepDraftType	ExtendedDraft RoundDraft NaturalDraft MixedDraft
SplitWhichSideToKeep	SplitKeepBoth SplitKeepPositiveOnly SplitKeepNegativeOnly
PolylineSegmentType	LineSegment ArcSegment SplineSegment AngularArcSegment
MessageSeverity	WarningMessage ErrorMessage

enum Constant	Parameters
	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 suc-
      cessful validation
```

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

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

:return: None

"""

param_dict = self.get_param_dict(param_values)


start_point = UDPPosition(0, 0, 0)

box = func_lib.CreateBox(start_point, [

  param_dict["x_size"],

  param_dict["y_size"],

  param_dict["z_size"]

])


# points on the middle of 4 vertical edges

points = [

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

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

  [param_dict["x_size"], param_dict["y_size"], param_dict["z_
  size"]/2],s

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

]


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


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

fillet_opt = UDPBLNDFilletOptions(True, BLNDFil-
letRadiusLaw.BLNDConstantRadius, fillet_rad, 0.0, True, BLNDFil-
letType.BLNDRound, 0.0, 0.0)
```

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

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

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

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

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

def GetPrimitiveTypeInfo(self):
    return primitive_info

def GetLengthParameterUnits(self):
    return length_units

def GetPrimitiveParametersDefinition2(self):
    return primitive_param_definitions

def AreParameterValuesValid2(self, error, udp_params):
```

```
    return True


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


def get_param_dict(self, param_values):
    """
    Function to return a dictionary of UDP parameter name and value
    (key: value) pairs
    :param param_values: list of udp parameter values (user input)
    generated by UDP Core
    :return: dict of parameter name and values
    """
    udm_param_def = self.GetPrimitiveParametersDefinition2()
    param_dict = {}
    for i, param in enumerate(udm_param_def):
      param_value = param_values[i].Data
      if str(param.PropType) != "Menu":
        param_dict[param.Name] = param_value
```

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

Introduction to CPython

An Ansys Electronics Desktop 2024R1 feature allows CPython to be used for standalone scripting, as an alternative to IronPython, to:

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

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

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

> **Important:**
>
> (Supported on both Windows and Linux in 24R1)
>
> Launching Ansys Electronics Desktop on a remote machine is not supported in this release. Initialize() will connect to a remote instance that is already running on the remote machine, but cannot launch a new instance on the remote machine.
>
> Connect() to an opened project in remote machine needs to use a share network path.

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 dir-
  ectories 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

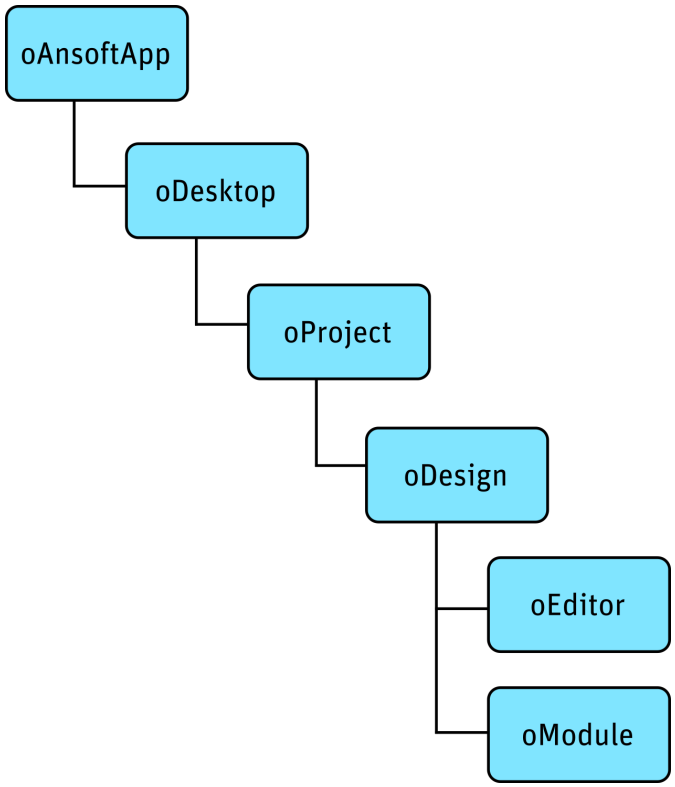
This page intentionally
left blank.

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 or VBscript to access the `Ansoft.ElectronicsDesktop` product.

In Iron Python, for example:

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

In VBscript, for example:

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

oDesktop

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

In Iron Python, for example:

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

In VBscript, for example:

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

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()
```

In VBscript, for example:

```
Set 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()
```

In VBscript, for example:

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

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

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

oEditor

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

In Iron Python, for example:

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

In VBscript, for example:

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

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

- 3D Modeler Editor Script Commands
- Schematic Editor Script Commands

> **Important:**
>
> There is no Reporter Editor object for `oEditor`. Reporter Editor commands are executed by `oDesign`.
>
> See: Reporter Editor Script Commands.

oModule

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

In IronPython, for example:

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

In VBscript, for example:

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

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

- Analysis Module Script Commands
- Mesh Operations Module Script Commands
- Optimetrics Module Script Commands

- Reduce Matrix Module Script Commands
- Solutions Module Script Commands

Example Script Opening

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

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

In VBscript, this would be:

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

GetActiveProject and GetActiveDesign for Wider Use

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

In IronPython:

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

In VBscript:

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

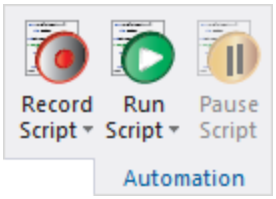
Running a Script

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

Within Electronics Desktop

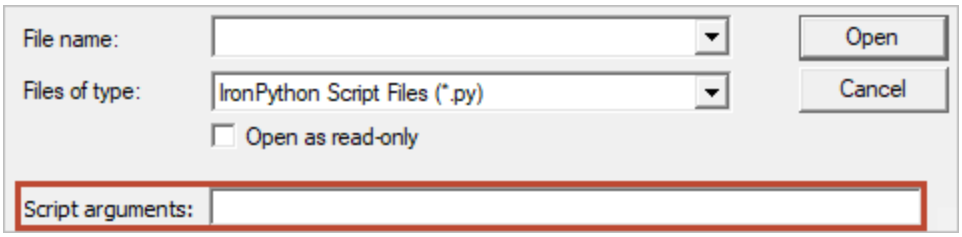
To run scripts in Electronics Desktop:

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



   The **Run Script** file browser appears.

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



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

4. Click **Open**.

   Electronics Desktop executes the script.

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

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

From the Command Line

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

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

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

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

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

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

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

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

Direct Launch

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

```
on error resume next

dim args

Set args = AnsoftScript.arguments

if(IsEmpty(args)) then

Set args = WSH.arguments

End if

on error goto 0

  // At this point, args has the arguments regardless of collection

msgbox "Count is " & args.Count

for i = 0 to args.Count - 1

msgbox args(i)

next
```

Recording a Script

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

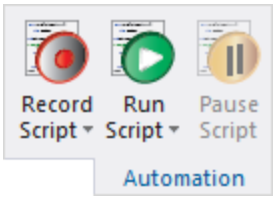
Scripts can be saved to an external file, or to the project.

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

Recording a Script to File

To record a script to file:

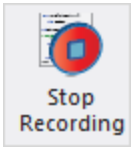
1. Click **Tools** > **Record Script to File**, or select the **Automation** tab and click the **Record Script** icon:



   A **Save As** file browser appears.

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

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



6. Perform the steps you want to record.

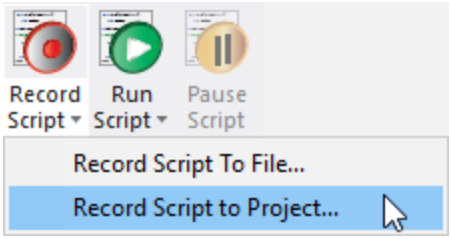
7. When you have finished recording the script, click **Stop Recording**, or select **Tools** > **Stop Script Recording**.

   The recorded script is saved to the folder you specified.

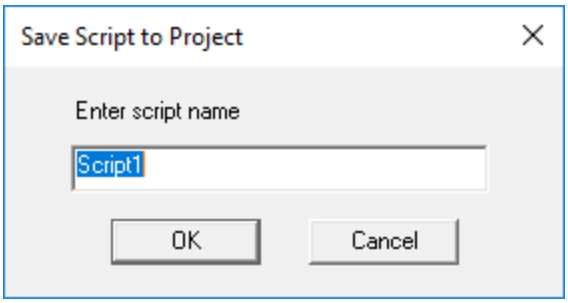
Recording a Script to a Project

To record a script to a project:

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



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



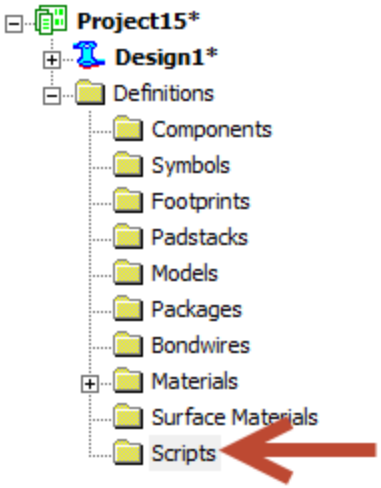
2. Enter a name for the script in the text box, then click **OK**.
3. Perform the steps you want to record.
4. When you have finished, click **Stop Recording**, or select **Tools** > **Stop Script Recording**.

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

Working with Project Scripts

Scripts can be recorded to a project.

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



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

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

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

Executing a Script from Within a Script

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

In IronPython:

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

In VBscript:

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

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

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

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

Electronics Desktop Scripting Conventions

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

Consult the following topics:

- Named Arguments
- Setting Numerical Values
- Layout Scripts and the Active Layer
- Scripts and Locked Layers
- Event Callback Scripting

Named Arguments

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

1. Named data, where name precedes data.

   For example:

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

2. Named Array, where name precedes array.

   For example:

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

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

   For example:

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

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

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

VBscript Example

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

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

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

```
oModule.EditImpedance "Imped1", Array("NAME:Imped1",
  "Resistance:=", "100")
```

IronPython Example

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

```
oModule.Edit("Port1",
  ["NAME:Port1",
    ["NAME:Properties",
    "PortSolver:=", "true",
    "Phase:=", "0deg",
    "Magnitude:=", "2mA",
    "Impedance:=", "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.

  In IronPython:

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

  In VBscript:

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

- Pass in a string containing the number with units.

  In IronPython:

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

  In VBscript:

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

- Pass in a variable name.

  In IronPython:

  ```
  var = 3.5

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

  In VBscript:

  ```
  dim var

  var = "3.5V"

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

Layout Scripts and the Active Layer

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

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

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

Scripts and Locked Layers

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

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

Event Callback Scripting

Event Callback scripting allows you to define custom JavaScript and VBScript routines that will run automatically after detecting a triggering event, such as placing a component or running a simulation. When defining an Event Callback script, specify one or more scripts that will be run after a particular event is detected.

A callback script can only access functions and other scripts defined by its callback definition. For example, a Simplorer callback script can call PropHost.GetValue — and all other PropHost functions — but only from scripts defined in the Property Dialog callback. As a result, "PropHost" is a Simplorer script item that is only visible in "Property" callback scripts. For more information, see Callback Scripting Using PropHost Object and Callback Scripting Using CompInstance Object.

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

Callback Event	Scripts Visible from the Event Callback Script	Functions Callable from the Visible Script
Place Component	CompInstance	**CompInstance.GetParentDesign()** — Returns a oDesign item that can be used to call Design functions. **CompInstance.GetPropserverName()** – Returns a CompInstance identification name that can be used in oEditor property-method scripts, such as GetPropertyValue(), SetPropertyValue(), etc. **CompInstance.GetComponentName()** — Returns the component name, e.g. "MS_TRL".
Simulate Component	CompInstance	**CompInstance.GetDesign()** — Returns the interface to the specified design simulation. **CompInstance.GetProgress()** – Returns the completion percentage (from 0 to 100) of the specified design simulation. **CompInstance.GetRunStatus()** — Returns the status number of the specified design simulation. **CompInstance.Abort()** — Aborts the specified design simulation.

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

This page intentionally
left blank.

2 - Object-Oriented Property Scripting

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

This topic covers the following:

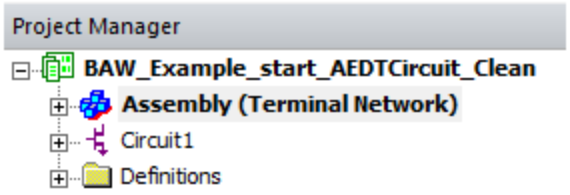
- Logic/syntax
- Basic attributes of project and design properties
- Example Scripts

Object-Oriented Scripting

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

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

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



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

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

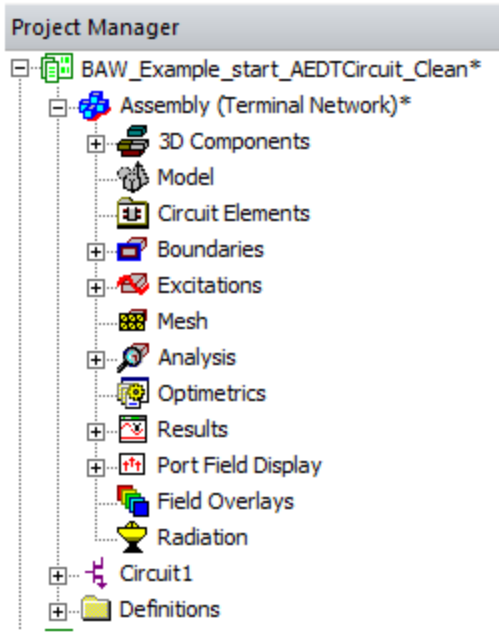
Once the objects have been defined, you can use GetChildNames() to learn what object instances exist for each. As an example, observe the Child Names of oProject, and you see a list of the Designs in the AEDT Project, per the GUI.

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

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

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

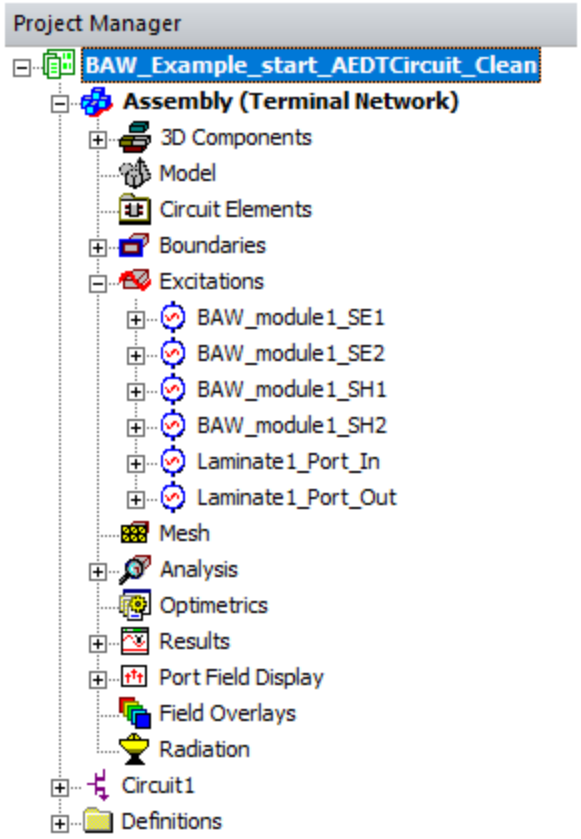
These names are what you would expect based on the Project Manager Layout:



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

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

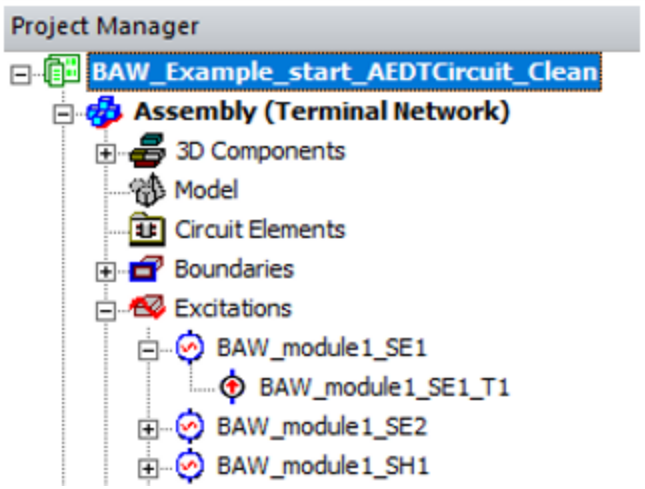
You now have an object, oExcitations defined to be the oDesign Child Object ‘Excitations’. What does this mean? If you expand the Excitations dialogue in the Project Manager, you expect that the Child Names of this object would be the names of the Excitations as defined, in this case six ports:




```
>>> oExcitations.GetChildNames()
['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.

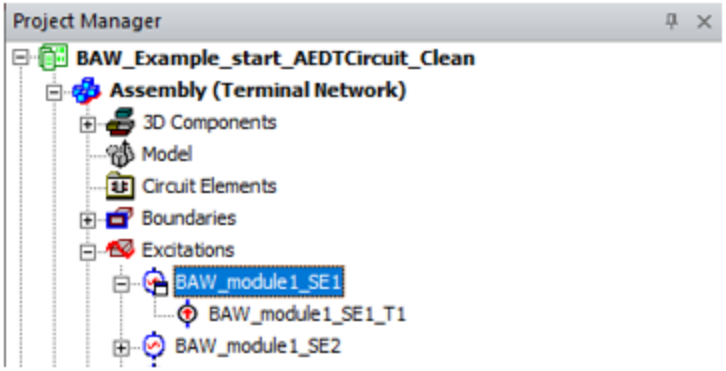
Expand the first port to see its expected Child Object, its Terminal, in the **Project Manager** Window:

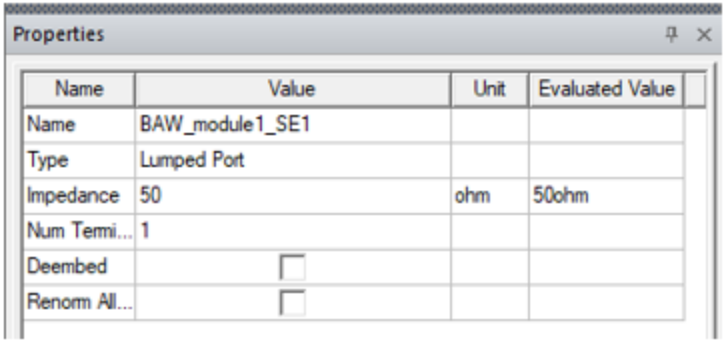


Through scripting, first define the Port Object (in this example, oPort) using the 'GetChildObject()' command for the first port, 'BAW_module1_SE1.' Then determine its Object Child Name, the terminal definition:

```
>>> 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:





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

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

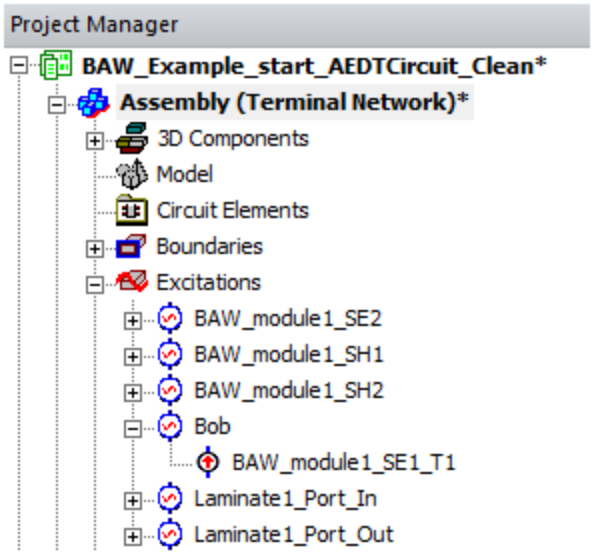
Once you identify the desired object and you know the desired property, you can access the value via GetPropValue(). For example, if you want to retrieve the name of the object oPort:

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

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

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

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



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

Material Properties and Examples

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

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

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

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

The Material Property of interest is "Bulk Conductivity." So you create a "Cond" object to store the value and use the GetPropValue function to obtain 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', 'Ori-
entation', 'Orientation/Choices', 'Model', 'Group', 'Display Wireframe', 'Material Appear-
ance', '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”, “PlanarEM”, “SIwave” in a 3D layout design.

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

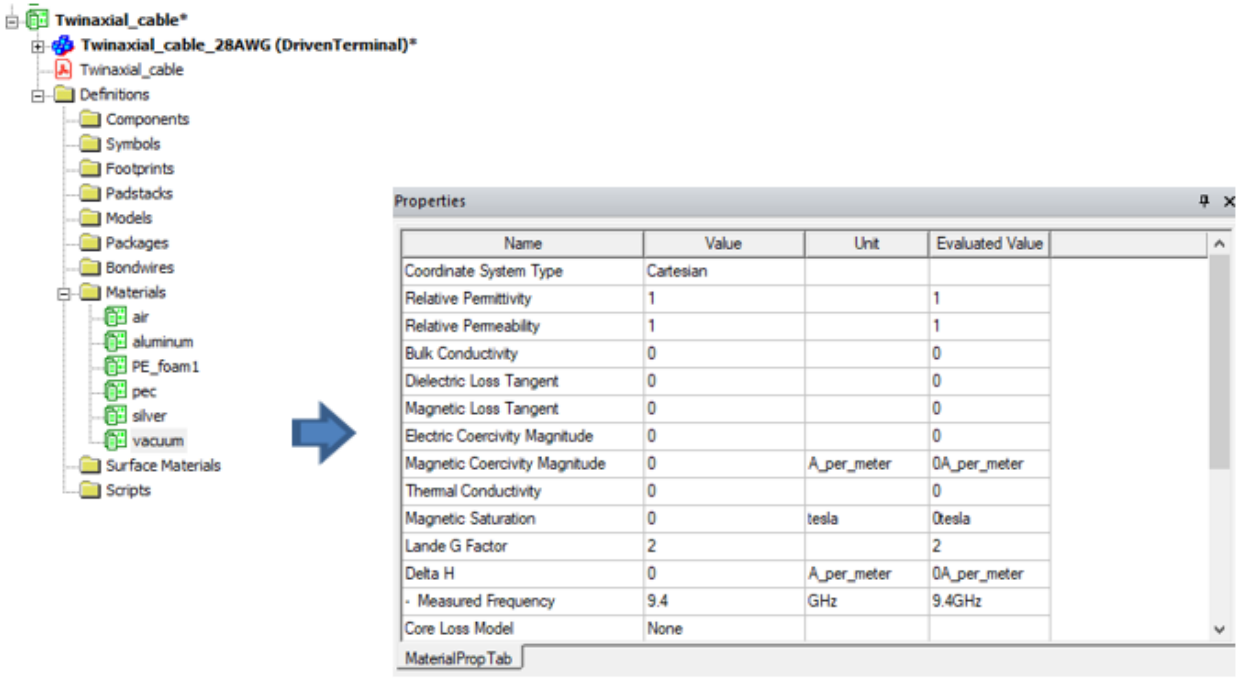
```
>>> setup.GetChildNames()
['Sweep', 'Sweep1', 'Sweep2']
>>> setup.GetChildTypes()
['Discrete', 'Interpolating']
>>> sweep1 = setup.GetChildObject("sweep")
```

Related Topics

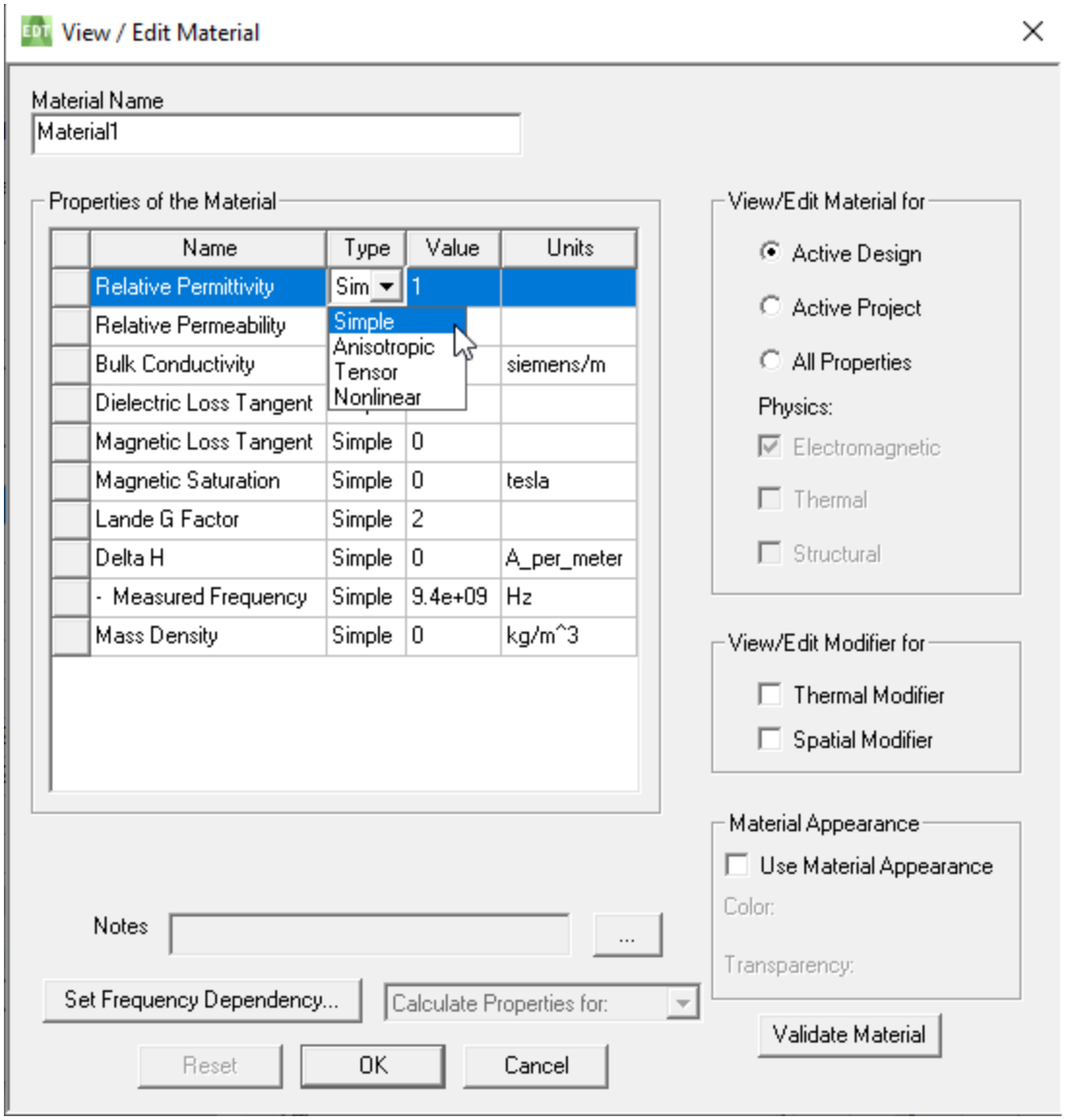
Example: GetPropNames and GetPropValues for Layered Impedance Boundary Script

Materials Scripting Support

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



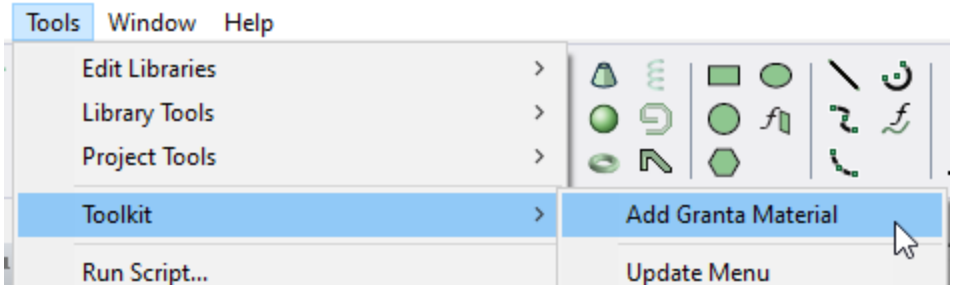
Some Material Properties like Relatively Permittivity may have values assigned as BH Curves or Tensors, as discussed in the Assigning Materials chapter of the online help.

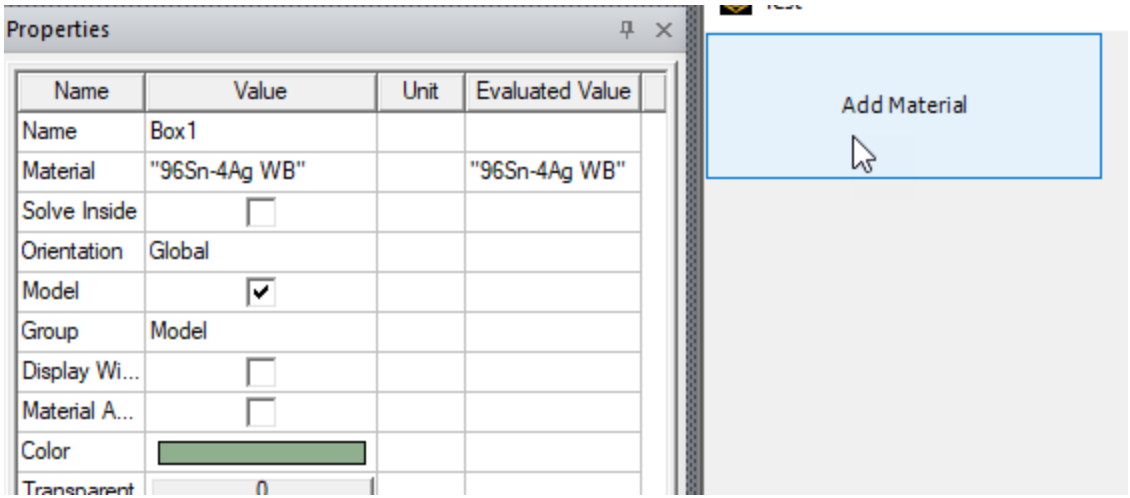


With this feature enabled you can then:

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

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





What is not supported:

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

Object Oriented Scripting for Materials

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

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

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

Examples showing change to material property type:

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

Examples showing change to a vector component value

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

```
True

>>> omat.GetPropNames()

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

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

True

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

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

Change choice property value

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

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

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

True

>>> omat.GetPropNames()
```

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

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

True

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

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

Change choice property value

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

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

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

True

>>> omat.GetPropNames()

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

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

>>>
```

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, and Reporter.

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

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

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")

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.

Callback Scripting Using PropHost 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 PropHost. The PropHost represents the PropServer (owner of properties) that contains the Property that is calling the Callback script. Therefore, the Callback script can use the PropHost's functions to query or set other properties in the same PropServer.

Definitions

<**propName**> = text string

<**value**> = double

<**valueText**> = text string

<**fileName**> = full path file name

<**choices**> = string containing menu choices separated by commas

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

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

<**bool**> is 1 for true or 0 for false

<propType>	<propTypeName>	Property Description
0	TextProp	Text
1	MenuProp	Menu
2	CheckboxProp	Checkbox
3	VariableProp	Variable
4	VPointProp	VPoint
5	V3DPointProp	V3DPoint
6	NumberProp	Number
7	ColorProp	Color
8	PointProp	Point
9	ValueProp	Value
10	ButtonProp	Button
11	SeparatorProp	Separator
12	NetlistProp	Netlist
13	FileNameProp	FileName

<tabType>	Description	Objects with this tab type
0	DefaultTab	
1	PassedParameterTab	Instances of designs, components, geometric objects
2	DefinitionParameterTab	Definitions of designs, components
3	LocalVariableTab	Definitions of designs, components
4	ProjectVariableTab	Projects
5	ConstantsTab	Projects
6	BaseElementTab	Geometric objects

7	ComponentTab	Designs, components
8	PropertyTab	
9	CircuitTab	
10	SystemTab	
11	HFSS3DLayoutTab	
12	HfssTab	HFSS objects
13	OptimetricsTab	Optimetrics data
14	AltraSimTab	
15	Report3DTab	Report3d
16	FieldsPostProcessorTab	Fieldspostprocessor
17	MeshSetupTab	Manual meshing setup
18	RadFieldSetupTab	Radiation field geometry setup
19	Geometry3DAttributeTab	Geometry3D
20	Geometry3DCmdTab	Geometry3D
21	Geometry3DPolylineTab	Geometry3D
22	Geometry3DCSTab	Geometry3D
23	Geometry3DPlaneTab	Geometry3D
24	Geometry3DPointTab	Geometry3D
25	Geometry3DListTab	Geometry3D
26	StandardPropTab	
27	PropDisplayPropTab	
28	CustomPropTab	
39512	Component	Designs, components

The topics for this section include:

ChangeProperty for HFSS and Maxwell

ChangeProperty for Schematic and Layout

ChangeProperty (Schematic Editor and Layout Editor)

Use: Changes to properties are scripted using the ChangeProperty command.

This command can be executed by the oEditor to change editor properites, by the oDesign to change design level properties, and by the oProject to change project level properties. The command can be used to create, edit, and/or remove properties. In HFSS and Maxwell, only Variable and Separator properties can be deleted.

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

Command: None

Syntax: ChangeProperty Array("Name:AllTabs", <PropTabArray>, <PropTabArray>, ...)

> ChangeProperty(<modulename>:<setup name>:<sweep name>)

Return Value: None

Parameters: <PropTabArray>

```
  Array("Name:<PropTab>",
     <PropServersArray>,
     <NewPropsArray>,
     <ChangedPropsArray>,
     <DeletedPropsArray>)


<PropServersArray>
  Array("Name:PropServers", <PropServer>,
```

```
   <PropServer>, ...)

<NewPropsArray>
   Array("Name:NewProps", <PropDataArray>,
   <PropDataArray>,...)

<ChangedPropsArray>
   Array("Name:ChangedProps",<PropDataArray>,
   <PropDataArray>, ...)

<DeletedPropsArray>
    Array("Name:DeletedProps", <PropName>,
   <PropName>, ...)

<PropDataArray>
   Array("NAME:<PropName>",
     "PropType:=", <PropType>,
     "NewName:=", <string>,
     "Description:=", <string>,
     "NewRowPosition:=", <int>,
```

```
    "ReadOnly:=", <bool>,
    "Hidden:=", <bool>,
    <PropTypeSpecificArgs>)
```

<PropType>

Type: string

Identifies the type of property when a new property is added. In HFSS and Maxwell, only separator properties and variable properties can be added.

```
"SeparatorProp"
"VariableProp"
"TextProp"
"NumberProp"
"ValueProp"
"CheckboxProp"
"MenuProp"
"PointProp"
"VPointProp"
"V3DPointProp"
"ButtonProp"
```

NewName

Specify the new name of a property if the property's name is being edited. In HFSS and Maxwell, the name can only be changed for separators and variables.

Description

Specify a description of the property. In HFSS and Maxwell, the description can only be changed for separators and variables.

NewRowPosition

Used to reorder rows in the **Property** dialog box.

In HFSS, this only applies to the **Project>Project Variables** panel and the **Hfss>DesignProperties** panel.

In Maxwell, this only applies to the **Project>Project Variables** panel and the **Maxwell3D>DesignProperties**, **Maxwell2D>Design Properties**, or **RMxprt>Design Properties** panels. Specify the new zero-based row index of the variable or separator.

ReadOnly

Used to mark a property as "read only" so it can not be modified. In HFSS and Maxwell, this flag can only be set for variables and separators.

Hidden

Used to hide a property so it can not be viewed outside of the **Property** dialog box. In HFSS and Maxwell, this flag can only be set for variables and separators.

```
<PropTypeSpecificArgs>
  SeparatorProp: no arguments
  TextProp: "Value:=", <string>
  NumberProp: "Value:=", <double>
  ValueProp: "Value:=", <value>
  CheckboxProp: "Value:=", <bool>
  MenuProp: "Value:=", <string>
  PointProp "X:=", <double>, "Y:=", <double>
  VPointProp: "X:=", <value>, "Y:=", <value>
  V3DPointProp: "X:=",<value>, "Y:=",<value>,
    "Z:=",<value>
  Material Button: "Material:=", <string>
  Color Button: "R:=",<int>,"G:=",<int>,"B:=",<int>
  Transparency Button: "Value:=", <double>
```

<PropTypeSpecificArgs> for VariableProps

Syntax:

"Value:=", <value>, <OptimizationFlagsArray>,

<TuningFlagsArray>, <SensitivityFlagsArray>,

<StatisticsFlagsArray>

Parameters:

<OptimizationFlagsArray>

Array("NAME:Optimization",
"Included:=", <bool>,
"Min:=", <value>,
"Max:=", <value>)

<Tuning flagsArray>

Array("NAME:Tuning",
"Included:=", <bool>,
"Step:=", <value>,
"Min:=", <value>,
"Max:=", <value>)

<SensitivityFlagsArray>

Array("NAME:Sensitivity",
"Included:=", <bool>,
"Min:=", <value>,
"Max:=", <value>,
"IDisp:=", <value>)

```
    <StatisticsFlagsArray>
      Array("NAME:Statistical",
"Included:=", <bool>,
"Dist:=", <Distribution>,
"StdD:=", <value>,
"Min:=", <value>,
"Max:=", <value>,
"Tol:=", <string>)
```

<Distribution>

Type: string

Value should be "Gaussian" or "Uniform"

StdD

Standard deviation.

Min

Low cut-off for the distribution.

Max

High cut-off for the distribution.

Tol

Tolerance for uniform distributions. Format is "<int>%".

Example: "20%".

VB Example: Adding a new project level variable "$width":

```
oProject.ChangeProperty Array("NAME:AllTabs",_
  Array("NAME:ProjectVariableTab",_
    Array("NAME:PropServers", "ProjectVariables"),_
    Array("NAME:NewProps",_
      Array("NAME:$width",_
"PropType:=", "VariableProp",_
"Value:=", "3mm",_
"Description:=", "my new variable"))))
```

VB Example: Deleting the design level variable "height":

```
oDesign.ChangeProperty Array("NAME:AllTabs",_
```

```
  Array("NAME:LocalVariableTab",_
    Array("NAME:PropServers", "DefinitionParameters"),_
    Array("NAME:DeletedProps", "height"))
```

VB Example: Changing a property's value. If the following command were executed, then the value of the property "XSize" of the PropServer "Box1:CreateBox:1" on the "Geometry3DCmdTab" tab would be changed. (oEditor is the Geometry3D editor in HFSS.)

```
oEditor.ChangeProperty Array("NAME:AllTabs",_
  Array("NAME:Geometry3DCmdTab",_
      Array("NAME:PropServers","Box1:CreateBox:1"),_
      Array("NAME:ChangedProps",_
Array("NAME:XSize", "Value:=", "1.4mil"))))
```

VB Example: Changing the Company Name, Design Name, the background color, and the Axis scaling in a Report.

```
Set oProject = oDesktop.SetActiveProject("wgcombiner")
Set oDesign = oProject.SetActiveDesign("HFSSDesign2")
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"))))

oModule.ChangeProperty Array("NAME:AllTabs", Array("NAME:Header",_ Array("NAME:PropServers", "XY
Plot1:Header"), _

Array("NAME:ChangedProps", Array("NAME:Design Name", _

"Value:=", "WG Combiner"))))

oModule.ChangeProperty Array("NAME:AllTabs", Array("NAME:General",_ Array("NAME:PropServers", "XY
Plot1:General"), _

Array("NAME:ChangedProps", Array("NAME:Back Color", _

"R:=", 128, "G:=", 255, "B:=", 255))))

oModule.ChangeProperty Array("NAME:AllTabs", Array("NAME:Axis", _ Array("NAME:PropServers", "XY
Plot1:AxisX"), _

Array("NAME:ChangedProps", Array("NAME:Axis Scaling", _

"Value:=", "Log"))))
```

PropHost Functions

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

GetEditor

GetPropHost

AddMenuProp

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

<table>
<tr><td>UI Access</td><td>NA</td></tr>
<tr><td>Parameters</td><td>
<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td><tabType></td><td>Integer</td><td>One of the following, where tab titles are shown in parentheses:
1 for PassedParameterTab ("Parameter Values")
2 for DefinitionParameterTab (Parameter Defaults")
3 for LocalVariableTab ("Variables" or "Local Variables")
4 for ProjectVariableTab ("Project variables")
5 for ConstantsTab ("Constants")
6 for BaseElementTab ("Symbol" or "Footprint")
7 for ComponentTab ("General")
28 for CustomTab ("Intrinsic Variables")
39500 for Quantities ("Quantities")
39506 for Signals ("Signals")
39512 for Component ("Component")</td></tr>
<tr><td><propName></td><td>String</td><td>Name of the new menu property</td></tr>
<tr><td><choices></td><td>String</td><td>A comma-separated list of menu choice strings.</td></tr>
<tr><td><initialChoice></td><td>String</td><td>One of the strings in the list of choices.</td></tr>
</table>
</td></tr>
<tr><td>Return Value</td><td>None</td></tr>
</table>

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

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

AddMenuProp2

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

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

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

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

AddProp

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

Command: None

Syntax: AddProp(<tabType>, <propType>, <propName>, <valueText>)

Return Value: None.

VB Example: `PropHost.AddProp(2, 3, "W1", "10mm"); creates a new VariableProp in the DefinitionParameters tab named W1 with value 10mm.`

<table>
<tr><td>UI Access</td><td>NA</td></tr>
<tr><td>Parameters</td><td>
<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td><tabType></td><td>Integer</td><td>One of the following, where tab titles are shown in parentheses:
1 for PassedParameterTab ("Parameter Values")
2 for DefinitionParameterTab (Parameter Defaults")
3 for LocalVariableTab ("Variables" or "Local Variables")
4 for ProjectVariableTab ("Project variables")
5 for ConstantsTab ("Constants")
6 for BaseElementTab ("Symbol" or "Footprint")
7 for ComponentTab ("General")
28 for CustomTab ("Intrinsic Variables")
39500 for Quantities ("Quantities")
39506 for Signals ("Signals")
39512 for Component ("Component")</td></tr>
<tr><td><propType></td><td>Integer</td><td>One of the following to add new definition properties (e.g., to component, design or project):
0 for TextProp (any string)
1 for Menu (one,two,three)
2 for CheckboxProp ("true" or "false" values)</td></tr>
</table>
</td></tr>
</table>

			3 for VariableProp (useful for Project and Local Variables, expression) 4 for VPointProp (x and y values, which can be expressions) 5 for V3DPointProp (x, y and z values, which can be expressions) 6 for NumberProp (simple number, no expression) 7 for ColorProp (takes an RGB value) 8 for PointProp (x and y values, no expressions) 11 for SeparatorProp (used for property list organization, the value is the separator "title") 13 for FileNameProp (special button property for filenames) 39506 for GenericProp (for generics in Twin Builder, SML and VHDL) 39500 for QuantityProp (for Twin Builder quantities, SML and VHDL) 39504 for SignalProp (for VHDL signals in Twin Builder) 39502 for VariablePropNU (like a `VariableProp`, but provides access to setting NetlistUnits in Component definition dialog for Designer; for Twin Builder, use `GenericProp` in `DefinitionParametersTab`, `QuantitiyProp` in `Quantities`, and `signalProp` in `Signals`)
	<propName>	String	Name of the new menu property.
	<valueText>	String	The property value
Return Value	None		

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

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

Note:

AddProp adds a new property using the <propName> preceded by a separator "--". For instance, a new property added with the <propName> "xxx" will be added with the name "--xxx". Consequently, all subsequent script functions that wish to access the added property must now use the name "--xxx".

For example:

PropHost.AddProp 1, 11, "Time_Domain_Options",""

Prophost.SetHidden "--Time_Domain_Options",1

propType	Property Type	Value Example
0	Text	"hello"
1	Menu	"one,two,three"
2	Checkbox	"true" or "false"
3	Variable	"34mm" or "2*val1"
4	VPoint	"x1,y1"
5	V3DPoint	"x1,y1,z1"
6	Number	"17" or "22pF"
7	Color	

8	Point	"34,27"
9	Value	"34mm" or "2*val1"
11	Separator	"colors"
13	Filename	"c:\temp\ne22.s2p"

AddProp2

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

<table>
<tr><td>UI Access</td><td>NA</td></tr>
<tr><td>Parameters</td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><tabType></td><td>Type</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><propType></td><td>String</td><td>One of the following to add new definition properties (e.g., to component,</td></tr>
</table>
</td></tr>
</table>

			design or project): TextProp (any string) CheckboxProp ("true" or "false" values) VariableProp (useful for Project and Local Variables, expression) VPointProp (x and y values, which can be expressions) V3DPointProp (x, y and z values, which can be expressions) NumberProp (simple number, no expression) ColorProp (takes an RGB value) PointProp (x and y values, no expressions) SeparatorProp (used for property list organization, the value is the separator "title") FileNameProp (special button property for filenames) GenericProp (for generics in Twin Builder, SML and VHDL) QuantityProp (for Twin Builder quantities, SML and VHDL) SignalProp (for VHDL signals in Twin Builder) VariablePropNU (like a `VariableProp`, but provides access to setting NetlistUnits in Component definition dialog for Designer; for Twin Builder, use `GenericProp` in `DefinitionParametersTab`, `QuantitiyProp` in `Quantities`, and `signalProp` in `Signals`)
	<propName>	String	Name of the new menu property.
	<valueText>	String	The property value
Return Value	None		

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

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

ExecuteScript

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

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

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

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

GetApplication

Returns the application object currently running the script.

UI Access	NA
Parameters	Name | Type | Description None | |
Return Value	Object The Application

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

VB Syntax	GetApplication
VB Example	`Dim oAnsoftApp2` `oAnsoftApp2 = prophost.GetApplication`

GetCallback

Finds named property and returns name of Callback script.

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

Python Syntax	GetCallback(<propName>)
Python Example	`a = PropHost.GetCallback( "W1")`

VB Syntax	GetCallback(<propName>)
VB Example	`a = PropHost.GetCallback( "W1")`

GetChangedProperty

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

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

	Returns "C" if the script was a Callback associated with the property named "C" and the script was called in response to the property "C" changing value.

Python Syntax	GetChangedProperty()
Python Example	`pn = PropHost.GetChangedProperty()`

VB Syntax	GetChangedProperty()
VB Example	`pn = PropHost.GetChangedProperty()`

GetDescription

Finds named property and returns description string

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

Python Syntax	GetDescription(<propName>)

Python Example	`a = PropHost.GetDescription( "W1")`

VB Syntax	GetDescription(<propName>)
VB Example	`a = PropHost.GetDescription( "W1")`

GetDesign

Use: Returns the interface to the specified design simulation.

Command: None

Syntax: GetDesign <DesignName>

Return Value: Interface to the specified design simulation.

Parameters: <DesignName>

Type: <string>

VB Example: `Set oDesign = oPropHost.GetDesign ("DesignerModel1")`

GetEditor

Use: Returns an interface to the editor requested IF the PropServer behind the PropHost is contained within that type of editor.

Command: None

Syntax: GetEditor(<editorName>)

Return Value: String

VB Example: `Set oLayout2 = PropHost.GetEditor("Layout"); returns the interface to the layout containing a selected component. This interface can be used to call Layout Scripting functions.`

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

GetEvaluatedText

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

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

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

VB Syntax	GetEvaluatedText (<propname>)
VB Example	`ret = prophost.GetEvaluatedText("R")`

GetFileName

Finds the full path name to propName.

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

> **Note:**
>
> Directory variables can be used in the property's value (e.g. $projectdir, $userlib, $syslib, $personallib) and these will be expanded to the correct path. GetFileName always returns a path string; if propName actually contains a variable expression, that expression is evaluated to a constant string before returning.

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

VB Syntax	GetFileName(<propName>)
VB Example	`a = PropHost.GetFileName("SubstrateFile")`

GetHidden

Finds named property and returns its Hidden flag.

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

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

VB Syntax	GetHidden(<propName>)
VB Example	`a = PropHost.GetHidden( "W1")`

GetProgress

Use: Returns the percentage (from 0 to 100) of the simulation completed.

Command: None

Syntax: GetProgress(<simProgress>)

Return Value: String

VB Example: `a = PropHost.GetDesign(<simProgress> );`

GetPropHost

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

Command: None

Syntax: GetPropHost()

Return Value: Returns interface to PropHost.

VB Example: `Set oPropHost2 = CompInstance.GetPropHost();`

Returns the interface to the properties of the compInstance.

This interface can be used to call PropHost functions; for more information see Callback Scripting Using PropHost Object.

Python Syntax	GetPropHost()
Python Example	`oPropHost2 = CompInstance.GetPropHost()`

GetPropServers

Returns array of objects that have properties showing on tabTypeName.

UI Access	NA
Parameters	Name | Type | Description None | |
Return Value	Array of propserver names.

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

	might contain "CompInst@CAP_;1".

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

GetPropTabType

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

UI Access	NA
Parameters	<table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><propName></td><td>String</td><td>Name of the property</td></tr></table>
Return Value	Returns string

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

VB Syntax	GetPropTabType(<propName>)

VB Example	`a = PropHost.GetPropTabType( "W1")`

GetReadOnly

Finds named property and returns its ReadOnly flag.

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

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

VB Syntax	GetReadOnly(<propName>)
VB Example	`a = PropHost.GetReadOnly( "W1");` `returns 1`

GetRunStatus

Use: Returns the status number of the specified design simulation.

Command: None

Syntax: GetRunStatus(<statusNumber>)

Return Value: Returns string.

VB Example: `a = PropHost.GetRunStatus(<statusNumber>);`

GetTabTypeName

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

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

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

VB Syntax	GetTabTypeName(<propName>)
VB Example	`a = PropHost.GetTabTypeName( "W1")`

GetText

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

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

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

VB Syntax	GetText(<propName>)
VB Example	`a = PropHost.GetText("C")`

GetValue

Use: Finds property in any tab and returns its value as a double.

Command: None

Syntax: GetValue(<propName>)

Return Value: Returns double.

VB Example: `a = PropHost.GetValue("C") ;`

> **Note:**
>
> Values are returned in SI units. Compound SI units are, in general, not supported. Temperature values are returned in Celsius

IsValueConstant

Determines whether the specified property is a constant or not. Returns True if propName is evaluated to be a constant; returns False if propName is evaluated to be an expression.

UI Access	NA
Parameters	Name: <propname> (String) — The name of the property.
Return Value	Integer; non-zero for constant, zero for variable or expression.

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

VB Syntax	IsValueConstant (<propname>)

VB Example	`ret = prophost.IsValueConstant("R")`

PropertyExists

Finds named property and returns its property type.

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

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

VB Syntax	PropertyExists(<propName>)
VB Example	`a = PropHost.PropertyExists( "W1")`

RemoveProp

Removes the named property from whichever tab it is found.

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

Return Value	None

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

VB Syntax	RemoveProp(<propName>)
VB Example	`PropHost.RemoveProp("W1")`

SetCallback

Finds named property and sets its Callback script.

UI Access	NA
Parameters	<table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><propName></td><td>String</td><td>Name of the property</td></tr><tr><td><scriptName></td><td>String</td><td>Name of a script stored in a script library.</td></tr></table>
Return Value	None

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

VB Syntax	SetCallback(<propName>, <scriptName>)
VB Example	`PropHost.SetCallback("W1", "SynchronizeResistors")`

SetDescription

Finds named property and sets its description text.

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

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

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

SetHidden

Finds named property and sets its Hidden flag.

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

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

VB Syntax	SetHidden(<propName>, <hidden>)
VB Example	`PropHost.SetHidden("W1", 1)`

SetReadOnly

Finds named property and sets its ReadOnly flag.

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

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

VB Syntax	SetReadOnly(<propName>, <hidden>)
VB Example	`PropHost.SetReadOnly("W1", 1)`

SetText

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

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

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

VB Syntax	SetText(<propName>, <text>)

VB Example	`PropHost.SetText("C", "22nF")`

SetValue

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

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

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

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

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()`

VB Syntax	GetArrayVariables
VB 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
Parameters	<table><tr><th>Name</th><th>Type</th><th>Description</th></tr><tr><td><PropTab></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") </td></tr></table>

			• ProjectVariableTab ("Project variables") • ConstantsTab ("Constants") • BaseElementTab ("Symbol" or "Footprint") • ComponentTab ("General") • Component("Component") • CustomTab ("Intrinsic Variables") • Quantities ("Quantities") • Signals ("Signals")
	<PropServer>	String	An object identifier, generally returned from another script method, such as CompInst@R;2;3
Return Value	Array of strings containing the names of the appropriate properties.		

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

VB Syntax	GetProperties <*PropTab*>, <*PropServer*>
VB Example	`oEditor.GetProperties "PassedParameterTab", "k"`

GetPropertyValue

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

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

UI Access	N/A
Parameters	<table><tr><th>Name</th><th>Type</th><th>Description</th></tr><tr><td><*PropTab*></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><*PropServer*></td><td>String</td><td>An object identifier, generally returned from another script method, such as <code>Comp:Inst@R;2;3</code></td></tr><tr><td><*PropName*></td><td>String</td><td>Name of the property.</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") ...

VB Syntax	GetPropertyValue (<*PropTab*>, <*PropServer*>, <*PropName*>)
VB 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()`

VB Syntax	GetVariables
VB 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")`

VB Syntax	GetVariableValue <*VarName*>
VB 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.

<table>
<tr><td>UI Access</td><td>N/A</td></tr>
<tr><td>Parameters</td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><propTab></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><propServer></td><td>String</td><td>An object identifier, generally returned from another script method, such as Comp‡nst@R;2;3</td></tr>
<tr><td><propName></td><td>String</td><td>Name of the property.</td></tr>
<tr><td><propValue></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")`

VB Syntax	SetPropertyValue <*propTab*>, <*propServer*>, <*propName*>, <*propValue*>
VB 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	<table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><*VarName*></td><td>String</td><td>Variable name.</td></tr><tr><td><*VarValue*></td><td>Value</td><td>New value for the variable.</td></tr></table>
Return Value	None.

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

VB Syntax	SetVariableValue <*VarName*>, <*VarValue*>
VB Example	`oProject.SetVariableValue "$Var1", "3mm"`

Example Use of Record Script and Edit Properties

A simple way to see how to format the string arguments for a design object or property of interest is to use the script recording command and then edit the property. Open the script file and look at the oEditor.ChangeProperty entry to see the string arguments.

```
Dim oAnsoftApp

Dim oDesktop

Dim oProject

Dim oDesign

Dim oEditor

Dim oModule

Set oAnsoftApp = CreateObject("Ansoft.ElectronicsDesktop")

Set oDesktop = oAnsoftApp.GetAppDesktop()

oDesktop.RestoreWindow

Set oProject = oDesktop.SetActiveProject("wg_combiner")

Set oDesign = oProject.SetActiveDesign("HFSSModel1")

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

oEditor.ChangeProperty Array("NAME:AllTabs", Array("NAME:Geometry3DAttributeTab", Array
("NAME:PropServers", _
```

```
  "Polyline1"), Array("NAME:ChangedProps", Array("NAME:Display Wireframe", "Value:=", true), _
  Array("NAME:Display Wireframe", "Value:=", false), Array("NAME:Transparent", "Value:=",
0.2))))
```

Additional Property Scripting Examples

The following is a sample VB script that uses the GetPropertyValue, SetPropertyValue, and GetProperties functions. The script gets all the properties of the first CreateBox command of "Box1". It then loops through the properties and for each one, shows the user the current value and asks if the value should be changed.

```
Dim all_props
Dim prop
all_props = oEditor.GetProperties("Geometry3DCmdTab",_
  "Box1:CreateBox:1")
For Each prop In all_props
  val = oEditor.GetPropertyValue("Geometry3DCmdTab",_
    "Box1:CreateBox:1", prop)
  new_val = InputBox("New Value of " + prop + ":",_
    "Current Value of '" + prop + "' is " + val, val)
  If new_val <> val Then
  oEditor.SetPropertyValue "Geometry3DCmdTab",_
    "Box1:CreateBox:1", prop, new_val
  val = _
    oEditor.SetPropertyValue("Geometry3DCmdTab",_
    "Box1:CreateBox:1", prop)
```

```
  MsgBox("Now the value of '" + prop + "' is " + val)
  End If
Next
```

The following is a sample VB script that creates a HFSS 3D Layout design, draws a rectangle in the layout editor and uses the GetPropertyValue, SetPropertyValue and GetProperties functions. The script gets all properties of the rectangle. It then loops through the properties and for each one, shows the user the current value and asks if the value should be changed. Note that the last call to GetPropertyValue in the script will fail if you change the name of the rectangle from the script.

```
Dim oAnsoftApp
Dim oDesktop
Dim oProject
Dim oDesign
Dim oEditor
Dim oModule
Set oAnsoftApp = CreateObject("Ansoft.ElectronicsDesktop")
Set oDesktop = oAnsoftApp.GetAppDesktop()
'
oDesktop.RestoreWindow
oDesktop.NewProject
Set oProject = oDesktop.GetActiveProject
'
' CREATE A RECTANGLE IN HFSS_3D_LAYOUT
```

```
'
oProject.InsertDesign "HFSS 3D Layout", "HFSS 3D Layout1", _
"C:\testinstall\Designer\syslib\PCB - SingleSided.asty", ""
Set oDesign = oProject.SetActiveDesign("HFSS 3D Layout1")
Set oEditor = oDesign.SetActiveEditor("Layout")
oEditor.CreateRectangle Array("NAME:Contents", _
"rectGeometry:=", Array("Name:=", _
"rect_1", "LayerName:=", "Top", "lw:=", _
"0mm", "Ax:=", "-22mm", "Ay:=", "20mm", "Bx:=", _
"29mm", "By:=", "-4mm", "ang:=", "0deg"))
'
' GET ALL PROPERTIES OF THE RECTANGLE
'
Dim all_props
Dim prop
Dim val
Dim new_val
'
all_props = oEditor.GetProperties("BaseElementTab","rect_1")
'
' LOOP OVER ALL PROPERTIES
```

```
'
For Each prop in all_props
val = oEditor.GetPropertyValue("BaseElementTab","rect_1",prop)
'
' DISPLAY VALUE TO THE USER
'
new_val = InputBox("New Value of "+prop+":",_
"Current Value of "+prop+" is "+val,val)
'
' CHANGE THE VALUE IF DESIRED
'
If new_val <> val Then
oEditor.SetPropertyValue _
"BaseElementTab","rect_1",prop,new_val
val = _
oEditor.GetPropertyValue("BaseElementTab","rect_1",prop)
MsgBox("Now the value of "+prop+" is "+val)
End If
'
Next
```

'

This page intentionally
left blank.

3 - Application Object Script Commands

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

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

General Application Script Commands

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

- GetAppDesktop

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

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

GetAppDesktop

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

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

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

VB Syntax	GetAppDesktop()
VB Example	`Set 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>
```

SetTempDirectory

ShowDockingWindow

Sleep

StopSimulations

SubmitJob

TileWindows

Related Topics:

Desktop Commands For Registry Values

ImportExport Tool Commands

AddMessage

Add a message with severity and context to message window.

UI Access	N/A
Parameters	

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

Return Value	None.

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

VB Syntax	AddMessage <*projectName*>, <*designName*>, <*severity*>, <*msg*>, <*category*>
VB Example	`oDesktop.AddMessage "Project1", "Circuit1", 0, "This is a test message", ""`

ClearMessages

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

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

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

Name	Type	Description
<projectName>	String	Name of the project from which to clear messages.
<designName>	String	Name of the design under the <projectName> from which to clear messages.
<startSeverity>	Integer	User-specified severity level at which messages start getting cleared: • 0 = informational • 1 = warning • 2 = error • 3 = fatal error
<stopSeverity>	Integer	*Optional*. User-specified stop severity level until which messages will be cleared. • 0 = informational

			• 1 = warning • 2 = error • 3 = fatal error If not specified, <stopSeverity> has a default value of 0.
Return Value	None.		

Python Syntax	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 mes-
sages

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

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

oDesktop.ClearMessages("Ex_EC_1_Asymmetrical_Conductor", "Maxwell3DDesign1", 0,
2)

# startSeverity = 0 and stopSeverity = 3; clears all the Info, Warning, Error,
and Fatal messages

oDesktop.ClearMessages("Ex_EC_1_Asymmetrical_Conductor", "Maxwell3DDesign1", 0,
3)


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

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

# startSeverity = 1 and stopSeverity = 1; clears all the Warning messages

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

# startSeverity = 1 and stopSeverity = 2; clears all the Warning and Error mes-
sages

oDesktop.ClearMessages("Ex_EC_1_Asymmetrical_Conductor","Maxwell3DDesign1",1, 2)

# startSeverity = 1 and stopSeverity = 3; clears all the Warning, Error, and
Fatal messages
```

```
oDesktop.ClearMessages("Ex_EC_1_Asymmetrical_Conductor","Maxwell3DDesign1",1, 3)


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


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

VB Syntax	ClearMessages <*projectName*>, <*designName*>, <*startSeverity*>, <*stopSeverity*>
VB Example	`Set oAnsoftApp = CreateObject("Ansoft.ElectronicsDesktop")` `Set oDesktop = oAnsoftApp.GetAppDesktop()` `oDesktop.ClearMessages "MyProject", "MyDesign", 0, 3`

CloseAllWindows

Closes all MDI child windows on the desktop.

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

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

VB Syntax	CloseAllWindows
VB Example	`oDesktop.CloseAllWindows`

CloseProject

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

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

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

VB Syntax	CloseProject <*ProjectName*>
VB Example	`oDesktop.CloseProject "MyProject"`

CloseProjectNoForce

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

UI Access	**File** > **Close**
Parameters	Name | Type | Description <ProjectName> | String | The name of the project already on the Desktop that is to be closed, without path or extension

Return Value	None

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

VB Syntax	CloseProjectNoForce <*ProjectName*>
VB 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`

VB Syntax	Count
VB Example	`Dim oAnsoftApp` `Dim oDesktop`

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

DeleteProject

Deletes a project from disk.

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

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

VB Syntax	DeleteProject <*ProjectName*>
VB Example	`oDesktop.DeleteProject "MyProject"`

DownloadJobResults

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

UI Access	Select Scheduler		
Parameters	Name	Type	Description
	<jobID>	String	Provide the job ID of the target job. The job ID must be able to be found in current .q (or .q.completed) file, or inside the “batchinfo” folder in projectPath. If the job ID is empty, the job ID in current .q (or .q.completed) file will be used.

	<projectPath>	String	A string of path to locate the project folder. The project file may be not necessary, but .q (or .q.completed) file or “batchinfo” folder with valid .jobid files are required.
	<resultPath>	String	A string giving the folder path for the download to save to.
	<Filters> (optional)	String	A string containing filters to download. The delimiter of file types is “;”. If no filter specified, the default filter “*” will be applied, which requests all files for download.
Return Value	A Boolean result about download complete or not		

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

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

DeleteRegistryEntry

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

UI Access	N/A		
Parameters	Name	Type	Description
	<pathToRegistry>	String	Path to Registry entry.

Return Value	Boolean: • **True** – Key has been deleted. • **False** – Key does not exist.

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

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

DoesRegistryValueExist

Determines whether a registry value exists.

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

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

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

EnableAutoSave

Enable or disable autosave feature.

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

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

VB Syntax	EnableAutoSave <*enable*>

VB Example	`oDesktop.EnableAutoSave True`

ExportOptionsFiles

Copies the options config files to the *DestinationDirectory*.

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

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

VB Syntax	ExportOptionsFiles <*DestinationDirectory*>
VB Example	`oDesktop.ExportOptionsFiles"D:/test/export/"`

GetActiveProject

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

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

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

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

VB Syntax	GetActiveProject
VB Example	`Set oProject = oDesktop.GetActiveProject`

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()`

VB Syntax	GetAutoSaveEnabled
VB 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()`

VB Syntax	GetBuildDateTimeString
VB Example	`dnt = oDesktop.GetBuildDateTimeString`

GetCustomMenuSet

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

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

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

VB Syntax	GetCustomMenuSet
VB Example	`Set oProject = oDesktop.GetCustomMenuSet`

GetDefaultUnit

Returns the default unit for a physical quantity.

UI Access	**Tools** > **Options** > **General Options** > **Default Units**. Note that this menu only displays units that can be changed, while the script can be used to view additional default units.
Parameters	Name | Type | Description <type> | String | String containing a type of measurement.

			Valid strings are (case insensitive): • "Acceleration" • "Angle" • "AngularAcceleration" • "AngularDamping" • "AngularSpeed" • "Capacitance" • "Conductance" • "Current" • "CurrentChangeRate" • "DataRate" • "DeltaH" (Magnetic Field Strength) • "Density" • "Flux" • "Force" • "Frequency" • "Inductance" • "Length" • "MagneticReluctance" • "Mass" • "MassFlowRate" • "MomentInertia" • "Power" • "Pressure" • "PressureCoefficient"

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

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

VB Syntax	GetDefaultUnit <type>
VB 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()`

VB Syntax	GetDesktopConfiguration
VB Example	`Set oProject = oDesktop.GetDesktopConfiguration`

GetDistributedAnalysisMachines

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

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

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

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

GetDistributedAnalysisMachinesForDesignType

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

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

Python Syntax	GetDistributedAnalysisMachinesForDesignType (<designTypeName>)
Python Example	`machineNames =oDesktop.` `GetDistributedAnalysisMachinesForDesignType` `("Circuit")`

VB Syntax	GetDistributedAnalysisMachinesForDesignType <designTypeName>

VB Example	`Set machineNames =oDesktop.` `GetDistributedAnalysisMachinesForDesignType` `("Circuit")`

GetExeDir

Returns the path where the executable is located.

UI Access	N/A
Parameters	None.
Return Value	String path where executable is located. Example: 'C:/Program Files/AnsysEM/v242/Win64/'

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

VB Syntax	GetExeDir
VB Example	`oDesktop.GetExeDir`

GetGDIObjectCount

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

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

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

GetLibraryDirectory

Get the path to the `SysLib` directory.

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

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

VB Syntax	GetLibraryDirectory
VB Example	`MsgBox oDesktop.GetLibraryDirectory`

VB Example:

```
------------------------------------------------------
message box returns the path in this example
------------------------------------------------------
Dim oAnsoftApp
Dim oDesktop
Dim oProject
Dim oDesign
Dim oEditor
Dim oModule
Set oAnsoftApp = CreateObject("Ansoft.ElectronicsDesktop")
Set oDesktop = oAnsoftApp.GetAppDesktop()
oDesktop.RestoreWindow
libdir = oDesktop.GetLibraryDirectory
msgbox(oDesktop.GetLibraryDirectory())
```

GetLocalizationHelper

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

Returns the object for the localization helper.

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

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

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

GetMessages

Get the messages from a specified project and design.

UI Access	NA
Parameters	Name | Type | Description <*ProjectName*> | String | Name of the project for which to collect messages. An incorrect project name results in no messages (design is ignored). An empty project name

			results in all messages (design is ignored)
	<*DesignName*>	String	Name of the design in the named project for which to collect messages. An incorrect design name results in no messages for the named project. An empty design name results in all messages for the named project
	<*Severity*>	Integer	Severity is 0-3, and is tied in to info/warning/error/fatal types as follows: • 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", "Circuit1", 1)`

VB Syntax	GetMessages <*ProjectName*>, <*DesignName*>, <*Severity*>
VB Examples	`Messages = oDesktop.GetMessages "MyProject", "Circuit1", 1`

GetPersonalLibDirectory

Get the path to the `PersonalLib` directory.

UI Access	N/A
Parameters	None.

Return Value	String path to the `PersonalLib` directory.

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

VB Syntax	GetPersonalLibDirectory
VB 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()`

VB Syntax	GetProcessID
VB 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()`

VB Syntax	GetProjectDirectory
VB 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()`

VB Syntax	GetProjectList
VB 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 VBScript methods.

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

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

VB Syntax	GetProjects
VB Example	`oDesktop.GetProjects`

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/Circuit/UpdateReportsDynamicallyOnEdits')`

VB Syntax	GetRegistryInt(<*KeyName*>)
VB Example	`num = oDesktop.GetRegistryInt("Desktop/Settings/ProjectOptions/Circuit/UpdateReportsDynamicallyOnEdits")`

GetRegistryString

Obtains registry key string value.

UI Access	N/A
Parameters	Name: *<KeyName>* — Type: String — Description: 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/Circuit')`

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

GetRunningInstancesMgr

Returns the object of the Running Instances Manager.

UI Access	N/A
Parameters	Name — Type — Description

	None		
Return Value	Object Running instances manager object		

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

VB Syntax	GetRunningInstancesMgr()
VB Example	`Set oRunningInstances = oDesktop.GetRunningInstancesMgr()`

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()`

VB Syntax	GetSchematicEnvironment
VB Example	`Set oProject = 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()`

VB Syntax	GetScriptingToolsHelper
VB 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()`

VB Syntax	GetSysLibDirectory
VB Example	`oDesktop.GetSysLibDirectory`

GetTempDirectory

Gets the path to the `Temp` directory.

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

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

VB Syntax	GetTempDirectory
VB Example	`oDesktop.GetTempDirectory`

GetUserLibDirectory

Gets the path to the UserLib directory.

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

Python Syntax	GetUserLibDirectory()

Python Example	`oDesktop.GetUserLibDirectory()`

VB Syntax	GetUserLibDirectory
VB 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()`

VB Syntax	GetVersion()
VB Example	`oDesktop.GetVersion`

IsFeatureEnabled

Returns a Boolean for whether a queried feature is enabled.

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

Python Syntax	IsFeatureEnabled()
Python Example	`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()`

VB Syntax	GetVersion()
VB Example	`results = oDesktop.IsFeatureEnabled(SF3519)`

KeepDesktopResponsive

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

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

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

VB Syntax	KeepDesktopResponsive <*TimeInMilliseconds*>
VB Example	`oDesktop.KeepDesktopResponsive 10000`

LaunchJobMonitor

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

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

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

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

NewProject

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

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

Python Syntax	NewProject()

Python Example	`oProject = oDesktop.NewProject()`

VB Syntax	NewProject
VB Example	`Set oProject = oDesktop.NewProject`

OpenAndConvertProject

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

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

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

Python Syntax	OpenAndConvertProject(*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

VB Syntax	OpenAndConvertProject(*filePath*, *legacyChoice*)
VB Example	`Set oProject = oDesktop.OpenAndConvertProject("c:\files\optimtee.hfss", 1)` **Note:** optimtee.hfss is gone after this code executes

OpenMultipleProjects

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

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

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

VB Syntax	OpenAndConvertProject <*filePath*>, < *legacyChoice*>

VB Example	`Set 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")`

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

OpenProjectWithConversion

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

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

PageSetup (Layout Editor)

Use: Specifies page setup for printing.

Command: **File>Page Setup**

Syntax: PageSetup <ArgArray>

Return Value: None.

Parameters: <Margins>: Page margins in implicit units of inches.

<Border>: Integer value indicating to draw border (1) or not to draw (0).

<DesignVars>: Integer value indicating to draw design vars (1) or not to draw (0).

VB Example:
```
Set oProject = oDesktop.GetActiveProject()
Set oDesign = oProject.GetActiveDesign()
Set oEditor = oDesign.GetActiveEditor()
oEditor.PageSetup Array("NAME:PageSetupData", "margins:=", Array("left:=", 550, "right:=", _
550, "top:=", 500, "bottom:=", 500), "border:=", 1, "DesignVars:=", 0)
```

PauseRecording

Temporarily stop script recording.

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

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

VB Syntax	PauseRecording
VB 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**
Parameters	Name | Type | Description <*Message*> | String | Any Text.
Return Value	None

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

VB Syntax	PauseScript <*Message*>
VB 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()`

VB Syntax	Print

VB Example	`oDesktop.Print`

QuitApplication

Exits the desktop.

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

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

VB Syntax	QuitApplication
VB 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()`

VB Syntax	RefreshJobMonitor
VB Example	`oDesktop.RefreshJobMonitor`

ResetLogging

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

UI Access	N/A

Parameters	<table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><*logFile*></td><td>String</td><td>Path to log file.</td></tr><tr><td><*logLevel*></td><td>Integer</td><td>Specified log level.</td></tr></table>
Return Value	None.

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

VB Syntax	ResetLogging <*logFile*>, <*logLevel*>
VB Examples	`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><*ArchiveFilePath*></td><td>String</td><td>Path to archived file</td></tr><tr><td><*ProjectFilePath*></td><td>String</td><td>Path to restore location</td></tr><tr><td><*OverwriteExistingFiles*></td><td>Boolean</td><td>True to overwrite an existing file of the same name; else False.</td></tr><tr><td><*OpenProjectAfterRestore*></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)`

VB Syntax	RestoreProjectArchive <*Archivefilepath*>, <*ProjectFilePath*>, <*OverwriteExistingFiles*>, <*OpenProjectAfterRestore*>
VB 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()`

VB Syntax	RestoreWindow
VB Example	`oDesktop.RestoreWindow`

ResumeRecording

Resume recording a script.

UI Access	N/A
Parameters	None.
Return Value	None

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

VB Syntax	ResumeRecording
VB 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.

UI Access	NA
Parameters	

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

Return Value	None

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

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

RunScript

Launches another script from within the script currently being executed.

UI Access	Tools>Run Script

Parameters

Name	Type	Description
<SCriptPath>	String	Name or full path of the script to execute. If the full path to the script is not specified, Twin Builder searches for the specified script in the following locations, in this order: 1. Personal library directory. This is the **PersonalLib** subdirectory in the project directory. The project directory can be specified in the **General Options** dialog box (click >**Tools** > **Options** > **General Options** to open this dialog box) under the **Project Options** tab. 2. User library directory. This is the **userlib** subdirectory in the library directory. The library directory can be specified **General Options** dialog in the box (click **Tools** > **Options** > **General Options** to to open this dialog box) under the **Project Options** tab. 3. System library directory. This is the syslib subdirectory in the library directory. The library dir-

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

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

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

RunScriptWithArguments

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

UI Access	NA
Parameters	Name | Type | Description <*ScriptPath*> | String | The name or full path of the script to execute. If the full path to the script is not specified, the software looks for the script in the following locations:

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

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

VB Syntax	RunScriptWithArguments <*ScriptPath*>, <*Arguments*>
VB Example	`oDesktop.RunScriptWithArguments` `"C:/Project/test2.vbs", "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.

<table>
<tr><td>UI Access</td><td colspan="3">Tools > Job Management > Select Scheduler.</td></tr>
<tr><td>Parameters</td><td colspan="3">
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><option></td><td>String</td><td>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</td></tr>
<tr><td><address (optional)></td><td>String</td><td>String specifying the IP address or hostname of the head node or for the remote host running the RSM service.</td></tr>
<tr><td><username (optional)></td><td>String</td><td>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.</td></tr>
<tr><td><forcePasswordEntry (optional)></td><td>String</td><td>Boolean used to force display of the Select Scheduler GUI to allow for password entry prior to job submission.</td></tr>
</table>
</td></tr>
<tr><td>Return Value</td><td colspan="3">The selected scheduler (if selection was successful, this string should match the input option string, although it</td></tr>
</table>

	could differ in upper/lowercase).

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

VB Syntax	Select Scheduler <*option*>, <*address*>, <*username*>, <*forcePasswordEntry*>
VB 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.

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

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

VB Syntax	SetActiveProject <*ProjectName*>
VB Example	`Set 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.

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

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

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

SetCustomMenuSet

Sets the custom menu set for Electronics Desktop.

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

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

VB Syntax	SetCustomMenuSet(<customMenuSet>)

VB Example	`Set oProject = oDesktop.SetCustomMenuSet "EM"`

SetDesktopConfiguration

Sets the desktop configuration.

UI Access	Navigate to **Tools** > **Options** > **General Options** > **General** > **Desktop Configuration**. Select a configuration using the **Set targeted configuration** drop-down menu.
Parameters	<table><tr><th>Name</th><th>Type</th><th>Description</th></tr><tr><td><configName></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')`

VB Syntax	SetDesktopConfiguration(<configName>)

VB Example	`Set oProject = oDesktop.SetDesktopConfiguration "SI"`

SetLibraryDirectory

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

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

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

VB Syntax	SetLibraryDirectory <*DirectoryPath*>
VB Example	`oDesktop.SetLibraryDirectory"c:\libraries"`

SetProjectDirectory

Sets the project directory path.

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

Return Value	None.

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

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

Python Syntax	SetRegistryFromFile(<*filePath*>)

Python Example	`oDesktop.SetRegistryFromFile('c:/temp/test.acf')`

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

SetRegistryInt

Sets registry key to an integer value.

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

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

VB Syntax	SetRegistryInt <*KeyName*> <*int*>

VB Example	`oDesktop.SetRegistryInt "Desktop/Settings/ProjectOptions/Circuit/Up-dateReportsDynamicallyOnEdits", 0`

SetRegistryString

Sets registry key to a string value.

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

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

VB Syntax	SetRegistryString <*KeyName*>, <*value*>
VB Example	`oDesktop.SetRegistryString "Desktop/ActiveDSOConfigurations/Circuit", "Local"`

SetSchematicEnvironment

Sets the schematic environment for Electronics Desktop.

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

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

VB Syntax	SetSchematicEnvironment(<schEnv>)
VB Example	`Set oProject = oDesktop.SetSchematicEnvironment 2`

SetTempDirectory

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

UI Access	N/A
Parameters	<table><tr><th>Name</th><th>Type</th><th>Description</th></tr><tr><td><*DirectoryPath*></td><td>String</td><td>The path to the Temp directory. This should be writeable by the user.</td></tr></table>

Return Value	None.

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

VB Syntax	SetTempDirectory <*DirectoryPath*>
VB Example	`oDesktop.SetTempDirectory "c:\tmp"`

ShowDockingWindow

Shows or hides a docking window.

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

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

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

VB Syntax	ShowDockingWindow <*windowName*> <*show*>
VB Example	`oDesktop.ShowDockingWindow "Message Manager" False`

Sleep

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

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

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

VB Syntax	Sleep <*TimeInMilliseconds*>
VB Example	`oDesktop.Sleep 1000`

SubmitJob

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

UI Access	**Tools** > **Job Management** > **Submit Job**.
Parameters	<table><tr><th>Name</th><th>Type</th><th>Description</th></tr><tr><td><*settingsPath*></td><td>String</td><td>Path to the settings file (exported from the Submit Job GUI) to use for submission.</td></tr><tr><td><*projectPath*></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><*design (optional)*></td><td>String</td><td>Name of the design to use for batch solve.</td></tr><tr><td><*setup (optional)*></td><td>String</td><td>Name of the specific setup to solve.</td></tr></table>
Return Value	Array of job ID strings (empty if no jobs submitted).

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")`

VB Syntax	SubmitJob <*settingsPath*>, <*projectPath*>, <*design*>, <*setup*>
VB 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.

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

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

VB Syntax	TileWindows <*TilingFlag*>
VB 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 in VB or Python.

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

1. Start Circuit.
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/Circuit", "test")
```

See the following subtopics:

DeleteRegistryEntry

DoesRegistryValueExist

GetRegistryInt

GetRegistryString

SetRegistryFromFile

SetRegistryInt

SetRegistryString

DeleteRegistryEntry

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

UI Access	N/A
Parameters	Name: *<pathToRegistry>*, Type: String, Description: Path to Registry entry.
Return Value	Boolean: • **True** – Key has been deleted. • **False** – Key does not exist.

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

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

DoesRegistryValueExist

Determines whether a registry value exists.

UI Access	N/A

Parameters	<table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><*KeyName*></td><td>String</td><td>Full name of registry key, including path.</td></tr></table>
Return Value	Boolean: • **True** – Key exists. • **False** – Key does not exist.

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

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

GetRegistryInt

Obtains registry key integer value.

UI Access	N/A
Parameters	<table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><*KeyName*></td><td>String</td><td>Full name of registry key, including path.</td></tr></table>
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/Circuit/UpdateReportsDynamicallyOnEdits')`

VB Syntax	GetRegistryInt(<*KeyName*>)
VB Example	`num = oDesktop.GetRegistryInt("Desktop/Settings/ProjectOptions/Circuit/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	`activeDSO = oDesktop.GetRegistryString('Desktop/ActiveDSOConfigurations/Circuit')`

Example	

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

SetRegistryFromFile

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

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

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

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

SetRegistryInt

Sets registry key to an integer value.

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

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

VB Syntax	SetRegistryInt <*KeyName*> <*int*>
VB Example	`oDesktop.SetRegistryInt "Desktop/Settings/ProjectOptions/Circuit/UpdateReportsDynamicallyOnEdits", 0`

SetRegistryString

Sets registry key to a string value.

UI Access	N/A

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

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

VB Syntax	SetRegistryString <*KeyName*>, <*value*>
VB Example	`oDesktop.SetRegistryString "Desktop/ActiveDSOConfigurations/Circuit", "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.

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

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

VB Syntax	ImportANF <*ANFfilename*>
VB Example	`Dim oAnsoftApp` `Dim oDesktop` `Dim oProject` `Dim oDesign` `Dim oEditor` `Dim oModule` `Dim oProjects` `Dim omachine` `Dim oTool` `Set oAnsoftApp = CreateObject("Ansoft.ElectronicsDesktop")` `Set oDesktop = oAnsoftApp.GetAppDesktop()` `oDesktop.RestoreWindow` `Set oTool = oDesktop.GetTool("ImportExport")` `oTool.ImportANF("%UserProfile%\Documents\HFSS Examples\package.anf")`

ImportANFv2

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

UI Access	**File** > **Import** > **ANF**.

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

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", "")`

VB Syntax	ImportANFv2 <*ANFfilename*>, <*outputPathName*>, <*controlFileName*>, <*cmpFileName*>
VB Example	`Dim oAnsoftApp` `Dim oDesktop` `Dim oProject` `Dim oDesign` `Dim oEditor`

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

ImportAutoCAD

Imports an AutoCAD file into a new project.

UI Access	**File** > **Import** > **AutoCAD**.
Parameters	

Name	Type	Description
<*dxfFileName*>	String	Full path of DXF file.
<*outputPathFileName*>	String	Full path of EDB file to create during import.
<*controlFileName*>	String	Full path of XML control file. Pass empty string if none.

Return Value	None.

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

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

ImportAWRMicrowaveOffice

Imports an AWRMicrowaveOffice file into a new project.

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

Python Syntax	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')

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

	oTool.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')

VB Syntax	ImportEDB *<edbFileName>*
VB Example	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><*extractaFileName*></td><td>String</td><td>Full path of Extracta file.</td></tr><tr><td><*outputPathName*></td><td>String</td><td>Full path of output file to be created.</td></tr><tr><td><*controlFileName*></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', '')`

VB Syntax	ImportExtracta <*extractaFileName*>, <*outputPathName*>, <*controlFileName*>
VB Example	`Set oTool = oDesktop.GetTool("ImportExport")` `oTool.ImportExtracta "C:/MyFiles/projectforimport.brd", "C:/MyFiles/project.edb", ""`

ImportGDSII

Imports a GDSII file into a new project.

UI Access	**File** > **Import** > **GDSII**.
Parameters	

Name	Type	Description
<*gdsiiFileName*>	String	Full path of GDSII file.
<*outputPathName*>	String	Full path of EDB file to create during import.
<*controlFileName*>	String	Optional. Full path of XML control file. Full path of “technology” (corresponding to `-t=technologyfile` argument in anstranslator). Pass empty string if none.
<*mapFileName*>	String	If technology file was used in place of the control file. Full path to either gdsii mapping file (corresponding to `-g=gdsMappingFile` argument in anstranslator) or XML control file. Otherwise, pass empty string.

Return Value	None.

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')`

VB Syntax	ImportGDSII <*gdsiiFileName*>, <*outputPathName*>, <*controlFileName*>, <*mapFileName*>
VB Example	`Set oTool = oDesktop.GetTool("ImportExport")`

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

ImportGerber

Imports a GERBER file into a new project.

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

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

VB Syntax	ImportGerber <*gerberFileName*>, <*outputPathName*>, <*controlFileName*>

VB Example	`Set oTool = oDesktop.GetTool("ImportExport")` `oTool.ImportGERBER "C:/Files/test.gbr", "C:/Files/test.aedb.edb", _` `"C:/Files/test.xml"`

ImportIDF

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

UI Access	placeholder

Parameters

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

	<ModelBoardAsRect>	bool	True or False
	<ModelDeviceAsRect>	bool	True or False
	<Cutoff>	bool	True or False
	<IncludeDrilledHoles>	bool	True or False
	<HoleDiameterCutoff>	string	Hole diameter size and unit
	<ReplaceDevices>	bool	True or False
	<CreatePointsOnBoardForInstances>	bool	True or False
Return Value	None.		

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

VB Syntax	ImportIDF <*idfFileName*>, <*outputPathName*>, <*controlFileName*>, <*libFileName*>
VB Example	`oDesign.ImportIDF Array("NAME:Settings", "Board:=", _"C:\Users\Model Files" "\brd_board.emn",` `"Library:=", _"C:\Users\Model Files" "\brd_board.emp", "Control:=", _"C:\Users\Model Files"`

	"\brd_board.xml", "Filters:=", Array("HeightExclude2D"), "CreateFilteredAsNonModel:=", _false, Array("NAME:definitionOverridesMap"), Array("NAME:instanceOverridesMap"), "HighSurfThickness:=", _"0.07mm", "LowSurfThickness:=", "0.07mm", "InternalLayerThickness:=", "0.07mm", "NumInternalLayer:=", _2, "HighSurfaceCopper:=", 30, "LowSurfaceCopper:=", 30, "InternalLayerCopper:=", _30, "TraceMaterial:=", "Cu-Pure", "SubstrateMaterial:=", "FR-4", "CreateBoard:=", _true, "ModelBoardAsRect:=", true, "ModelDeviceAsRect:=", false, "Cutoff:=", _false, "IncludeDrilledHoles:=", true, "ReplaceDevices:=", false)

ImportIDFandMerge

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

UI Access	**File** > **Import** > **IDF**. Enable **Merge with existing** and select a project/design.
Parameters	

Name	Type	Description
<idfFileName>	String	Full path of GERBER file.
<libFileName>	String	Optional. Full path of library file. Pass empty string if none.
<controlFileName>	String	Optional. Full path of XML control file. Pass empty string if none.
<project>	String	Name of project to merge with.
<design>	String	Name of design to merge with.

Return Value	None.

Python Syntax	ImportIDFandMerge(<*idfFileName*>, <*libPathName*>, <*controlFileName*>, <*project*>, <*design*>)
Python Example	`oDesktop.RestoreWindow()` `Set oTool = oDesktop.GetTool('ImportExport')` `oTool.ImportIDFandMerge('C:/Files/test.brd', 'C:/Files/test.aedb.lib',` `'C:/Files/test.xml', 'Project1', 'Design12')`

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

ImportIDX

Imports and IDX model into a new Icepak project.

UI Access	**Icepak** > **Import IDX**		
Parameters	Name	Type	Description
	<*NAME*>	string	Settings
	<*Board*>	bool	Full path of .idx file
	<*Library*>	string	NA
	<*Control*>	string	NA
	<*Filters*>	list	HeightExclude2D
	<*CreateFilteredAsNonModel*>	bool	True or False
	<*NAME*>	string	definitionOverridesMap

	<NAME>	string	instanceOverridesMap
	<HighSurfThickness>	string	High surface layer thickness value and unit
	<LowSurfThickness>	string	Low surface layer thickness value and unit
	<InternalLayerThickness>	string	Internal surface layer thickness value and unit
	<NumInternalLayer>	int	Number of internal layers value
	<HighSurfaceCopper>	int	High surface percent coverage value
	<LowSurfaceCopper>	int	Low surface percent coverage value
	<InternalLayerCopper>	int	Internal layer percent coverage value
	<TraceMaterial>	string	Trace material name
	<SubstrateMaterial>	int	Substrate material name
	<CreateBoard>	bool	True or False
	<ModelBoardAsRect>	bool	True or False
	<ModelDeviceAsRect>	bool	True or False
	<Cutoff>	bool	True or False
	<ReplaceDevices>	bool	True or False
Return Value	None		

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

VB Syntax	ImportIDX (<*NAME*>, <*Board*>, <*Library*>, <*Control*>, <*Filters*>, <*CreateFilteredAsNonModel*>, <*NAME*>, <*NAME*>, <*HighSurfThickness*>, <*LowSurfThickness*>, <*InternalLayerThickness*>, <*NumInternalLayer*>, <*HighSurfaceCopper*>, <*LowSurfaceCopper*>, <*InternalLayerCopper*>, <*TraceMaterial*>, <*SubstrateMaterial*>, <*CreateBoard*>, <*ModelBoardAsRect*>, <*ModelDeviceAsRect*>, <*Cutoff*>, <*ReplaceDevices*>)
VB Example	`oDesign.ImportIDX Array("NAME:Settings", "Board:=", _"C:\Users\Model files" _` `"\PCB-00278_A.idx", "Library:=", "", "Control:=", "", "Filters:=",`

	Array(_"HeightExclude2D"), "CreateFilteredAsNonModel:=", false, Array("NAME:definitionOverridesMap"), Array("NAME:instanceOverridesMap"), "HighSurfThickness:=", _0.07mm", "LowSurfThickness:=", "0.07mm", "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.

UI Access	**File** > **Import** > **IPC2581**.
Parameters	

Name	Type	Description
<*ipcFileName*>	String	Full path of IPC2581 file.
<*outputPathName*>	String	Full path of EDB file to create during import.
<*controlFileName*>	String	Optional. Full path of XML control file. Pass empty string if none.

Return Value	None.
Python Syntax	ImportIPC(<*ipcFileName*>, <*outputPathName*>, <*controlFileName*>)
Python Example	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')

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

ImportODB

Imports an ODB++ file into a new project.

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

Python Syntax	ImportODB(<*odbFileName*>, <*outputPathName*>, <*controlFileName*>)
Python Example	oDesktop.RestoreWindow() Set oTool = oDesktop.GetTool('ImportExport')

	`oTool.ImportODB('C:/Files/test.odb', 'C:/Files/test.aedb', 'C:/Files/test.xml')`

VB Syntax	ImportODB <*odbFileName*>, <*outputPathName*>, <*controlFileName*>
VB Example	`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.

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

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')

VB Syntax	ImportXFL <*xflFileName*>, <*outputPathName*>, <*controlFileName*>
VB Example	Set oTool = oDesktop.GetTool("ImportExport") oTool.ImportXFL "C:/Files/test.xfl", "C:/Files/test.aedb", _ "C:/Files/test.xml

This page intentionally
left blank.

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:

```
Set 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()`

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

GetRunningInstanceByProcessID

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

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

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

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

GetRunningInstancesMgr

Returns the object of the Running Instances Manager.

UI Access	N/A
Parameters	Name | Type | Description None	

Return Value	Object Running instances manager object

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

VB Syntax	GetRunningInstancesMgr()
VB Example	`Set oRunningInstances = oDesktop.GetRunningInstancesMgr()`

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:

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

Dataset Script Commands:

AddDataset

DeleteDataset

EditDataset

ExportDataset

HasDataset

ImportDataset

Other Project Object Script Commands:

AddMaterial

AnalyzeAll

ChangeProperty

ClearMessages

CloneMaterial

Close

CopyDesign

CutDesign

AddDataset

Adds a dataset. This can be executed by the oProject 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]]])

VB Syntax	AddDataset <*DatasetDataArray*>
VB Example	`oProject.AddDatasetArray("NAME:ds1",` `  Array("NAME:Coordinates",` `  Array("NAME:Coordinate", "X:=", 1, "Y:=", 2,` `  Array("NAME:Coordinate", "X:=", 3, "Y:=", 4),` `  Array("NAME:Coordinate", "X:=", 5, "Y:=", 7),` `  Array("NAME:Coordinate", "X:=", 6, "Y:=", 20)))`

AddMaterial

Adds a local material.

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

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")]`

VB Syntax	AddMaterial Array("NAME:<MaterialName>", <MatProperty>, <MatProperty>, ...)
VB Example	`oDefinitionManager.AddMaterial Array(` `"permittivity:=", "2.2", "0.002")`

	oDefinitionManager.AddMaterial Array("NAME:Material2",_ "dielectric_loss_tangent:=", "44", Array("NAME:saturation_mag",_ "property_type:=", "AnisoProperty",_ "unit:=", "Gauss",_ "component1:=", "11", _ "component2:=", "22", _ "component3:=", "33"), _ "delta_H:=", "44Oe")

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()

VB Syntax	AnalyzeAll
VB Example	`oProject.AnalyzeAll`

ChangeProperty

Changes the properties of an object in the history tree.

UI Access	Right-click an object in the History Tree and select **Properties**.
Parameters	<table><tr><th>Name</th><th>Type</th><th>Description</th></tr><tr><td><*propertyArgs*></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 **Automation** tab.</td></tr></table>
Return Value	None.

Python Syntax	ChangeProperty(<*propertyArgs*>)
Python Example	**Example Changing the Position of a Box:** `oEditor.ChangeProperty(` `[` `"NAME:AllTabs",` `[` `"NAME:Geometry3DCmdTab",` `[` `"NAME:PropServers" ,` `"Box1:CreateBox:1"`

```
        ],
        [
            "NAME:ChangedProps",
            [
                "NAME:Position",
                "X:="           , "0.35in",
                "Y:="           , "0.55in",
                "Z:="           , "0in"
            ]
        ]
    ]
])
```

Example 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
```

	``]]]]) ```

VB Syntax	ChangeProperty <*propertyArgs*>
VB Example	**Example Changing the Position of a Box:** ``` oEditor.ChangeProperty Array("NAME:AllTabs", Array("NAME:Geometry3DCmdTab", Array("NAME:PropServers" , "Box1:CreateBox:1"), Array("NAME:ChangedProps", Array("NAME:Position", "X:=" , "0.35in", "Y:=" , "0.55in", "Z:=" , "0in")))) ``` **Example Changing a Box's Material and Wireframe Display:** ``` oEditor.ChangeProperty(Array("NAME:AllTabs", Array("NAME:Geometry3DAttributeTab", Array("NAME:PropServers", "Box1"), ```

```
        Array("NAME:ChangedProps",
            Array("NAME:Material",
            "Value:=", "\"vacuum\""
            ),
            Array("NAME:Display Wireframe",
            "Value:=", True
            )
        )
    )
)
```

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()`

VB Syntax	ClearMessages
VB Example	`oProject.ClearMessages`

CloneMaterial

Clones a local material.

UI Access	N/A
Parameters	<table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><*matName*></td><td>String</td><td>Name of existing material.</td></tr><tr><td><*newName*></td><td>String</td><td>Name for newly cloned material.</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")`

VB Syntax	CloneMaterial <*matName*>, <*newName*>
VB 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()`

VB Syntax	Close
VB Example	`oProject.Close`

CopyDesign

Copies a specified design.

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

Return Value	None.

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

VB Syntax	CopyDesign <*DesignName*>
VB 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**.		
Parameters	Name	Type	Description
	<*DesignName*>	String	Name of the design.
Return Value	None.		

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

VB Syntax	CutDesign <*DesignName*>
VB Example	`oProject.CutDesign "SimplorerDesign1"`

DeleteDataset

Deletes a specified dataset. This can be executed by the oProject 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')`

VB Syntax	DeleteDataset <*DatasetName*>
VB Example	`oProject.DeleteDataset "$ds1"`

DeleteDesign

Deletes a specified design in the project.

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

Python Syntax	DeleteDesign (<*DesignName*>)
Python Example	`oProject.DeleteDesign("CircuitDesign2")`

VB Syntax	DeleteDesign <*DesignName*>
VB Example	`oProject.DeleteDesign "CircuitDesign2"`

DeleteToolObject

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

UI Access	N/A
Parameters	Name | Type | Description

	<*ObjectName*>	String	Name of the tool object.
Return Value	None.		

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

VB Syntax	DeleteToolObject <*ObjectName*>
VB Example	`oProject.DeleteToolObject "object1"`

EditDataset

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

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

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

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

VB Syntax	EditDataset <*OriginalName*> <*DatasetDataArray*>
VB Example	oProject.EditDataset "ds1" Array("NAME:ds2",_ Array("NAME:Coordinates",Array("NAME:Coordinate", "X:=", 1, "Y:=", 2), Array("NAME:Coordinate", "X:=", 3, "Y:=", 4)))

EditMaterial

Modifies an existing material.

<table>
<tr><td>UI Access</td><td>View/Edit Materials 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><OriginalName></td><td>String</td><td>Name of the material before editing.</td></tr>
<tr><td><MatProperties></td><td>Array</td><td>Structured array containing material properties:
<pre>
[
  "NAME:<New material name>",
  "CoordinateSystemType:=", <string>,
  "BulkOrSurfaceType:=" , <integer>,
  [
    "NAME:PhysicsTypes",
    "set:=" , <array containing string physics types>
  ],
  <Optional ModifierDataArray>,
  "permeability:=" , <string containing value>,
  "conductivity:=" , <string containing value>,
  "thermal_conductivity:=", <string containing value>,
  "mass_density:=" , <string containing value>,
  "specific_heat:=" , <string containing value>,
</pre>
</td></tr>
</table>
</td></tr>
</table>

		"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
        ]
     ]
  ])
```

VB Syntax	EditMaterial <*OriginalName*>, <*MatProperties*>
VB Example	**Without Modifiers:** `oDefinitionManager.EditMaterial "alumina_92pct",`

```
Array(
"NAME:alumina_92pct",
"CoordinateSystemType:=", "Cartesian",
"BulkOrSurfaceType:=", 1,
Array("NAME:PhysicsTypes",
"set:=", Array("Electromagnetic","Thermal","Structural")),
"permittivity:=", "9.3",
"dielectric_loss_tangent:=", "0.008",
"thermal_conductivity:=", "26",
"mass_density:=", "3720",
"specific_heat:=", "790",
"youngs_modulus:=", "267000000000",
"poissons_ratio:=", "0.26",
"thermal_expansion_coeffcient:=", "7.2e-006"
)
```

With Thermal Modifier:

```
oDefinitionManager.EditMaterial "copper",
  Array("NAME:copper",
  "CoordinateSystemType:=", "Cartesian",
  "BulkOrSurfaceType:=" , 1,
    Array("NAME:PhysicsTypes",
```

```
    "set:=" , Array("Electromagnetic","Thermal","Structural")),
  Array("NAME:ModifierData",
    Array("NAME:ThermalModifierData",
    "modifier_data:=" , "thermal_modifier_data",
      Array("NAME:all_thermal_modifiers",
        Array("NAME:one_thermal_modifier",
        "Property::=" , "permittivity",
        "Index::=" , 0,
        "prop_modifier:=" , "thermal_modifier",
        "use_free_form:=" , True,
        "free_form_value:=" , "if(Temp > 1000cel, 1, if(Temp < -273.15cel, 1, 1))"
        )
      )
    )
  ),
  "permeability:=" , "0.999991",
  "conductivity:=" , "58000000",
  "thermal_conductivity:=" , "400",
  "mass_density:=" , "8933",
  "specific_heat:=" , "385",
```

```
  "youngs_modulus:=" , "120000000000",
  "poissons_ratio:=" , "0.38",
  "thermal_expansion_coefficient:=" , "1.77e-05"
)
```

ExportDataset

Exports a dataset to a named file. This can be executed by the oProject 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')`

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

ExportMaterial

Exports a local material to a library.

UI Access	**Export to Library** command in the material editor.
Parameters	<table><tr><th>Name</th><th>Type</th><th>Description</th></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>
Return Value	None.

Python Syntax	ExportMaterial (<*ExportData*>, <*LibraryLocation*>)
Python Example	`oDefinitionManager.ExportMaterial (["NAME:mylib",_` `"Material1", "Material2", "Material3"], "PersonalLib")`

VB Syntax	ExportMaterial <*ExportData*>, <*LibraryLocation*>
VB Example	`oDefinitionManager.ExportMaterial Array("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()`

VB Syntax	GetActiveDesign
VB Example	`Set 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()`

VB Syntax	GetArrayVariables
VB Example	`oProject.GetArrayVariables` `oDesign.GetArrayVariables`

GetChildNames [Project]

Returns the names of the project's child objects.

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

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

VB Syntax	GetChildNames (<*type*>)

VB 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	Name | Type | Description <*path*> | String | The path may include multiple generations (for example, "designObject/moduleObj/SetupObject"). See Object Path.
Return Value	Object of a found child.

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

VB Syntax	GetChildObject(<*path*>)
VB Example	`Set oProject = oDesktop.GetActiveproject()` `Set oDesign = oProject.GetChildObject(“TeeModel”)` `Set oVariable = oProject.GetChildObject("VariableName")` `Set 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()`

VB Syntax	GetChildTypes
VB Example	`oProject.GetChildTypes`

GetConfigurableData (Project)

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

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

GetDefinitionManager

Gets the `DefinitionManager` object.

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

Python Syntax	GetDefinitionManager()
Python Example	`oDefinitionManager = oProject.GetDefinitionManager()`

VB Syntax	GetDefinitionManager
VB Example	`Set 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()`

VB Syntax	GetDependentFiles
VB Example	`files = oProject.GetDependentFiles`

GetDesign

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

UI Access	N/A

Parameters	<table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><*designName*></td><td>String</td><td>Name of the design.</td></tr></table>
Return Value	Object of the specified design.

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

VB Syntax	GetDesign <*designName*>
VB 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()`

VB Syntax	GetDesigns
VB 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()`

VB Syntax	GetEDBHandle

VB 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()`

VB Syntax	GetLegacyName
VB 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()`

VB Syntax	GetName
VB 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()`

VB Syntax	GetObjPath
VB 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()`

VB Syntax	GetPath
VB 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.

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

Python Syntax	GetPropEvaluatedValue (*<PropName>*)
Python Example	`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.

UI Access	N/A
Parameters	Name | Type | Description *<includeReadOnly>* | Boolean | (Optional)

			• True – Include read only props. • False – Do not include read only props.
Return Value	Empty array.		

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

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

Python Syntax	GetPropSIValue (<*PropName*>)
Python Example	`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.

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

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

VB Syntax	GetPropValue <*propPath*>
VB Example	`oProject.GetPropValue "TeeModel/offset"`

GetProperties

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

UI Access	N/A
Parameters	

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

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

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

VB Syntax	GetProperties <*PropTab*>, <*PropServer*>
VB Example	`oEditor.GetProperties "PassedParameterTab", "k"`

GetPropertyValue

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

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

UI Access	N/A
Parameters	See table below

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 `Comp‡nst@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")` `...`

VB Syntax	GetPropertyValue (<*PropTab*>, <*PropServer*>, <*PropName*>)
VB 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()`

VB Syntax	GetTopDesignList
VB 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`.

UI Access	N/A

Parameters	<table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><*VarName*></td><td>String</td><td>Name of the variable to access.</td></tr></table>
Return Value	String represents the value of the variable.

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

VB Syntax	GetVariableValue <*VarName*>
VB 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()`

VB Syntax	GetVariables
VB Example	`oProject.GetVariables` `oDesign.GetVariables`

ImportDataset

Imports a dataset from a named file. This can be executed by the oProject 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.

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

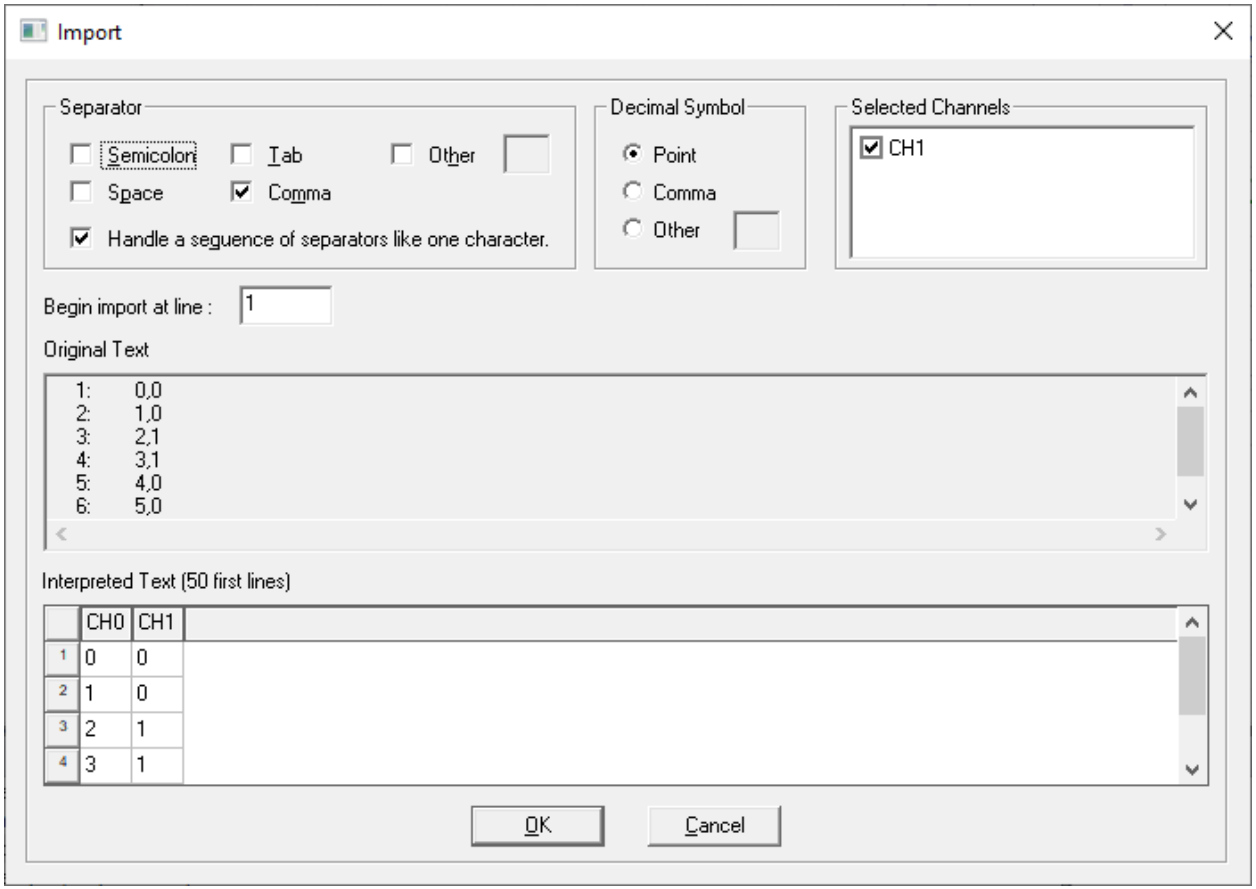
Python Syntax	ImportDataset (<*datasetFileFullPath*>,<*optionalDatasetName*>)
Python Example	`oProject.ImportDataset('e:\tmp\dsdata.tab')` `oProject.ImportDataset('e:\tmp\dsdata.tab', 'MyDatasetName')`

VB Syntax	ImportDataset <*datasetFileFullPath*>, <*optionalDatasetName*>
VB Example	`oProject.ImportDataset "e:\tmp\dsdata.tab"` `oProject.ImportDataset "e:\tmp\dsdata.tab", "MyDatasetName"`

Note About File Types:

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

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



InsertDesign

Inserts a new design in the project. For Circuit and HFSS 3D Layout scripts, the last argument will always be empty.

UI Access	**Project** > **[Insert Circuit Design / Insert Circuit Netlist / Insert HFSS 3D Layout Design]**.
Parameters	

Name	Type	Description
<*DesignType*>	String	Design type. Valid types: "Circuit", "Nexxim Circuit", "System" , "Nexxim Netlist", "HFSS3D".
<*DesignName*>	String	Design name.
<*TechnologyFile*>	String	The path to the Circuit technology file to be used in the design. Use a pair of empty double quotes ("") for none.
<*SubcircuitID*>	String	(Optional) Must be preceded by a colon if included along with the Technology File name. No colon is necessary when the subcircuit ID is omitted.
""	None	Empty argument.

Return Value	None.

Python Syntax	InsertDesign (<*DesignType*>, <*DesignName*>, <*TechnologyFile*>:<*SubCircuitID*>,'')
Python Example	`oProject.InsertDesign('Nexxim Circuit','MyDesigner', 'C:\\Program Files\\AnsysEM\\Designer\\', '')`

VB Syntax	InsertDesign <*DesignType*>, <*DesignName*>, <*TechnologyFile*>:<*SubCircuitID*>,""
VB Example	`oProject.InsertDesign "Nexxim Circuit","MyDesigner", "C:\Program Files\AnsysEM\Designer\", ""`

InsertDesignWithWorkflow

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

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

Python Syntax	InsertDesignWithWorkflow(<*type*>, <*workflowName*>, <*specName*>, <*fileName*>, <*libLoc*>, <*stationaryPath*>)
Python Example	`oProject.InsertDesignWithWorkflow ("Circuit Design", "Serial Design", "PCIe3 Stressed", "LongChannel", "SysLib", "C:\\Program Files\\AnsysEM\\v242\\Win64\\syslib\\MS - RT_duroid 6010 (Er=10.2) 0.010 inch, 0.5 oz copper.asty")`

VB Syntax	InsertDesignWithWorkflow <*type*>, <*workflowName*>, <*specName*>, <*fileName*>, <*libLoc*>, <*stationaryPath*>
VB Example	`oProject.InsertDesignWithWorkflow "Circuit Design", "Serial Design", _ "PCIe3 Stressed", "LongChannel", "SysLib", _ "C:\Program Files\AnsysEM\v242\Win64\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()`

VB Syntax	Paste

VB 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()`

VB Syntax	Redo
VB 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()`

VB Syntax	RemoveAllUnusedDefinitions
VB Example	`oProject.RemoveAllUnusedDefinitions`

RemoveMaterial

Removes a material from a library.

UI Access	**Remove Material(s)** command in the material editor
Parameters	See table below
Return Value	None.

Name	Type	Description
<*MaterialName*>	String	Name of the material to be removed.
<*IsProjectMaterial*>	Boolean	If True, assumes the material is a project material. The last two parameters will be ignored. If False, the material is not a project material.
<*LibraryName*>	String	Name of the user or personal library where the material resides.
<*LibraryLocation*>	String	Location of library. Valid options:"UserLib" or "PersonalLib".

Python Syntax	RemoveMaterial (<*MaterialName*>, <*IsProjectMaterial*>, <*LibraryName*>, <*LibraryLocation*>)

Python Example	`oDefinitionManager.RemoveMaterial ([` `"Material1", false, "mo0907","UserLib"])`

VB Syntax	RemoveMaterial <*MaterialName*>, <*IsProjectMaterial*>, <*LibraryName*>, <*LibraryLocation*>
VB Example	`oDefinitionManager.RemoveMaterial` `"Material1", false, "mo0907","UserLib"`

RemoveUnusedDefinitions

Removes any unused project definitions.

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

Python Syntax	RemoveUnusedDefinitions(<*Definitions*>)
Python Example	`oProject.RemoveUnusedDefinitions(` `  [` `  [`

```
        "NAME:Materials",
        "Al-Extruded"
      ],
      [
        "NAME:SurfaceMaterials",
        "Steel-oxidised-surface"
      ]
    ])
```

VB Syntax	RemoveUnusedDefinitions <*Definitions*>
VB Example	`oProject.RemoveUnusedDefinitions Array(Array("NAME:Materials", "Al-Extruded"), Array ("NAME:SurfaceMaterials", _ "Steel-oxidised-surface"))`

Rename

Renames the project and saves it. Similar to SaveAs().

UI Access	**Edit** > **Rename**.

Parameters

Name	Type	Description
<*NewName*>	String	Desired name of the project. The path is optional.
<*OverWriteOk*>	Boolean	• True - overwrite the file on disk if it exists. • False - prevent overwrite.

Return Value	None.

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

VB Syntax	Rename <*NewName*>,<*OverWriteOK*>
VB Example	`oProject.Rename "c:\projects\MyProject.aedt", true`

RunToolkit

Runs a Python toolkit script, applying it to the active design. The toolkit script itself may have prerequisites, such as sweeps defined, or specific model characteristics, such as port definitions.

> **Important:**
>
> Full scripting for cable modeling is not supported and no arguments are allowed in the script. The **RunToolkit** command will not automatically create a cable bundle, but will simply open the **Cable Modeling** dialog box. You will have to manually input your parameters.

UI Access	[**Circuit**] > **Toolkit** > [**Script Name**].
Parameters	

Name	Type	Description
<*LibraryType*>	String	Name of the library in which the script is located.
<*ToolkitName*>	String	Name of the Python script.

	<*ToolkitArg*>	Array	Structured array containing arguments required by the toolkit script.
Return Value	User-defined solution(s) and/or output(s).		

Python Syntax	RunToolkit(<*LibraryType*>, <*ToolkitName*>, <*ToolkitArg*>)
Python Example	`oDesign.RunToolkit("SysLib", "Hearing Aid Compliance", [])`

VB Syntax	RunToolkit <*LibraryType*>, <*ToolkitName*>, <*ToolkitArg*>
VB Example	`oDesign.RunToolkit "SysLib", "Wavelength Calculator", Array()` `oDesign.RunToolkit "SysLib", "Hearing Aid Compliance", Array()`

Save

Saves the active project.

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

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

VB Syntax	Save
VB 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.

UI Access	**File** > **Save As**.
Parameters	

Name	Type	Description
<*NewName*>	String	The desired name of the project, with directory and extension.
<*OverWriteOK*>	Boolean	True to overwrite the file of the same name, if it exists. False to prevent overwrite.
<*DefaultAction*>	String	For Schematic/Layout projects only. Otherwise omit. See note below. Valid actions: ef_overwrite , ef_copy_no_overwrite, ef_make_path_absolute, or empty string.
<*OverwriteActions*>	Array	For Schematic/Layout projects only. Otherwise omit. See note below. Structured array: Array("Name: <Action>", <FileName>, <FileName>, ...) Valid actions: ef_overwrite , ef_copy_no_overwrite, ef_make_path_absolute, or empty string.

Return Value	None.

Python Syntax	For non-Schematic/Layout project: SaveAs (<*NewName*> <*OverWriteOK*>) For Schematic/Layout project: SaveAs (<*NewName*> <*OverWriteOK*> <*DefaultAction*> <*OverrideActions*>)
Python Example	`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']]])`

VB Syntax	For non-Schematic/Layout project: SaveAs <*NewName*> <*OverWriteOK*> For Schematic/Layout project: SaveAs <*NewName*> <*OverWriteOK*> <*DefaultAction*> <*OverrideActions*>
VB Examples	`oProject.SaveAs "D:/projects/project1.aedt", true` `----` `oProject.SaveAs` `  "F:\Designer Projects\TA33097\HighSpeedChannel.aedt", true, "ef_overwrite", Array ("NAME:OverrideActions",` `Array("NAME:ef_copy_no_overwrite", Array("NAME:Files","$PROJECTDIR/circuit_models.inc")),` `Array("NAME:ef_make_path_absolute", Array("NAME:Files","$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	Name | Type | Description *<projectName>* | String | The desired name of the project, with directory and extension.
Return Value	None.

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

VB Syntax	SaveAsStandAloneProject <*projectName*>
VB Examples	`oProject.SaveAsStandAloneProject "D:/projects/project1.aedt"`

SaveProjectArchive

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

UI Access	**File** > **Archive**.
Parameters	

Name	Type	Description
<*archiveFilePath*>	String	Path to archived file.
<*IncludeExternalFiles*>	Boolean	True to include external files; False to exclude.
<*IncludeResultsFiles*>	Boolean	True to include simulation files associated with the project; False to exclude.
<*AdditionalFiles*>	Array	Additional specified files to include.
<*ArchiveNotes*>	String	String describe the archive.

Return Value	None.

Python Syntax	SaveProjectArchive(<*archivefilepath*>, <*IncludeExternalFiles*>, <*IncludeResultsFiles*>, <*AdditionalFiles*>, <*ArchiveNotes*>)
Python	`oProject.SaveProjectArchive("C:\\Users\\Documents\\Ansoft\\Project27.aedtz", True,`

Example	`False, [], ""`)

VB Syntax	SaveProjectArchive <*archivefilepath*>, <*IncludeExternalFiles*>, <*IncludeResultsFiles*>, <*AdditionalFiles*>, <*ArchiveNotes*>
VB Example	`oProject.SaveProjectArchive "C:\Documents\OptimTee.aedtz", true,  false, Array(), "My notes"`

SetActiveDefinitionEditor

Obtains a specified definition editor.

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

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

VB Syntax	SetActiveDefinitionEditor <*EditorName*>, <*DefinitionName*>
VB Example	`oProject.SetActiveDefinitionEditor "SymbolEditor",_` `"Simplorer Elements\Basic Elements\Circuit\Passive Elements:R"`

SetActiveDesign

Sets a design to be the active design.

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

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

VB Syntax	SetActiveDesign <*DesignName*>
VB Example	`Set oDesign = oProject.SetActiveDesign "SimplorerDesign2"`

SetPropValue [Project]

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

UI Access	Edit Properties on ProjectTree objects.
Parameters	<table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><*propPath*></td><td>String</td><td>A child object's property path. See: Property Function.</td></tr><tr><td><*newValue*></td><td>String</td><td>New property value.</td></tr></table>
Return Value	Boolean: • **True** – property found. • **False** – property not found.

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

VB Syntax	SetPropValue <*propPath*>, <*newValue*>
VB 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><td>UI Access</td><td>N/A</td></tr>
<tr><td>Parameters</td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><propTab></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><propServer></td><td>String</td><td>An object identifier, generally returned from another script method, such as Comp‡nst@R;2;3</td></tr>
<tr><td><propName></td><td>String</td><td>Name of the property.</td></tr>
<tr><td><propValue></td><td>String</td><td>The value for the property</td></tr>
</table>
</td></tr>
<tr><td>Return Value</td><td>None.</td></tr>
</table>

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

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

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')`

VB Syntax	SetVariableValue <*VarName*>, <*VarValue*>
VB 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()`

VB Syntax	SimulateAll
VB 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()`

VB Syntax	Undo
VB 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()`

VB Syntax	UpdateDefinitions
VB Example	`oProject.UpdateDefinitions`

This page intentionally
left blank.

7 - Design Object Script Commands

Design object commands should be executed by the oDesign object.

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

For example:

```
Set oDesign = oProject.SetActiveDesign("CircuitDesign1")
```

Conventions Used in this Chapter

`<ModuleName>` is a placeholder for any one of the following modules:

- Analysis Module – "AnalysisSetup"
- Data Module – “DataBlock”
- Design Verification Module – "DV"
- Optimetrics Module – "Optimetrics"
- Output Variable Module – "OutputVariable"
- Reporter Module – "ReportSetup"
- User Defined Documents Module – "UserDefinedDocuments"
- User Defined Solutions Module – "UserDefinedSolutionModule"

AddDataset

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

<table>
<tr><td>UI Access</td><td colspan="3">Project > Datasets > Add.</td></tr>
<tr><td rowspan="2">Parameters</td><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><<i>DatasetDataArray</i>></td><td>Array</td><td>Array("NAME:<DatasetName>",
Array("NAME:Coordinates", <CoordinateArray>,</td></tr>
</table>

	<table><tr><td></td><td></td><td><CoordinateArray>, ...)</td></tr><tr><td><*DatasetName*></td><td>String</td><td>Name of the dataset.</td></tr><tr><td><*CoordinateArray*></td><td>Array</td><td>Array("NAME:Coordinate", "X:=", <double>, "Y:=",<double>)</td></tr></table>
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

	``]]]) ```

VB Syntax	AddDataset <*DatasetDataArray*>
VB Example	``` oProject.AddDatasetArray("NAME:ds1", Array("NAME:Coordinates", Array("NAME:Coordinate", "X:=", 1, "Y:=", 2, Array("NAME:Coordinate", "X:=", 3, "Y:=", 4), Array("NAME:Coordinate", "X:=", 5, "Y:=", 7), Array("NAME:Coordinate", "X:=", 6, "Y:=", 20))) ```

AddDesignVariablesForDynamicLink

Creates design variables based on any in a Circuit, HFSS, 3D Layout, Q3D, or 2D extractor project that is connected to a Circuit or 3D Layout design through a dynamic link.

UI Access	• In the **Schematic Editor** or **Layout Editor**, right click the new instance of the linked design and click **Add Design Variables for Dynamic Link**. • In the **Project Manager**, right click on [**Project Name**] > [**Design Name**] > [**Dynamic Link Design**] and click **Add Design Variables for Dynamic Link**.
Parameters	

Name	Type	Description
<*InstanceID*>	String	Instance ID for the dynamic link component

Return Value	None

Python Syntax	AddDesignVariablesForDynamicLink (<*InstanceID*>)
Python Example	`oDesign = oProject.SetActiveDesign("Circuit1")` `oDesign.AddDesignVariablesForDynamicLink ("6")`

VB Syntax	AddDesignVariablesForDynamicLink <*InstanceID*>
VB Example	`Set oDesign = oProject.SetActiveDesign("Circuit1")` `oDesign.AddDesignVariablesForDynamicLink  "6"`

AddDynamicLink

Adds a dynamic link component to a Circuit or HFSS 3D Layout design. Circuit designs can accept dynamic links from HFSS, Q3D, 2D Extractor, and SIwave.

> **Note:**
>
> HFSS 3D Layout designs can only accept dynamic links from HFSS.

UI Access	• Open the Circuit or HFSS 3D Layout design to which you want to add the link. On the **Component Libraries** menu, open the **Symbols** tab, and then click **Import Models**. Select the appropriate icon from the panel of icons. The **Import Components** dialog opens. • If the source for a dynamic link is an HFSS design, in the Project Manager drag and drop the design onto a

<table>
<tr><td></td><td>Circuit or HFSS 3D Layout design to create the link.</td></tr>
<tr><td>Parameters</td><td>
<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td><designName></td><td>String</td><td>Design source for the dynamic link. Optional, defaults to the first design in the project pointed to by projectPath.</td></tr>
<tr><td><projectPath></td><td>String</td><td>Project source for the dynamic link. Optional, defaults to the current active project.</td></tr>
<tr><td><linkName></td><td>String</td><td>Name applied to the dynamic link. Optional, defaults to the designName.</td></tr>
<tr><td><solutionName></td><td>String</td><td>Solution in source design which will be referenced by the dynamic link. Optional, defaults to the first setup:sweep.</td></tr>
<tr><td><transLinePort ></td><td>String</td><td>For HFSS designs only, port name for transmission lines. It is created only if it is not empty. Optional, defaults to “”. This value is ignored for non-HFSS design types.</td></tr>
<tr><td><ReduceMatrix></td><td>String</td><td>For Q3D and 2D extractor designs, reduce matrix which will be referenced by the dynamic link. This is ignored for other design types. Optional, defaults to “Original”.</td></tr>
<tr><td><EnableHFSSCableModeling></td><td>Boolean</td><td>For 2D extractor designs only, flag to create cable models. This is ignored for other designs types. Optional, defaults to false.</td></tr>
<tr><td><Description></td><td>String</td><td>Description of the dynamic link's source. Optional.</td></tr>
</table>
</td></tr>
<tr><td>Return Value</td><td>None.</td></tr>
</table>

Python Syntax	AddDynamicLink (<*designName*>, <*projectPath*>, <*linkName*>, <*solutionName*>, <*transLinePort*>, <*reducedMatrix*>, <*enableCableModeling*>, <*description*>)
Python Example	`oDesign = oProject.SetActiveDesign("Circuit1")` `oDesign.AddDynamicLink("TeeModel","$PersonalLib\\TeeWrongFile.aedt","TeeLink","Setup1 : LastAdaptive","Port1","",False,"desc")`

VB Syntax	AddDynamicLink <*designName*>, <*projectPath*>, <*linkName*>, <*solutionName*>, <*transLinePort*>, <*reducedMatrix*>, <*enableCableModeling*>, <*description*>
VB Example	`Set oDesign = oProject.SetActiveDesign("Circuit1")` `oDesign.AddDynamicLink "TeeModel", "$PersonalLib\\TeeWrongFile.aedt", "TeeLink", "Setup1 : LastAdaptive", "Port1", "", False, "desc"`

AddModelingProperties

Use: Add a modeling property to a design

Command: None

Syntax: AddModelingProperties <design>

Return Value: None

Parameters: <design>

Type: string

VB Example: oDesign.AddModelingProperties <design>

Analyze

Solves a single solution setup and all of its frequency sweeps.

UI Access	Right-click a solution setup in the project tree, and select **Analyze**.
Parameters	<table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><*setup*></td><td>String</td><td>Setup name.</td></tr><tr><td>isBlocking</td><td>Boolean</td><td>An optional arg that defaults to `true`. If it's `false`, the command immediately returns while the analysis or analyses run in the background. The status of the</td></tr></table>

	analyses can be checked with the `AreThereSimulationsRunning` command.
Return Value	Integer value: • **0** – Success • **Else** – Error

Python Syntax	Analyze (<*setup*>)
Python Example	`oDesign.Analyze("Setup1", isBlocking)`

VB Syntax	Analyze <*setup*>
VB Example	`oDesign.Analyze "Setup1"`

AnalyzeAll [design]

Runs all solution setups and Optimetrics setups for the current design instance.

UI Access	N/A		
Parameters	Name	Type	Description
	isBlocking	Boolean	An optional arg that defaults to `true`. If it's `false`, the command immediately returns while the analysis or analyses run in the background. The status of the analyses can be checked with the `AreThereSimulationsRunning` command.
Return Value	None		

Python Syntax	AnalyzeAll()
Python Example	`oDesign.AnalyzeAll(isBlocking)`

VB Syntax	AnalyzeAll
VB Example	`oDesign.AnalyzeAll`

AnalyzeAllNominal

Runs all defined setups.

UI Access	Right-click **Analysis** in the project tree, and select **Analyze All**.
Parameters	Name: isBlocking; Type: Boolean; Description: An optional arg that defaults to `true`. If it's `false`, the command immediately returns while the analysis or analyses run in the background. The status of the analyses can be checked with the `AreThereSimulationsRunning` command.
Return Value	None

Python Syntax	AnalyzeAllNominal()
Python Example	`oDesign.AnalyzeAllNominal()`

VB Syntax	AnalyzeAllNominal
VB Example	`oDesign.AnalyzeAllNominal`

ChangePortProperty [Nexxim]

Use: Change a port property

Command: None

Syntax: ChangePortProperty <Port Name> <Port Info> <Port Properties>

Return Value: None

Parameters: `<Port Name> - Type: string`

<Port Info> - Type: array

<Port Properties> - Type:array

VB Example:

```
oDesign.ChangePortProperty "Port2", Array("NAME:Port2", "IIPortName:=", "Port2", "SymbolType:=",
_
0), Array(Array("NAME:Properties", Array("NAME:NewProps", Array("NAME:term", "PropType:=", _
"TextProp", "OverridingDef:=", true, "Value:=", "Model1")), Array("NAME:ChangedProps", Array
("NAME:TerminationData", "Value:=", _
"Zo"), Array("NAME:pnum", "Value:=", "2"), Array("NAME:noisetemp", "Value:=", "16.85")))
```

Python Syntax	ChangePortProperty(<portName> [<portInfo>] [<portProperties>])
Python Exampl-	`oDesign.ChangePortProperty( "Port2", ["NAME:Port2","IIPortName:=", "Port2", "Sym-bolType:=",0,`

e	[["NAME:Properties", ["NAME:NewProps",["NAME:term", "PropType:=","TextProp", "OverridingDef:=", true, "Value:=","Model1"]], ["NAME:ChangedProps", ["NAME:TerminationData","Value:=", "Zo"], ["NAME:pnum", "Value:=", "2"], ["NAME:noisetemp", "Value:=", "16.85"]]])

ChangeProperty

Changes the properties of an object in the history tree.

UI Access	Right-click an object in the History Tree and select **Properties**.
Parameters	<table><tr><th>Name</th><th>Type</th><th>Description</th></tr><tr><td><*propertyArgs*></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 **Automation** tab.</td></tr></table>
Return Value	None.

Python Syntax	ChangeProperty(<*propertyArgs*>)
Python Example	**Example Changing the Position of a Box:** `oEditor.ChangeProperty(` `[` `"NAME:AllTabs",` `[` `"NAME:Geometry3DCmdTab",` `[`

```
            "NAME:PropServers" ,
            "Box1:CreateBox:1"
        ],
        [
            "NAME:ChangedProps",
            [
                "NAME:Position",
                "X:="          , "0.35in",
                "Y:="          , "0.55in",
                "Z:="          , "0in"
            ]
        ]
    ]
])
```

Example 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
```

	``]]]]) ```

VB Syntax	ChangeProperty <*propertyArgs*>
VB Example	**Example Changing the Position of a Box:** ``` oEditor.ChangeProperty Array("NAME:AllTabs", Array("NAME:Geometry3DCmdTab", Array("NAME:PropServers" , "Box1:CreateBox:1"), Array("NAME:ChangedProps", Array("NAME:Position", "X:=" , "0.35in", "Y:=" , "0.55in", "Z:=" , "0in")))) ``` **Example Changing a Box's Material and Wireframe Display:** ``` oEditor.ChangeProperty(Array("NAME:AllTabs", Array("NAME:Geometry3DAttributeTab", Array("NAME:PropServers", ```

```
        "Box1"
        ),
        Array("NAME:ChangedProps",
            Array("NAME:Material",
            "Value:=", "\"vacuum\""
            ),
            Array("NAME:Display Wireframe",
            "Value:=", True
            )
        )
    )
)
```

ChangeSourceProperty [Nexxim]

Use: Change a source property

Command: None

Syntax: ChangeSourceProperty <Source_Name> <Source_Data>

Return Value: None

Parameters: <Source_Name>

Type: string

<Source_Data>

Type: array

VB Example:

```
oDesign.ChangeSourceProperty "Sinusoidal1", Array(Array("NAME:Properties",
Array("NAME:NewProps", _
```

```
Array("NAME:noise", "PropType:=", "TextProp", "OverridingDef:=", true, "UserDef:=", true,
"Value:=", "SourceNoise1"), _
Array("NAME:mod", "PropType:=", "TextProp", "OverridingDef:=", true, "UserDef:=", true,
"Value:=", "SourceModulation1"))), _
Array("NAME:DataBlocks", Array("NAME:NewDataBlk", Array("NAME:DataBlock", "Name:=",
"SourceNoise1", _
"noisedata:=", Array("FDEV=1Hz ", "RFUP=1 ", "RFLO=2 ", "NCOR=2 2 ")), _
Array("NAME:DataBlock", "Name:=", "SourceModulation1", "ModulationSourceDatas:=", Array( _
"IS95", "Butterworth", "Br=1.2288MHz", "Dly=0.5", "Iasc=1", "Qasc=1")))))
```

Python Syntax	ChangeSourceProperty(<sourceName>,[<Properties])
Python Example	`oDesign.ChangeSourceProperty( "Sinusoidal1",` `  [` `  [` `  "NAME:Properties",` `  [` `  "NAME:NewProps",` `  [` `  "NAME:noise",` `  "PropType:=", "TextProp",`

```
            "OverridingDef:=", true,
            "UserDef:=", true,
            "Value:=", "SourceNoise1"
          ],
          [
            "NAME:mod",
            "PropType:=", "TextProp",
            "OverridingDef:=", true,
            "UserDef:=", true,
            "Value:=", "SourceModulation1"
          ]
        ]
      ],
      [
        "NAME:DataBlocks",
        [
          "NAME:NewDataBlk",
          [
            "NAME:DataBlock",
            "Name:=", "SourceNoise1",
            "noisedata:=",
            [
```

```
                    "FDEV=1Hz ",
                    "RFUP=1",
                    "RFLO=2 ",
                    "NCOR=2 2 "
                ]
            ],
            [
              "NAME:DataBlock",
              "Name:=", "SourceModulation1"
              "ModulationSourceDatas:=",
               [
                   "IS95",
                   "Butterworth",
                   "Br=1.2288MHz",
                   "Dly=0.5",
                   "Iasc=1",
                   "Qasc=1"
               ]
            ]
        ]
    ]
```

	])

CopyEyeItemAsCommand

Use: Make a QuickEye copy of a VerifEye analysis, or a VerifEye copy of a QuickEye analysis.

Command: Select an analysis in the Project tree, then right-click and select **Copy As QuickEye** or **Copy As VerifEye**.

Syntax: CopyEyeItemAsCommand <itemPathList>

Return Value: Copy of the analysis.

Parameters: <itemPathList>

Type: Array of strings // Type of analysis from which to copy.

VB Example:

```
oDesign.CopyEyeItemAsCommand Array("eye_diagram_sch|Nexxim1|Analysis|VerifEyeAnalysis")
oDesign.CopyEyeItemAsCommand Array("eye_diagram_sch|Nexxim1|Analysis|QuickEyeAnalysis")
```

Python Syntax	CopyEyeItemAsCommand <itemPathList>
Python Example	`oDesign.CopyEyeItemAsCommand( ["eye_diagram_sch|Nexxim1|Analysis|VerifEyeAnalysis"])` `oDesign.CopyEyeItemAsCommand( ["eye_diagram_sch|Nexxim1|Analysis|QuickEyeAnalysis"])`

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

VB Example: oDesign.CopyItemCommand Array("Project1|Nexxim1|Analysis|DCAnalysis2")

CreateReport

Creates a new report with a single trace and adds it to the **Results** branch in the project tree.

<table>
<tr><td>UI Access</td><td>Right-click on Results > Create [Type] Report</td></tr>
<tr><td>Parameters</td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><ReportName></td><td>String</td><td>Name of report.</td></tr>
<tr><td><ReportType></td><td>String</td><td>Type of report.
Possible values are:
"Modal S Parameters" - Only for Driven Modal solution-type problems with ports.
"Terminal S Parameters" - Only for Driven Terminal solution-type problems with ports.
"Eigenmode Parameters" - Only for Eigenmode solution-type problems.
"Fields"
"Far Fields" - Only for problems with radiation or PML boundaries.
"Near Fields" - Only for problems with radiation or PML boundaries.</td></tr>
</table>
</td></tr>
</table>

		"Emission Test"
<*DisplayType*>	String	Type of display. If ReportType is "Modal S Parameters", "Terminal S Parameters", or "Eigenmode Parameters", then set to one of the following: `"Rectangular Plot", "Polar Plot", "Radiation Pattern", "Smith Chart", "Data Table", "3D Rectangular Plot", "3D Rectangular Bar Plot" or "3D Polar Plot".` If <ReportType> is "Fields", then set to one of the following: `"Rectangular Plot", "Polar Plot", "Radiation Pattern", "Rectangular Contour Plot", "Data Table", "3D Rectangular Plot", or "3D Rectangular Bar Plot".` If <ReportType> is "Far Fields" or "Near Fields", then set to one of the following: `"Rectangular Plot", "Radiation Pattern", "Rectangular Contour Plot", "Data Table", "3D Rectangular Plot", "3D Rectangular Bar Plot" or "3D Polar Plot"` If <ReportType> is "Emission Test", then set to one of the following: `"Rectangular Plot" or "Data Table"`
<*SolutionName*>	String	Name of the solution as listed in the **Modify Report** dialog box.
<*ContextArray*>	Array	Context for which the expression is being evaluated. This can be an empty string if there is no context. `Array("Domain:=", <DomainType>)` `<DomainType> ex. "Sweep" or "Time"` `Array("Context:=", <GeometryType>)` `<GeometryType>` `ex. "Infinite Sphere`*n*`", "Sphere`*n*`", "Polyline`*n*`"`

	<FamiliesArray>	Array	Contains sweep definitions for the report. `Array("<VariableName>:= ", <ValueArray>)` `<ValueArray>` `Array("All") or Array("Value1", "Value2", ..."Valuen")` `examples of <VariableName> "Freq", "Theta", "Distance"`
	<ReportDataArray>	Array	This array contains the report quantity and X, Y, and (Z) axis definitions. `Array("X Component:=", <VariableName>, "Y Component:=", <VariableName> | <ReportQuantityArray>)` `<ReportQuantityArray>` `ex. Array("dB(S(Port1, Port1))")`
Return Value	None.		

Python Syntax	CreateReport(*<ReportName>*, *<ReportType>*, *<DisplayType>*, *<SolutionName>*, *<ContextArray>*, *<FamiliesArray>*, *<ReportDataArray>*, *<ExtendedInfo>*)
Python Example	`Three examples are given below: (nominal, Optimetrics, spectral)` **Nominal Example:** `oModule.CreateReport( "XY Plot 2", "Standard",` `"Rectangular Plot", "TR",` `[` `"NAME:Context",`

```
                "SimValueContext:=", [], 1, 0, 2, 0, false,
                false, -1, 1, 0, 1, 1, "", 0, 0]
        ],
        [
                "Time:="       , ["All"],
                "aaa:="        , ["Nominal"]
        ],
        [
                "X Component:=", "Time",
                "Y Component:=", ["E1.I"]
        ],
        [])
```

Optimetrics Example:

```
oModule.CreateReport ("XY Plot 3", "Standard",
        "Rectangular Plot", "TR",
        [
                "NAME:Context",
                "OptiSetup:=", "ParametricSetup1", "SimValueContext:=",
                [
                        1, 0, 2, 0, false, false, 0, 1,0, 1, 1, "", 0, 0
                ]
        ],
        [
                "Time:=", ["All"], "aaa:=", ["Nominal"]
        ],
        [
                "X Component:="  , "Time",
                "Y Component:="  , ["E1.I"]
        ],
        [])
```

Spectral Example:

```
oModule.CreateReport ("XY Plot 4", "Standard",
       "Rectangular Plot", "TR",
       [
              "NAME:Context",
              "SimValueContext:=",
              [
                     2, 0, 2, 0, false, false, -1, 1, 0, 1, 1, "",
                     0, 0, "CG", false, "0", "KP",false, "0", "MH",
                     false, "100", "TE", false, "40ms", "TH", false,
                     "40", "TS", false, "0ns", "UF", false, "0",
                     "WT", false, "0", "WW", false, "100"
              ]
       ],
       [
              "Spectrum:="   , ["All"], "aaa:=", ["Nominal"]
       ],
       [
              "X Component:=","Spectrum",
              "Y Component:=", ["mag(E1.I)"]
       ],
       [])
```

Partial Discharge Example

```
oModule.CreateReport("Partial Discharge Plot 2",
       "Partial Discharge", "Rectangular Plot",
       "PartialDischarge1 : PartialDischarge", [],
       [
              "GasPressure:=", ["All"],
              "Freq:="       , ["All"],
              "bend_angle:=" , ["All"]
       ],
```

```
        [
                "X Component:=", "GasPressure",
                "Y Component:=", ["Power"]
        ])
```

Example 3D Rectangular Bar Plot with Change Bar properties

```
oModule = oDesign.GetModule("ReportSetup")
oModule.CreateReport("S Parameter Plot 4", "Modal Solution Data",
        "3D Rectangular Bar Plot", "Setup1 : Sweep", [],
        [
                "Freq:="        , ["All"],
                "UU:="          , ["All"],
                "ZZZ:="         , ["Nominal"]
        ],
        [
                "X Component:=", "Freq",
                "Y Component:=", "UU",
                "Z Component:=", ["dB(S(wDipole1_1_p1,wDipole1_1_p1))"]
        ])

oModule.ChangeProperty(
        [
                "NAME:AllTabs",
                [
```

```
                        "NAME:Bar",
                        [
                                "NAME:PropServers"
                                "S Parameter Plot 4:Traces-dB(S(wDipole1_1_p1,wDipole1_1_p1))"
                        ],
                        [
                                "NAME:ChangedProps",
                                [
                                        "NAME:Transparency",
                                        "Value:=", "0.5"
                                ]
                        ]
                ]
        ])

oModule.ChangeProperty(
        [
                "NAME:AllTabs",
                [
                        "NAME:Bar",
                        [
                                "NAME:PropServers",
                                "S Parameter Plot 4:Traces-dB(S(wDipole1_1_p1,wDipole1_1_p1))"
                        ],
```

```
                                "NAME:PropServers",
                                "S Parameter Plot 4:Traces-dB(S(wDipole1_1_p1,wDipole1_1_p1))"
                        ],
                        [
                                "NAME:ChangedProps",
                                [
                                        "NAME:Show Outline",
                                        "Value:=", True
                                ]
                        ]
                ]
        ])

oModule.ChangeProperty(
        [
                "NAME:AllTabs",
                [
                        "NAME:Bar",
                        [
                                "NAME:PropServers",
                                "S Parameter Plot 4:Traces-dB(S(wDipole1_1_p1,wDipole1_1_p1))"
                        ],
                        [
                                "NAME:ChangedProps",
                                [
                                        "NAME:Outline Color",
                                        "R:=", 0,
                                        "G:=", 0,
                                        "B:=", 160
                                ]
```


VB Syntax	CreateReport <*ReportName*>, <*ReportType*>, <*DisplayType*>, <*SolutionName*>, <*ContextArray*>, <*FamiliesArray*>, <*ReportDataArray*>, <*ExtendedInfo*
VB Example	`oModule.CreateReport "3D Cartesian Plot1", "Far Fields",_` `  "3D Cartesian Plot", "Setup1 : LastAdaptive", _` `  Array("Context:=", "Infinite Sphere1", "Domain:=", "Sweep"), _` `  Array("Theta:=", Array("All"), "Phi:=", Array("All"), _` `  "Freq:=", Array("10GHz")), _` `  Array("X Component:=", "Theta", _` `  "Y Component:=", "Phi", _` `  "Z Component:=", Array("rETotal")), _` `  Array()` `oModule.CreateReport "ReptSmithFreq",_` `  "Modal Solution Data", "Smith Plot", "Setup1 : Sweep1", _` `  Array("Domain:=", "Sweep"), _` `  Array("Freq:=", Array("All")), _` `  Array("Polar Component:=", _` `  Array("ln(Y(LumpPort1,LumpPort1))")), _` `  Array()` `oModule.CreateReport "Rectangular Contour Plot 2", "Far Fields", _` `  "Rectangular Contour Plot", "Setup1 : LastAdaptive", Array("Context:=", _` `  "Infinite Sphere1"), Array("_u:=", Array("All"), "_v:=", Array("All"), "Freq:=",` `Array( _`

```
  "5GHz")), Array("X Component:=", "_u", "Y Component:=", "_v", "Z Component:=", Array
( _
  "rETotal")), Array()

oModule.CreateReport "Rept2DRectFreq",_
  "Modal Solution Data", "XY Plot", _
  "Setup1 : Sweep1", _
  Array("Domain:=", "Sweep"), _
  Array("Freq:=", Array("All")), _
  Array("X Component:=", "Freq", _
  "Y Component:=", _ Array("dB(S(LumpPort1,LumpPort1))")), _
  Array()
```

DeleteDataset

Deletes a specified dataset. This can be executed by the oProject 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')`

VB Syntax	DeleteDataset <*DatasetName*>

VB Example	`oProject.DeleteDataset "$ds1"`

DeleteDesignInstance

Use: Delete the design instance

Command: None

Syntax: DeleteDesignInstance <instance_name>

Return Value: None

Parameters: <instance_name>

Type: string

VB Example: oDesign.DeleteDesignInstance <instance_name>

Python Syntax	DeleteDesignInstance(<instanceName>)
Python Example	`oProject.DeleteDesignInstance("Circuit1")`

DeleteOutputVariable

Deletes an existing output variable. The variable can only be deleted if it is not in use by any traces.

UI Access	**Circuit** > **Results** > **Output Variables**. In the **Output Variables** window, click **Delete**.
Parameters	Name | Type | Description <*OutputVarName*> | String | Name of the output variable.

Return Value	None.

Python Syntax	DeleteOutputVariable (<*OutputVarName*>)
Python Example	`oModule = oDesign.GetModule("OutputVariable")` `oModule.DeleteOutputVariable ("testNew")`

VB Syntax	DeleteOutputVariable <*OutputVarName*>
VB Example	`Set oModule = oDesign.GetModule("OutputVariable")` `oModule.DeleteOutputVariable "testNew"`

DeletePort [Nexxim]

Use: Delete a port

Command: None

Syntax: DeletePort <Port_Name>

Return Value: None

Parameters: `<Port_Name>`

Type: string

VB Example: `oDesign.DeletePort "Port1"`

Python Syntax	DeletePort(<portName>)

Python Example	`oDesign.DeletePort("Port1")`

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, DeleteFieldVariation, and DeleteLinked Variation.

UI Access	Right-click on **Results**, select **Browse Solutions...**, click **Delete** button in the dialog.
Parameters	<table><tr><th>Name</th><th>Type</th><th>Description</th></tr><tr><td><*SoluParams*></td><td>Array</td><td>Structured array. `Array(<DataSpecifierArray>, ...)`</td></tr><tr><td><*DataSpecifierArray*></td><td>Array</td><td>Structured array. `Array(<DesignVariationKey>, <SetupName>, <SolnName>)`</td></tr></table>
Return Value	None.

Python Syntax	DeleteSolutionVariation(<*SoluParams*>)
Python Example	`oModule.DeleteSolutionVariation([` `  ["width='2in'", "Setup1", "Adaptive_1"],` `  ["width='2in'", "Setup1", "Sweep1"]` `])`

VB Syntax	DeleteSolutionVariation <*SoluParams*>
VB Example	`oModule.DeleteSolutionVariation Array( _` `  Array("width=’2in’", "Setup1", "Adaptive_1"), _` `  Array("width=’2in’", "Setup1", "Sweep1"))`

DeleteSource [Nexxim]

Use: Delete the source

Command: None

Syntax: DeleteSource <srcname>

Return Value: None

Parameters: <srcname>

Type: string

VB Example: `oDesign.DeleteSource "srcname"`

Python Syntax	DeleteSource(<sourceName>)
Python Example	`oDesign.DeleteSource("VoltageSource1")`

EditDataset

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

UI Access	**Project** > **Datasets** > **Edit**.

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

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` `  ]` `  ]` `]` `)`

VB Syntax	EditDataset <*OriginalName*> <*DatasetDataArray*>
VB Example	`oProject.EditDataset "ds1" Array("NAME:ds2",_` `Array("NAME:Coordinates",Array("NAME:Coordinate",` `"X:=", 1, "Y:=", 2), Array("NAME:Coordinate",` `"X:=", 3, "Y:=", 4)))`

EditImportData

Use: Edit an imported solution

Command: None

Syntax: EditImportData <oldname> <newlink> <newpath> <newname> <data> <file>

Return Value: None

Parameters: <oldname>

Type: string

<newlink>

Type: int

<newpath>

Type: string

<newname>

Type: string

<data>

Type: array

<file>

Type: string

VB Example: `oDesign.EditImportData <oldname> <newlink> <newpath> <newname> <data> <file>`

EditNotes

Updates the notes for the design.

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

Python Syntax	EditNotes(<*DesignNotes*>)
Python Example	`oDesign.EditNotes("My design notes.")`

VB Syntax	EditNotes <*DesignNotes*>
VB Example	`oDesign.EditNotes "My design notes."`

EditOutputVariable

Changes the name or expression of an existing output variable.

<table>
<tr><td>UI Access</td><td>N/A</td></tr>
<tr><td>Parameters</td><td>
<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td><OrigVarName></td><td>String</td><td>Name of the original output variable.</td></tr>
<tr><td><NewExpression></td><td>String</td><td>New value to assign to the variable.</td></tr>
<tr><td><NewVarName></td><td>String</td><td>New name of the variable if any, else pass empty string.</td></tr>
<tr><td><SolutionName></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><ReportType> or <SolutionType></td><td>String</td><td>The name of the report type as seen in the output variable UI.

For Layout Editor, use the solution type.</td></tr>
<tr><td><ContextArray> or <DomainArray></td><td>Array</td><td>Structured array containing context for which the output variable expression is being evaluated.

<code>Array("Context:=", <string>)</code>

For Layout Editor, use Domain array:

<code>Array("Domain:=", <string>)</code></td></tr>
</table>
</td></tr>
<tr><td>Return Value</td><td>None</td></tr>
</table>

Python Syntax	EditOutputVariable (<*OrigVarName*>, <*NewExpression*>, <*NewVarName*>, <*SolutionName*>, <*ReportType* | *SolutionType*>, <*ContextArray* | *DomainArray*>)
Python Example	`oModule = oDesign.GetModule("OutputVariable")` `oModule.EditOutputVariable ("test", "normalize(R1_0.V)", "testNew", "TR", "Standard", [])`

VB Syntax	EditOutputVariable <*OrigVarName*>, <*NewExpression*>, <*NewVarName*>, <*SolutionName*>, <*ReportType* | *SolutionType*>, <*ContextArray* | *DomainArray*>
VB Example	`Set oModule = oDesign.GetModule("OutputVariable")` `oModule.EditOutputVariable "test", "dB(S(WavePort1,WavePort1)) ", "testNew", "Setup1 : LastAdaptive", "Modal Solution Data", Array("Domain:=", "Sweep")`

ExportDataset

Exports a dataset to a named file. This can be executed by the oProject 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')`

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

ExportForSpice

Exports matrix solution data to a file in a format suitable for Spice analysis. Available only for Driven Terminal solution types with ports. Output in an appropriate format will be generated for each of the non-empty file names provided.

UI Access	N/A

Parameters

Name	Type	Description
<*DesignVariation*>	String	Desgin variation key.
<*SoluSelections*>	Array	Array of selected solutions.
<*SpiceType*>	Integer	0 - PSpice 2 - Maxwell Spice
<*BandWidth*>	Integer	0 - Low (narrow) band width.
<*FWSFile*>	String	Full wave spice file name.
<*LumpedElementFile*>	String	Lumped element file name.
<*PoleZeroSpiceFile*>	String	Pole zero spice file name.
<*PoleZeroMatlabFile*>	String	Pole zero Matlab file name.
<*PartialFractionFile*>	String	Partial fraction file name.
<*FittingError*>	Double	Optional. The accuracy to use in fitting the pole zero model.
<*MaximumOrder*>	Integer	Optional. Maximum number of poles in rational function expansion.
<*UseCommonGround*>	Integer	Optional. 1 to use common ground, 0 otherwise.
<*EnforcePassivity*>	Integer	Optional. 1 to enforce passivity, 0 otherwise.

Return Value	None.

Python Syntax	ExportForHSpice(<*DesignVariation*>, <*SoluSelections*>, <*SpiceType*>, <*BandWidth*>, <*FWSFile*>, <*LumpedElementFile*>, <*PoleZeroSpiceFile*>, <*PoleZeroMatlabFile*>, <*PartialFractionFile*>, [Optional <*FittingError*>], [Optional <*MaximumOrder*>], [Optional <*UseCommonGround*>], [Optional <*EnforcePassivity*>])
Python Example	`oModule.ExportForSpice("width='2in'",`

	["Setup1:Sweep1"], 2, 0, "c:\mydir\Sweep1.fws", "", "", "", "")

VB Syntax	ExportForHSpice <*DesignVariation*>, <*SoluSelections*>, <*SpiceType*>, <*BandWidth*>, <*FWSFile*>, <*LumpedElementFile*>, <*PoleZeroSpiceFile*>, <*PoleZeroMatlabFile*>, <*PartialFractionFile*>, [Optional <*FittingError*>], [Optional <*MaximumOrder*>], [Optional <*UseCommonGround*>], [Optional <*EnforcePassivity*>]
VB Example	oModule.ExportForSpice "width='2in'", _ Array("Setup1:Sweep1"), 2, 0, _ "c:\mydir\Sweep1.fws", "", "", "", ""

ExportNMFData

*Use:*Exports s-parameters in neutral file format.

*Command:*In the **matrix** tab of the **Solution** dialog box, click **Export->S-Parameter**. Then select the **Neutral Model** file format.

Syntax: ExportNetworkData <SolutionName> <FileName> <ReduceMatrix><Reference Impedance> <FrequencyArray> <DesignVariation> <Format> <Length> <PassNumber>

*Return Value:*None

*Parameters:*SolutionName>

Type: <String>

Format: <SetupName>:<SolutionName>

Solution that is exported.

```
<FileName>
     Type:<String>
     The name of the file. The file extension will determine the file format.
<ReduceMatrix>
     Type: <string>
     Either "Original" or one of the reduce matrix setup name
<Reference Impedance>
     Type: <Double>
     Reference impedance
<FrequencyArray>
     Type: <Array of doubles>
     Value: Array(<double>,<double>….)
     Frequency points in the sweep that is exported.
<DesignVariation>
   Type: <String>
   Design variation at which the solution is exported.
<Format>
     Type: <String>
     Values: "MagPhase", "RealImag", "DbPhase".
     The format in which the sparameters will be exported.
<Length>
```

Type:<String>

Length for exporting sparameters.

<PassNumber>

Type:<integer>

Pass number.

VB Example:

```
oDesign.ExportNMFData "Setup7 : LastAdaptive", "C:/temp/neu.nmf", "Original", _ 50, Array
(10000000, 20000000, 30000000, 40000000, 50000000, 60000000, 70000000, _ 80000000, 90000000,
100000000), "", "MagPhase","7meter"
```

Python Syntax	ExportNMFData("", # Empty string. 1, #The number 1. "Name", # This is the full path of the file from which the solution is loaded "ExportFile", # full path of file to export to ["NAME:Frequencies"],

```
#optional, if none defined all frequencies are used
        ["NAME:NMFOptions",
#Export NMF options object
        "DataTypes:=", ["S"],
#DataTypes can be "S", "Y","Z", "G", and "Z0", for S , Y, Z matrix, Gamma and Z0 (zero)
        "DisplayFormat:=", "MA",
#DisplayFormat "MA", "RI", "DB"
        "FileType:=", "",
# Export File Type
2 - Spreadsheet(*.tab)
3 - Touchstone(*.sNp)
4 - Citifile(*.cit)
6 - Neutral format(*.nmf)
7 - Matlab format(*.m)
          "Renormalize:=", false,
#Renormalize true/false
        "RefImpedance:=",50,
# Reference Impedance
        "Precision:=", 8,
# Number of digits Precision
        "Variables:=", ["FF", "cap", "Rs"]
```

	# Array of variables "Variations:=", ["", "", ""] # Array of variations to export solutions for ["NAME:ConstantVars"] #Array of variables that are constant, can be empty ["NAME: DependentVars"] #Array of variables that are dependent, can be empty "MatrixSize:=", 2, #Matrix size, optional (used in nmf file header) "CreateNPortModel:=", true #Create a model based on the exported file true/false])
Python Example	oTool = oDesktop.GetTool("NdExplorer") oTool.ExportNetworkData("", True, "C:/.../100OHM.S2P", "$SYSLIB/Test.s2p", "", ["NAME:Frequencies", 500000000, 1000000000, 10000000000,

```
                25000000000,
                50000000000,
                75000000000,

                100000000000
        ],
        [
                "NAME:TSOptions",
                "DataTypes:="           , ["S"],
                "DisplayFormat:="       , "DB",
                "FileType:="            , 3,
                "Renormalize:="         , True,
                "RefImpedance:="        , 50,
                "Precision:="           , 6,
                "UseMultipleCores:="    , False,
                "NumberOfCores:="       , 1,
                "Comments:="            , True,
                "Noise:="               , False
        ])
```

ExportNetlist [Nexxim]

Exports a netlist solution.

UI Access	None
Parameters	<table><tr><th>Name</th><th>Type</th><th>Description</th></tr><tr><td><*solution*></td><td>String</td><td>The Circuit solution in the design. For Maxwell Circuit, the solution must be an empty string, "".</td></tr><tr><td><*fileName*></td><td>String</td><td>File and path to the netlist file. For Maxwell Circuit, the file name must be a SPH file.</td></tr></table>
Return Value	None.

Python Syntax	ExportNetlist(<setupName>,<fileName>)
Python Example	`oDesign.ExportNetlist("QuickEyeAnalysis", "SolutionData")`

VB Syntax	ExportNetlist <*solution*> <*fileName*>
VB Example	`oDesign.ExportNetlist "QuickEyeAnalysis" "SolutionData"`

ExportReport

> **Note:**
>
> The ExportReport script command has been replaced by the script command ExportToFile. ExportReport remains in order to retain backward compatibility for existing scripts, but it is strongly recommended that you now use ExportToFile.

Export a report to a data file.

Command: None

Syntax: ExportReport <ReportName>, <FileName>, <FileExtension>

Return Value: None

Parameters: <ReportName>

Type: string

<Filename>

Type: string

<FileExtension>

Type: string

VB Example:

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

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

Set oProject = oDesktop.SetActiveProject("BJTinverter")
```

```
Set oDesign = oProject.SetActiveDesign("Nexxim1")
oDesign.ExportReport "Data Table 1", "table_test", "csv"
```

Python Syntax	ExportReport(<ReportName>, <FileName>)
Python Example	`oDesign.ExportReport(` `"Plot1", "c:\report1.dat")`

GetActiveEditor

Get the name of the active editor.

Command: None

Syntax: GetActiveEditor <object_variable>

Return Value: String

Parameters: <object_variable>

Type: string

VB Example: oDesign.GetActiveEditor <object_variable>

Note that GetActiveEditor returns NULL if the editor is open in a non-active window.

Python Syntax	GetActiveEditor(<object_variable>)
Python Example	`oDesign.GetActiveEditor(editor_string)` *

GetAllPorts

Get all ports in a design.

UI Access	NA
Parameters	Name | Type | Description None | |
Return Value	Ports in the design

Python Syntax	GetAllPorts()
Python Example	`oEditor.GetAllPorts()`

VB Syntax	GetAllPorts
VB Example	`Set oEditor = oDesign.SetActiveEditor("SchematicEditor")` `oEditor.GetAllPorts()`

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", and "Variable". Default returns

			Module names.
Return Value	Names of children for the queried object).		
Python Syntax	GetChildNames (<*type*>)		
Python Example	`oDesign.GetChildNames("Variable")`		

VB Syntax	GetChildNames <*type*>
VB Example	`oDesign.GetChildNames "Variable"`

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")

VB Syntax	GetChildObject <*path*>
VB 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()

VB Syntax	GetChildTypes
VB Example	`oDesign.GetChildTypes`

GetConfigurableData

Obtains configurable data of a specified type

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

UI Access	N/A
Parameters	Name | Type | Description *<type>* | String | Specified type.
Return Value	None.

Python Syntax	GetConfigurableData(*<type>*)
Python Example	`oDesign.GetConfigurableData("model")`

VB Syntax	GetConfigurableData *<type>*
VB Example	`oDesign.GetConfigurableData "model"`

GetData

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

Python Syntax	GetData()
Python Example	oDesign.GetData()

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()`

VB Syntax	GetDesignType

VB Example	`Set oDesign = oProject.GetActiveDesign` `oDesign.GetDesignID`

GetDesignName

Returns the name of the active design.

UI Access	N/A
Parameters	None.
Return Value	String indicating the name of the active design.

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

VB Syntax	GetDesignName
VB Example	`Set oDesign = oProject.GetActiveDesign` `oDesign.GetDesignName`

GetDesignType

Returns the design type of the active design.

UI Access	N/A

Parameters	None.
Return Value	String indicating the design type of the active design ("Circuit Design", "Circuit Netlist","EMIT", "HFSS 3D Layout Design", "HFSS", "HFSS-IE", "Icepak", "Maxwell 2D", "Maxwell 3D", "Q2D Extractor", "Q3D Extractor", "RMxprt", or "Twin Builder").

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

VB Syntax	GetDesignType
VB Example	`Set oDesign = oProject.GetActiveDesign` `oDesign.GetDesignType`

GetEditor

Use: Returns an interface to the editor requested IF the PropServer behind the PropHost is contained within that type of editor.

Command: None

Syntax: GetEditor(<editorName>)

Return Value: String

VB Example: `Set oLayout2 = PropHost.GetEditor("Layout"); returns the interface to the layout containing a selected component. This interface can be used to call Layout Scripting functions.`

Python Syntax	GetEditor(<editorName>)

Python Example	`oEditor = GetEditor("SchematicEditor")`

GetModule

Returns the IDispatch for the specified module.

UI Access	N/A
Parameters	<table><tr><th>Name</th><th>Type</th><th>Description</th></tr><tr><td><*modulename*></td><td>String</td><td>One of the following: - Analysis Module – "AnalysisSetup" - Data Module – “DataBlock” - Design Verification Module – "DV" - Optimetrics Module – "Optimetrics" - Output Variable Module – "OutputVariable" - Reporter Module – "ReportSetup" - User Defined Documents Module – "UserDefinedDocuments" - User Defined Solutions Module – "UserDefinedSolutionModule" </td></tr></table>
Return Value	Module IDispatch

Python Syntax	GetModule (<*modulename*>)
Python Example	`oModule = oDesign.GetModule("SimSetup")`

VB Syntax	GetModule <*modulename*>
VB Example	`set oModule = oDesign.GetModule "SimSetup"`

GetName

Returns the design ID and design name of the active design, in that order separated by a semicolon. If the active design is a sub-circuit, it also returns the parent ID and parent design name, listed before the active design information.

To get the *only* the design's name without the parent design name or design IDs, see oDesign.GetDesignName().

UI Access	N/A
Parameters	None.
Return Value	String indicating the name and ID of the active design and parent design (if the active design is a sub-circuit).

Python Syntax	GetName()
Python Example	oDesign.GetName( ) for a subcircuit *EMDesign2* in a parent design *EMDesign1* design_name = oDesign.GetName() 1;EMDesign1/2;EMDesign2 In the returned string: `1` is the parent design ID; `EMDesign1` is the parent design name. `2` is the active design ID; `EMDesign2` is the active design name.

VB Syntax	GetName

VB Example	`design_name = oDesign.GetName`

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()`

VB Syntax	GetNoteText
VB 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()`

VB Syntax	GetObjPath
VB 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.

UI Access	N/A

Parameters

Name	Type	Description
<*OutputVarName*>	String	Name of the output variable.
<*IntrinsicVariation*>	String	A set of intrinsic variable value pairs to use when evaluating the output expression. HFSS example: `"Freq='20GHz' Theta='20deg' Phi='30deg'"`
<*SolutionName*>	String	Name of the solution as listed in the output variable UI. For example, `"Setup1 : Last Adaptive"`.
<*ReportType*>	String	The name of the report type as seen in the output variable UI.

			Possible values in Layout/Schematic Editor are: • "Standard" • "Load Pull" • "Constellation" • "Data Table" • "Eye Diagram" • "Statistical"
	<*ContextArray*>	Array	Structured array containing context for which the output variable expression is being evaluated. Can be empty. `Array("Context:=", <string>)`
Return Value	Double value of the output variable.		

Python Syntax	GetOutputVariableValue(<*OutputVarName*>, <*IntrinsicVariation*>, <*SolutionName*>, <*ReportTypeName*>, <*ContextArray*>)
Python Example	`Val = oDesign.GetOutputVariableValue("test","Freq = '20Ghz' Theta='20deg' Phi='30deg'", "TR", "Standard", [])`

VB Syntax	GetOutputVariableValue <*OutputVarName*>, <*IntrinsicVariation*>, <*SolutionName*>, <*ReportTypeName*>, <*ContextArray*>
VB Example	`Val = oDesign.GetOutputVariableValue` `"test","Freq = '20Ghz' Theta='20deg'`

	`Phi='30deg'", "TR", "Standard", Array()`

GetOutputVariables

Returns the list of output variables.

UI Access	N/A
Parameters	None.
Return Value	Array containing all output variables.

Python Syntax	GetOutputVariables()
Python Example	`oDesign.GetOutputVariables()`

VB Syntax	GetOutputVariables
VB Example	`oDesign.GetOutputVariables`

GetPropEvaluatedValue

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

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

UI Access	N/A

Parameters	<table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><*PropName*></td><td>String</td><td>Name of the property.</td></tr></table>
Return Value	String value of the evaluated value.

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

GetPropHost

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

Command: None

Syntax: GetPropHost()

Return Value: Returns interface to PropHost.

VB Example: `Set oPropHost2 = CompInstance.GetPropHost();`

Returns the interface to the properties of the compInstance.

This interface can be used to call PropHost functions; for more information see Callback Scripting Using PropHost Object.

Python Syntax	GetPropHost()
Python Example	`oPropHost2 = CompInstance.GetPropHost()`

GetPropNames [Design]

Returns array containing names of property. For designs, always returns an empty array because the design has no property.

UI Access	N/A
Parameters	Name: *<includeReadOnly>* — Type: Boolean — Description: (Optional). • True - include read only property. • False - do not include read only property.
Return Value	Empty array.
Python Syntax	GetPropNames(*<includeReadOnly>*)
Python Example	`oDesign.GetPropNames()`

VB Syntax	GetPropNames *<includeReadOnly>*
VB Example	`oDesign.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><*PropName*></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 [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><*propPath*></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')`

VB Syntax	GetPropValue(<*propPath*>)
VB Example	`val = oDesign.GetPropValue("offset/SIValue")` `disp = oDesign.GetPropValue("Results/S Parameter Plot 1/Display Type")` `arr = oDesign.GetPropValue("Results/S Parameter Plot 1/Display Type/Choices")`

GetProperties

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

UI Access	N/A
Parameters	<table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><*PropTab*></td><td>String</td><td>One of the following, where tab titles are shown in parentheses:</td></tr></table>

			• PassedParameterTab ("Parameter Values") • DefinitionParameterTab (Parameter Defaults") • LocalVariableTab ("Variables" or "Local Variables") • ProjectVariableTab ("Project variables") • ConstantsTab ("Constants") • BaseElementTab ("Symbol" or "Footprint") • ComponentTab ("General") • Component("Component") • CustomTab ("Intrinsic Variables") • Quantities ("Quantities") • Signals ("Signals")
	<*PropServer*>	String	An object identifier, generally returned from another script method, such as CompInst@R;2;3
Return Value	Array of strings containing the names of the appropriate properties.		

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

VB Syntax	GetProperties <*PropTab*>, <*PropServer*>
VB Example	`oEditor.GetProperties "PassedParameterTab", "k"`

GetPropertyValue

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

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

<table>
<tr><td>UI Access</td><td>N/A</td></tr>
<tr><td>Parameters</td><td>
<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td><PropTab></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><PropServer></td><td>String</td><td>An object identifier, generally returned from another script method, such as <code>Comp$nst@R;2;3</code></td></tr>
<tr><td><PropName></td><td>String</td><td>Name of the property.</td></tr>
</table>
</td></tr>
<tr><td>Return Value</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")` `...`

VB Syntax	GetPropertyValue (<*PropTab*>, <*PropServer*>, <*PropName*>)
VB Example	`selectionArray = oEditor.GetSelections` `for k in selectionArray:` `val = oEditor.GetPropertyValue("PassedParameterTab", k, "R")` `...`

GetResultsDirectory [Nexxim]

Gives access to the directory containing the specified Nexxim solution files.

Command: None

Syntax: GetResultsDirectory ([in] BSTR solname, [in] BSTR variation, [in] BOOL finalDir, [out, retval] BSTR* dirname)

Return Value: <dirname>

Type: String // Directory name. (The returned directory path does not contain a trailing slash.).

Parameters: <solname>

Type: string

<variation>

Type: string

<finalDir>

Type: boolean // True if the final directory is desired; False if the working/temporary directory is desired.

VB Example: Dim dirName

Set oDesign = oProject.GetActiveDesign() dirName = oDesign.GetResultsDirectory ("Trans", "res='1000'", false)

Python Syntax	dirname = GetResultsDirectory (< solname>, <variation>,<finalDir>,)
Python Example	`dirName = oDesign.GetResultsDirectory("Trans", "res='1000'", False)`

GetSourceData (Layout Editor)

Takes a source name as input and returns a VARIANT array containing a data block of the source data.

Command: None

Syntax: GetSourceData

Return Value: Type: Array of i/o block data

Parameters: <SourceName>

Type: string

VB Example: Dim sourceDataArr

```
sourceDataArr = oDesign.GetSourceData ("SourceName")
```

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")`

VB Syntax	GetVariableValue <*VarName*>
VB 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()`

VB Syntax	GetVariables
VB Example	`oProject.GetVariables` `oDesign.GetVariables`

GetVariationVariableValue

Returns the value for a specified variation's variable.

UI Access	N/A
Parameters	Name | Type | Description <*VariationString*> | String | The name of the design variation. <*VariableName*> | String | The name of the variable.
Return Value	Returns a double precision value in SI units, interpreted to mean the value of the variable contained in the variation string.

Python Syntax	GetVariationVariableValue(<*VariationString*>, <*VariableName*>)
Python Example	`oDesign.GetVariationVariableValue('x_size = 2mm y_size = 1mm', 'y_size')`

VB Syntax	GetVariationVariableValue <*VariationString*>, <*VariableName*>
VB Example	`varval = oDesign.GetVariationVariableValue "x_size = 2mm y_size = 1mm", "y_size"`

ImportData [Nexxim]

Imports a network solution

Command: None

Syntax: ImportData <data> <filename> <linktofile>

Return Value: None

Parameters: <data>

Type: string // data to be imported

<filename>

Type: string // name of the file

<linktofile>

Type: int // link to the file

VB Example: `oDesign.ImportData <data> <filename> <linktofile>`

Python Syntax	ImportData(<data>,<file>, <customization>)
Python Example	oDesign.ImportData(["NAME:NPortData", "Description:=" , "", "ImageFile:=" , "", "SymbolPinConfiguration:=", 0, ["NAME:PortInfoBlk"], ["NAME:PortOrderBlk"], "filename:=" , "E:/R18Examples/Signal Integrity/models/Serial_Channel/connector_model.s4p", "numberofports:=" , 4, "sssfilename:=" , "", "sssmodel:=" , False, "PortNames:=" , ["Port1","Port2","Port3","Port4"],

```
        "domain:="                , "frequency",
        "datamode:="              , "Link",
        "devicename:="            , "",
        "SolutionName:="  , "connector_model",
        "displayformat:=" , "MagnitudePhase",
        "datatype:="              , "SMatrix",
        [
           "NAME:DesignerCustomization",
           "DCOption:="           , 0,
           "InterpOption:="       , 0,
           "ExtrapOption:="       , 1,
           "Convolution:="        , 0,
           "Passivity:="          , 0,
           "Reciprocal:="         , False,
           "ModelOption:="        , "",
           "DataType:="           , 1
        ],
        [
           "NAME:NexximCustomization",
           "DCOption:="           , 3,
```

```
        "InterpOption:="       , 1,
        "ExtrapOption:="       , 3,
        "Convolution:="                , 0,
        "Passivity:="          , 0,
        "Reciprocal:="         , False,
        "ModelOption:="                , "",
        "DataType:="           , 2
    ],
    [
        "NAME:HSpiceCustomization",
        "DCOption:="           , 1,
        "InterpOption:="       , 2,
        "ExtrapOption:="       , 3,
        "Convolution:="        , 0,
        "Passivity:="          , 0,
        "Reciprocal:="         , False,
        "ModelOption:="        , "",
        "DataType:="           , 3
    ],
    "NoiseModelOption:="       , "External"
], "", False)
```

ImportDataFilePath [Nexxim]

Returns the file path of the imported solution given an imported solution name. If the solution is not found it returns an empty string.

Command: None

Syntax: ImportDataFilePath <solution_name>

Return Value: File path of imported solution, empty string if solution not found in the design.

Type: string

Parameters: <solution_name>

Type: string

VB Example: oDesign.ImportDataFilePath <solution_name>

Python Syntax	ImportDataFilePath(<solution_name>)
Python Example	`solution_path_variable = oDesign.ImportDataFilePath("MySolution")`

ImportDataset

Imports a dataset from a named file. This can be executed by the oProject 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.

UI Access	**Project** > **Datasets** > **Import**.

Parameters	Name	Type	Description
	<datasetFileFullPath>	String	The full path to the file containing the dataset values. *.tab files recommended (see note below).
	<optionalDatasetName>	String	*Optional*. User-defined dataset name.
Return Value	None.		

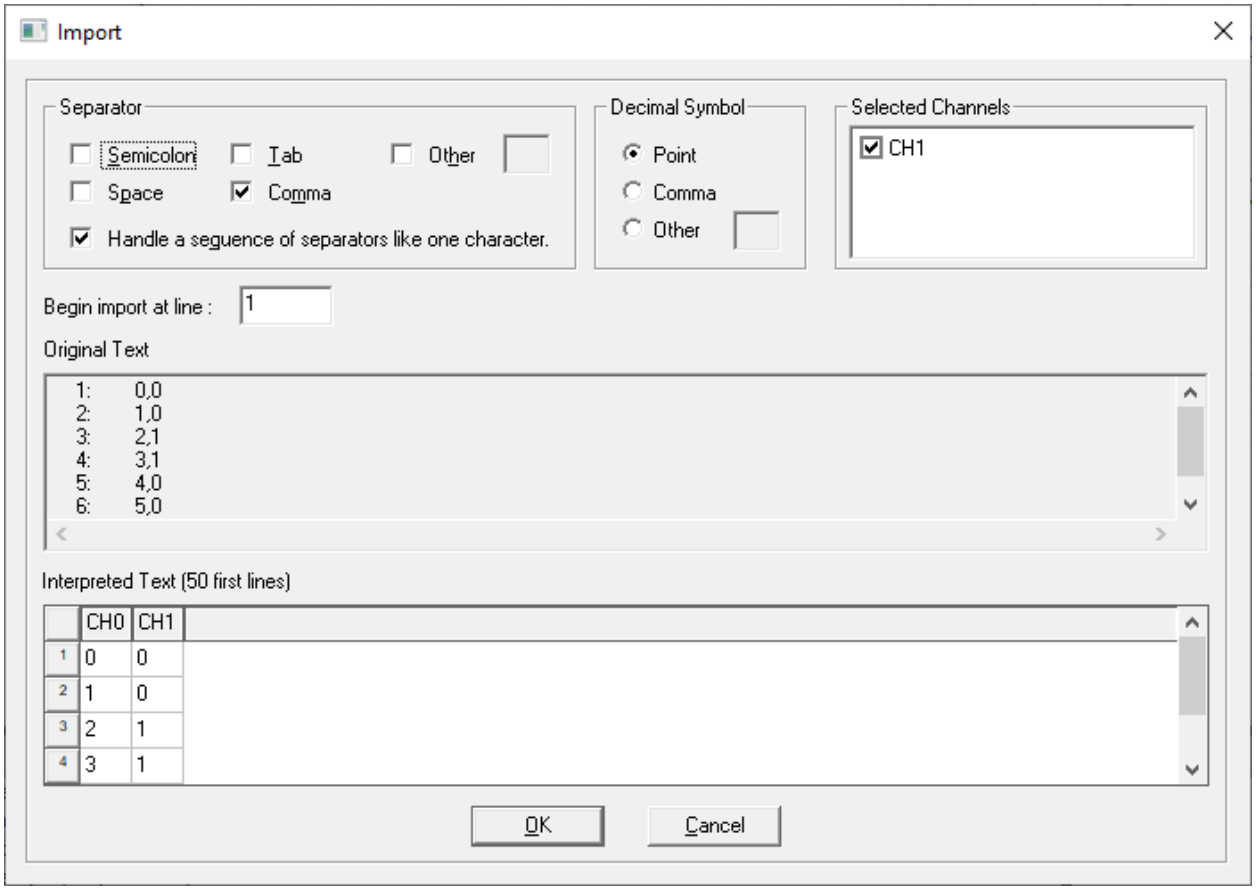
Python Syntax	ImportDataset (*<datasetFileFullPath>*,*<optionalDatasetName>*)
Python Example	`oProject.ImportDataset('e:\tmp\dsdata.tab')` `oProject.ImportDataset('e:\tmp\dsdata.tab', 'MyDatasetName')`

VB Syntax	ImportDataset *<datasetFileFullPath>*, *<optionalDatasetName>*
VB Example	`oProject.ImportDataset "e:\tmp\dsdata.tab"` `oProject.ImportDataset "e:\tmp\dsdata.tab", "MyDatasetName"`

Note About File Types:

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

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



InsertDesign

Inserts a new design in the project. For Circuit and HFSS 3D Layout scripts, the last argument will always be empty.

UI Access	**Project** > **[Insert Circuit Design / Insert Circuit Netlist / Insert HFSS 3D Layout Design]**.
Parameters	<table><tr><th>Name</th><th>Type</th><th>Description</th></tr><tr><td><*DesignType*></td><td>String</td><td>Design type. Valid types: "Circuit", "Nexxim Circuit", "System" , "Nexxim Netlist", "HFSS3D".</td></tr><tr><td><*DesignName*></td><td>String</td><td>Design name.</td></tr><tr><td><*TechnologyFile*></td><td>String</td><td>The path to the Circuit technology file to be used in the design. Use a pair of empty double quotes ("") for none.</td></tr><tr><td><*SubcircuitID*></td><td>String</td><td>(Optional) Must be preceded by a colon if included along with the Technology File name. No colon is necessary when the subcircuit ID is omitted.</td></tr><tr><td>""</td><td>None</td><td>Empty argument.</td></tr></table>
Return Value	None.

Python Syntax	InsertDesign (<*DesignType*>, <*DesignName*>, <*TechnologyFile*>:<*SubCircuitID*>,'')
Python Example	`oProject.InsertDesign('Nexxim Circuit','MyDesigner', 'C:\\Program Files\\AnsysEM\\Designer\\', '')`

VB Syntax	InsertDesign <*DesignType*>, <*DesignName*>, <*TechnologyFile*>:<*SubCircuitID*>,""
VB Example	`oProject.InsertDesign "Nexxim Circuit","MyDesigner", "C:\Program Files\AnsysEM\Designer\", ""`

PasteDesign (Design Object)

Pastes a design that has already been copied to the clipboard into another design.

UI Access	**Edit** > **Paste**.
Parameters	<table><tr><th>Name</th><th>Type</th><th>Description</th></tr><tr><td><*pasteOption*></td><td>Integer</td><td>One of the following: **0** – Link to the existing design that was copied **1** – Create a new copy of the design and keep the original layers of the design being copied **2** – Create a new copy of the design and merge the layers of the design being copied. </td></tr></table>
Return Value	None.

Python Syntax	PasteDesign(<*pasteOption*>)
Python Example	`oProject.CopyDesign('DesignB')` `oDesign = oProject.SetActiveDesign('DesignA')` `oDesign.PasteDesign(1)`

VB Syntax	PasteDesign <*pasteOption*>
VB Example	`oProject.CopyDesign "B"` `Set oDesign = oProject.SetActiveDesign("A")` `oDesign.PasteDesign 1`

PasteItemCommand

Paste tree items, such as Altrasim Solution and Solve Setups pasted to the Analysis Tree, or Frequency Sweeps pasted to the Solve Setup Tree.

Command: Right-click on the Analysis item in the Project tree and select Paste.

Syntax: PasteItemCommand <ItemPath>

Return Value: None

Parameters: <ItemPath>

Type: string

VB Example: `oDesign.PasteItemCommand "Project1|Nexxim1|Analysis"`

Python Syntax	PasteItemCommand(<ItemPath>)
Python Example	`oDesign.PasteItemCommand("Project1|Nexxim1|Analysis")`

Redo [Design]

Reapplies the last design-level command.

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

Python Syntax	Redo()
Python Example	`oDesign.Redo()`

VB Syntax	Redo
VB Example	`oDesign.Redo`

RemoveImportData (Layout Editor)

Remove an imported solution

Command: None

Syntax: RemoveImportData <solution_name>

Return Value: None

Parameters: `<solution_name>`

Type: string

VB Example: `oDesign.RemoveImportData <solution_name>`

RenameDesignInstance

Renames a design instance.

UI Access	Right-click a design instance in the project tree, and then click **Rename** on the shortcut menu.
Parameters	Name | Type | Description <*OldName*> | String | The current name of the design, which must be the design on which this com-

			mand is invoked.
	<*NewName*>	String	The new name for the design.
Return Value	None.		

Python Syntax	RenameDesignInstance (<*OldName*>, <*NewName*>)
Python Example	`oDesign.RenameDesignInstance("Design1", "Design2")`

VB Syntax	RenameDesignInstance <*OldName*>, <*NewName*>
VB Example	`oDesign.RenameDesignInstance "Design1", "Design2"`

RenameImportData [Nexxim]

Rename an imported solution

Command: None

Syntax: RenameImportData <oldname> <newname>

Return Value: None

Parameters: <oldname>

Type: string

<newname>

Type: string

VB Example: oDesign.RenameImportData <oldname> <newname>

Python Syntax	RenameImportData(<oldName>,<newName>)
Python Example	`oDesign.RenameImportData("Solution", "ConnectorData")`

RenameSource [Nexxim]

Use: Rename the source

Command: None

Syntax: RenameSource <oldname>, <newname>

Return Value: None

Parameters: <oldname>

Type: string

<newname>

Type: string

VB Example: `oDesign.RenameSource "oldname", "newname"`

Python Syntax	RenameSource(<oldName>,<newName>)
Python Example	`oDesign.RenameSource("VoltageSource1", "MySource1")`

RunToolkit

Runs a Python toolkit script, applying it to the active design. The toolkit script itself may have prerequisites, such as sweeps defined, or specific model characteristics, such as port definitions.

> **Important:**
>
> Full scripting for cable modeling is not supported and no arguments are allowed in the script. The **RunToolkit** command will not automatically create a cable bundle, but will simply open the **Cable Modeling** dialog box. You will have to manually input your parameters.

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

Python Syntax	RunToolkit(<*LibraryType*>, <*ToolkitName*>, <*ToolkitArg*>)
Python Example	`oDesign.RunToolkit("SysLib", "Hearing Aid Compliance", [])`

VB Syntax	RunToolkit <*LibraryType*>, <*ToolkitName*>, <*ToolkitArg*>
VB Example	`oDesign.RunToolkit "SysLib", "Wavelength Calculator", Array()` `oDesign.RunToolkit "SysLib", "Hearing Aid Compliance", Array()`

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')`

VB Syntax	SetActiveEditor <*EditorName*>
VB Example	`oDesign.SetActiveEditor "3D Modeler"`

SetPropValue [Design]

Sets the property value for the active property object.

UI Access	Edit properties on Project Tree objects.
Parameters	Name | Type | Description <*propPath*> | String | A child object's property path. See: Object Property Script Function Summary. <*Value*> | String | New property value.
Return Value	Boolean:

	• **True** - property found and the new value is valid. • **False** - property not found.

Python Syntax	SetPropValue(<*propPath*>, <*Value*>)
Python Example	`oDesign.SetPropValue("offset", "12mm")` `oDesign.SetPropValue("Results/S Parameter Plot 1/Display Type", "Rectangular Plot")`

VB Syntax	SetPropValue <*propPath*>, <*Value*>
VB 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.

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 Comp‡nst@R;2;3
	<propName>	String	Name of the property.
	<propValue>	String	The value for the property
Return Value	None.		

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

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

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')`

VB Syntax	SetVariableValue <*VarName*>, <*VarValue*>
VB Example	`oProject.SetVariableValue "$Var1", "3mm"`

SimulateLink

Runs linked simulation.

UI Access	N/A
Parameters	Name | Type | Description

	<setup>	String	Setup name.
	<linkedDesign>	Object	Optional. Linked design.
	<linkType>	String	Optional. Type of linked design.
	<msg>	String	Optional. Messages to add.
Return Value	None.		

Python Syntax	SimulateLink(*<setup>*, *<linkedDesign>*, *<linkType>*, *<msg>*)
Python Example	`oDesign.SimulateLink("1_4GHz")`

VB Syntax	SimulateLink *<setup>*, *<linkedDesign>*, *<linkType>*, *<msg>*
VB Example	`oDesign.SimulateLink "1_4GHz"`

StartAnalysis [Nexxim]

Simulates all setups

Command: None

Syntax: StartAnalysis

Return Value: None

Parameters: None

VB Example: `oDesign.StartAnalysis`

Python Syntax	StartAnalysis()
Python Example	`oDesign.StartAnalysis()`

StopSimLink

Stops linked simulation.

UI Access	N/A
Parameters	<table><tr><th>Name</th><th>Type</th><th>Description</th></tr><tr><td><*simId*></td><td>Integer</td><td>ID of specified simulation.</td></tr><tr><td><*abort*></td><td>Boolean</td><td>Whether to stop the running simulation: **True** - abort the running simulation. **False** - do not abort running simulation. </td></tr></table>
Return Value	None.

Python Syntax	StopSimLink(<*simId*>, <*abort*>)
Python Example	`oDesign.StopSimLink(18, True)`

VB Syntax	StopSimLink <*simId*>, <*abort*>
VB Example	`oDesign.StopSimLink 18, true`

Undo [Design]

Cancels the last design-level command.

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

Python Syntax	Undo()
Python Example	`oDesign.Undo()`

VB Syntax	Undo
VB Example	`oDesign.Undo`

UseCircuitSParameterDefinition

Selects whether the circuit or high-frequency definition of S-parameters is used.

Command: None

Syntax: UseCircuitSparameterDefinition <boolean_operator>

Return Value: None

Parameters: `<boolean_operator>`

Type: bool

VB Example: `oDesign.UseCircuitSparameterDefinition true`

```
oDesign.UseCircuitSparameterDefinition true
```

Python Syntax	UseCircuitSparameterDefinition(True)
Python Example	`oDesign.UseCircuitSparameterDefinition(True)`

ValidateLink

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

Python Syntax	ValidateLink()
Python Example	oDesign.ValidateLink()

This page intentionally
left blank.

8 - Output Variable Script Commands

Output variable commands should be executed by the "OutputVariable" module.

First, obtain the output variable module from oDesign and use it for output variable commands:

```
Set oModule = oDesign.GetModule("OutputVariable")
oModule.<CommandName><args>
```

The commands are:

CreateOutputVariable

Adds a new output variable. Different forms of this command are executed for different design types.

> **Note:**
>
> Output variables are associated with a name and an expression. The name is not permitted to collide with design variable, sim value, or other output variable names. It cannot have spaces or any arithmetic or other operators in it. Expression definitions cannot be cyclic. For example, A = 2*B, B=3*A is not allowed.

UI Access	**Circuit** > **Results** > **Output Variables**.

Parameters

Name	Type	Description
<*OutputVarName*>	String	Name of the new output variable.
<*Expression*>	Value	Value to assign to the variable.
<*SolutionName*>	String	Name of the solution, as seen in the output variable UI.
<*reportType*> or <*solutionType*>	String	The name of the report type as seen in the output variable UI. For Layout Editor, use the solution type.
<*ContextArray*> or <*DomainArray*>	Array	Structured array containing context for which the output variable expression is being evaluated. `Array("Context:=", <string>)` For Layout Editor, use Domain array: `Array("Domain:=", <string>)`

Return Value

None.

Python Syntax

CreateOutputVariable (<*OutputVarName*>, <*Expression*>, <*SolutionName*>, <*reportType* | *solutionType*>, <*contextArray* | *domainArray*>)

Python Example

HFSS Example:

```
oModule.CreateOutputVariable("Var" & OutputQuantity, OutputQuantity, Solution, "Far Fields", ["Context:=", InfiniteSphere, "Domain:=", "Sweep"])
```

Layout Editor Example :

	Set oModule = oDesign.GetModule("OutputVariable") oModule.CreateOutputVariable("test","mag(S(WavePort1,WavePort1))", "Setup1 : LastAdaptive", "Modal Solution Data", ["Domain:=", "Sweep"])

VB Syntax	CreateOutputVariable <*OutputVarName*>, <*Expression*>, <*SolutionName*>, <*reportType* | *solutionType*>, <*contextArray* | *domainArray*>
VB Example	**HFSS Example:** oModule.CreateOutputVariable "Var_" & OutputQuantity, OutputQuantity, Solution, "Far Fields", Array("Context:=", InfiniteSphere, "Domain:=", "Sweep") **Layout Editor Example :** Set oModule = oDesign.GetModule("OutputVariable") oModule.CreateOutputVariable "test","mag(S(WavePort1,WavePort1))", "Setup1 : LastAdaptive", "Modal Solution Data", Array("Domain:=", "Sweep")

DeleteOutputVariable

Deletes an existing output variable. The variable can only be deleted if it is not in use by any traces.

UI Access	**Circuit** > **Results** > **Output Variables**. In the **Output Variables** window, click **Delete**.
Parameters	<table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><*OutputVarName*></td><td>String</td><td>Name of the output variable.</td></tr></table>
Return Value	None.

Python Syntax	DeleteOutputVariable (<*OutputVarName*>)
Python Example	`oModule = oDesign.GetModule("OutputVariable")` `oModule.DeleteOutputVariable ("testNew")`

VB Syntax	DeleteOutputVariable <*OutputVarName*>
VB Example	`Set oModule = oDesign.GetModule("OutputVariable")` `oModule.DeleteOutputVariable "testNew"`

DoesOutputVariableExist

Verifies whether or not a named output variable exists.

UI Access	N/A
Parameters	Name | Type | Description <*outputVariableName*> | String | The output variable name.
Return Value	Boolean True if the variable exists; False if it does not.

Python Syntax	DoesOutputVariableExist(<*outputVariableName*>)
Python Example	`oProject = oDesktop.GetActiveProject()`

	oDesign = oProject.GetActiveDesign() oModule = oDesign.GetModule("OutputVariable") oModule.DoesOutputVariableExist("MyTestVar")

VB Syntax	DoesOutputVariableExist(<*outputVariableName*>)
VB Example	Set oModule = oDesign.GetModule "OutputVariable" oModule.DoesOutputVariableExist "MyTestVar"

EditOutputVariable

Changes the name or expression of an existing output variable.

UI Access	N/A

Parameters

Name	Type	Description
<*OrigVarName*>	String	Name of the original output variable.
<*NewExpression*>	String	New value to assign to the variable.
<*NewVarName*>	String	New name of the variable if any, else pass empty string.
<*SolutionName*>	String	Name of the solution as seen in the output variable UI. For example, "Setup1 : Last Adaptive".
<*ReportType*> or <*SolutionType*>	String	The name of the report type as seen in the output variable UI. For Layout Editor, use the solution type.
<*ContextArray*> or <*DomainArray*>	Array	Structured array containing context for which the output variable expression is being evaluated.

			`Array("Context:=", <string>)` For Layout Editor, use Domain array: `Array("Domain:=", <string>)`
Return Value	None		

Python Syntax	EditOutputVariable (<*OrigVarName*>, <*NewExpression*>, <*NewVarName*>, <*SolutionName*>, <*ReportType* | *SolutionType*>, <*ContextArray* | *DomainArray*>)
Python Example	`oModule = oDesign.GetModule("OutputVariable")` `oModule.EditOutputVariable ("test", "normalize(R1_0.V)", "testNew", "TR", "Standard", [])`

VB Syntax	EditOutputVariable <*OrigVarName*>, <*NewExpression*>, <*NewVarName*>, <*SolutionName*>, <*ReportType* | *SolutionType*>, <*ContextArray* | *DomainArray*>
VB Example	`Set oModule = oDesign.GetModule("OutputVariable")` `oModule.EditOutputVariable "test", "dB(S(WavePort1,WavePort1)) ", "testNew", "Setup1 : LastAdaptive", "Modal Solution Data", Array("Domain:=", "Sweep")`

ExportOutputVariables

Exports output variables to a file.

UI Access	Click on **Export** in the **Output Variables** dialog.

Parameters	<table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><*FileName*></td><td>String</td><td>Name of the file include path.</td></tr></table>
Return Value	Boolean: • **True** - output variable successfully exported. • **False** - error when export output variables.

Python Syntax	ExportOutputVariables(<*FileName*>)
Python Example	`oModule.ExportOutputVariables("C:/output_var.aoutvar")`

VB Syntax	ExportOutputVariables <*FileName*>
VB Example	`oModule.ExportOutputVariables "C:/output_var.aoutvar"`

GetOutputVariables

Returns the list of output variables.

UI Access	N/A
Parameters	None.
Return Value	Array containing all output variables.

Python Syntax	GetOutputVariables()

Python Example	`oDesign.GetOutputVariables()`

VB Syntax	GetOutputVariables
VB 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.

UI Access: N/A

Parameters

Name	Type	Description
<*OutputVarName*>	String	Name of the output variable.
<*IntrinsicVariation*>	String	A set of intrinsic variable value pairs to use when evaluating the output expression. HFSS example: `"Freq='20GHz' Theta='20deg' Phi='30deg'"`
<*SolutionName*>	String	Name of the solution as listed in the output variable UI. For example, `"Setup1 : Last Adaptive"`.
<*ReportType*>	String	The name of the report type as seen in the output variable UI. Possible values in Layout/Schematic Editor are: • "Standard" • "Load Pull" • "Constellation"

			• "Data Table" • "Eye Diagram" • "Statistical"
	<*ContextArray*>	Array	Structured array containing context for which the output variable expression is being evaluated. Can be empty. `Array("Context:=", <string>)`
Return Value	Double value of the output variable.		

Python Syntax	GetOutputVariableValue(<*OutputVarName*>, <*IntrinsicVariation*>, <*SolutionName*>, <*ReportTypeName*>, <*ContextArray*>)
Python Example	`Val = oDesign.GetOutputVariableValue("test","Freq = '20Ghz' Theta='20deg' Phi='30deg'", "TR", "Standard", [])`

VB Syntax	GetOutputVariableValue <*OutputVarName*>, <*IntrinsicVariation*>, <*SolutionName*>, <*ReportTypeName*>, <*ContextArray*>
VB Example	`Val = oDesign.GetOutputVariableValue` `"test","Freq = '20Ghz' Theta='20deg'` `Phi='30deg'", "TR", "Standard", Array()`

ImportOutputVariables

Imports output variables from a file.

UI Access	Click on **Import** in the **Output Variables** dialog.

Parameters	<table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><*FileName*></td><td>String</td><td>Name of the file include path.</td></tr></table>
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")`

VB Syntax	ImportOutputVariables <*FileName*>
VB Example	`oModule.ImportOutputVariables "C:/output_var.aoutvar"`

SimValueContext

SimValueContext holds context information for a trace, and describes how data for a trace should be extracted from the simulation. SimValueContext contains a list of 14 required initial values:

```
SimValueContext (
Domain ID, Calculation Type, Number of Cycles, Rise Time,
Step, Impulse, Context ID, Window Width,
Window Type, TDR Kaiser Parameter, Hold Time, DeviceName,
TDR Step Time, DR Maximum Time )
```

For example, the following indicates a trace in the Time Domain, Standard Calculation with the number of cycles being 2:

```
"SimValueContext:=", Array(1, 0, 2, 0, false, false, -1, 1, 0, 1, 1, "", 0, 0)
```

Additional, context-specific values may follow the required values, as described in subsection 15 below.

1. Domain ID

No Domain	0
Time Domain	1
Spectrum Domain	2
Sweep Domain	3
Device Domain	4
SinglePt Domain	5
LoadPull Domain	6
Transient Domain	7
Budget Domain	8
NetworkFunction Domain	9
Oscillator Domain	55802
Noise Domain	55803
Transfer Function Domain	55807
Time Frequency Domain	55808
Transient Time Domain	55809
Periodic AC Domain	55818
UI Domain	55819
Eye Measurement Domain	55823

Initial Response Domain	55824
Peak Distortion Domain	55825

2. Calculation Type

Standard Calculation	0
Device2_DCIV	1
Device3_DCIV_Output	2
Device3_DCIV_Input	3
Device3_DCIV_Transfer	4
Device3_DCIV_Reverse	5
Device2_ACLoad	6
Device3_ACLoad_Output	7
Device3_ACLoad_Input	8
Device3_ACLoad_Transfer	9
Device3_ACLoad_Reverse	10
Constellation	11
EyeDiagram	12
FreeX (Statistic Report)	13

3. **Number of Cycles** — Used in Time Domain in HarmonicBalance analysis.

4. **Rise Time** — Not used by Designer/Nexxim.

5. **Step** — Not used by Designer/Nexxim.

6. **Impulse** — Not used by Designer/Nexxim.
7. **Context ID** — Not used by Designer/Nexxim.
8. **Window Width** — Not used by Designer/Nexxim.
9. **Window Type** — Not used by Designer/Nexxim.
10. **TDR Kaiser Parameter** — Not used by Designer/Nexxim.
11. **Hold Time** — Not used by Designer/Nexxim.
12. **DeviceName** — Not used by Designer/Nexxim.
13. **TDR Step Time** — Not used by Designer/Nexxim.
14. **TDR Maximum Time** — Not used by Designer/Nexxim.
15. **Context-specific values** — Used in Time Domain in HarmonicBalance analysis.

Context-specific values are entered in the format "key, true/false, keyvalue", where:

— **"key"** is the name of the key being set.

— **"true/false"** indicates whether the key is a string value.

— **"keyvalue"** is the value of the key.

— The order of the context keys is not significant.

— Context keys have software defaults that will be used if not provided in the script.

VB Example:

```
"SimValueContext:=", Array(1, 0, 2, 0, false, false, -1, 1, 0, 1, 1, "", 0, 0,
"DE", false, "0",
"DP", false, "20000000",
"DT", false, "0.001",
```

```
"WE", false, "10ns",
"WM", false, "10ns",
"WN", false, "0ns",
"WS", false, "0ns"))
```

a. Plotting Range for Time domain in Transient and QuickEye analysis:

Description	Key Name	Is a string?	Key Value
Start Time	WS	False	0ns
Stop Time	WE	False	10ns
Minimum Time	WM	False	0ns
Maximum Time	WN	False	10ns
Is Thinning Enabled?	DE	False	0
Dy/dx Tolerance	DT	False	0.001
Number of points	DP	False	20000000

b.Transient report context for Spectral domain in Transient analysis:

Description	Key Name	Is a string?	Key Value
Start Time	TS	False	0ns
Stop Time	TE	False	10ns

Max Harmonics	MH	False	100
Max Frequency	MF	False	*
Window type	WT	False	0
Width Percentage	WW	False	100
Kaiser Parameter	KP	False	0
Adjust Coherent Gain	CG	False	0

* Script can specify either MH or MF. If neither is specified, Max Harmonics is set to 100. If both are specified, MF is used.

Window Type	**ID**
Rectangular	0
Bartlett	1
Blackman	2
Hamming	3
Hanning	4
Kaiser	5
Welch	6
Weber	7
Lanzcos	8

c. Eyeprobe index context for UI domain, Time domain, Eye 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

9 - Reporter Editor Script Commands

Reporter commands should be executed by the oDesign object.

For example:

```
Set oDesign = Project.SetActiveDesign("Circuit1")
Set oModule = oDesign.GetModule("ReportSetup")
```

All Report and Trace properties can be edited using the **ChangeProperty** commands. This includes Title properties, General properties, and Background properties such as border color, fonts, X and Y axis scaling, and number display.

> **Note:**
>
> When you execute **Tools** > **Record Script**, operations performed in the Reporter are automatically recorded.

The list of commands is as follows:

UpdateAllReports

UpdateReports

UpdateTraces

UpdateTracesContextandSweeps

AddAllEyeMeasurements

Displays all the eye measurements in tabular format

UI Access	Right-click the report and select **Trace Characteristics** > **Add All Eye Measurements**
Parameters	Name: *<ReportName>* — Type: String — Description: Name of the report.
Return Value	None.

Python Syntax	AddAllEyeMeasurements(*<ReportName>*)
Python Example	`oModule.AddAllEyeMeasurements("DQS Eye")`

VB Syntax	AddAllEyeMeasurements *<ReportName>*
VB Example	`oModule.AddAllEyeMeasurements "DQS Eye"`

AddCartesianLimitLine

Adds a limit line to a report on the X axis.

<table>
<tr><td>UI Access</td><td>Report2D > Add Limit Line> Specify Points...</td></tr>
<tr><td>Parameters</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.</td></tr>
<tr><td><Def></td><td>Array</td><td>Structured array:
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>
)</td></tr>
</table>
</td></tr>
<tr><td>Return Value</td><td>None.</td></tr>
</table>

Python Syntax	AddCartesianLimitLine (<*ReportName*>, <*Def*>)
Python Example	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"]

	)

VB Syntax	AddCartesianLimitLine <*ReportName*>, <*Def*>
VB Example	oModule.AddCartesianLimitLine "Project Outputs", Array("NAME:CartesianLimitLine", Array("NAME:XValues",0, 2, 5, 7, 10, 15), "XUnits:=", "s", Array("NAME:YValues",0.05, 0.3, 0.65, 0.825, 0.95, 1), "YUnits:=", "mV", "YAxis:=", "Y1")

AddCartesianLimitLineFromCurve

Adds a limit line to a report from selected curve on the plot.

UI Access	**Report2D** > **Add Limit Line** > **From Selected Curve...**
Parameters	<table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><*ReportName*></td><td>String</td><td>Name of the report.</td></tr><tr><td><*LimitLineParams*></td><td>String</td><td>Structured array. Array("NAME:CartesianLimitLineFromCurve", "TraceName:=", <string name of selected trace>, "CurveName:=", <string name of selected curve>,</td></tr></table>

			"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

	])

VB Syntax	AddCartesianLimitLineFromCurve<*ReportName*>, <*LimitLineParams*>
VB Example	oModule.AddCartesianLimitLineFromCurve _ "Variables Plot 2", _ Array("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.

UI Access	**Report2D** > **Add Limit Line** > **Specify Equation...**
Parameters	See table below

Name	Type	Description
<*ReportName*>	String	Name of the report.
<*LimitLineParams*>	Array	Structured array. Array("NAME:CartesianLimitLineFromEquation", "YAxis:=", <integer Y-Axis number>,

			"Start:=", <string start frequency with unit>, "Stop:=", <string end frequency with unit>, "Step:=", <string frequency resolution with unit>, "Equation:=", <string specified equation>)
Return Value	None.		

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"])

VB Syntax	AddCartesianLimitLineFromEquation <*ReportName*>, <*LimitLineParams*>
VB Example	oModule.AddCartesianLimitLineFromEquation _ "S Parameter Plot 1", _

	Array("NAME:CartesianLimitLineFromEquation", _ "YAxis:=", 1, _ "Start:=", "9GHz", "Stop:=", "11GHz", "Step:=", "0.2GHz", _ "Equation:=", "x+1")

AddCartesianXMarker

Adds a marker to a report on the X axis.

UI Access	**Report2D** > **Marker** > **Add X Marker**.
Parameters	Name / Type / Description <*ReportName*> / String / Name of report. <*MarkerName*> / String / Marker name, including any trailing number. <*XValue*> / Double / X coordinate.
Return Value	None

Python Syntax	AddCartesianXMarker (<*ReportName*>, <*MarkerName*>, <*XValue*>)
Python Example	oModule.AddCartesianXMarker ("XY Plot 1", "MX1", 0)

VB Syntax	AddCartesianXMarker <*ReportName*>, <*MarkerName*>, <*XValue*>
VB Example	oModule.AddCartesianXMarker "XY Plot1", "MX1", 0

AddCartesianYMarker

Adds a marker to a report on the Y axis.

UI Access	**Report2D** > **Marker** > **Add Y Marker**.
Parameters	

Name	Type	Description
<*ReportName*>	String	Name of report.
<*MarkerName*>	String	Marker name, including any trailing number.
<*AxisName*>	String	Name of axis.
<*YValue*>	Double	Y coordinate.
<*CurveName*>	String	Name of curve.

Return Value	None

Python Syntax	AddCartesianYMarker (<*ReportName*>, <*MarkerName*>, <*AxisName*>, <*YValue*>, <*CurveName*>)
Python Example	`oModule.AddCartesianYMarker("XY Plot 1", "MY1", "Y1", 0,` `"dB() : Setup1 : Sweep1")`

VB Syntax	AddCartesianYMarker <*ReportName*>, <*MarkerName*>, <*AxisName*>, <*YValue*>, <*CurveName*>
VB Example	`oModule.AddCartesianYMarker "XY Plot 1", "MY1", "Y1", 0, _` `"dB() : Setup1 : Sweep1"`

AddCartesianYMarkerToStack

Adds a marker to a stacked report on the Y axis.

UI Access	N/A
Parameters	

Name	Type	Description
<*ReportName*>	String	Name of report.
<*MarkerName*>	String	Marker name, including any trailing number.
<*AxisName*>	String	Name of axis.
<*YValue*>	Double	Y coordinate.
<*CurveName*>	String	Name of curve.
<*StackOption*>	Array	"Current" to create marker on the current stack. "All" to create marker on all stack.

Return Value	None

Python Syntax	AddCartesianYMarkerToStack (<*ReportName*>, <*MarkerName*>, <*AxisName*>, <*YValue*>, <*CurveName*>, <*StackOption*>)
Python Example	`oModule.AddCartesianYMarker("XY Plot 1", "MY1", "Y1", 0,` `"dB() : Setup1 : Sweep1", ["All"])`

VB Syntax	AddCartesianYMarkerToStack <*ReportName*>, <*MarkerName*>, <*AxisName*>, <*YValue*>, <*CurveName*>, <*StackOption*>
VB Example	`oModule.AddCartesianYMarkerToStack "XY Plot 1", "MY1", "Y1", 0, _` `"dB() : Setup1 : Sweep1", Array("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	Value for the primary solution.
<MarkerName2>	String	Marker name, including any trailing number, for the second marker.
<CurveName2>	String	Full trace name for the second marker.
<PrimarySweepValue2>	String	Value for the primary solution.

Return Value	None

Python Syntax	AddDeltaMarker(*<ReportName>*, *<MarkerName1>*, *<CurveName1>*, *<PrimarySweepValue1>*, *<MarkerName2>*, *<CurveName2>*, *<PrimarySweepValue2>*)
Python Example	`oModule.AddDeltaMarker("VerifEye Eye Plot 1", "m1",` `"AEYEPROBE(OutputEye) : VerifEyeAnalysis : __Amplitude=\'0.49865788706322V\' : Cartesian",` `"0.062", "m2",` `"AEYEPROBE(OutputEye) : VerifEyeAnalysis : __Amplitude=\'0.294343491063066V\' : Cartesian",` `"0.146")`

VB Syntax	AddDeltaMarker <*ReportName*>, <*MarkerName1*>, <*CurveName1*>, <*PrimarySweepValue1*>, <*MarkerName2*>, <*CurveName2*>, <*PrimarySweepValue2*>
VB Example	`oModule.AddDeltaMarker "VerifEye Eye Plot 1", "m1",` `"AEYEPROBE(OutputEye) : VerifEyeAnalysis : __Amplitude=\'0.49865788706322V\' : Cartesian",` `"0.062", "m2",` `"AEYEPROBE(OutputEye) : VerifEyeAnalysis : __Amplitude=\'0.294343491063066V\' : Cartesian",` `"0.146"`

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")`

VB Syntax	AddMarker <*ReportName*>, <*MarkerName*>, <*CurveName*>, <*PrimarySweepValue*>
VB Example	`Set oModule = oDesign.GetModule("ReportSetup")` `oModule.AddMarker "XY Plot1","m1",` `"mag(S(Port1 Port1)) : Setup1:` `LastAdaptive : Cartesian", "0.3in"`

AddNote

Adds a note at a specified location to a given report.

UI Access	Right-click on the plot and select **Add Note**.
Parameters	

Name	Type	Description
<*ReportName*>	String	Name of report
<*NoteDataArray*>	Array	Structured array. `Array("NAME:<StringDataName>", <NoteArray>)`
<*NoteArray*>	Array	Structured array: `Array("NAME:<NoteDataSourceName>",` `"SourceName:=", <string source name>,` `"HaveDefaultPos:=", <boolean True for position 0,0;`

	False to specify below>, "DefaultXPos:=", <int X position for note; 0 for default>, "DefaultYPos:=", <int Y position for note; 0 for default>, "String:=", <string note text>)
Return Value	None

Python Syntax	AddNote (<*ReportName*>, <*NoteDataArray*>)
Python Example	oModule.AddNote("XY Plot1", ["NAME:NoteDataSource", "SourceName:=", "Note1", "HaveDefaultPos:=", False, "DefaultXPos:=", 1996, "DefaultYPos:=", 3177, "String:=", "This is a note."])

VB Syntax	AddNote <*ReportName*> <*NoteDataArray*>
VB Example	`oModule.AddNote "XY Plot1", _` `Array("NAME:NoteDataSource", _` `  "SourceName:=", "Note1", _` `  "HaveDefaultPos:=", false, _` `  "DefaultXPos:=", 1996, _` `  "DefaultYPos:=", 3177, _` `  "String:=", "This is a note.")`

AddTraceCharacteristics

Adds a trace characteristics field to the legend on a report.

UI Access	**Report2D** > **Trace Characteristics** > **All**. This opens the **Add Trace Characteristics** dialog box.

Parameters

Name	Type	Description
<*ReportName*>	String	Name of report.
<*FunctionName*>	String	The function name. See the **Functions** column of the **Add Trace Characteristics** dialog box.
<*FunctionArgs*>	Array	Array containing string values for any function arguments. Pass empty array if no arguments exist. To see which argument(s) a function takes, see the bottom of the **Add Trace Characteristics** dialog box. For function with one argument: `Array(<value>)` 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 Example	`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"])`

VB Syntax	AddTraceCharacteristics *<ReportName>*, *<FunctionName>*, *<FunctionArgs>*, *<RangeArgs*
VB Example	`oModule.AddTraceCharacteristics "XY Plot 2", "delaytime", Array("0"), Array("Full")` `oModule.AddTraceCharacteristics "Differential S-parameters", "prms", Array("0", "0"), Array("Full")` `oModule.AddTraceCharacteristics "Rept2DRectFreq", "distortion", Array("0"), Array( "Specified", "19.5GHz", "20.4GHz")`

AddTraces

Creates a new trace and adds it to the specified report.

<table>
<tr><td>UI Access</td><td colspan="3">Modify Report > Add Trace.</td></tr>
<tr><td rowspan="5">Parameters</td><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><ReportName></td><td>String</td><td>Name of Report</td></tr>
<tr><td><SolutionName></td><td>String</td><td>Name of the solution as listed in the Modify Report dialog box.</td></tr>
<tr><td><ContextArray></td><td>Array</td><td>Context for which the expression is being evaluated. This can be an empty string if there is no context.

Array("Domain:=", <DomainType>)

<DomainType> ex. "Sweep" or "Time"

Array("Context:=", <GeometryType>)

<GeometryType> ex. "Infinite Spheren", "Spheren", "Polylinen"</td></tr>
<tr><td><FamiliesArray></td><td>Array</td><td>Contains sweep definitions for the report.

Array("<VariableName>:= ", <ValueArray>)

<ValueArray>

Array("All") or Array("Value1", "Value2", ..."Valuen")

examples of <VariableName> "Freq", "Theta", "Distance"</td></tr>
<tr><td><ReportDataArray></td><td>Array</td><td>This array contains the report quantity and X, Y, and (Z) axis definitions.

Array("X Component:=", <VariableName>,

"Y Component:=", <VariableName> | <ReportQuantityArray>)</td></tr>
</table>

			<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)"]], [])

VB Syntax	Add Traces <*ReportName*> <*SolutionName*> <*ContextArray*> <*FamiliesArray*> <*ReportDataArray*>
VB Example	oModule.AddTraces "XY Plot1", "Setup1 : Sweep1", _ Array("Domain:=", "Time", "HoldTime:=", 1, "RiseTime:=", 0, _ "StepTime:=", 6.24999373748E-012, "Step:=", false, _

	"WindowWidth:=", 1, _ "WindowType:=", 0, "KaiserParameter:=", 1, _ "MaximumTime:=", 6.2437437437444E-009), _ Array("Time:=", Array("All"), "OverridingValues:=", _ Array("0s","6.24999373748188e-012s", ...)), _ Array("X Component:=", "Time", _ "Y Component:=", Array("TDRZ(WavePort1)")), _ Array()

ApplyReportTemplate

Applies settings to a report from a template file.

UI Access	Right-click on a report, select **Report Templates** > **Apply Settings**.
Parameters	<table><tr><th>Name</th><th>Type</th><th>Description</th></tr><tr><td><*ReportName*></td><td>String</td><td>Name of the report to apply settings to.</td></tr><tr><td><*TemplateFile*></td><td>String</td><td>Template file name with path.</td></tr><tr><td><*PropertyType*></td><td>String</td><td>Property types to apply. ("Graphical" | "Data" | "All")</td></tr></table>
Return Value	None.

Python Syntax	ApplyReportTemplate(<*ReportName*>, <*TemplateFile*>, <*PropertyType*>)
Python Example	oModule.ApplyReportTemplate("Variables Plot 1", "C:/MyTemplate.rpt", "Graphical")

VB Syntax	ApplyReportTemplate <*ReportName*>, <*TemplateFile*>, <*PropertyType*>
VB Example	`oModule.ApplyReportTemplate "Variables Plot 1", "C:/MyTemplate.rpt", "Graphical"`

ChangeProperty[ReportSetup]

Change the properties for a Report.

UI Access	Double-click on a report to change its properties.
Parameters	<table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><*PropertyArray*></td><td>Array</td><td>Varies, depending on the properties associated with the select object command.</td></tr></table>
Return Value	None.

Python Syntax	ChangeProperty(<*PropertyArray*>)
Python Example	`oModule = oDesign.GetModule("ReportSetup")` `  oModule.ChangeProperty(` `  [` `  "NAME:AllTabs",` `  [` `  "NAME:General",`

```
      [
      "NAME:PropServers",
      "XY Plot 1:General"
      ],
      [
      "NAME:ChangedProps",
      [
      "NAME:Use Scientific Notation",
      "Value:=", True
      ]
      ]
   ]
])

oModule = oDesign.GetModule("ReportSetup")
   oModule.ChangeProperty(
      [
      "NAME:AllTabs",
      [
        "NAME:Legend",
        [
```

```
        "NAME:PropServers",
        "S Parameter Plot 1:Legend"
        ],
        [
        "NAME:ChangedProps",
        [
        "NAME:Legend Name",
        "Value:=" , "Ansys"
        ]
    ]
    ]
])
    oModule.ChangeProperty(
    [
        "NAME:AllTabs",
        [
        "NAME:General",
        [
        "NAME:PropServers",
        "S Parameter Plot 1:General"
```

```
        ],
      [
        "NAME:ChangedProps",
        [
          "NAME:Auto Scale Fonts",
          "Value:=" , False
        ]
      ]
    ]
])
```

VB Syntax	ChangeProperty <*PropertyArray*>
VB Example	oModule.ChangeProperty Array("NAME:AllTabs", Array("NAME:Cartesian", Array ("NAME:PropServers", _ "XY Plot 1:CartesianDisplayTypeProperty"), Array("NAME:ChangedProps", Array("NAME:Show X Scrollbar", "Value:=", _ true)))) oModule.ChangeProperty Array("NAME:AllTabs", Array("NAME:General", Array("NAME:PropServers", _ "XY Plot 1:General"), Array("NAME:ChangedProps", Array("NAME:Use Scientific Notation", "Value:=", _ true))))

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")`

VB Syntax	ClearAllMarkers <*ReportName*>
VB Example	`oModule.ClearAllMarkers "XY Plot 1"`

ClearAllTraceCharacteristics

Clears all trace characteristics from the legend in a report.

UI Access	**Report2D** > **Trace Characteristics** > **Clear All**.
Parameters	Name | Type | Description *<PlotName>* | String | Name of the plot

Return Value	None

Python Syntax	ClearAllTraceCharacteristics(<*PlotName*>)
Python Example	`oModule.ClearAllTraceCharacteristics("XY Plot 1")`

VB Syntax	ClearAllTraceCharacteristics <*PlotName*>
VB Example	`oModule.ClearAllTraceCharacteristics "XY Plot 1"`

CloneReportsFromDatasetSolution

Clones a report for a 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")`

VB Syntax	CloneReportsFromDatasetSolution <*ReportsToClone*>, <*SoluNameToUse*>
VB Example	`oModule.CloneReportsFromDatasetSolution _` `Array("Rectangular Plot1"), "DatasetSolution_rsm_5812"`

CopyPlotSettings

Copies settings of a specified plot.

UI Access	Right-click a report, select **Copy**
Parameters	Name | Type | Description <*ReportName*> | String | Name of specified report.
Return Value	None.

Python Syntax	CopyPlotSettings(<*ReportName*>)
Python Example	`oModule.CopyPlotSettings("Plot 1")`

VB Syntax	CopyPlotSettings <*ReportName*>
VB Example	`oModule.CopyPlotSettings "Plot 1"`

CopyReportDefinitions

Copy the definition of a report for paste operations.

UI Access	Select a report in the Project tree, right-click and select **Copy Definition**
Parameters	Name | Type | Description <*ReportsArray*> | Array | Names of reports from which to copy the definitions
Return Value	None

Python Syntax	CopyReportDefinitions(<*ReportsArray*>)
Python Example	`oModule.CopyReportDefinitions(["Transmission", "Reflection"])`

VB Syntax	CopyReportDefinitions <*ReportsArray*>
VB Example	`oModule.CopyReportDefinitionsv _` `Array("Transmission", "Reflection")`

CopyReportsData

Copy all data corresponding to the specified reports.

UI Access	Select a report in the Project tree, right-click and select **Copy Data**
Parameters	Name | Type | Description <*ReportsArray*> | Array | Names of reports from which to copy data

Return Value	None

Python Syntax	CopyReportsData (<*ReportsArray*>)
Python Example	`oModule.CopyReportsData (["Transmission","Reflection"])`

VB Syntax	CopyReportsData <*ReportsArray*>
VB Example	`oModule.CopyReportsData _` `Array("Transmission","Reflection")`

CopyTraceDefinitions

Copy trace definitions for a paste operation.

UI Access	Select a trace in the Project tree, right-click and select **Copy Definition**
Parameters	<table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><*ReportName*></td><td>String</td><td>Name of Report</td></tr><tr><td><*TracesArray*></td><td>Array</td><td>Trace definitions to copy</td></tr></table>
Return Value	None

Python Syntax	CopyTraceDefinitions(<*ReportName*>, <*TracesArray*>)
Python Example	`oModule.CopyTraceDefinitions ("Transmission",`

	["mag(S(Port1,Port2))"])

VB Syntax	CopyTraceDefinitions <*ReportName*>, <*TracesArray*>
VB Example	oModule.CopyTraceDefinitions "Transmission", _ Array("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))"])

VB Syntax	CopyTracesData <*ReportName*>, <*TracesArray*>
VB Example	oModule.CopyTracesData _

	`"C11", Array("mag(S(Port1,Port2))")`

CreateReport

Creates a new report with a single trace and adds it to the **Results** branch in the project tree.

UI Access	Right-click on **Results** > **Create [Type] Report**

Parameters

Name	Type	Description
<ReportName>	String	Name of report.
<ReportType>	String	Type of report. Possible values are: "Modal S Parameters" - Only for Driven Modal solution-type problems with ports. "Terminal S Parameters" - Only for Driven Terminal solution-type problems with ports. "Eigenmode Parameters" - Only for Eigenmode solution-type problems. "Fields" "Far Fields" - Only for problems with radiation or PML boundaries. "Near Fields" - Only for problems with radiation or PML boundaries. "Emission Test"
<DisplayType>	String	Type of display. If ReportType is "Modal S Parameters", "Terminal S Parameters", or "Eigenmode Parameters", then set to one of the following: `"Rectangular Plot", "Polar Plot", "Radiation Pattern", "Smith Chart", "Data Table", "3D Rectangular`

		Plot", "3D Rectangular Bar Plot" or "3D Polar Plot". If <ReportType> is "Fields", then set to one of the following: "Rectangular Plot", "Polar Plot", "Radiation Pattern", "Rectangular Contour Plot", "Data Table", "3D Rectangular Plot", or "3D Rectangular Bar Plot". If <ReportType> is "Far Fields" or "Near Fields", then set to one of the following: "Rectangular Plot", "Radiation Pattern", "Rectangular Contour Plot", "Data Table", "3D Rectangular Plot", "3D Rectangular Bar Plot" or "3D Polar Plot" If <ReportType> is "Emission Test", then set to one of the following: "Rectangular Plot" or "Data Table"
<*SolutionName*>	String	Name of the solution as listed in the **Modify Report** dialog box.
<*ContextArray*>	Array	Context for which the expression is being evaluated. This can be an empty string if there is no context. Array("Domain:=", <DomainType>) <DomainType> ex. "Sweep" or "Time" Array("Context:=", <GeometryType>) <GeometryType> ex. "Infinite 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	CreateReport(<*ReportName*>, <*ReportType*>, <*DisplayType*>, <*SolutionName*>, <*ContextArray*>, <*FamiliesArray*>, <*ReportDataArray*>, <*ExtendedInfo*>)
Python Example	Three examples are given below: (nominal, Optimetrics, spectral) **Nominal Example:** `oModule.CreateReport( "XY Plot 2", "Standard",` `"Rectangular Plot", "TR",` `[` `"NAME:Context",` `"SimValueContext:=", [], 1, 0, 2, 0, false,` `false, -1, 1, 0, 1, 1, "", 0, 0]` `],` `[` `"Time:=" , ["All"],` `"aaa:=" , ["Nominal"]` `],`

```
        [
                "X Component:=", "Time",
                "Y Component:=", ["E1.I"]
        ],
        [])
```

Optimetrics Example:

```
oModule.CreateReport ("XY Plot 3", "Standard",
        "Rectangular Plot", "TR",
        [
                "NAME:Context",
                "OptiSetup:=", "ParametricSetup1", "SimValueContext:=",
                [
                        1, 0, 2, 0, false, false, 0, 1,0, 1, 1, "", 0, 0
                ]
        ],
        [
                "Time:=", ["All"], "aaa:=", ["Nominal"]
        ],
        [
                "X Component:="  , "Time",
                "Y Component:="  , ["E1.I"]
        ],
        [])
```

Spectral Example:

```
oModule.CreateReport ("XY Plot 4", "Standard",
        "Rectangular Plot", "TR",
        [
                "NAME:Context",
                "SimValueContext:=",
```

```
                [
                        2, 0, 2, 0, false, false, -1, 1, 0, 1, 1, "",
                        0, 0, "CG", false, "0", "KP",false, "0", "MH",
                        false, "100", "TE", false, "40ms", "TH", false,
                        "40", "TS", false, "0ns", "UF", false, "0",
                        "WT", false, "0", "WW", false, "100"
                ]
        ],
        [
                "Spectrum:="   , ["All"], "aaa:=", ["Nominal"]
        ],
        [
                "X Component:=","Spectrum",
                "Y Component:=", ["mag(E1.I)"]
        ],
        [])
```

Partial Discharge Example

```
oModule.CreateReport("Partial Discharge Plot 2",
        "Partial Discharge", "Rectangular Plot",
        "PartialDischarge1 : PartialDischarge", [],
        [
                "GasPressure:=", ["All"],
                "Freq:="       , ["All"],
                "bend_angle:=" , ["All"]
        ],
        [
                "X Component:=", "GasPressure",
                "Y Component:=", ["Power"]
        ])
```

Example 3D Rectangular Bar Plot with Change Bar properties

```
oModule = oDesign.GetModule("ReportSetup")
oModule.CreateReport("S Parameter Plot 4", "Modal Solution Data",
       "3D Rectangular Bar Plot", "Setup1 : Sweep", [],
       [
              "Freq:="       , ["All"],
              "UU:="         , ["All"],
              "ZZZ:="        , ["Nominal"]
       ],
       [
              "X Component:=", "Freq",
              "Y Component:=", "UU",
              "Z Component:=", ["dB(S(wDipole1_1_p1,wDipole1_1_p1))"]
       ])

oModule.ChangeProperty(
       [
              "NAME:AllTabs",
              [
                     "NAME:Bar",
                     [
                            "NAME:PropServers"
                            "S Parameter Plot 4:Traces-dB(S(wDipole1_1_p1,wDipole1_1_p1))"
                     ],
                     [
```

```
                                "NAME:ChangedProps",
                                [
                                        "NAME:Transparency",
                                        "Value:=", "0.5"
                                ]
                        ]
                ]
        ])

oModule.ChangeProperty(
        [
                "NAME:AllTabs",
                [
                        "NAME:Bar",
                        [
                                "NAME:PropServers",
                                "S Parameter Plot 4:Traces-dB(S(wDipole1_1_p1,wDipole1_1_p1))"
                        ],
                        [
                                "NAME:ChangedProps",
                                [
                                        "NAME:Filled",
                                        "Value:=", False
                                ]
                        ]
```

```
hangeProperty(
        [
                "NAME:AllTabs",
                [
                        "NAME:Bar",
                        [
                                "NAME:PropServers",
                                "S Parameter Plot 4:Traces-dB(S(wDipole1_1_p1,wDipole1_1_p1))"
                        ],
                        [
                                "NAME:ChangedProps",
                                [
                                        "NAME:Outline Color",
                                        "R:=", 0,
                                        "G:=", 0,
                                        "B:=", 160
                                ]
                        ]
                ]
        ])

oModule.ChangeProperty(
        [
                "NAME:AllTabs",
                [
                        "NAME:Bar",
                        [
                                "NAME:PropServers",
                                "S Parameter Plot 4:Traces-dB(S(wDipole1_1_p1,wDipole1_1_p1))"
                        ],
                        [
```


VB Syntax	CreateReport <*ReportName*>, <*ReportType*>, <*DisplayType*>, <*SolutionName*>, <*ContextArray*>, <*FamiliesArray*>, <*ReportDataArray*>, <*ExtendedInfo*
VB Example	`oModule.CreateReport "3D Cartesian Plot1", "Far Fields",_` `  "3D Cartesian Plot", "Setup1 : LastAdaptive", _` `  Array("Context:=", "Infinite Sphere1", "Domain:=", "Sweep"), _` `  Array("Theta:=", Array("All"), "Phi:=", Array("All"), _` `  "Freq:=", Array("10GHz")), _` `  Array("X Component:=", "Theta", _` `  "Y Component:=", "Phi", _` `  "Z Component:=", Array("rETotal")), _` `  Array()` `oModule.CreateReport "ReptSmithFreq",_` `  "Modal Solution Data", "Smith Plot", "Setup1 : Sweep1", _` `  Array("Domain:=", "Sweep"), _` `  Array("Freq:=", Array("All")), _` `  Array("Polar Component:=", _` `  Array("ln(Y(LumpPort1,LumpPort1))")), _` `  Array()` `oModule.CreateReport "Rectangular Contour Plot 2", "Far Fields", _` `  "Rectangular Contour Plot", "Setup1 : LastAdaptive", Array("Context:=", _` `  "Infinite Sphere1"), Array("_u:=", Array("All"), "_v:=", Array("All"), "Freq:=",` `Array( _`

```
  "5GHz")), Array("X Component:=", "_u", "Y Component:=", "_v", "Z Component:=", Array
( _
  "rETotal")), Array()

oModule.CreateReport "Rept2DRectFreq",_
  "Modal Solution Data", "XY Plot", _
  "Setup1 : Sweep1", _
  Array("Domain:=", "Sweep"), _
  Array("Freq:=", Array("All")), _
  Array("X Component:=", "Freq", _
  "Y Component:=", _ Array("dB(S(LumpPort1,LumpPort1))")), _
  Array()
```

CreateReport [Designer]

*Use:*Creates a new report with a single trace and adds it to the **Results** branch in the project tree.

*Command:*Product Menu>Results>Create <type> Report

*Syntax:*CreateReport <ReportName> <ReportType> <DisplayType> <SolutionName> <ContextArray> <FamilesArray> <ReportDataArray>

*Return Value:*None

Parameters:<ReportName>

Type: <string>

Name of Report.

<ReportType>

Type: <string>

Possible values are:

"Standard" - For most plot types.

"Load Pull" - For load pull plots.

"Constellation" - For constellation plots.

"Data table" - For data tables.

"Eye Diagram" - For eye diagrams.

"Statistical" - For statistical plots.

`<DisplayType>`

Type: <string>

Possible values are:

"Rectangular Plot", "Polar Plot", "Radiation Pattern", "Smith Chart", "Data Table", "3D Rectangular Plot", "3D Polar Plot", or "Rectangular Stacked Plot".

`<SolutionName>`

Type: <string>

Name of the solution as listed in the **Modify Report** dialog box.

For example: "Setup1 : Last Adaptive"

`<ContextArray>`

Type: Array of strings

Context for which the expression is being evaluated. This can be an empty string if there is no context.

Array("Domain:=", <DomainType>)

`<DomainType>`

ex. "Sweep" or "Time"

`Array("Context:=", <SimValueContext>)`

Context for the trace. For more information see SimValueContext.

`<FamiliesArray>`

Type: Array of strings

Contains sweep definitions for the report.

Array("<VariableName>:= ", <ValueArray>)

`<ValueArray>`

Array("All") or Array("Value1", "Value2", ..."Value*n*")

examples of <VariableName>

"Freq", "Theta", "Distance"

`<ReportDataArray>`

Type: Array of strings

This array contains the report quantity and X, Y, and (Z) axis definitions.

Array("X Component:=", <VariableName>, "Y Component:=", <VariableName> | <ReportQuantityArray>)

<ReportQuantityArray>

ex. Array("dB(S(Port1, Port1))")

VB Example:

```
oModule.CreateReport "XY Stacked Plot 1", "Standard", "Rectangular Stacked Plot", _
"LinearFrequency", Array("NAME:Context", "SimValueContext:=", Array(3, 0, 2, 0, _
false, false, -1, 1, 0, 1, 1, "", 0, 0)), Array("F:=", Array("All")), Array("X Component:=", _
"F", "Y Component:=", Array("dB(S(Port1,Port1))", "dB(S(Port1,Port2))", _
"dB(S(Port2,Port1))", "dB(S(Port2,Port2))")), Array()
```

VB Example:

```
oModule.CreateReport "Data Table 1", "Standard", "Data Table", "LinearFrequency",_
Array("NAME:Context", "SimValueContext:=", Array( _
3, 0, 2, 0, false, false, -1, 1, 0, 1, 1, "", 0, 0)), Array("F:=", Array("All")), Array("X Com-
ponent:=", _
"F", "Y Component:=", Array("dB(S(Port1,Port1))")), Array()
```

VB Example:

```
oModule.CreateReport "3D Rectangular Plot 1", "Standard", "3D Rectangular Plot", _
```

```
"LinearFrequency", Array("NAME:Context", "SimValueContext:=", Array(3, 0, 2, 0, _
false, false, -1, 1, 0, 1, 1, "", 0, 0)), Array("F:=", Array("All")), Array("X Component:=", _
"F", "Y Component:=", "F", "Z Component:=", Array("dB(S(Port1,Port1))")), Array()
```

VB Example:

```
oModule.CreateReport "3D Rectangular Plot 2", "Standard", "3D Rectangular Plot", _
"LinearFrequency", Array("NAME:Context", "SimValueContext:=", Array(3, 0, 2, 0, _
false, false, -1, 1, 0, 1, 1, "", 0, 0)), Array("F:=", Array("All")), Array("X Component:=", _
"F", "Y Component:=", "F", "Z Component:=", Array("dB(S(Port1,Port1))")), Array()
```

Python Syntax	CreateReport(<data_block>)
Python Example	oModule = oDesign.GetModule("ReportSetup") oModule.CreateReport("QuickEye Voltage Plot 1", "Eye Diagram", "Rectangular Plot", "QuickEyeAnalysis", ["NAME:Context", "SimValueContext:=" , [1,0,2,0,False,False, -1,1,0,1,1,"",0,0,"DE",False,"0","DP",False, "20000000","DT",False,"0.001","NUMLEVELS",False,"1","PCID", False,"3","PID",False,"0", "WE",False,"49.99995us","WM",False,"49.99995us","WN",False, "0ps","WS",False,"0ps"]

```
],
[
  "Time:=" , ["All"]
],
[
  "Component:=" , ["aeyeprobe3"]
],
[
  "Unit Interval:=" , "1e-009s",
  "Offset:=" , "0",
  "Auto Delay:=" , True,
  "Manual Delay:=" , "0ps",
  "AutoCompCrossAmplitude:=", True,
  "CrossingAmplitude:=" , "0mV",
  "AutoCompEyeMeasurementPoint:=", True,
  "EyeMeasurementPoint:=" , "5e-010s"
])
```

CreateReportFromTemplate

Creates a report from a saved template.

UI Access	**[product]** > **Results** > **Report Templates** > **PersonalLib** > [TemplateName]
Parameters	Name | Type | Description <*TemplatePath*> | String | Path to report template
Return Value	None

Python Syntax	CreateReportFromTemplate(<*TemplatePath*>)
Python Example	`oModule.CreateReportFromTemplate(` `"C:/MyHFSSProjects/PersonalLib/ReportTemplates/TestTemplate.rpt")`

VB Syntax	CreateReportFromTemplate <*TemplatePath*>
VB 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.

UI Access	NA
Parameters	Name | Type | Description <*ReportType*> | String | Report type name as input parameter <*DisplayType*> | String | Display type name as input parameter <*SolutionName*> | String | Solution name as input parameter <*SimValueCtxt*> | String | A context name, or array of string that encoded the context(I). <*CategoryName*> | String | Category name as input parameter

	<PointSet>	Array	Array of strings(II)
	<CommonComponentsOfTraces>	Array	Array of strings(III)
	<ExtTraceInfo>	Array	Array of strings(IV)
Return Value	None.		

Python Syntax	CreateReportOfAllQuantities(*<ReportNameArg>*, *<ReportType>*, *<DisplayType>*, *<SolutionName>*, *<SimValueCtxt>*, *<CategoryName>*, *<PointSet>*, *<CommonComponentsOfTraces>*, *<ExtTraceInfo>*)
Python Example	`oModule.CreateReportOfAllQuantities("Smith Chart all", "Modal Solution Data", "Smith Chart", "Setup1 : LastAdaptive", [],"S Parameter",["Freq:=", ["All"], "offset:=", ["All"],"a:=", ["Nominal"],"b:=", ["Nominal"]],[],[])`

VB Syntax	CreateReportOfAllQuantities *<ReportNameArg>*, *<ReportType>*, *<DisplayType>*, *<SolutionName>*, *<SimValueCtxt>*, *<CategoryName>*, *<PointSet>*, *<CommonComponentsOfTraces>*, *<ExtTraceInfo>*
VB Example	`oModule.CreateReportOfAllQuantities "Smith Chart all", _ "Modal Solution Data", "Smith Chart", "Setup1 : LastAdaptive", _ Array(),"S Parameter", _`

	`Array("Freq:=", Array("All"), "offset:=", Array("All"), _` `"a:=", Array("Nominal"),"b:=", Array("Nominal")), _` `Array(), Array()`

DeleteMarker

*Use:*Deletes the specified marker.

UI Access	[product] > **Fields** > **Fields** > **Marker** > **Delete Marker**.
Parameters	Name / Type / Description <*MarkerName*> / String / Name of the marker.
Return Value	None.

Python Syntax	DeleteMarker(<*MarkerName*>)
Python Example	`oModule.DeleteMarker("m1")`

VB Syntax	DeleteMarker <*MarkerName*>
VB Example	`oModule.DeleteMarker "m1"`

DeleteAllReports

Deletes all existing reports.

UI Access	Right-click the report to delete in the project tree, and then click **Delete All Reports** on the shortcut menu.
Parameters	None.
Return Value	None.

Python Syntax	DeleteAllReports()
Python Example	`oModule.DeleteAllReports()`

VB Syntax	DeleteAllReports
VB Example	`oModule.DeleteAllReports`

DeleteReports

Deletes an existing report or reports.

UI Access	Right-click the report to delete in the project tree, and then click **Delete** on the shortcut menu
Parameters	Name | Type | Description <*ReportNameArray*> | Array | Array of report names to be deleted
Return Value	None.

Python Syntax	DeleteReports(<*ReportNameArray*>)

Python Example	`oModule.DeleteReports (["Rept2DRectFreq"])`

VB Syntax	DeleteReports <*ReportNameArray*>
VB Example	`oModule.DeleteReports Array("Rept2DRectFreq")`

DeleteTraceCharacteristics

Deletes a trace characteristics field from a report

UI Access	N/A
Parameters	Name / Type / Description <*ReportName*> / String / Name of the report to delete from. <*TraceCharsNames*> / Array / Array of trace characteristics to delete.
Return Value	None.

Python Syntax	DeleteTraceCharacteristics(<*ReportName*>, <*TraceCharsNames*>)
Python Example	`oModule.DeleteTraceCharacteristics("Variables Plot 1", ["lowercutoff", "gain-crossover"])`

VB Syntax	DeleteTraceCharacteristics <*ReportName*>, <*TraceCharsNames*>
VB Example	`oModule.DeleteTraceCharacteristics "Variables Plot 1", _` `Array("lowercutoff", "gaincrossover")`

DeleteTraces

Deletes an existing traces or traces.

<table>
<tr><td>UI Access</td><td>Right-click the Trace to delete in the project tree, and then click Delete on the shortcut menu</td></tr>
<tr><td>Parameters</td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><*TraceSelection*></td><td>Array</td><td>Structured array define selections.
<code>Array("<ReportName>:=", <TracesArray>, <TracesArray>,...)</code></td></tr>
<tr><td><*ReportName*></td><td>String</td><td>Name of Report</td></tr>
<tr><td><*TracesArray*></td><td>Array</td><td>Contains the traces to delete within a report
<code>Array(<Trace>, <Trace>, ...)</code></td></tr>
<tr><td><*Trace*></td><td>String</td><td>A specific trace that the user wishes to delete</td></tr>
</table>
</td></tr>
<tr><td>Return Value</td><td>None.</td></tr>
</table>

Python Syntax	DeleteTraces(<*TraceSelection*>)
Python Example	`oModule.DeleteTraces (["XY Plot 1:=",` `["dB(S(LumpPort1,LumpPort1))"],` `"XY Plot 2:=", ["Mag_E"]])`

VB Syntax	DeleteTraces <*TraceSelection*>

VB Example	`oModule.DeleteTraces Array("XY Plot 1:=", _` `Array("dB(S(LumpPort1,LumpPort1))"), )` `"XY Plot 2:=", Array("Mag_E"))`

DoesSupportTraceCharacteristics

Determines whether trace characteristics is supported in a specified display type.

UI Access	N/A
Parameters	Name | Type | Description *<DisplayType>* | String | Specified display type to check.
Return Value	Integer • **1** - trace characteristics is supported. • **0** - trace characteristics not supported.

Python Syntax	DoesSupportTraceCharacteristics(*<DisplayType>*)
Python Example	`oModule.DoesSupportTraceCharacteristics("Rectangular Plot")`

VB Syntax	DoesSupportTraceCharacteristics *<DisplayType*
VB Example	`oModule.DoesSupportTraceCharacteristics "Rectangular Plot"`

DumpAllReportsData

Dumps all reports data to an Ansoft report data file.

UI Access	N/A
Parameters	Name | Type | Description *<FileName>* | String | File name with path.
Return Value	None.

Python Syntax	DumpAllReportsData(*<FileName>*)
Python Example	`oModule.DumpAllReportsData("C:/ReportsData.rdat")`

VB Syntax	DumpAllReportsData *<FileName>*
VB Example	`oModule.DumpAllReportsData "C:/ReportsData.rdat"`

EditCartesianXMarker

Edits an XMarker value.

UI Access	N/A
Parameters	Name | Type | Description *<ReportName>* | String | Name of report. *<MarkerName>* | String | Marker name, including any trailing number. *<XValue>* | Double | X coordinate.
Return Value	None.

Python Syntax	EditCartesianXMarker (<*ReportName*>, <*MarkerName*>, <*XValue*>)
Python Example	`oModule.EditCartesianXMarker ("XY Plot 1", "MX1", 0)`

VB Syntax	EditCartesianXMarker <*ReportName*>, <*MarkerName*>, <*XValue*>
VB Example	`oModule.EditCartesianXMarker "XY Plot1", "MX1", 0`

EditCartesianYMarker

Edits a YMarker value.

UI Access	N/A
Parameters	

Name	Type	Description
<*ReportName*>	String	Name of report.
<*MarkerName*>	String	Marker name, including any trailing number.
<*AxisName*>	String	Name of axis.
<*YValue*>	Double	Y coordinate.
<*CurveName*>	String	Name of curve.

Return Value	None

Python Syntax	EditCartesianYMarker (<*ReportName*>, <*MarkerName*>, <*AxisName*>, <*YValue*>, <*CurveName*>)
Python Example	`oModule.EditCartesianYMarker("XY Plot 1", "MY1", "Y1", 0,` `"dB() : Setup1 : Sweep1")`

VB Syntax	EditCartesianYMarker <*ReportName*>, <*MarkerName*>, <*AxisName*>, <*YValue*>, <*CurveName*>
VB Example	`oModule.EditCartesianYMarker "XY Plot 1", "MY1", "Y1", 0, _` `"dB() : Setup1 : Sweep1"`

EditMarker

Edits a marker on a report.

UI Access	N/A
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	EditMarker(<*ReportName*>, <*MarkerName*>, <*CurveName*>, <*PrimarySweepValue*>)
Python Example	`oModule.EditMarker("XY Plot 1", "m5",` `"GS1.VAL : TR4 : Cartesian", "3.61599999999997s")`

VB Syntax	EditMarker <*ReportName*>, <*MarkerName*>, <*CurveName*>, <*PrimarySweepValue*>

VB Example	`Set oModule = oDesign.GetModule("ReportSetup")` `oModule.EditMarker "XY Plot1","m1",` `"mag(S(Port1 Port1)) : Setup1:` `LastAdaptive : Cartesian", "0.3in"`

ExportEyeMaskViolation

Exports eye mask violations to a file.

UI Access	N/A
Parameters	Name | Type | Description *<ReportName>* | String | Name of specified report. *<ExportFileName>* | String | File name to export to.
Return Value	N/A

Python Syntax	ExportEyeMaskViolation(*<ReportName>*, *<ExportFileName>*)
Python Example	`oModule.ExportEyeMaskViolation("Variables Plot 1", "C:/eyemask1.csv")`

VB Syntax	ExportEyeMaskViolation *<ReportName>*, *<ExportFileName>*
VB Example	`oModule.ExportEyeMaskViolation "Variables Plot 1", "C:/eyemask1.csv"`

ExportImageToFile [Reporter]

Exports a report image in a specified format. In Release 23.1, this command is fully supports -ng (non-graphical) mode.

UI Access	N/A
Parameters	<table><tr><th>Name</th><th>Type</th><th>Description</th></tr><tr><td><*ReportName*></td><td>String</td><td>Name of report to be exported.</td></tr><tr><td><*FileName*></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><*Width*></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><*Height*></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></table>
Return Value	None.

Python Syntax	ExportImageToFile(<*ReportName*>, <*FileName*>, <*Width*>, <*Height*>)
Python Example	`oModule.ExportImageToFile(` `"Rectangular Contour Plot 1",` `"D:/work/2015/UV-Export/pp1.gif",` `0, 0)`

VB Syntax	ExportImageToFile <*ReportName*>, <*FileName*>, <*Width*>, <*Height*>
VB Example	`oModule.ExportImageToFile _` `"Rectangular Contour Plot 1", _` `"D:/work/2015/UV-Export/pp1.gif", 0, 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 supports export overlay of polar plot 3D with existing transformation (scaling, rotation and translation) in -ng (non-graphical) mode.

<table>
<tr><td>UI Access</td><td colspan="3">Modeler > Export.</td></tr>
<tr><td rowspan="5">Parameters</td><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><path></td><td>String</td><td>Full file path including extension.</td></tr>
<tr><td><width></td><td>Integer</td><td>Width in pixels (use 0 for default).</td></tr>
<tr><td><height></td><td>Integer</td><td>Height in pixels (use 0 for default).</td></tr>
<tr><td><Parameters></td><td>Array</td><td>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>

			"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 then AutoFit is true, it makes sure color key does not overlap field plot. It is true 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`

```
  ("C:/Users/jdoe/Desktop/export.bmp",
  1920,
  1080,
[
  "NAME:SaveImageParams",
    "ShowAxis:=" , "True",
    "ShowGrid:=" , "True",
    "ShowRuler:=" , "True",
    "ShowRegion:=" , "Default",
    "Selections:=" , "",
    "FieldPlotSelections:=" , "",
    "FitToSelections:=" , "",
    "FitToFieldPlotSelections:=", "",
    "AutoFit:=" , "True",
    "Orientation:=" , ""
])
```

VB Syntax	ExportModelImageToFile <*path*> <*width*> <*height*> <*Parameters*>
VB Example	`oEditor.ExportModelImageToFile` `"C:/Users/jdoe/Desktop/export.bmp",` `1383,`

```
        512,
        Array("NAME:SaveImageParams",
                "ShowAxis:=", "True",
                "ShowGrid:=", "True",
                "ShowRuler:=", "True",
                "ShowRegion:=", "Default",
                "Selections:=", "",
                "FitToFieldPlotSelections:=", "",
                "AutoFit:="             , "True",
                "Orientation:=", ""
```

ExportModelMeshToFile

Exports geometry model to a 3D model file (e.g. *.obj, *.wrl, etc.).

UI Access	N/A
Parameters	

Name	Type	Description
<*filePath*>	String	Full file path, including extension *.obj, *.wrl, etc
<*selections*>	Array	Selected parts.

Return Value	None.

Python Syntax	ExportModelMeshToFile <*filePath*>, <*selections*>)

Python Example	`oEditor.ExportModelMeshToFile("E:/MyDir/scriptRun/2Selected-ng.obj", ["BotCover-","AveragingVolumeAtPeakRMSEfieldLocation"])`

VB Syntax	ExportModelMeshToFile <*filePath*>, <*selections*>
VB 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")

VB Syntax	ExportPlot3DToFile <*plotName*> <*Path*>
VB Example	oModule.ExportPlot3DToFile("rE Plot 1", "D:/projects/test2-output/rEPlot1.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

VB Example:

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

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

Set oProject = oDesktop.SetActiveProject("BJTinverter")
Set oDesign = oProject.SetActiveDesign("Nexxim1")
oDesign.ExportReport "Data Table 1", "table_test", "csv"
```

Python Syntax	ExportReport(<ReportName>, <FileName>)
Python Example	`oDesign.ExportReport(` `"Plot1", "c:\report1.dat")`

ExportReportDataToFile

Exports report data to a file.

UI Access	N/A
Parameters	Name | Type | Description <*ReportName*> | String | Name of specified report. <*ExportFileName*> | String | Name of export file. File extenstion "rdat" is expected.
Return Value	None.

Python Syntax	ExportReportDataToFile(<*ReportName*>, <*ExportFileName*>)
Python Example	`oModule.ExportReportDataToFile("Plot 1", "C:/Plot1data.rdat")`

VB Syntax	ExportReportDataToFile <*ReportName*>, <*ExportFileName*>
VB Example	`oModule.ExportReportDataToFile "Plot 1", "C:/Plot1data.rdat"`

ExportTableToFile

Exports a marker table from a report to a file.

UI Access	Right-click on a plot, select **Marker** > **Export Marker Table...** or **Export Delta Marker Table...**
Parameters	<table><tr><th>Name</th><th>Type</th><th>Description</th></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.</td></tr><tr><td><*TableType*></td><td>String</td><td>Type of marker table to export. "Marker" or "DeltaMarker".</td></tr></table>
Return Value	None.

Python Syntax	ExportTableToFile(<*ReportName*>, <*ExportFileName*>, <*TableType*>)
Python Example	`oModule.ExportTableToFile("Plot 1", "C:/Marker.csv", "Marker")`

VB Syntax	ExportTableToFile <*ReportName*>, <*ExportFileName*>, <*TableType*>
VB Example	`oModule.ExportTableToFile "Plot 1", "C:/Marker.csv", "Marker"`

ExportToFile

> **Note:**
>
> The ExportToFile script command has replaced the script command ExportReport. ExportReport remains in order to retain backward compatibility for existing scripts, but it is strongly recommended that you now use ExportToFile.

From a data table or plot, generates text format, comma delimited, tab delimited, or .dat type output files.

<table>
<tr><td>UI Access</td><td>Right-click on report name in the project tree and select Export Data.</td></tr>
<tr><td>Parameters</td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><ReportName></td><td>String</td><td>Name of report to be exported.</td></tr>
<tr><td><FileName></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><UnitSpec></td><td>String</td><td>For example, “kV, Mhz, yd”</td></tr>
<tr><td><UseTraceNumberFormat></td><td>Boolean</td><td>"True", "False"</td></tr>
</table>
</td></tr>
<tr><td>Return Value</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)

VB Syntax	ExportToFile <*ReportName*>, <*FileName*>
VB Example	`oModule.ExportToFile "rETotal", "C:/Documents/rETotal.csv"`

ExportToFile [Reporter]

From a data table or plot, generates text format, comma delimited, tab delimited, .dat, or .rdat type output files.

UI Access	Right-click on report name in the Project tree and select **Export**.
Parameters	<table><tr><th>Name</th><th>Type</th><th>Description</th></tr><tr><td><ReportName></td><td>String</td><td>Name of the report</td></tr><tr><td><FileName></td><td>String</td><td>Path and File Name Supported formats<table><tr><td>.txt</td><td>Post processor format file</td></tr><tr><td>.csv</td><td>Comma-delimited data file</td></tr><tr><td>.tab</td><td>Tab-separated file</td></tr><tr><td>.dat</td><td>Ansys plot data file</td></tr><tr><td>.rdat</td><td>Ansys report data file</td></tr></table></td></tr></table>
Return Value	None

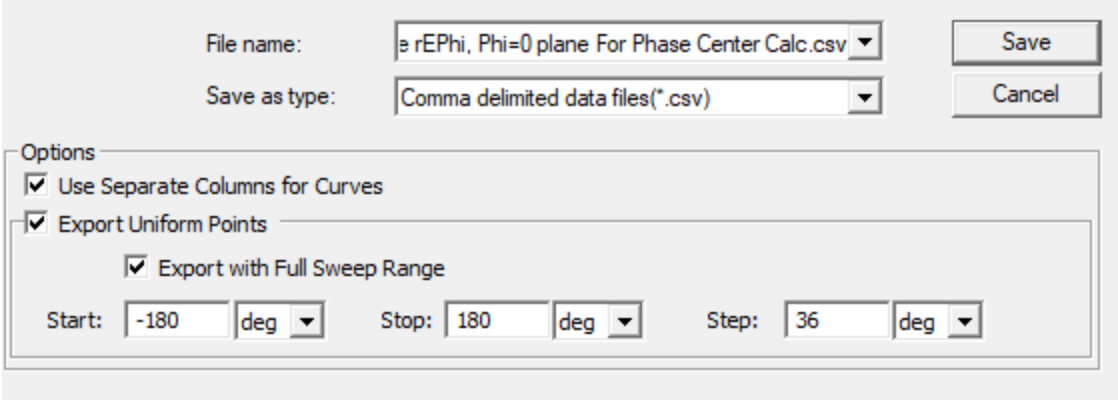
Python Syntax	ExportToFile (<ReportName>, <FileName>)
Python Example	`oModule.ExportToFile(` `"Plot1", "c:\report1.dat")`

VB Syntax	ExportToFile <ReportName>, <FileName>
VB 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.



UI Access	Right-click on report name in the project tree and select **Export Data**.
Parameters	

Name	Type	Description
<ReportName>	String	Name of report to be exported.
<FileName>	String	Full path of the exported image file name; with extension of .txt - Post processor format file .csv - Comma-delimited data file .tab - Tab-separated file

			.dat - Ansys plot data file
	<SweepRange>	String	Start, stop, and step range with units, for sweep.
	<UnitSpec>	String	For example, “kV, Mhz, yd”
	<UseTraceNumberFormat>	Boolean	"True", "False"
Return Value	None.		

Python Syntax	ExportUniformPointsToFile(<*ReportName*>, <*FileName*>, <SweepRange>)
Python Example	`oModule.ExportUniformPointsToFile("S Parameter Table 2", "D:/MyFiles/cftt.csv", "4GHz", "5GHz", "1GHz", False, " kV, MHz, yd ", False)`

VB Syntax	ExportUniformPointsToFile <*ReportName*>, <*FileName*>
VB Example	`oModule.ExportUniformPointsToFile "rETheta and rEPhi [dB]", "C:/MyFiles/rETheta and rEPhi [dB].csv", "-180deg", "180deg", "36deg"`

GetAllCategories

Get all available category names (not including variable and output-variables) in a solution for a report type and display type, returned as an array of text strings.

UI Access	NA		
Parameters	Name	Type	Description
	<*ReportType*>	String	Report type name.
	<*DisplayType*>	String	Name of display type.
	<*SolutionName*>	String	Name of solution.

	<*SimValueCtxt*>	Array	A context name, or array of strings that encode the contexts.
Return Value	Array of text strings		

Python Syntax	GetAllCategories(<*ReportType*>, <*DisplayType*>, <*SolutionName*>, <*SimValueCtxt*>)
Python Example	`oModule.GetAllCategories("Far Fields",` `"Rectangular Plot", "Setup1 : LastAdaptive", "Infinite Sphere1")`

VB Syntax	GetAllCategories <*ReportType*>, <*DisplayType*>, <*SolutionName*>, <*SimValueCtxt*>
VB 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.

UI Access	NA		
Parameters	Name	Type	Description
	<*ReportType*>	String	Report type name.
	<*DisplayType*>	String	Display type name.
	<*SolutionName*>	String	Name of solution.
	<*SimValueCtxt*>	Array	A context name, or array of string that encoded the contexts(I).

	<*CategoryName*>	String	A category name as input parameter. a category name returned in GetAllCategories() or "Variables", or "Output Variables"
Return Value	Array of text strings		

Python Syntax	GetAllQuantities(<*ReportType*>,<*DisplayType*>, <*SolutionName*>, <*SimValueCtxt*>, <*CategoryName*>)
Python Example	`oModule.GetAllQuantities(` `"Far Fields", "Rectangular Plot",` `"Setup1 : LastAdaptive",` `"Infinite Sphere1", "Gain")`

VB Syntax	GetAllQuantities <*ReportType*>,<*DisplayType*>, <*SolutionName*>, <*SimValueCtxt*>, <*CategoryName*>
VB 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()`

VB Syntax	GetAllReportNames
VB Example	`oModule.GetAllReportNames`

GetAvailableDisplayTypes

Retrieves all supported display types in report type as an array of text strings.

UI Access	N/A
Parameters	Name | Type | Description <*ReportType*> | String | Report type name.
Return Value	Array of text strings

Python Syntax	GetAvailableDisplayTypes(<*ReportType*>)
Python Example	`oModule.GetAvailableDisplayTypes("Far Fields")`

VB Syntax	GetAvailableDisplayTypes <*ReportType*>
VB 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()`

VB Syntax	GetAvailableReportTypes
VB Example	`oModule.GetAvailableReportTypes`

GetAvailableSolutions

Gets all available solutions in report type as an array of text strings.

UI Access	N/A

Parameters	<table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><*ReportType*></td><td>String</td><td>Report type name.</td></tr></table>
Return Value	Array of text strings

Python Syntax	GetAvailableSolutions(<*ReportType*>)
Python Example	`oModule.GetAvailableSolutions("Far Fields")`

VB Syntax	GetAvailableSolutions <*ReportType*>
VB 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()`

VB Syntax	GetChildNames
VB 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.

UI Access	NA		
Parameters	Name	Type	Description
	<*ObjectPath*>	String	Report Name or an object path beginning with a report name.
Return Value	A ReportSetup(Results) Module Child Object, [ReportSetup(Results) Module Child Objects]		

Python Syntax	GetChildObject(<*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")`

VB Syntax	GetChildObject <*ObjectPath*>
VB Example	`set oRpt = oRptModule.GetChildObject "S Parameter Plot 1"`

	`set oTrace = oRptModule.GetChildObject "S Parameter Plot 1/dB(S(Port1,Port1))"` `set oAxisX = oRptModule.GetChildObject "S Parameter Plot 1/AxisX"`

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()`

VB Syntax	GetChildTypes
VB 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	<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><*TraceName*></td><td>String</td><td>Name of specified trace.</td></tr></table>
Return Value	Array of string containing the PropServer name.

Python Syntax	GetCurvePropServerName(<*ReportName*>, <*TraceName*>)
Python Example	`oModule.GetCurvePropServerName("Plot 1", "Phase")`

VB Syntax	GetCurvePropServerName <*ReportName*>, <*TraceName*>
VB Example	`oModule.GetCurvePropServerName "Plot 1", "Phase"`

GetDisplayType

Gets the display type of a specified report.

UI Access	NA
Parameters	<table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><*ReportName*></td><td>String</td><td>Report name</td></tr></table>
Return Value	String containing display type.

Python Syntax	GetDisplayType(<*ReportName*>)
Python Example	`oModule.GetDisplayType("Design Plot 1")`

VB Syntax	GetDisplayType <*ReportName*>
VB Example	`oModule.GetDisplayType "Design Plot 1"`

GetDynLinkIntrinsicVariables

Gets variable names from a trace included in dynamic link outputs.

UI Access	N/A
Parameters	Name | Type | Description <*TraceName*> | String | Trace name with its report name.
Return Value	Array of variable names

Python Syntax	GetDynLinkIntrinsicVariables(<*TraceName*>)
Python Example	`oModule.GetDynLinkIntrinsicVariables("Plot 1:Trace")`

VB Syntax	GetDynLinkIntrinsicVariables <*TraceName*>
VB Example	`oModule.GetDynLinkIntrinsicVariables "Plot 1:Trace")`

GetDynLinkQtyValueState

Gets the state of the quantity values from a source dynamic linked trace.

UI Access	N/A
Parameters	<table><tr><th>Name</th><th>Type</th><th>Description</th></tr><tr><td><*TraceName*></td><td>String</td><td>Name of specified source trace.</td></tr><tr><td><*QtyName*></td><td>String</td><td>Name of specified quantity value.</td></tr></table>
Return Value	Array of strings containing states.

Python Syntax	GetDynLinkQtyValueState(<*TraceName*>, <*QtyName*>)
Python Example	`oModule.GetDynLinkQtyValueState("Plot 1:Trace1", "")`

VB Syntax	GetDynLinkQtyValueState <*TraceName*>, <*QtyName*>
VB Example	`oModule.GetDynLinkQtyValueState "Plot 1:Trace1", ""`

GetDynLinkTraces

Gets the names of the dynamic linked traces.

UI Access	N/A
Parameters	<table><tr><th>Name</th><th>Type</th><th>Description</th></tr><tr><td><*SoluName*></td><td>String</td><td>Name of the source solution. If empty, refer to current solution.</td></tr></table>
Return Value	Array of strings containing trace names.

Python Syntax	GetDynLinkTraces(<*SoluName*>)
Python Example	`oModule.GetDynLinkTraces("")`

VB Syntax	GetDynLinkTraces <*SoluName*>
VB Example	`oModule.GetDynLinkTraces ""`

GetDynLinkVariableValues

Gets the values of a variable from a dynamic linked trace.

UI Access	N/A
Parameters	Name | Type | Description <*TraceName*> | String | Name of specified dynamic linked trace, with its report name. <*VarName*> | String | Name of specified variable.
Return Value	Array of strings containing variable values.

Python Syntax	GetDynLinkVariableValues(<*TraceName*>, <*VarName*>)
Python Example	`oModule.GetDynLinkVariableValues("Plot 1:Trace", "Var1")`

VB Syntax	GetDynLinkVariableValues <*TraceName*>, <*VarName*>
VB Example	`oModule.GetDynLinkVariableValues "Plot 1:Trace", "Var1"`

GetName

Returns the design ID and design name of the active design, in that order separated by a semicolon. If the active design is a sub-circuit, it also returns the parent ID and parent design name, listed before the active design information.

To get the *only* the design's name without the parent design name or design IDs, see oDesign.GetDesignName().

UI Access	N/A
Parameters	None.
Return Value	String indicating the name and ID of the active design and parent design (if the active design is a sub-circuit).

Python Syntax	GetName()
Python Example	oDesign.GetName( ) for a subcircuit *EMDesign2* in a parent design *EMDesign1* design_name = oDesign.GetName() 1;EMDesign1/2;EMDesign2 In the returned string: `1` is the parent design ID; `EMDesign1` is the parent design name. `2` is the active design ID; `EMDesign2` is the active design name.

VB Syntax	GetName

VB Example	`design_name = oDesign.GetName`

GetObjPath [Design]

Obtains the path to the design.

UI Access	N/A
Parameters	None.
Return Value	String containing the path to the design.

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

VB Syntax	GetObjPath
VB 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>UI Access</td><td>N/A</td></tr>
<tr><td>Parameters</td><td>
<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td><PropTab></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><PropServer></td><td>String</td><td>An object identifier, generally returned from another script method, such as <code>Comp:inst@R;2;3</code></td></tr>
<tr><td><PropName></td><td>String</td><td>Name of the property.</td></tr>
</table>
</td></tr>
<tr><td>Return Value</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") ...

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

VB Syntax	GetPropNames
VB Example	`oModule.GetPropNames`

GetPropValue [Report Setup]

Gets the property value for a Report, or reports' child object.

UI Access	N/A
Parameters	Name | Type | Description *<PropPath>* | String | A child object's property path. See property path discussion here.
Return Value	String containing the property value.

Python Syntax	GetPropValue(*<PropPath>*)
Python Example	`oModule.GetPropValue("S Parameter Plot 1/Display Type")`

VB Syntax	GetPropValue *<PropPath>*
VB Example	`oModule.GetPropValue "S Parameter Plot 1/Display Type"`

GetQtyExpressionsForSourceTrace

Gets the quantity expressions from a specified source trace.

UI Access	N/A
Parameters	Name | Type | Description <*SourceTraceName*> | String | Name of specified source trace.
Return Value	Array of strings containing quantity expressions.

Python Syntax	GetQtyExpressionsForSourceTrace(<*SourceTraceName*>)
Python Example	`oModule.GetQtyExpressionsForSourceTrace("Plot 1:Trace1")`

VB Syntax	GetQtyExpressionsForSourceTrace <*SourceTraceName*>
VB Example	`oModule.GetQtyExpressionsForSourceTrace "Plot 1:Trace1"`

GetReportTraceNames

Gets the names of existing trace names in a plot.

UI Access	N/A
Parameters	Name | Type | Description <*PlotName*> | String | Name of specified plot.
Return Value	Array of strings containing trace names.

Python Syntax	GetReportTraceNames(<*PlotName*>)
Python Example	`oModule.GetReportTraceNames("SParameter Plot 1")`

VB Syntax	GetReportTraceNames <*PlotName*>
VB 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")`

VB Syntax	GetReportSummaryForRegressionTesting <*ReportName*>
VB 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	<table><tr><th>Name</th><th>Type</th><th>Description</th></tr><tr><td><*ReportType*></td><td>String</td><td>Report type name.</td></tr><tr><td><*DisplayType*></td><td>String</td><td>Display type name.</td></tr><tr><td><*SolutionName*></td><td>String</td><td>Name of solution.</td></tr></table>
Return Value	Array of text strings

Python Syntax	GetSolutionContexts(<*ReportType*>, <*DisplayType*>, <*SolutionName*>)
Python Example	`oModule.GetSolutionContexts(` `"Far Fields", "Rectangular Plot", "Setup1:LastAdaptive")`

VB Syntax	GetSolutionContexts <*ReportType*>, <*DisplayType*>, <*SolutionName*>
VB 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.

<table>
<tr><td>UI Access</td><td>N/A</td></tr>
<tr><td>Parameters</td><td>
<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td><reportTypeArg></td><td>String</td><td>Report type name as input parameter.</td></tr>
<tr><td><solutionNameArg></td><td>String</td><td>Solution name as input parameter.</td></tr>
<tr><td><simValueCtxtArg></td><td>Structured Array</td><td>Same as ContextArray values created in the relevant CreateReport script.</td></tr>
<tr><td><familiesArg></td><td>Array of Strings</td><td>Same as FamiliesArray values created in the relevant CreateReport script.</td></tr>
<tr><td><expressionArg></td><td>String or Array of Strings</td><td>Text string or array of text strings; valid expression, may validate it as the data-table Y-component.</td></tr>
</table>
</td></tr>
<tr><td>Return Value</td><td>ARRAY of ISolutionDataResultComInterface objects, containing:
GetSweepNames()
GetSweepUnits()
GetSweepValues()
IsPerQuantityPrimarySweep()
GetDataExpressions() – should be the same as <expressionArg>
GetPerQuantityPrimarySweepValues()
IsDataComplex()
GetDataUnits()
GetRealDataValues()
GetImagDataValues()
ReleaseData()
GetDesignVariableNames()</td></tr>
</table>

	• 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))'])`

VB Syntax	GetSolutionDataPerVariation(reportTypeArg, solutionNameArg, simValueCtxtArg, familiesArg, expressionArg)
VB Example	`set solutionData = oModule.GetSolutionDataPerVariation(`

	"Modal Solution Data", "Setup1 : Sweep", Array("Domain:=", "Sweep"), Array("Freq:=", Array("All"), "offset:=", Array("All")), Array("S(Port1,Port1)", "dB(S(Port1,Port3))"))

GetDataExpressions

Returns data expressions.

> **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	GetDataExpressions()
Python Example	`expressions = oModule.GetDataExpressions()`

VB Syntax	GetDataExpressions()
VB Example	`dim expressions` `expressions = oModule.GetDataExpressions()`

GetDataUnits

Returns text string containing units.

> **Important:**
>
> This is a member function of ISolutionDataResultComInterface object, which is the element of the returned array from function GetSolutionDataPerVariation.

UI Access	N/A
Parameters	Name | Type | Description <expressionString> | String | Can be returned from GetDataExpressions()
Return Value	Text string of units; empty if no units

Python Syntax	GetDataUnits(<expressionString>)
Python Example	`oModule.GetDataUnits(expressions)`

VB Syntax	GetDataUnits(<expressionString>)

VB 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()`

VB Syntax	GetDesignVariableNames()
VB Example	`dim names` `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.

UI Access	N/A
Parameters	Name | Type | Description <varName> | String | Can be returned from GetDesignVariableNames()
Return Value	Text string of units; empty if no units.

Python Syntax	GetDesignVariableUnits(<varName>)
Python Example	`units = oModule.GetDesignVariableUnits('Variable Name')`

VB Syntax	GetDesignVariableUnits(<varName>)
VB Example	`dim units` `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.

UI Access	N/A
Parameters	<varName> (String) — Can be returned from GetDesignVariableNames() <siValue> (Boolean) — True to return SI-value; False to return by GetDesignVariableUnits()
Return Value	Double value

Python Syntax	GetDesignVariableValue(<varName>, <siValue>)
Python Example	`value = oModule.GetDesignVariableValue('varName',1)`

VB Syntax	GetDesignVariableValue(<varName>, <siValue>)
VB Example	`dim value` `value = oModule.GetDesignVariableValue("varName", True)`

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()`

VB Syntax	GetDesignVariationKey()
VB Example	`dim key` `set key = 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.

UI Access	N/A
Parameters	<table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><expressionString></td><td>String</td><td>Can be returned from GetDataExpressions()</td></tr><tr><td><siValue></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	GetImagDataValues(<expressionString>,<siValue>)
Python Example	`imaginaryvalues = oModule.GetImagDataValues('expression',1)`

VB Syntax	GetImagDataValues(<expressionString>,<siValue>)
VB Example	`dim imaginaryvalues` `imaginaryvalues = oModule.GetImagDataValues("expression",True)`

GetPerQuantityPrimarySweepValues

Returns per quantity primary sweep values.

Important:

This 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><expressionString></td><td>String</td><td>Can be returned from GetDataExpressions().</td></tr><tr><td><siValue></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)`

VB Syntax	GetPerQuantitySweepValues(<expressionString>, <siValue>)
VB Example	`dim sweepvalues` `sweepvalues = oModule.GetPerQuantitySweepValues("0.111,0.201,0.345,0.231", True)`

GetRealDataValues

Returns array of real data values.

Important:

This 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><expressionString></td><td>String</td><td>Can be returned from GetDataExpressions()</td></tr><tr><td><siValue></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)`

VB Syntax	GetRealDataValues(<expressionString>,<siValue>)
VB Example	`dim realvalues` `realvalues = oModule.GetRealDataValues("expression",True)`

GetSweepNames

Returns array of text strings containing primary sweep name(s).

Important:

This 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()`

VB Syntax	GetSweepNames()
VB Example	`dim sweepnames` `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')`

VB Syntax	GetSweepUnits(<sweepName>)
VB Example	`dim sweepunits` `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.

UI Access	N/A
Parameters	<table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><sweepName></td><td>String</td><td>Primary sweep name</td></tr><tr><td><siValue></td><td>Boolean</td><td>True to return SI-value; False to return by GetSweepUnits().</td></tr></table>
Return Value	Array of doubles

Python Syntax	GetSweepValues(<sweepName>, <siValue>)
Python Example	`sweepvalues = oModule.GetSweepValues('Sweep 1', True)`

VB Syntax	GetSweepValues(<sweepName>, <siValue>)
VB Example	`dim sweepvalues` `sweepvalues = oModule.GetSweepValues("Sweep 1", True)`

IsDataComplex

Returns whether an expression is complex.

Important:

This is a member function of ISolutionDataResultComInterface object, which is the element of the returned array from function GetSolutionDataPerVariation.

UI Access	NA
Parameters	Name | Type | Description <expressionString> | String | Can be returned from GetDataExpressions().
Return Value	Boolean (True if expression is Complex data; False if not)

Python Syntax	IsDataComplex(<expressionString>)
Python Example	`oModule.IsDataComplex('.001,.234,.455,.434')`

VB Syntax	IsDataComplex(<expressionString>)
VB 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()`

VB Syntax	IsPerQuantityPrimarySweep()
VB Example	`dim var` `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()`

VB Syntax	ReleaseData()
VB Example	`oModule.ReleaseData()`

GroupPlotCurvesByGroupingStrategy

Groups curves in a Stacked Plot automatically based on a curve grouping strategy.

UI Access	N/A
Parameters	

Name	Type	Description
<ReportName>	String	Name of report.
<GroupStrategy>	String	Strategy for grouping, "Single", "By Trace" or "By Units".

Return Value	None.

Python Syntax	GroupPlotCurvesByGroupingStrategy(<*ReportName*>, <*GroupStrategy*>)
Python Example	`oModule.GroupPlotCurvesByGroupingStrategy("Transient Plot 1", "By Trace")`

VB Syntax	GroupPlotCurvesByGroupingStrategy <*ReportName*>, <*GroupStrategy*>
VB Example	`oModule.GroupPlotCurvesByGroupingStrategy "Transient Plot 1", "By Trace"`

ImportIntoReport

Imports .tab, .csv, and .dat format files into a report.

<table>
<tr><td>UI Access</td><td colspan="3">Right-click on report name in the Project tree and select Import....</td></tr>
<tr><td>Parameters</td><td colspan="3">
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><*ReportName*></td><td>String</td><td>Name of the Report</td></tr>
<tr><td><*FileName*></td><td>String</td><td>Path and File Name
<table>
<tr><td>.csv</td><td>Comma-delimited data file</td></tr>
<tr><td>.tab</td><td>Tab-separated file</td></tr>
<tr><td>.dat</td><td>Ansys plot data file</td></tr>
</table>
</td></tr>
</table>
</td></tr>
<tr><td>Return Value</td><td colspan="3">None</td></tr>
</table>

Python Syntax	ImportIntoReport (<*ReportName*>, <*FileName*>)
Python Example	`oDesign.ImportIntoReport ("Plot1", "c:\report1.dat")`

VB Syntax	ImportIntoReport <*ReportName*>, <*FileName*>
VB 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")`

VB Syntax	ImportReportDataIntoReport <*ReportName*>, <*FileName*>
VB Example	`oModule.ImportReportDataIntoReport "Plot 1", "C:/Plot1data.rdat"`

MovePlotCurvesToGroup

In a Stacked Plot move curve(s) from its stack(s) to an existing stack. Here term ‘group’ is synonymous to ‘stack’ in the context of cartesian stacked plot.

UI Access	N/A
Parameters	<table><tr><th>Name</th><th>Type</th><th>Description</th></tr><tr><td><*ReportName*></td><td>String</td><td>Name of report.</td></tr><tr><td><*CurveArray*></td><td>Array</td><td>Array of curve names to move.</td></tr><tr><td><*StackName*></td><td>String</td><td>Name of stack to move to.</td></tr></table>
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")`

VB Syntax	MovePlotCurvesToGroup <*ReportName*>, <*CurveArray*>, <*StackName*>
VB Example	`oModule.MovePlotCurvesToGroup _` `"XY Stacked Plot 1", _` `Array("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><th>Name</th><th>Type</th><th>Description</th></tr><tr><td><*ReportName*></td><td>String</td><td>Name of report.</td></tr><tr><td><*CurveArray*></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"])`

VB Syntax	MovePlotCurvesToNewGroup <*ReportName*>, <*CurveArray*>
VB Example	`oModule.MovePlotCurvesToNewGroup _` `"XY Stacked Plot 1", _` `Array("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()`

VB Syntax	OpenWindowForAllReports
VB Example	`oModule.OpenWindowForAllReports`

OpenWindowForReports

Opens windows for specified reports.

UI Access	N/A
Parameters	Name | Type | Description <*ReportNames*> | Array | Array of strings containing report names.
Return Value	None.

Python Syntax	OpenWindowForReports(<*ReportNames*>)

Python Example	`oModule.OpenWindowForReports(["Report1", "Report2"])`

VB Syntax	OpenWindowForReports <*ReportNames*>
VB Example	`oModule.OpenWindowForReports Array("Report1", "Report2")`

PastePlotSettings

Paste plot settings to a specified report.

UI Access	Right-click a report, select **Paste**
Parameters	

Name	Type	Description
<*ReportName*>	String	Name of specified report.
<*PropTypeToApply*>	String	Property type to paste. "Graphical", "Data", or "All".

Return Value	None.

Python Syntax	PastePlotSettings(<*ReportName*>, <*PropTypeToApply*>)
Python Example	`oModule.PastePlotSettings("Plot 1", "Graphical")`

VB Syntax	PastePlotSettings <*ReportName*>, <*PropTypeToApply*>
VB Example	`oModule.PastePlotSettings "Plot 1", "Graphical"`

PasteReports

Paste copied reports to results in the current project.

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

Python Syntax	PasteReports ()
Python Example	`oModule.PasteReports()`

VB Syntax	PasteReports
VB 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()`

VB Syntax	PasteReportsWithLegacyNames
VB 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")`

VB Syntax	PasteTraces <*ReportName*>

VB Example	`oModule.PasteTraces "XY Plot1"`

PasteTracesWithLegacyNames

Pastes copied traces to a named plot using legacy name definitions.

UI Access	N/A
Parameters	Name | Type | Description <*ReportName*> | String | Name of plot
Return Value	None

Python Syntax	PasteTracesWithLegacyNames (<*ReportName*>)
Python Example	`oModule.PasteTracesWithLegacyNames("XY Plot1")`

VB Syntax	PasteTracesWithLegacyNames <*ReportName*>
VB Example	`oModule.PasteTracesWithLegacyNames "XY Plot1"`

RenameReport

Renames an existing report.

UI Access	Select a report on the Project tree, right-click and select **Rename**

Parameters	<table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><*OldReportName*></td><td>String</td><td>Old Report Name</td></tr><tr><td><*NewReportName*></td><td>String</td><td>New Report Name</td></tr></table>
Return Value	None.

Python Syntax	RenameReport (<*OldReportName*>, <*NewReportName*>)
Python Example	`oModule.RenameReport("XY Plot1", "Reflection")`

VB Syntax	RenameReport <*OldReportName*>, <*NewReportName*>
VB Example	`oModule.RenameReport "XY Plot1", "Reflection"`

RenameTrace

To rename a trace in a plot

UI Access	N/A
Parameters	<table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><*ReportName*></td><td>String</td><td>Name of report.</td></tr><tr><td><*TraceName*></td><td>String</td><td>Name of Trace</td></tr><tr><td><*NewName*></td><td>String</td><td>New trace name.</td></tr></table>
Return Value	None.

Python Syntax	RenameTrace(<*ReportName*>, <*TraceName*>, <*NewName*>)
Python Example	`oModule.RenameTrace ("XY Plot1",` `"dB(S(WavePort1,WavePort1))1", "Port1dbS")`

VB Syntax	RenameTrace <*ReportName*>, <*TraceName*>, <*NewName*>
VB Example	`oModule.RenameTrace "XY Plot1", _` `"dB(S(WavePort1,WavePort1))1", _` `"Port1dbS"`

ResetPlotSettings

Resets plot settings to defaults.

UI Access	Right-click on a plot, select **Edit** > **Reset Plot Settings**
Parameters	Name | Type | Description <*PlotName*> | String | Name of specified plot.
Return Value	None.

Python Syntax	ResetPlotSettings(<*PlotName*>)
Python Example	`oModule.ResetPlotSettings("Differential S-parameters")`

VB Syntax	ResetPlotSettings <*PlotName*>
VB Example	`oModule.ResetPlotSettings("Differential S-parameters")`

SavePlotSettingsAsDefault

Saves report plot settings as default.

UI Access	**Report Templates** > **Save Settings as Default**
Parameters	Name | Type | Description <*PlotName*> | String | Name of plot to use for plot defaults.
Return Value	None.

Python Syntax	SavePlotSettingsAsDefault ("<*PlotName*>")
Python Example	`oModule.SavePlotSettingsAsDefault ("XY Plot1")`

VB Syntax	SavePlotSettingsAsDefault "<*PlotName*>"
VB Example	`oModule.SavePlotSettingsAsDefault "XY Plot 1"`

SetLinkOutputTraces

Specifies dynamic link output traces from the current design.

UI Access	Right-click the **Results**, select **Link Output...**
Parameters	Name | Type | Description

	<*TraceArray*>	Array	Array of traces to set. `Array("<ReportName>:=", <array of trace names>,` `"<ReportName>:=", <array of trace names>,...)`
Return Value	None.		

Python Syntax	SetLinkOutputTraces(<*TraceArray*>)
Python Example	`oModule.SetLinkOutputTraces(` `[` `  "Plot 1:=", ["Trace1"],` `  "Plot 2:=", ["Trace1"]` `])`

VB Syntax	SetLinkOutputTraces <*TraceArray*>
VB Example	`oModule.SetLinkOutputTraces _` `Array("Plot 1:=", Array("Trace1"), _` `  "Plot 2:=", Array("Trace1"))`

SetPropValue [Report Setup]

Sets the property value for report module child object.

UI Access	Select **Edit Properties** on Report objects.
Parameters	<table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><*PropPath*></td><td>String</td><td>A child object's property path. See property path discussion here.</td></tr><tr><td><*NewValue*></td><td>String, Number, or Boolean</td><td>New value data type is depending on the property type,</td></tr></table>
Return Value	True if the property is found and the new value is valid. Otherwise return False.

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")`

VB Syntax	SetPropValue <*PropPath*>, <*NewValue*>
VB 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.

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

	<ReportName>	String	Name of report.
	<GroupName>	String	Stack group name.
Return Value	None.		

Python Syntax	UnGroupPlotCurvesInGroup(*<ReportName>*, *<GroupName>*)
Python Example	`oModule.UngroupPlotCurvesInGroup("S Parameter Plot 3", "Stack 1")`

VB Syntax	UnGroupPlotCurvesInGroup *<ReportName>*, *<GroupName>*
VB Example	`oModule.UngroupPlotCurvesInGroup "S Parameter Plot 3", "Stack 1"`

UpdateAllReports

Updates all reports in the **Results** branch in the project tree.

UI Access	Right-click on **Results** in the project tree, select **Update All Reports**
Parameters	None
Return Value	None

Python Syntax	UpdateAllReports()
Python Example	`oModule.UpdateAllReports()`

VB Syntax	UpdateAllReports
VB Example	`oModule.UpdateAllReports`

UpdateReports

Updates specified reports.

UI Access	N/A
Parameters	Name | Type | Description <*ReportNames*> | Array | Array of strings containing report names.
Return Value	None.

Python Syntax	UpdateReports(<*ReportNames*>)
Python Example	`oModule.UpdateReports (["XY Plot 1", "XY Plot 4"])`

VB Syntax	UpdateReports <*ReportNames*>
VB Example	`oModule.UpdateReports Array("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><td>UI Access</td><td>In Report dialog, click Apply Traces button</td></tr>
<tr><td>Parameters</td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><ReportName></td><td>String</td><td>Name of Report.</td></tr>
<tr><td><TraceNames></td><td>Array</td><td>Array of strings containing trace names.</td></tr>
<tr><td><SolutionName></td><td>String</td><td>Name of the solution.</td></tr>
<tr><td><ContextArray></td><td>Array</td><td>Context for which the expression is being evaluated. This can be an empty string if there is no context.
Array("Domain:=", <DomainType>)
<DomainType> ex. "Sweep" or "Time"
Array("Context:=", <GeometryType>)
<GeometryType> ex. "Infinite Spheren", "Spheren", "Polylinen"</td></tr>
<tr><td><FamiliesArray></td><td>Array</td><td>Contains sweep definitions for the report.
Array("<VariableName>:= ", <ValueArray>)
<ValueArray>
Array("All") or Array("Value1", "Value2", ..."Valuen")
examples of <VariableName> "Freq", "Theta", "Distance"</td></tr>
<tr><td><ReportDataArray></td><td>Array</td><td>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))")</td></tr>
<tr><td><ExtTraceInfo></td><td>Array</td><td>Optional. Array defines extended trace information.</td></tr>
</table>
</td></tr>
</table>

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)"]], [])

VB Syntax	UpdateTraces <*ReportName*>, <*SolutionName*>, <*ContextArray* >, <*FamiliesArray*>, <*ReportDataArray*>
VB Example	`oModule.UpdateTraces "XY Plot1", Array("dB(S(WavePort1,WavePort1))"), _` `"Setup1 : Sweep1", Array("Domain:=", "Time", "HoldTime:=", 1, _` `"RiseTime:=", 0, "StepTime:=", 0, "Step:=", false, _` `"WindowWidth:=", 1, "WindowType:=", 0, "KaiserParameter:=",1, _` `"MaximumTime:=", 0), Array("Time:=", Array("All")), _` `Array("X Component:=", "Time",_` `"Y Component:=", Array("dB(S(WavePort1,WavePort1))")), _` `Array()`

UpdateTracesContextAndSweeps

Edits sweeps and context of multiple traces without affecting their component expressions.

UI Access	**Modify Report** with multiple traces selected.		
Parameters	Name	Type	Description
	<*ReportName*>	String	Name of Report.
	<*TraceNames*>	Array	Array of strings containing trace names.
	<*SolutionName*>	String	Name of the solution as listed in the **Modify Report** dialog box. For example: `"Setup1 : Last Adaptive"`
	<*ContextArray*>	Array	Context for which the expression is being evaluated. This can be an empty string if there is no context. ex. "Sweep" or "Time"

	<PointSet>	Array	Point set for the selected traces, for example, X and Y values for the plot.
Return Value	None		

Python Syntax	UpdateTracesContextAndSweeps(<*ReportName*>, <*TraceNames*>, <*SolutionName*>, <*ContextArray*>, <*PointSet*>)
Python Example	`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"]])`

VB Syntax	UpdateTracesContextAndSweeps <*ReportName*>, <*TraceNames*>, <*SolutionName*>, <*ContextArray*>, <*PointSet*>

VB Example	oModule.UpdateTracesContextAndSweeps _ "Active S Parameter Quick Report",_ Array("dB(ActiveS(Port1:1))", "dB(ActiveS(Port2:1))"),_ "Setup1 : Sweep1", Array(), _ Array("Freq:=", _ Array("9GHz", "9.05GHz", "9.1GHz", _ "9.15GHz", "9.2GHz", _ "9.25GHz", "9.3GHz", "9.35GHz", _ "9.4GHz", "9.45GHz", "9.5GHz", _ "9.55GHz", _ "9.6GHz", "9.65GHz", "9.7GHz", _ "9.75GHz", "9.8GHz", "9.85GHz", _ "9.9GHz", "9.95GHz", "10GHz"),_ "offset:=", Array("All"))

10 - Analysis Setup Module Script Commands

Circuit analysis setup commands should be executed by the Analysis module, referred to in Circuit scripts as the "AnalysisSetup" module.

Set oModule = oDesign.GetModule("AnalysisSetup")

ClearLinkedData

CopySetup

DeleteDrivenSweep

DeleteSetups

EditCircuitSettings

EditSetup

ExportCircuit

GetSetupCount

GetSetups

GetSweepCount

GetSweeps

InsertFrequencySweep

InsertSetup

InsertSetup [Transient]

PasteSetup

PasteSweep

RenameDrivenSweep

RenameSetup

RevertAllToInitial

RevertSetupToInitial

CopySetup

Copy the specified Optimetrics setup.

UI Access	NA
Parameters	Name | Type | Description <*SetupName*> | String | Name of the setup.
Return Value	None.

Python Syntax	CopySetup (<*SetupName*>)
Python Example	`oModule.CopySetup ("OptimizationSetup1")`

VB Syntax	CopySetup <*SetupName*>
VB Example	`oModule.CopySetup "OptimizationSetup1"`

CopyDrivenSetup

Copies a driven setup.

UI Access	N/A
Parameters	Name | Type | Description <*SetupName*> | String | Name of the setup.
Return Value	None.

Python Syntax	CopyDrivenSetup(<*SetupName*>)
Python Example	`oModule.CopyDrivenSetup("Setup1")`

VB Syntax	CopyDrivenSetup <*SetupName*>
VB Example	`oModule.CopyDrivenSetup "Setup1"`

CopyEigenSetup

Copies an eigen-analysis setup.

UI Access	N/A
Parameters	Name | Type | Description <*SetupName*> | String | Name of the setup.
Return Value	None.

Python Syntax	CopyEigenSetup(<*SetupName*>)

Python Example	`oModule.CopyEigenSetup("Setup1")`

VB Syntax	CopyEigenSetup <*SetupName*>
VB Example	`oModule.CopyEigenSetup "Setup1"`

CopySweep

Copies a frequency sweep from an analysis setup.

UI Access	Right-click on a sweep, select **Copy**
Parameters	Name / Type / Description <*SetupName*> / String / Name of the setup to which the sweep belongs. <*SweepName*> / String / Name of the sweep.
Return Value	None.

Python Syntax	CopySweep (<*SetupName*>, <*SweepName*>)
Python Example	`oModule.CopySweep("Setup1", "Sweep2")`

VB Syntax	CopySweep <*SetupName*>, <*SweepName*>
VB Example	`oModule.CopySweep "Setup1", "Sweep2"`

DeleteDrivenSweep

Deletes a driven frequency sweep.

UI Access	N/A
Parameters	Name | Type | Description <*SetupName*> | String | Name of the setup to which the sweep belongs. <*SweepName*> | String | Name of the sweep.
Return Value	None.

Python Syntax	DeleteDrivenSweep (<*SetupName*>, <*SweepName*>)
Python Example	`oModule.DeleteDrivenSweep("Setup1", "Sweep2")`

VB Syntax	DeleteDrivenSweep <*SetupName*>, <*SweepName*>
VB Example	`oModule.DeleteDrivenSweep "Setup1", "Sweep2"`

DeleteSetups

Deletes one or more solution setups, which are specified by an array of solution setup names.

UI Access	Right-click a solution setup in the project tree and then click **Delete** on the shortcut menu, or delete selected solution setups in the **List** dialog box.
Parameters	Name | Type | Description <*SetupArray*> | Array | Array of solution setup names.

Return Value	None.

Python Syntax	DeleteSetups (<*SetupArray*>)
Python Example	`oModule.DeleteSetups(["Setup1", "Setup2"])`

VB Syntax	DeleteSetups <*SetupArray*>
VB Example	`oModule.DeleteSetups Array("Setup1", "Setup2")`

DoesSweepSetupExists

Determines whether a specified sweep exists.

UI Access	N/A
Parameters	<table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><*SetupName*></td><td>String</td><td>Name of the setup to which the sweep belongs.</td></tr><tr><td><*SweepName*></td><td>String</td><td>Name of the sweep.</td></tr></table>
Return Value	Integer: • **1** - sweep exists. • **0** - sweep does not exist.

Python Syntax	DoesSweepSetupExists (<*SetupName*>, <*SweepName*>)
Python Example	`oModule.DoesSweepSetupExists("Setup1", "Sweep2")`

VB Syntax	DoesSweepSetupExists <*SetupName*>, <*SweepName*>
VB Example	`oModule.DoesSweepSetupExists "Setup1", "Sweep2"`

EditCircuitSettings

Use: Exports equivalent circuit data.

Command: Right-click the **Analysis** folder, and then choose **Edit Circuit Settings.**

Syntax: EditCircuitSettings <ExportSettings>

Return Value: None

Parameters: <ExportSettings>

See ExportCircuit for details.

VB Example:
```
oModule.EditCircuitSettings Array("NAME:CircuitData", "MatrixName:=", "Original", _
"NumberOfCells:=", "1", "UserHasChangedSettings:=", true, "IncludeCap:=", true, "IncludeCond:=",
_
true, Array("NAME:CouplingLimits", "CouplingLimitType:=", "By Fraction", "CapFraction:=", 0.01,
_
"IndFraction:=", 0.01, "ResFraction:=", 0.01, "CondFraction:=", 0.01), "IncludeDCR:=", false, _
"IncudeDCL:=", false, "IncludeACR:=", false, "IncludeACL:=", false, "ADDResistance:=", true)
```

VB Example:
```
oModule.EditCircuitSettings Array("NAME:CircuitData", "MatrixName:=", "Original", _
"1", "UserHasChangedSettings:=", true, "IncludeCap:=", true, "IncludeCond:=", true, Array _
("NAME:CouplingLimits", "CouplingLimitType:=", "By Fraction", "CapFraction:=",0.01, _
```

```
"IndFraction:=", 0.01, "ResFraction:=", 0.01, "CondFraction:=", 0.01), "IncludeR:=", false, _
"IncludeL:=", false, "ExportDistributed:=", true, "LumpedLength:=", "7meter", "RiseTime:=", "1s")
```

EditDrivenSweep

Modifies a driven frequency sweep.

<table>
<tr><td>UI Access</td><td>N/A.</td></tr>
<tr><td>Parameters</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 solution setup containing the sweep to be edited.</td></tr>
<tr><td><SweepName></td><td>String</td><td>Name of the sweep to be edited.</td></tr>
<tr><td><Attributes></td><td>Array</td><td>Structured array.
<pre>Array("NAME:<SweepName>",
  "IsEnabled:=", <boolean>,
  "SetupType:=", <SetupType>,
  <FrequencyInformation>,
  "Type:=", <SweepType>,
  <SaveFieldsList>
  <DCExtrapInfo>
  "SMatrixOnlySolveMode:=", <"Manual">|"Auto">,
  "SMatrixOnlySolveAbove:=", "<real><units>")</pre></td></tr>
</table>
</td></tr>
<tr><td>Return Value</td><td>None.</td></tr>
</table>

Python Syntax	EditDrivenSweep (<*SetupName*>, <*SweepName*>, <*Attributes*>)
Python Example	`oModule.EditDrivenSweep("Setup1", "Sweep3",` `["NAME:Sweep3", "IsEnabled:=", True,` `"SetupType:=", "SinglePoints",` `"ValueList:=", ["1GHz", "2GHz", "3GHz"],` `"Type:=", "Discrete",` `"SaveFieldsList:=", [False, False, False],` `"ExtrapToDC:=", False])`

VB Syntax	EditDrivenSweep <*SetupName*>, <*SweepName*>, <*Attributes*>
VB Example	`oModule.EditDrivenSweep "Setup1", "Sweep3", _` `Array("NAME:Sweep3", "IsEnabled:=", true, _` `"SetupType:=", "SinglePoints", _` `"ValueList:=", Array("1GHz", "2GHz", "3GHz"), _` `"Type:=", "Discrete", _` `"SaveFieldsList:=", Array(false, false, false), _` `"ExtrapToDC:=", false)`

EditFrequencySweep

Modifies an existing frequency sweep. For HFSS-IE use EditSweep [HFSS-IE]

<table>
<tr><td>UI Access</td><td>Double-click a frequency sweep in the Project Manager to modify its settings.</td></tr>
<tr><td>Parameters</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 solution setup containing the sweep to be edited.</td></tr>
<tr><td><SweepName></td><td>String</td><td>Name of the sweep to be edited.</td></tr>
<tr><td><Attributes></td><td>Array</td><td>Structured array.
<pre>Array("NAME:<SweepName>",
  "IsEnabled:=", <boolean>,
  "SetupType:=", <SetupType>,
  <FrequencyInformation>,
  "Type:=", <SweepType>,
  <SaveFieldsList>
  <DCExtrapInfo>
  "SMatrixOnlySolveMode:=", <"Manual">|"Auto">,
  "SMatrixOnlySolveAbove:=", "<real><units>")</pre></td></tr>
</table>
</td></tr>
<tr><td>Return Value</td><td>None.</td></tr>
</table>

<table>
<tr><td>Python Syntax</td><td>EditFrequencySweep (<SetupName>, <SweepName>, <Attributes>)</td></tr>
<tr><td>Python Example</td><td>
<pre>oModule.EditFrequencySweep("Setup1", "Sweep3",
["NAME:Sweep3", "IsEnabled:=", True,
"SetupType:=", "SinglePoints",</pre></td></tr>
</table>

	"ValueList:=", ["1GHz", "2GHz", "3GHz"], "Type:=", "Discrete", "SaveFieldsList:=", [False, False, False], "ExtrapToDC:=", False])

VB Syntax	EditFrequencySweep <*SetupName*>, <*SweepName*>, <*Attributes*>
VB Example	oModule.EditFrequencySweep "Setup1", "Sweep3", _ Array("NAME:Sweep3", "IsEnabled:=", true, _ "SetupType:=", "SinglePoints", _ "ValueList:=", Array("1GHz", "2GHz", "3GHz"), _ "Type:=", "Discrete", _ "SaveFieldsList:=", Array(false, false, false), _ "ExtrapToDC:=", false)

EditSetup

Modifies an existing solution setup.

UI Access	Double-click a solution setup in the project tree to modify its settings.
Parameters	<table><tr><th>Name</th><th>Type</th><th>Description</th></tr><tr><td><*SetupName*></td><td>String</td><td>Name of the solve setup being edited.</td></tr><tr><td><*Attributes*></td><td>Array</td><td>Structured array. Array("NAME:<NewSetupName>", <NamedParameters>)</td></tr></table>

	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("EddyCurrent",
[
```

```
"NAME:EddyCurrent",
"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"
  ])
```

VB Syntax	EditSetup <*SetupName*>, <*Attributes*>

VB Example

```
oModule.EditSetup "Setup1",
Array("NAME:NewSetup",
"AdaptiveFreq:=", "1GHz",
"EnableDistribProbTypeOption:=", false,
"SaveFields:=", "true",
"Enabled:=", true,
Array("NAME:Cap",
        "MaxPass:=", 10,
        "MinPass:=", 1,
        "MinConvPass:=", 2,
        "PerError:=", 1,
        "PerRefine:=", 30,
        "AutoIncreaseSolutionOrder:=", false,
        "SolutionOrder:=", "Normal"),
Array("NAME:DC",
        "Residual:=", 1E-005,
        "SolveResOnly:=", false,
        Array("NAME:Cond",
                "MaxPass:=", 10,
                "MinPass:=", 1,
                "MinConvPass:=", 1,
```

	"PerError:=", 1, "PerRefine:=", 30), Array("NAME:Mult", "MaxPass:=", 1, "MinPass:=", 1, "MinConvPass:=", 1, "PerError:=", 1, "PerRefine:=", 30)), Array("NAME:AC", "MaxPass:=", 10, "MinPass:=", 1, "MinConvPass:=", 2, "PerError:=", 1, "PerRefine:=", 30))

ExportCircuit

Use: Export equivalent circuit data.

Command: Right-click a setup in the project tree or the **Analysis** folder and choose **Export Circuit**.

Syntax: ExportCircuit <Solution>, <Variation>, <FileName>, <ExportSettings>, <ModelName>, <Freq>

Return Value: none

Parameters: <Solution>

```
  <SetupName>:<SolutionName>
  <SetupName>
    Type: <string>
    Name of the setup where circuit is being exported
  <SolutionName>
    Type: <String>
    Name of the solution.
<Variation>
  Type: <string>
  The variation where circuit is being exported
<FileName>
  Type: <string>
  The name of the file where circuit is being exported
<ModelName>
  Type: <String>
  Model name or name of the sub circuit (Optional). If not specified then <FileName> is con-
  sidered as model name.
<Freq>
  Type: <double>
  Sweep frequency in hz.
<ExportSettings>
  Array("NAME:CircuitData", "MatrixName:=", _
```

<ReduceMatrix>, "NumberOfCells:=", <NumCell>, "UserHasChangedSettings:=", <User-ChangedSettings>, "IncludeCap:=", <IncludeCap>, "IncludeCond:=", <IncludeCond>, Array ("NAME:CouplingLimits", <CouplingLimitsArray>, "IncludeDCR:=", <IncludeDCR>, "IncludeDCL:=", <IncludeDCL>, "IncludeACR:=", <IncludeACR>, "IncludeACL:=", <IncludeACL>, "ADDResistance:=", <AddResistance>)

Parameters:

<ReduceMatrix>

Type: <string>

One of the reduced matrix setup or "Original"

<NumCell>

Type: <string>

Number of cells in export. Can be a variable.

<UserChangedSettings>

Type: <bool>

Whether user changed settings or use default settings.wirebond

<IncludeCap>

Type: <bool>

Flag indicates whether to export Capacitance matrix.

<IncludeCond>

Type: <bool>

Flag indicates whether to export Conductance matrix.

<IncludeDCR>

```
  Type: <bool>
  Flag indicates whether to export DC resistance matrix.
<IncludeDCL>
  Type: <bool>
  Flag indicates whether to export DC Inductance matrix.
<IncludeACR>
  Type: <bool>
  Flag indicates whether to export AC resistance matrix.
<IncludeACL>
  Type: <bool>
  Flag indicates whether to export AC inductance matrix.
<AddResistance>
  Type: <bool>
  Adds the DC and AC resistance.
```

Note:

You cannot export both AC and DC matrices unless **AddResistance** is selected.

```
<CouplingLimitsArray>
  Array("NAME:CouplingLimits", "CouplingLimitType:=", <CouplingLimitType>, _
  <CouplingLimitsParameters>, 0.01, "CondFraction:=", 0.01)
  <CouplingLimitType> = "None"
```

```
    Argument not needed
  <CouplingLimitType> = "ByFraction"
    <CouplingLimitsParameters>
    "CapFraction:=", <Fraction>, "IndFraction:=", <Fraction>,
    "ResFraction:=", <Fraction>,"CondFraction:=",<Fraction>,
    Parameters:
    <Fraction>
```

Type: <double>

Fraction of the self term

```
  <CouplingLimitType> = "ByValue"
    <CouplingLimitsParameters>
    "CapLimit:=", <Limit>, "IndLimit:=", <Limit>, "ResLimit:=", <Limit>,
    "CondLimit:=", <Limit>,
    Parameters:
    <Limit>
    Type: <string>
    Value of the limit.
```

VB Example:

```
oModule.ExportCircuit
"Setup1 : LastAdaptive", "", "C:/Project/Q3D/FourNets.cir", Array("NAME:CircuitData", _
```

```
"MatrixName:=", "Original", "NumberOfCells:=", "1", "UserHasChangedSettings:=", true, _
"IncludeCap:=", true, "IncludeCond:=", true, Array("NAME:CouplingLimits", "CouplingLimitType:=",
_
"By Fraction", "CapFraction:=", 0.01, "IndFraction:=", 0.01, "ResFraction:=", 0.01, _
"CondFraction:=", 0.01), "IncludeDCR:=", false, IncudeDCL:=", false, "IncludeACR:=", false, _
"IncludeACL:=", false, "ADDResistance:=", true), "", 2000000000000
```

GetSetupCount

Gets the number of analysis setups in a design.

UI Access	N/A
Parameters	None.
Return Value	Integer containing number of setups.

Python Syntax	GetSetupCount ()
Python Example	`oModule.GetSetupCount()`

VB Syntax	GetSetupCount
VB Example	`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()`

VB Syntax	GetSetups
VB Example	`oModule.GetSetups`

GetSetups (Layout Editor)

Get setups.

Command: None

Syntax: GetSetups <setups>

Return Value: String array of setup names

Parameters: <setups>

Type: string

VB Example:

```
oSetup = oDesign.GetModule("SolveSetups")
setups = oSetup.GetSetups()
```

GetSetupNames

Gets the names of far field and near field radiation setups in a design.

UI Access	N/A
Parameters	Name | Type | Description <*RadiationType*> | String | Type of radiation.
Return Value	Array of strings containing setup names.

Python Syntax	GetSetupNames(<*RadiationType*>)
Python Example	`oModule.GetSetupNames("Infinite Sphere")`

VB Syntax	GetSetupNames <*RadiationType*>
VB Example	`oModule.GetSetupNames "Infinite Sphere"`

GetSweepCount

Gets the number of sweeps in a specified analysis setup.

UI Access	N/A
Parameters	Name | Type | Description <*SetupName*> | String | Name of specified setup.
Return Value	Integer containing number of sweeps for the named setup.

Python Syntax	GetSweepCount (<*SetupName*>)
Python Example	`oModule.GetSweepCount("Setup1")`

VB Syntax	GetSweepCount <*SetupName*>
VB Example	`oModule.GetSweepCount "Setup1"`

GetSweeps

Gets the names of all sweeps in a given analysis setup.

UI Access	N/A
Parameters	Name | Type | Description <*SetupName*> | String | Name of specified setup.
Return Value	Array of sweep names.

Python Syntax	GetSweeps (<*SetupName*>)
Python Example	`oModule.GetSweeps("Setup1")`

VB Syntax	GetSweeps <*SetupName*>

VB Example	`oModule.GetSweeps "Setup1"`

GetSweeps (Layout Editor)

Get sweeps.

Command: None

Syntax: GetSweeps <sweep>

Return Value: String array of sweeps

Parameters: <sweep>

Type: string

VB Example:

```
oSetup = oDesign.GetModule("SolveSetups")
setups = oSetup.GetSetups()
sweep = oSetup.GetSweeps(setups[0])
```

InsertDrivenSweep

Adds a frequency sweep to a Driven solution-type setup.

UI Access	N/A.

Parameters

Name	Type	Description
<*SetupName*>	String	Name of the solution setup into which the sweep will be inserted.
<*Attributes*>	Array	Structured array. `Array("NAME:<string name of sweep>",` `"IsEnabled:=", <boolean>` `"RangeType:=", <"LinearStep" | "LinearCount" | "LogScale">,`

			<LinearStepArray> "RangeStart:=", "<real>GHz", "RangeEnd:=", "<real>GHz", "RangeStep:=", "<real>GHz", <LinearCountArray> "RangeStart:=", "<real>GHz", "RangeEnd:=", "<real>GHz", "RangeCount:=", <int>, <LogScaleArray> RangeStart:=", "<real>GHz", "RangeEnd:=", "<real>GHz", "RangeCount:=", <int>, "RangeSamples:=", <int>, <SweepTypeArray> "Type:=", <"Interpolating" |"Fast" | "Discrete">, <InterpolatingParametersArray> "SaveFields:=", <boolean>, "SaveRadFields:=", <boolean>, "InterpTolerance:=", <real>, "InterpMaxSolns:=", <int>,

			"InterpMinSolns:=", <int>, "InterpMinSubranges:=", <int>, "ExtrapToDC:=", <boolean>, "InterpUseS:=", <boolean>, "InterpUsePortImped:=", <boolean>, "InterpUsePropConst:=", <boolean>, "UseDerivativeConvergence:=", true, "InterpDerivTolerance:=", <real>, "UseFullBasis:=", <boolean>, "EnforcePassivity:=", <boolean>, "PassivityErrorTolerance:=", <real>) "EnforceCausality:=", <boolean>, "UseQ3DForDCSolve:=" , <boolean>, "SMatrixOnlySolveMode:=", <"Manual">|"Auto">, "SMatrixOnlySolveAbove:=", "<real><units>" <FastParametersArray> "SaveFields:=", <boolean>, "SaveRadFields:=", <boolean>, "GenerateFieldsForAllFreqs:=", <boolean>, "ExtrapToDC:=", <boolean>) <DiscreteParametersArray>

			"SaveFields:=", <boolean>, SaveRadFields:=", <boolean>, "ExtrapToDC:=", <boolean>)

Return Value	None.

Python Syntax	InsertDrivenSweep(<*SetupName*>, <*Attributes*>)
Python Example	oModule.InsertDrivenSweep("Setup1", ["NAME:Sweep1", "IsEnabled:=" , True, "RangeType:=" , "LinearCount", "RangeStart:=" , "2.5GHz", "RangeEnd:=" , "7.5GHz", "RangeCount:=" , 401, "Type:=" , "Interpolating", "SaveFields:=" , False, "SaveRadFields:=" , False, "InterpTolerance:=" , 0.5, "InterpMaxSolns:=" , 250,

```
    "InterpMinSolns:="      , 0,
    "InterpMinSubranges:="  , 1,
    "InterpUseS:="          , True,
    "InterpUsePortImped:="  , False,
    "InterpUsePropConst:="  , True,
    "UseDerivativeConvergence:=", False,
    "InterpDerivTolerance:=", 0.2,
    "UseFullBasis:="        , True,
    "EnforcePassivity:="    , True,
    "PassivityErrorTolerance:=", 0.0001,
    "SMatrixOnlySolveMode:=", "Manual",
    "SMatrixOnlySolveAbove:=", "1MHz"
  ])
```

VB Syntax	InsertDrivenSweep <*SetupName*>, <*Attributes*>
VB Example	Discrete Sweep: oModule.InsertDrivenSweep "Setup1", Array("NAME:Sweep", "IsEnabled:=", true, "RangeType:=", "LinearStep", "RangeStart:=", "1GHz", "RangeEnd:=", "10GHz", "RangeStep:=", "0.1GHz", "Type:=", "Discrete",

```
"SaveFields:=", false,
"SaveRadFields:=", false,
"ExtrapToDC:=", false)

Fast Sweep:

oModule.InsertDrivenSweep "Setup1",
Array("NAME:Sweep2", "IsEnabled:=", false,
"RangeType:=", "LinearStep",
"RangeStart:=", "1GHz",
"RangeEnd:=", "10GHz",
"RangeStep:=", "0.1GHz",
"Type:=", "Fast",
"SaveFields:=", true,
"SaveRadFields:=", false,
"GenerateFieldsForAllFreqs:=", false,
"ExtrapToDC:=", false)

Interpolating Sweep with additional setups:

oModule.InsertDrivenSweep "Setup1", Array("NAME:Sweep3", _

"IsEnabled:=", true, "SetupType:=", _

"LinearStep", "StartValue:=", "0GHz", _

"StopValue:=", "2.5GHz", "StepSize:=", "0.005GHz", _

"Type:=", "Interpolating", _

"SaveFields:=", false, _

"InterpTolerance:=", 0.5, _

"InterpMaxSolns:=", 50, "InterpMinSolns:=", 0, _

"InterpMinSubranges:=", 1, _
```

```
"ExtrapToDC:=", true, "MinSolvedFreq:=", "0.005GHz", _
"InterpUseS:=", true, _
"InterpUseT:=", false, "InterpUsePortImped:=", false, _
"InterpUsePropConst:=", true, "UseFullBasis:=", true) _
Array( "SetupType:=", "LogScale", "StartValue:=", "11GHz", "StopValue:=", _
"12GHz", "Count:=", 91, _
"SetupType:=", "LinearCount", "StartValue:=", "13GHz", _
"StopValue:=", "100GHz", "Count:=", 91)
Discrete sweeps with linear step and log scale:
oModule.InsertDrivenSweep "Setup1", Array("NAME:Sweep2", _
"IsEnabled:=", true, "SetupType:=", "LinearStep", _
"StartValue:=", "0.005GHz", >"StopValue:=", "2.5GHz", _
"StepSize:=", "0.005GHz", "Type:=", "Discrete", _
"SaveFields:=", false, "ExtrapToDC:=", false)

oModule.InsertDrivenSweep "Setup1", Array("NAME:Sweep3", _
"IsEnabled:=", true, "SetupType:=", "LogScale", _
"StartValue:=", "1GHz", "StopValue:=", "10GHz", _
"SamplesPerDecade:=", 4, "Type:=", "Discrete", _
"SaveFields:=", false, "ExtrapToDC:=", false)
A Fast sweep, specified using the starting and stopping frequencies and the step
count:
```

	oModule.InsertDrivenSweep "Setup1", Array("NAME:Sweep4", _ "IsEnabled:=", true, "SetupType:=", "LinearCount", _ "StartValue:=", "1GHz", "StopValue:=", "10GHz", _ "Count:=", 3, "Type:=", "Fast", _ "SaveFields:=", true, "ExtrapToDC:=", false)

InsertFrequencySweep

Adds a frequency sweep to a Driven solution-type setup.

UI Access	Right-click an analysis setup, then select **Add Frequency Sweep...**.
Parameters	See table below

Name	Type	Description
<SetupName>	String	Name of the solution setup into which the sweep will be inserted.
<Attributes>	Array	Structured array. Array("NAME:<string name of sweep>", "IsEnabled:=", <boolean> "RangeType:=", <"LinearStep" | "LinearCount" | "LogScale">, <LinearStepArray> "RangeStart:=", "<real>GHz", "RangeEnd:=", "<real>GHz", "RangeStep:=", "<real>GHz", <LinearCountArray>

			"RangeStart:=", "<real>GHz", "RangeEnd:=", "<real>GHz", "RangeCount:=", <int>, <LogScaleArray> RangeStart:=", "<real>GHz", "RangeEnd:=", "<real>GHz", "RangeCount:=", <int>, "RangeSamples:=", <int>, <SweepTypeArray> "Type:=", <"Interpolating" |"Fast" | "Discrete">, <InterpolatingParametersArray> "SaveFields:=", <boolean>, "SaveRadFields:=", <boolean>, "InterpTolerance:=", <real>, "InterpMaxSolns:=", <int>, "InterpMinSolns:=", <int>, "InterpMinSubranges:=", <int>, "ExtrapToDC:=", <boolean>, "InterpUseS:=", <boolean>, "InterpUsePortImped:=", <boolean>, "InterpUsePropConst:=", <boolean>,

			"UseDerivativeConvergence:=", true, "InterpDerivTolerance:=", <real>, "UseFullBasis:=", <boolean>, "EnforcePassivity:=", <boolean>, "PassivityErrorTolerance:=", <real>) "EnforceCausality:=", <boolean>, "UseQ3DForDCSolve:=" , <boolean>, "SMatrixOnlySolveMode:=", <"Manual">|"Auto">, "SMatrixOnlySolveAbove:=", "<real><units>" <FastParametersArray> "SaveFields:=", <boolean>, "SaveRadFields:=", <boolean>, "GenerateFieldsForAllFreqs:=", <boolean>, "ExtrapToDC:=", <boolean>) <DiscreteParametersArray> "SaveFields:=", <boolean>, SaveRadFields:=", <boolean>, "ExtrapToDC:=", <boolean>)
Return Value	None.		

Python Syntax	InsertFrequencySweep(<*SetupName*>, <*Attributes*>)
Python Example	oModule.InsertFrequencySweep("Setup1", ["NAME:Sweep1", "IsEnabled:=" , True, "RangeType:=" , "LinearCount", "RangeStart:=" , "2.5GHz", "RangeEnd:=" , "7.5GHz", "RangeCount:=" , 401, "Type:=" , "Interpolating", "SaveFields:=" , False, "SaveRadFields:=" , False, "InterpTolerance:=" , 0.5, "InterpMaxSolns:=" , 250, "InterpMinSolns:=" , 0, "InterpMinSubranges:=" , 1, "InterpUseS:=" , True, "InterpUsePortImped:=" , False, "InterpUsePropConst:=" , True, "UseDerivativeConvergence:=", False, "InterpDerivTolerance:=", 0.2,

```
    "UseFullBasis:="        , True,
    "EnforcePassivity:="    , True,
    "PassivityErrorTolerance:=", 0.0001,
    "SMatrixOnlySolveMode:=", "Manual",
    "SMatrixOnlySolveAbove:=", "1MHz"
  ])
```

VB Syntax	InsertFrequencySweep <*SetupName*>, <*Attributes*>
VB Example	Discrete Sweep: oModule.InsertFrequencySweep "Setup1", Array("NAME:Sweep", "IsEnabled:=", true, "RangeType:=", "LinearStep", "RangeStart:=", "1GHz", "RangeEnd:=", "10GHz", "RangeStep:=", "0.1GHz", "Type:=", "Discrete", "SaveFields:=", false, "SaveRadFields:=", false, "ExtrapToDC:=", false) Fast Sweep: oModule.InsertFrequencySweep "Setup1", Array("NAME:Sweep2", "IsEnabled:=", false, "RangeType:=", "LinearStep", "RangeStart:=", "1GHz",

```
"RangeEnd:=", "10GHz",
"RangeStep:=", "0.1GHz",
"Type:=", "Fast",
"SaveFields:=", true,
"SaveRadFields:=", false,
"GenerateFieldsForAllFreqs:=", false,
"ExtrapToDC:=", false)

Interpolating Sweep with additional setups:

oModule.InsertFrequencySweep "Setup1", Array("NAME:Sweep3", _

"IsEnabled:=", true, "SetupType:=", _

"LinearStep", "StartValue:=", "0GHz", _

"StopValue:=", "2.5GHz", "StepSize:=", "0.005GHz", _

"Type:=", "Interpolating", _

"SaveFields:=", false, _

"InterpTolerance:=", 0.5, _

"InterpMaxSolns:=", 50, "InterpMinSolns:=", 0, _

"InterpMinSubranges:=", 1, _

"ExtrapToDC:=", true, "MinSolvedFreq:=", "0.005GHz", _

"InterpUseS:=", true, _

"InterpUseT:=", false, "InterpUsePortImped:=", false, _

"InterpUsePropConst:=", true, "UseFullBasis:=", true) _

Array( "SetupType:=", "LogScale", "StartValue:=", "11GHz", "StopValue:=", _

"12GHz", "Count:=", 91, _

"SetupType:=", "LinearCount", "StartValue:=", "13GHz", _
```

```
"StopValue:=", "100GHz", "Count:=", 91)
Discrete sweeps with linear step and log scale:
oModule.InsertFrequencySweep "Setup1", Array("NAME:Sweep2", _
"IsEnabled:=", true, "SetupType:=", "LinearStep", _
"StartValue:=", "0.005GHz", >"StopValue:=", "2.5GHz", _
"StepSize:=", "0.005GHz", "Type:=", "Discrete", _
"SaveFields:=", false, "ExtrapToDC:=", false)

oModule.InsertFrequencySweep "Setup1", Array("NAME:Sweep3", _
"IsEnabled:=", true, "SetupType:=", "LogScale", _
"StartValue:=", "1GHz", "StopValue:=", "10GHz", _
"SamplesPerDecade:=", 4, "Type:=", "Discrete", _
"SaveFields:=", false, "ExtrapToDC:=", false)
A Fast sweep, specified using the starting and stopping frequencies and the step
count:
oModule.InsertFrequencySweep "Setup1", Array("NAME:Sweep4", _
"IsEnabled:=", true, "SetupType:=", "LinearCount", _
"StartValue:=", "1GHz", "StopValue:=", "10GHz", _
"Count:=", 3, "Type:=", "Fast", _
"SaveFields:=", true, "ExtrapToDC:=", false)
```

InsertSetup

Adds a new solution setup.

<table>
<tr><td>UI Access</td><td>[product] > Analysis Setup > Add Solution Setup.</td></tr>
<tr><td>Parameters</td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><SetupType></td><td>String</td><td>Type of setup to be inserted.</td></tr>
<tr><td><Attributes></td><td>Array</td><td>Structured array.
Array("NAME:<SetupName>", <NamedParameters>)
<Named Parameters>
"AdaptMultipleFreqs:=", <boolean>,
"Frequency:=", <string>,
"MaxDeltaS:=", <float>,
"PortsOnly:=", <boolean>,
"UseMatrixConv:=", <boolean>,
If UseMatrixCov is True,
[
"NAME:Matrix Convergence",
<Convergence Array>
],
MaximumPasses:=", <integer>,
"MinimumPasses:=", <integer>,
"MinimumConvergedPasses:=", <integer>,</td></tr>
</table>
</td></tr>
</table>

			"PercentRefinement:=", <integer>, "IsEnabled:=", <boolean>, "BasisOrder:=", <integer>, "DoLambdaRefine:=", <boolean>, "DoMaterialLambda:=", <boolean>, "SetLambdaTarget:=", <boolean>, "Target:=", <float>, "PortAccuracy:=", <integer>, "SaveRadFieldsOnly:=", <boolean>, "SaveAnyFields:=", <boolean>, "ListsForFields:=", <array of string>, "IESolverType:=", <string>, "LambdaTargetForIESolver:=", <float> "UseDefaultLambdaTgtForIESolver:=", <boolean>
Return Value	None.		

Python Syntax	InsertSetup(<*SetupType*>, <*Attributes*>)
Python Example	Maxwell 3D Eddy Current Setup `oModule = oDesign.GetModule("AnalysisSetup")`

```
oModule.InsertSetup("EddyCurrent",
[
   "NAME:EddyCurrent_03",
   "Enabled:="            , True,
  [
     "NAME:MeshLink",
     "ImportMesh:="         , False
  ],
  "MaximumPasses:="       , 4,
  "MinimumPasses:="       , 2,
  "MinimumConvergedPasses:=", 1,
  "PercentRefinement:="   , 30,
  "SolveFieldOnly:="      , False,
  "PercentError:="        , 0.1,
  "SolveMatrixAtLast:="   , True,
  "UseNonLinearIterNum:=" , False,
  [
    "NAME:ExpressionCache",
    [
      "NAME:CacheItem",
      "Title:="              , "eddy_loss1",
```

```
        "Expression:="          , "eddy_loss",
        "Intrinsics:="          , "Phase=\'0deg\'",
        "ReportType:="          , "Fields",
        [
          "NAME:ExpressionContext"
        ]
      ]
    ],
    "UseCacheFor:="         , ["Pass"],
    "UseIterativeSolver:="  , False,
    "RelativeResidual:="    , 0.0001,
    "NonLinearResidual:="   , 0.0001,
    "SmoothBHCurve:="       , False,
    "Frequency:="           , "200Hz",
    "HasSweepSetup:="       , False,
    "UseHighOrderShapeFunc:=", False,
    "UseMuLink:="           , False,
    "LossAdaptiveCtrl:="    , "0.3"
])
```

HFSS simulation

```
oModule.InsertSetup("HfssDrivenAuto",
["NAME:Setup1",
  "Enabled:=", True,
  "AutoSolverSetting:=", "Balanced",
  ["NAME:Sweeps",
          ["NAME:Sweep",
            "RangeType:=", "LinearStep",
            "RangeStart:=", "1GHz",
            "RangeEnd:=", "10GHz",
            "RangeStep:=", "1GHz"
            ]
  ],
  "SaveRadFieldsOnly:=", False,
  "SaveAnyFields:=", True,
  "Type:=", "Discrete"
  ])

oModule.InsertSetup("HfssDrivenAuto",
        [
                "NAME:Setup2",
                "SolveType:="           , "Auto",
```

```
                "IsEnabled:="           , True,
        [
                "NAME:MeshLink",
                "ImportMesh:="          , False
        ],
        "AutoSolverSetting:="   , "Balanced",
        [
                "NAME:Sweeps",
                [
                        "NAME:Sweep",
                        "RangeType:="           , "LinearCount",
                        "RangeStart:="          , "0GHz",
                        "RangeEnd:="            , "10GHz",
                        "RangeCount:="          , 501
                ]
        ],
        "SaveRadFieldsOnly:="   , False,
        "SaveAnyFields:="       , True,
        "InfiniteSphereSetup:=" , "Infinite Sphere1",
        "ListsForFields:="      , ["Objectlist2"],
```

```
        "Type:="                , "Interpolating"
        ])


oModule.InsertSetup("HfssDriven",
["NAME:Setup3",
        "AdaptMultipleFreqs:=", False,
        "Frequency:=", "5GHz",
        "MaxDeltaS:=", 0.02,
        "PortsOnly:=", False,
        "UseMatrixConv:=", False,
        "MaximumPasses:=", 6,
        "MinimumPasses:=", 1,
        "MinimumConvergedPasses:=", 1,
        "PercentRefinement:=", 30,
        "IsEnabled:=", True,
        "BasisOrder:=", 1,
        "DoLambdaRefine:=", True,
        "DoMaterialLambda:=", True,
        "SetLambdaTarget:=", False,
        "Target:=", 0.3333,
        "UseMaxTetIncrease:=", False,
```

```
        "PortAccuracy:=", 2,
        "UseABCOnPort:=", False,
        "SetPortMinMaxTri:=", False,
        "UseDomains:=", True,
        "UseIterativeSolver:=", False,
        "IterativeResidual:=", 1E-06,
        "DDMSolverResidual:=", 0.0001,
        "EnhancedLowFreqAccuracy:=", True,
        "SaveRadFieldsOnly:=", False,
        "SaveAnyFields:=", True,
        "IESolverType:=", "Auto",
        "LambdaTargetForIESolver:=", 0.15,
        "UseDefaultLambdaTgtForIESolver:=", True,
        "SkipIERegionSolveDuringAdaptivePasses:=", True
        "RayDensityPerWavelength:=", 4,
        "MaxNumberOfBounces:="  , 5,
        "InfiniteSphereSetup:=" , "Infinite Sphere1",
        "SkipSBRSolveDuringAdaptivePasses:=", True,
        "PTDUTDSimulationSettings:=", "PTD Correction + UTD Rays",
        "PTDEdgeDensity:="      , 20
```

```
                                                    ])
```

An HFSS with Hybrid and Arrays setup

```
import ScriptEnv
ScriptEnv.Initialize("Ansoft.ElectronicsDesktop")
oDesktop.RestoreWindow()
oProject = oDesktop.SetActiveProject("Project1")
oDesign = oProject.SetActiveDesign("HFSSDesign1")
oModule = oDesign.GetModule("AnalysisSetup")
oModule.InsertSetup("HfssDriven",
  [
    "NAME:Setup1",
    "SolveType:="           , "Single",
    "Frequency:="           , "5GHz",
    "MaxDeltaE:="           , 0.1,
    "MaximumPasses:="       , 6,
    "MinimumPasses:="       , 1,
    "MinimumConvergedPasses:=", 1,
    "PercentRefinement:="   , 30,
    "IsEnabled:="           , True,
    [
      "NAME:MeshLink",
```

```
    "ImportMesh:="          , False
  ],
  "BasisOrder:="          , 1,
  "DoLambdaRefine:="      , True,
  "DoMaterialLambda:="    , True,
  "SetLambdaTarget:="     , False,
  "Target:="              , 0.3333,
  "UseMaxTetIncrease:="   , False,
  "DrivenSolverType:="    , "Direct Solver",
  "EnhancedLowFreqAccuracy:=", False,
  "SaveRadFieldsOnly:="   , False,
  "SaveAnyFields:="       , True,
  "IESolverType:="        , "Auto",
  "LambdaTargetForIESolver:=", 0.15,
  "UseDefaultLambdaTgtForIESolver:=", True,
  "IE Solver Accuracy:="  , "Balanced",
  "RayDensityPerWavelength:=", 4,
  "MaxNumberOfBounces:="  , 5,
  "RadiationSetup:="      , "Infinite Sphere1",
  "PTDUTDSimulationSettings:=", "None",
```

```
    "EnableSBRSelfCoupling:=", False,
    "UseSBRAdvOptionsGOBlockage:=", False,
    "UseSBRAdvOptionsWedges:=", False,
    "SkipSBRSolveDuringAdaptivePasses:=", False,
    "UseSBREnhancedRadiatedPowerCalculation:=", True
  ])
```

An SBR+ Setup with Fast Frequency Looping

```
oModule.InsertSetup("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
        ])
```

SBR+ Setup with Enhanced Radiated Power Calculation

```
oModule.InsertSetup("HfssDriven",
  [
    "NAME:Setup2",
```

```
    "IsEnabled:="           , True,
    [
      "NAME:MeshLink",
      "ImportMesh:="          , False
    ],
    "IsSbrRangeDoppler:="   , False,
    "RayDensityPerWavelength:=", 4,
    "MaxNumberOfBounces:="  , 5,
    "EnableCWRays:="        , False,
    "RadiationSetup:="      , "Infinite Sphere1",
    "PTDUTDSimulationSettings:=", "None",
    "FastFrequencyLooping:=", False,
    "UseSBRAdvOptionsGOBlockage:=", False,
    "UseSBRAdvOptionsWedges:=", False,
    [
      "NAME:Sweeps",
      [
        "NAME:Sweep",
        "RangeType:="           , "LinearStep",
        "RangeStart:="          , "1GHz",
        "RangeEnd:="            , "10GHz",
```

```
        "RangeStep:="           , "1GHz"
      ]
    ],
    "ComputeFarFields:="    , True,
    "UseSBREnhancedRadiatedPowerCalculation:=", True,
    "IsGOBlockageEnabled:=" , False,
    "GOBlockageSurfaceSelfBlock:=", False
  ])
```

Insert 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.InsertSetup("HfssRFDischarge",
  [
    "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:="  , False,
      "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:="          , "Argon"
    "Save Electron Density:=", True,
```

```
    "Is Pulsed Signal:="    , True,
    "Duty Cycle:="          , 50
  ])
```

Setup with UseMatrixConv is True and Matrix Convergence Entry Specified as All

```
oModule.InsertSetup("Setup1",
  [
    "NAME:Setup1",
    "SolveType:="           , "Single",
    "Frequency:="           , "10GHz",
    "MaxDeltaS:="           , 0.02,
    "UseMatrixConv:="       , True,
    [
      "NAME:Matrix Convergence",
      "AllEntries:="          , True,
      "MagLimit:="            , "0.01",
      "PhaseLimit:="          , "2deg",
      "MagMinThreshold:="     , 0.01
    ],
    "MaximumPasses:="       , 12,
    "MinimumPasses:="       , 1,
    "MinimumConvergedPasses:=", 1,
```

```
    "PercentRefinement:="   , 30,
    "IsEnabled:="           , True,
    [
      "NAME:MeshLink",
      "ImportMesh:="          , False
    ],
    "BasisOrder:="          , 1,
    "DoLambdaRefine:="      , True,
    "DoMaterialLambda:="    , True,
    "SetLambdaTarget:="     , False,
    "Target:="              , 0.3333,
    "UseMaxTetIncrease:="   , False,
    "PortAccuracy:="        , 2,
    "UseABCOnPort:="        , False,
    "SetPortMinMaxTri:="    , False,
    "DrivenSolverType:="    , "Direct Solver",
    "EnhancedLowFreqAccuracy:=", False,
    "SaveRadFieldsOnly:="   , False,
    "SaveAnyFields:="       , True,
    "IESolverType:="        , "ACA",
```

```
    "LambdaTargetForIESolver:=", 0.15,
    "UseDefaultLambdaTgtForIESolver:=", True,
    "IE Solver Accuracy:="  , "Balanced",
    "InfiniteSphereSetup:=" , ""
  ])
```

Setup with UseMatrixConv as Entries and MeshLink

```
oModule = oDesign.GetModule("AnalysisSetup")
oModule.InsertSetup("Setup1",
  [
    "NAME:Setup1",
    "SolveType:="           , "Single",
    "Frequency:="           , "10GHz",
    "MaxDeltaS:="           , 0.02,
    "UseMatrixConv:="       , True,
    [
      "NAME:Matrix Convergence",
      [
        "NAME:Entries",
        [
             "NAME:Entry",
             "Port1:="          , "A[1,1]P1",
```

```
            "Port2:="          , "A[1,1]P1",
            "MagLimit:="               , "0.011",
            "PhaseLimit:="             , "5deg"
        ],
        [
            "NAME:Entry",
            "Port1:="          , "A[4,4]P2",
            "Port2:="          , "A[1,1]P2",
            "MagLimit:="               , "0.041",
            "PhaseLimit:="             , "5deg"
        ]
      ]
    ],
    "MaximumPasses:="       , 1,
    "MinimumPasses:="       , 1,
    "MinimumConvergedPasses:=", 1,
    "PercentRefinement:="   , 30,
    "IsEnabled:="           , True,
    [
      "NAME:MeshLink",
```

```
        "ImportMesh:="          , True,
        "Project:="             , "This Project*",
        "Product:="             , "",
        "Design:="              , "unit",
        "Soln:="                , "Setup1 : LastAdaptive",
        [
          "NAME:Params",
          "Airbox_dist:="         , "9.9931mm",
          "Scan_Theta:="          , "0deg",
          "Scan_phi:="            , "0deg",
          "VirtualObject_dist:="  , "2.9979mm",
          "coax_inner_rad:="      , "0.125mm",
          "coax_outer_rad:="      , "0.425mm",
          "cut_off:="             , "1.7mm",
          "feedX:="               , "-0.4mm",
          "feedY:="               , "2mm",
          "feed_length:="         , "1mm",
          "patchX:="              , "9.65mm",
          "rot:="                 , "45deg",
          "subH:="                , "0.8mm",
          "subX:="                , "15mm",
```

```
        "subY:="                , "15mm"
      ],
      "ForceSourceToSolve:="  , False,
      "PreservePartnerSoln:=" , False,
      "PathRelativeTo:="      , "TargetProject",
      "ApplyMeshOp:="         , True,
      "AdaptPort:="           , False
    ],
    "BasisOrder:="          , 1,
    "DoLambdaRefine:="      , False,
    "DoMaterialLambda:="    , True,
    "SetLambdaTarget:="     , False,
    "Target:="              , 0.3333,
    "UseMaxTetIncrease:="   , False,
    "PortAccuracy:="        , 2,
    "UseABCOnPort:="        , False,
    "SetPortMinMaxTri:="    , False,
    "DrivenSolverType:="    , "Domain Decomposition",
    "IterativeResidual:="   , 0.0001,
    "DDMSolverResidual:="   , 0.0001,
```

```
    "EnhancedLowFreqAccuracy:=", False,
    "SaveRadFieldsOnly:="   , False,
    "SaveAnyFields:="       , True,
    "IESolverType:="        , "Auto",
    "LambdaTargetForIESolver:=", 0.15,
    "UseDefaultLambdaTgtForIESolver:=", True,
    "IE Solver Accuracy:="  , "Balanced",
    "InfiniteSphereSetup:=" , ""
  ])
```

Example from design with UseMatrixConv True for AllDiagEntries

```
    "UseMatrixConv:="       , True,
    [
      "NAME:Matrix Convergence",
      "AllDiagEntries:="      , True,
      "DiagonalMag:="         , "0.03",
      "DiagonalPhase:="       , "4rad",
      "MagMinThreshold:="     , 0.01,
      "AllOffDiagEntries:="   , True,
      "OffDiagonalMag:="      , "0.05",
      "OffDiagonalPhase:="    , "6deg",
      "MagMinThreshold:="     , 0.01
```

```
    ],
```

Example from Modal design with UseMatrixConv True for Entries

```
    "UseMatrixConv:="       , True,
    [
      "NAME:Matrix Convergence",
      [
        "NAME:Entries",
        [
            "NAME:Entry",
            "Port1:="          , "Port1",
            "Port2:="          , "Port1",
            "MagLimit:="               , "0.02",
            "PhaseLimit:="             , "5deg"
        ],
        [
            "NAME:Entry",
            "Port1:="          , "Port3",
            "Port2:="          , "Port2",
            "MagLimit:="               , "0.03",
            "PhaseLimit:="             , "5deg"
```

```
        ]
      ]
    ],
```

Example from Design with Periodic Ports and UseMaxtrixConv is True

```
    "UseMatrixConv:="       , True,
    [
      "NAME:Matrix Convergence",
      [
        "NAME:Entries",
        [
          "NAME:Entry",
          "Port1:="               , "Incident_Port1",
          "Port2:="               , "Incident_Port1",
          "MagLimit:="            , "0.11",
          "PhaseLimit:="          , "5deg"
        ],
        [
          "NAME:Entry",
          "Port1:="               , "Incident_Port2",
          "Port2:="               , "Incident_Port2",
          "MagLimit:="            , "0.22",
```

```
            "PhaseLimit:="          , "5deg"
          ]
        ]
      ],
```

Example Terminal Design with UseMatrixConv = True and Expression Cache

```
oProject = oDesktop.SetActiveProject("term-2x2_1_parasolid")
oDesign = oProject.SetActiveDesign("unit")
oModule = oDesign.GetModule("AnalysisSetup")
oModule.InsertSetup("Setup1",
  [
    "NAME:Setup1",
    "SolveType:="           , "Single",
    "Frequency:="           , "10GHz",
    "MaxDeltaS:="           , 0.02,
    "UseMatrixConv:="       , True,
    [
      "NAME:Matrix Convergence",
      [
        "NAME:Entries",
        [
```

```
            "NAME:Entry",
            "Port1:="          , "P2",
            "Port2:="          , "P1",
            "MagLimit:="               , "0.05",
            "PhaseLimit:="             , "10deg"
        ]
      ]
    ],
    "MaximumPasses:="       , 15,
    "MinimumPasses:="       , 1,
    "MinimumConvergedPasses:=", 1,
    "PercentRefinement:="   , 30,
    "IsEnabled:="           , True,
    [
      "NAME:MeshLink",
      "ImportMesh:="          , False
    ],
    "BasisOrder:="          , 1,
    "DoLambdaRefine:="      , True,
    "DoMaterialLambda:="    , True,
    "SetLambdaTarget:="     , False,
```

```
    "Target:="              , 0.3333,
    "UseMaxTetIncrease:="   , False,
    "PortAccuracy:="        , 2,
    "UseABCOnPort:="        , False,
    "SetPortMinMaxTri:="    , False,
    "DrivenSolverType:="    , "Iterative Solver",
    "IterativeResidual:="   , 0.0001,
    "DDMSolverResidual:="   , 0.0001,
    "EnhancedLowFreqAccuracy:=", False,
    "SaveRadFieldsOnly:="   , False,
    "SaveAnyFields:="       , True,
    [
      "NAME:ExpressionCache",
      [
        "NAME:CacheItem",
        "Title:="               , "dB_AxialRatioValue_1",
        "Expression:="          , "dB(AxialRatioValue)",
        "Intrinsics:="          , "Phi=\'0deg\' Theta=\'0deg\'",
        "ReportType:="          , "Far Fields",
        [
```

```
            "NAME:ExpressionContext",
            "Context:="             , "Infinite Sphere1"
          ]
        ],
        [
          "NAME:CacheItem",
          "Title:="               , "dB_St_Diff1_Diff1__1",
          "Expression:="          , "dB(St(Diff1,Diff1))",
          "Intrinsics:="          , "",
          "ReportType:="          , "Terminal Solution Data",
          [
            "NAME:ExpressionContext",
            "Diff:="                , "Differential Pairs"
          ]
        ],
        [
          "NAME:CacheItem",
          "Title:="               , "dB_St_P1_P1__1",
          "Expression:="          , "dB(St(P1,P1))",
          "Intrinsics:="          , "",
          "ReportType:="          , "Terminal Solution Data",
```

```
          [
            "NAME:ExpressionContext",
            "Diff:="                , "Terminals"
          ]
]
     ],
     "CacheSaveKind:="       , "Delta",
     "ConstantDelta:="       , "0s",
     "IESolverType:="        , "Auto",
     "LambdaTargetForIESolver:=", 0.15,
     "UseDefaultLambdaTgtForIESolver:=", True,
     "IE Solver Accuracy:="  , "Balanced",
     "InfiniteSphereSetup:=" , ""
])
```

Example HFSS Hybrid and Arrays with Beta Parallel Component Mesh Adapt

```
oModule.InsertSetup("HfssDriven",
   [
     "NAME:Setup2",
     "SolveType:="           , "Single",
     "Frequency:="           , "5GHz",
```

```
    "MaxDeltaS:="           , 0.02,
    "UseMatrixConv:="       , False,
    "MaximumPasses:="       , 6,
    "MinimumPasses:="       , 1,
    "MinimumConvergedPasses:=", 1,
    "PercentRefinement:="   , 30,
    "ComponentAdaptOption:=", "Fully Independent",
    "PerformFullDesignSolveAtLastPass:=", True,
    "IsEnabled:="           , True,
    [
      "NAME:MeshLink",
      "ImportMesh:="          , False
    ],
    "BasisOrder:="          , 1,
    "DoLambdaRefine:="      , True,
    "DoMaterialLambda:="    , True,
    "SetLambdaTarget:="     , False,
    "Target:="              , 0.3333,
    "UseMaxTetIncrease:="   , False,
    "PortAccuracy:="        , 2,
    "UseABCOnPort:="        , False,
```

```
    "SetPortMinMaxTri:="    , False,
    "DrivenSolverType:="    , "Direct Solver",
    "EnhancedLowFreqAccuracy:=", False,
    "SaveRadFieldsOnly:="   , False,
    "SaveAnyFields:="       , True,
    "IESolverType:="        , "Auto",
    "LambdaTargetForIESolver:=", 0.15,
    "UseDefaultLambdaTgtForIESolver:=", True,
    "IE Solver Accuracy:="  , "Balanced",
    "InfiniteSphereSetup:=" , "",
    "MaxPass:="             , 10,
    "MinPass:="             , 1,
    "MinConvPass:="         , 1,
    "PerError:="            , 1,
    "PerRefine:="           , 30
  ])
```

VB Syntax	InsertSetup <*SetupType*>, <*Attributes*>
VB	An HFSS Driven project solution with Auto Solution:

Example

```
oModule.InsertSetup "HfssDrivenAuto", Array("NAME:Setup1", "IsEnabled:=", true, _
"AutoSolverSetting:=",  "Balanced", _
Array("NAME:Sweeps", Array("NAME:Sweep", "RangeType:=", "LinearCount", _
"RangeStart:=",  "1GHz", "RangeEnd:=", "10GHz", "RangeCount:=", "451")), _
"SaveRadFieldsOnly:=",  false, "SaveAnyFields:=", false, "Type:=", "Interpolating")
```

A Driven solution type with a mesh link. References to dependent solve in old scripts are converted to mesh link form.

```
oModule.InsertSetup "HfssDriven",
Array("NAME:Setup1",
"Frequency:=", "1GHz",
"MaxDeltaE:=", 0.1,
"MaximumPasses:=", 6,
"MinimumPasses:=", 1,
"MinimumConvergedPasses:=", 1,
"PercentRefinement:=", 30,
"IsEnabled:=", true,
Array("NAME:MeshLink",
"Project:=", "Tee.aedt",
"Design:=", "TeeModel",
"Soln:=", "Setup1 : LastAdaptive",
Array("NAME:Params", "offset:=", "0in"),
"ForceSourceToSolve:=", false,
```

```
"PreservePartnerSoln:=", false,
"PathRelativeTo:=", "SourceProduct",
"ApplyMeshOp:=", true),
"BasisOrder:=", 1,
"UseIterativeSolver:=", false,
"DoLambdaRefine:=", false,
"DoMaterialLambda:=", true,
"SetLambdaTarget:=", false,
"Target:=", 0.3333,
"UseMaxTetIncrease:=", false,
"MaxTetIncrease:=", 1000000,
"EnableSolverDomains:=", false,
"ThermalFeedback:=", false,
"UsingConstantDelta:=", 0,
"ConstantDelta:=", "0s",
"NumberSolveSteps:=", 1)
A Driven solution type with ports:
oModule.InsertSetup "HfssDriven",
Array("NAME:Setup2",
"Frequency:=", "1GHz",
```

```
"PortsOnly:=", false,
"MaxDeltaS:=", 0.02,
"UseMatrixConv:=", false,
"MaximumPasses:=", 6,
"MinimumPasses:=", 1,
"MinimumConvergedPasses:=", 1,
"PercentRefinement:=", 30,
"IsEnabled:=", true,
"BasisOrder:=", 1,
"UseIterativeSolver:=", true,
"IterativeResidual:=", 0.0001,
"DoLambdaRefine:=", true,
"DoMaterialLambda:=", false,
"SetLambdaTarget:=", false,
"Target:=", 0.3333,
"UseMaxTetIncrease:=", false,
"MaxTetIncrease:=", 1000000,
"PortAccuracy:=", 2,
"UseABCOnPort:=", true,
"SetPortMinMaxTri:=", false,
"EnableSolverDomains:=", false,
```

```
"ThermalFeedback:=", false,
"UsingConstantDelta:=", 0,
"ConstantDelta:=", "0s",
"NumberSolveSteps:=", 1)
An Eigenmode solution type:
oModule.InsertSetup "HfssEigen",
Array("NAME:Setup2",
"MinimumFrequency:=", "1.77347GHz",
"NumModes:=", 1,
"MaxDeltaFreq:=", 10,
"ConvergeOnRealFreq:=", true,
"MaximumPasses:=", 3,
"MinimumPasses:=", 1,
"MinimumConvergedPasses:=", 1,
"PercentRefinement:=", 30,
"IsEnabled:=", true,
"BasisOrder:=", 1,
"UseIterativeSolver:=", false,
"DoLambdaRefine:=", true,
"DoMaterialLambda:=", true,
```

```
"SetLambdaTarget:=", false,
"Target:=", 0.2,
"UseMaxTetIncrease:=", false,
"MaxTetIncrease:=", 1000000,
"UsingConstantDelta:=", 0,
"ConstantDelta:=", "0s",
"NumberSolveSteps:=", 1)
A Driven solution type with ports and matrix convergence:
oModule.InsertSetup "HfssDriven",
Array("NAME:Setup2",
"Frequency:=", "1GHz",
"PortsOnly:=", false,
"MaxDeltaS:=", 0.02,
"UseMatrixConv:=", true,
Array("NAME:ConvergenceMatrix",
"AllDiagEntries:=", true,
"MagMinThreshold:=", 0.01,
"Entry:=", Array("Port1:=", "abc", "ModeNum1:=", 1)),
"MaximumPasses:=", 6,
"MinimumPasses:=", 1,
"MinimumConvergedPasses:=", 1,
```

```
"PercentRefinement:=", 30,
"IsEnabled:=", true,
"BasisOrder:=", 1,
"UseIterativeSolver:=", false,
"DoLambdaRefine:=", true,
"DoMaterialLambda:=", true,
"SetLambdaTarget:=", false,
"Target:=", 0.3333,
"UseMaxTetIncrease:=", false,
"MaxTetIncrease:=", 1000000,
"PortAccuracy:=", 2,
"UseABCOnPort:=", true,
"SetPortMinMaxTri:=", false,
"EnableSolverDomains:=", false,
"ThermalFeedback:=", false,
"UsingConstantDelta:=", 0,
"ConstantDelta:=", "0s",
"NumberSolveSteps:=", 1)
A Driven solution type with Multi-Frequencies specified:
oModule.InsertSetup "HfssDriven", Array("NAME:Setup2", "AdaptMultipleFreqs:=", true, _
```

```
Array("NAME:MultipleAdaptiveFreqsSetup", "1.1GHz:=", Array( _
0.02), "10GHz:=", Array(0.02), "11GHz:=", Array(0.02), "12.5GHz:=", Array(0.02),
"15GHz:=", Array( _
0.02), "20GHz:=", Array(0.02)), "MaximumPasses:=", 6, "MinimumPasses:=", 1, "Min-
imumConvergedPasses:=",  _
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:=", false, "UseIterativeSolver:=",  _
false, "SaveRadFieldsOnly:=", false, "SaveAnyFields:=", true, "IESolverType:=",  _
"Auto", "LambdaTargetForIESolver:=", 0.15, "UseDefaultLambdaTgtForIESolver:=",  _
true, "RayDensityPerWavelength:=", 4, "MaxNumberOfBounces:=", 5)
An Driven Solution type with Broadband specified:
oModule.InsertSetup "HfssDriven", Array("NAME:Setup1", "AdaptMultipleFreqs:=", true, _
Array("NAME:MultipleAdaptiveFreqsSetup", Array("NAME:Broadband", "Low:=",  _
"2GHz", "High:=", "20GHz")), "MaximumPasses:=", 6, "MinimumPasses:=", 1, "Min-
imumConvergedPasses:=",  _
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:=", false, "UseIterativeSolver:=",  _
```

```
false, "SaveRadFieldsOnly:=", false, "SaveAnyFields:=", true, "IESolverType:=",  _
"Auto", "LambdaTargetForIESolver:=", 0.15, "UseDefaultLambdaTgtForIESolver:=",  _
true, "RayDensityPerWavelength:=", 4, "MaxNumberOfBounces:=", 5)
oModule.InsertFrequencySweep "Setup1", Array("NAME:Sweep", "IsEnabled:=", true,
"RangeType:=",  _
"LinearCount", "RangeStart:=", "2GHz", "RangeEnd:=", "20GHz", "RangeCount:=",  _
451, "Type:=", "Interpolating", "SaveFields:=", false, "SaveRadFields:=",  _
false, "InterpTolerance:=", 0.5, "InterpMaxSolns:=", 250, "InterpMinSolns:=",  _
0, "InterpMinSubranges:=", 1, "ExtrapToDC:=", false, "InterpUseS:=", true, "Inter-
pUsePortImped:=",  _
false, "InterpUsePropConst:=", true, "UseDerivativeConvergence:=", false, "Inter-
pDerivTolerance:=",  _
0.2, "UseFullBasis:=", true, "EnforcePassivity:=", true, "PassivityErrorTolerance:=",
_
0.0001)
```

InsertSetup [Transient]

Add a new solution setup to a Transient design

Command: **HFSS>Analysis Setup>Add Solution Setup**

Syntax: InsertSetup <SetupType>, <AttributesArray>

Return Value: None

Parameters: <SetupType>

Type: <string>

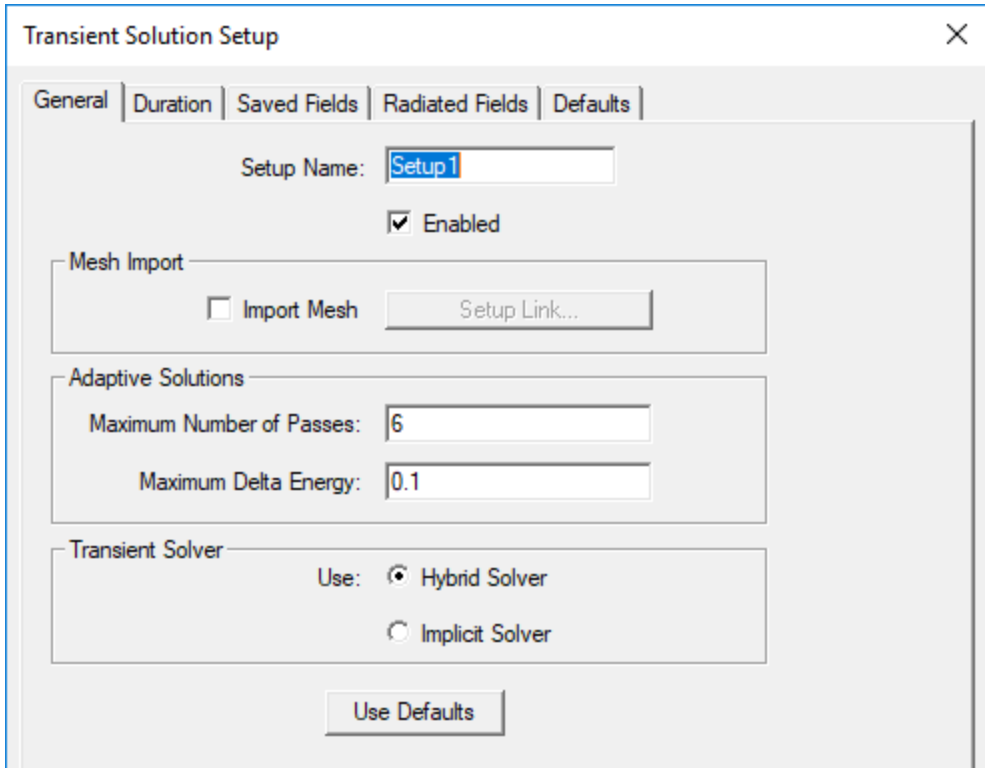
"HfssTransient". Must match the HFSS solution type.

`<AttributesArray>`

Array("NAME:<SetupName>", <Named Parameters>)

`<Named Parameters>`

The parameters correspond to fields in the **Setup** dialog box. For example, here is default dialog for an HFSS Transient driven Composite Excitation setup with one excitation assigned. To see the required parameters for a specific set of parameters and their format, use the record script function, and view the resulting script in a text editor.

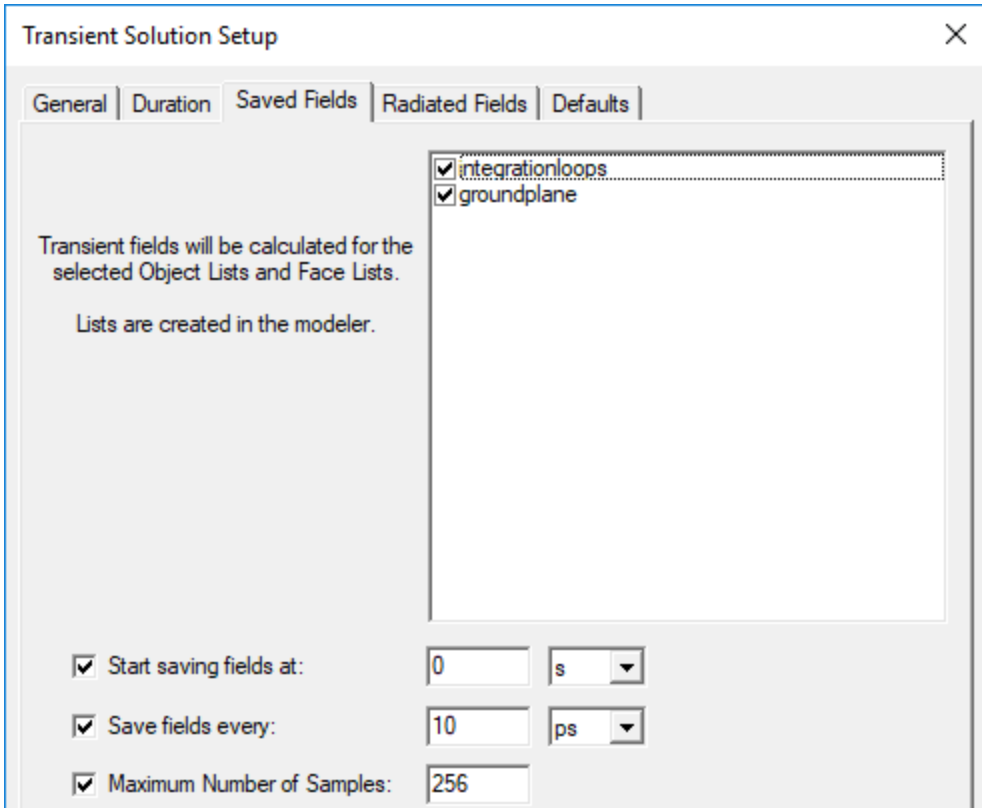


The first part of a vbs script corresponds to the fields and selections on the **General** tab:

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

oModule.InsertSetup "HfssTransient", Array("NAME:Setup1", "Frequency:=", "5GHz", "MaxDeltaE:=",
0.1, _"MaximumPasses:=", 6, "UseImplicitSolver:=", true, "IsEnabled:=", true,
```

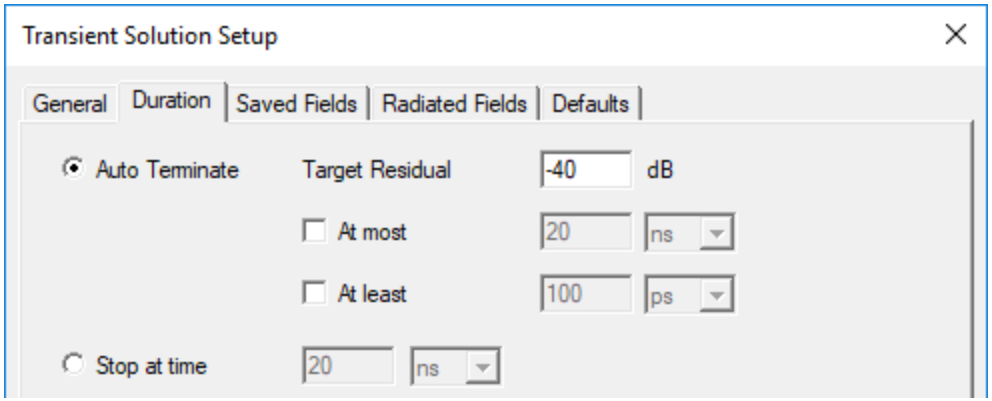
The next parameters correspond to fields and selections on the **Saved Fields** tab:




```
Array("NAME:Transient", "ListsForFields:=", Array("integrationloops", "groundplane"), _
"UseSaveCount:=", 1, "SaveCount:=", 256, _
"UseSaveStart:=", 1, "SaveStart:=", "0s", _
"UseSaveDelta:=", 1, "SaveDelta:=", "10ps", _
"SaveRadFields:=", 0, "SaveFDRadFields:=", 0, _
```

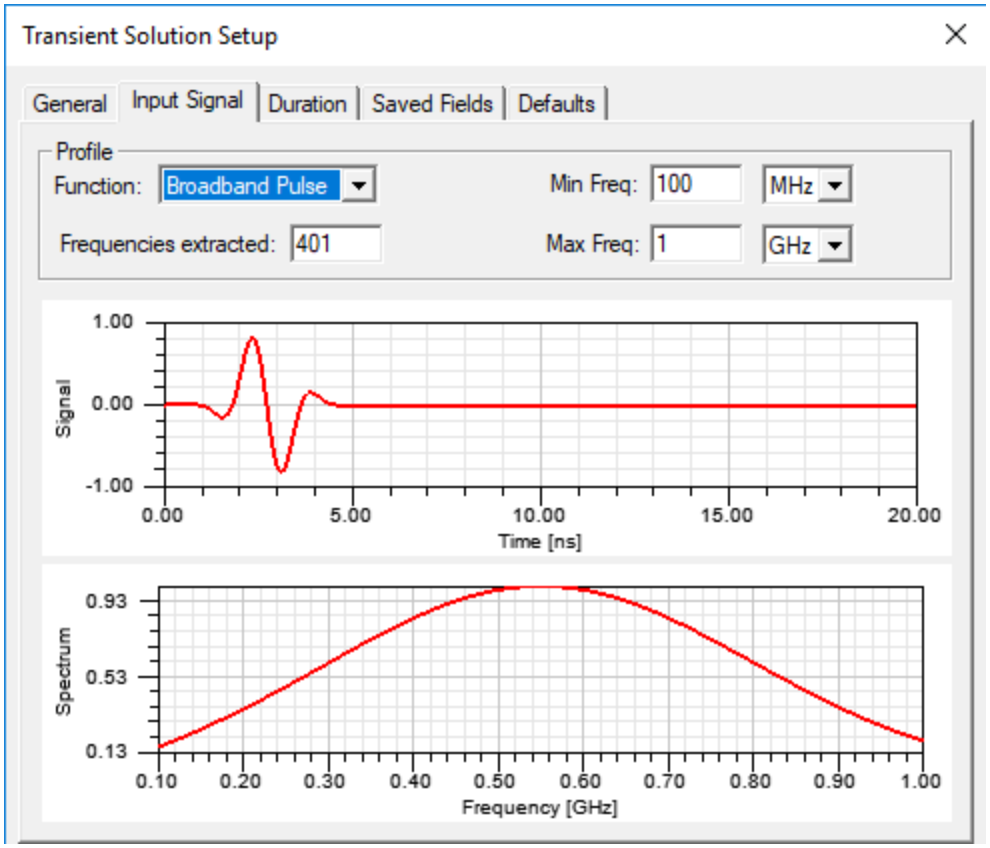
You must create one or more face lists and select them to enable the ability to Save fields at specified intervals.

The next part corresponds to fields on the **Duration** tab.




```
"UseAutoTermination:=", 1, "SteadyStateCriteria:=", 0.01, _
"UseMinimumDuration:=", 0, "TerminateOnMaximum:=", 0))
```

The choices made for solver, Transient solution type as Network Analysis or Composite Excitation, Mesh Linking, and whether you use Expression Cash, Derivatives, and HPC will all affect the parameters array for a particular setup. The Network Analysis solution type includes the **Input Signal** tab.



The vbs script for this setup includes parameters for this information.

```
Array("NAME:Transient", "TimeProfile:=", "Broadband Pulse", "HfssFrequency:=", "5GHz", _
"MinFreq:=", "100MHz", "MaxFreq:=", "1GHz", _
"NumFreqsExtracted:=", 401, _
"SweepMinFreq:=", "100MHz", "SweepMaxFreq:=", "1GHz",
```

To see the parameters for a specific Setup and their format, use the record script function, and view the resulting script in a text editor. For descriptions of the parameters, see the Online help for the setup. See the examples below.

VB Example: Transient Solution Type

```
Set oModule = oDesign.GetModule("AnalysisSetup")
oModule.InsertSetup "HfssTransient", _
Array("NAME:Setup1", _
"Frequency:=", "1GHz", _
"MaxDeltaE:=", 0.1, _
"MaximumPasses:=", 20, _
"IsEnabled:=", true, _
"BasisOrder:=", -1, _
"NoAdditionalRefinementOnImport:=", true, _
Array("NAME:Transient", _
"UseAutoTermination:=", 1, _
"SteadyStateCriteria:=", 0.001, _
"UseMinimumDuration:=", 0, _
"TerminateOnMaximum:=", 1, _
"UseMaxTime:=", 1, _
"MaxTime:=", "20000ps"))
```

*VB Example:*Transient Network Solution Type

```
Dim oAnsoftApp
Dim oDesktop
Dim oProject
Dim oDesign
Dim oEditor
Dim oModule
Set oAnsoftApp = CreateObject("Ansoft.ElectronicsDesktop")
Set oDesktop = oAnsoftApp.GetAppDesktop()
oDesktop.RestoreWindow
Set oProject = oDesktop.SetActiveProject("transient")
Set oDesign = oProject.SetActiveDesign("HFSSDesign1")
Set oModule = oDesign.GetModule("AnalysisSetup")
oModule.InsertSetup "HfssTransient", Array("NAME:Setup1", "Frequency:=", "5GHz", "MaxDeltaS:=",
_
0.02, "MaximumPasses:=", 20, "UseImplicitSolver:=", true, "IsEnabled:=", true, "BasisOrder:=", _
-1, Array("NAME:Transient", "TimeProfile:=", "Broadband Pulse", "HfssFrequency:=", _
"5GHz", "MinFreq:=", "100MHz", "MaxFreq:=", "1GHz", "NumFreqsExtracted:=", 401, "Sweep-
MinFreq:=", _
"100MHz", "SweepMaxFreq:=", "1GHz", "ListsForFields:=", Array("Facelist1"), "UseSaveCount:=", _
1, "SaveCount:=", 256, "UseSaveStart:=", 1, "SaveStart:=", "0s", "UseSaveDelta:=", _
1, "SaveDelta:=", "10ps", "SaveRadFields:=", 0, "SaveFDRadFields:=", 0, "UseAutoTermination:=",
_
1, "SteadyStateCriteria:=", 0.01, "UseMinimumDuration:=", 0, "TerminateOnMaximum:=", _
```

```
1, "UseMaxTime:=", 1, "MaxTime:=", "20ns"))
```

Python Syntax	InsertSetup("HfssTransient", [<parameters>])
Python Example	oModule.InsertSetup("HfssTransient", ["NAME:Setup2", "Frequency:=" , "5GHz", "MaxDeltaS:=" , 0.02, "MaximumPasses:=" , 20, "UseImplicitSolver:=" , True, "IsEnabled:=" , True, "BasisOrder:=" , -1, ["NAME:Transient", "TimeProfile:=" , "Broadband Pulse", "HfssFrequency:=" , "5GHz", "MinFreq:=" , "100MHz", "MaxFreq:=" , "1GHz", "NumFreqsExtracted:=" , 401, "SweepMinFreq:=" , "100MHz", "SweepMaxFreq:=" , "1GHz", "ListsForFields:=" , ["Facelist1"], "UseSaveCount:=" , 1, "SaveCount:=" , 256, "UseSaveStart:=" , 1, "SaveStart:=" , "0s", "UseSaveDelta:=" , 1, "SaveDelta:=" , "10ps", "SaveRadFields:=" , 0, "SaveFDRadFields:=" , 0, "UseAutoTermination:=" , 1,

	"SteadyStateCriteria:=" , 0.01, "UseMinimumDuration:=" , 0, "TerminateOnMaximum:=" , 0]])

PasteDrivenSetup

Pastes a driven setup.

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

Python Syntax	PasteDrivenSetup()
Python Example	`oModule.PasteDrivenSetup()`

VB Syntax	PasteDrivenSetup
VB 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()`

VB Syntax	PasteEigenSetup
VB Example	`oModule.PasteEigenSetup`

PasteSetup

Use: Paste a solve setup.

Syntax: PasteSetup

Return Value: None

VB Example: `oModule.PasteSetup`

PasteSweep

Pastes a copied sweep.

UI Access	Right-click on a setup, select **Paste**

Parameters	Name: <*SetupName*> — Type: String — Description: Name of setup where the copied sweep is pasted to.
Return Value	None.

Python Syntax	PasteSweep (<*SetupName*>)
Python Example	`oModule.PasteSweep("Setup1")`

VB Syntax	PasteSweep <*SetupName*>
VB Example	`oModule.PasteSweep "Setup1"`

RenameDrivenSweep

Renames an existing frequency sweep in HFSS. For HFSS-IE use RenameSweep.

UI Access	Right-click a frequency sweep in the Project Manager and then click **Rename** on the shortcut menu.
Parameters	Name: <*SetupName*> — Type: String — Description: Name of setup to which the sweep belongs. Name: <*OldSweepName*> — Type: String — Description: Name of the sweep to be renamed. Name: <*NewSweepName*> — Type: String — Description: New name for the sweep.
Return Value	None.

Python Syntax	RenameDrivenSweep (<*SetupName*>, <*OldSweepName*>, <*NewSweepName*>)

Python Example	`oModule.RenameDrivenSweep("Setup1", "Sweep1", "MySweep")`

VB Syntax	RenameDrivenSweep <*SetupName*>, <*OldSweepName*>, <*NewSweepName*>
VB Example	`oModule.RenameDrivenSweep "Setup1", "Sweep1", "MySweep"`

RenameSetup

Renames an existing solution setup.

UI Access	Right-click a solution setup in the Project Manager and then click **Rename** on the shortcut menu.
Parameters	

Name	Type	Description
<*OldSetupName*>	String	Name of the solution setup being renamed.
<*NewSetupName*>	String	New name for the solution setup.

Return Value	None.

Python Syntax	RenameSetup (<*OldSetupName*>, <*NewSetupName*>)
Python Example	`oModule.RenameSetup("Setup1", "MySetup")`

VB Syntax	RenameSetup <*OldSetupName*>, <*NewSetupName*>
VB Example	`oModule.RenameSetup "Setup1", "MySetup"`

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")`

VB Syntax	ResetToTimeZero <*setupName*>
VB 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()`

VB Syntax	RevertAllToInitial
VB 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()`

VB Syntax	RevertAllToZeroDisplacement
VB Example	`oModule.RevertAllToZeroDisplacement`

RevertSetupToInitial

Marks the current mesh for a solution setup as invalid. This will force the next simulation to begin with the initial mesh.

UI Access	Right-click a solution setup in the Project Manager and then click **Rename** on the shortcut menu.
Parameters	Name / Type / Description <*SetupName*> / String / Name of specified setup.
Return Value	None.

Python Syntax	RevertSetupToInitial (<*SetupName*>)
Python Example	`oModule.RevertSetupToInitial("Setup1")`

VB Syntax	RevertSetupToInitial <*SetupName*>
VB Example	`oModule.RevertSetupToInitial "Setup1"`

RevertSetupToZeroDisplacement

Reverts the displacement values of objects for a specified setup to zero.

UI Access	N/A
Parameters	Name / Type / Description <*SetupName*> / String / Name of specified setup.
Return Value	None.

Python Syntax	RevertSetupToZeroDisplacement (<*SetupName*>)
Python Example	`oModule.RevertSetupToZeroDisplacement("Setup1")`

VB Syntax	RevertSetupToZeroDisplacement <*SetupName*>
VB 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	Name | Type | Description <*SetupName*> | String | Name of the setup to be solved
Return Value	None

Python Syntax	SolveSetup (<*SetupName*>)
Python Example	`oModule.SolveSetup ("Setup1")`

VB Syntax	SolveSetup <*SetupName*>
VB Example	`oModule.SolveSetup "Setup1"`

This page intentionally
left blank.

11 - 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.

CopySetup

DeleteSetups [Optimetrics]

DistrbutedAnalyzeSetup

EditSetup

EditSetup [Parametric]

EditSetup [Optimization]

EditSetup [Sensitivity]

EditSetup [Statistical]

EnableSetup

ExportDXConfigFile

ExportOptimetricsProfile

ExportOptimetricsResult

ExportParametricResults

ExportParametricSetupTable

ExportRespSurfaceMinMaxTable

The topics for this section include:

CopySetup

Copy the specified Optimetrics setup.

UI Access	NA
Parameters	Name | Type | Description <*SetupName*> | String | Name of the setup.
Return Value	None.

Python Syntax	CopySetup (<*SetupName*>)
Python Example	`oModule.CopySetup ("OptimizationSetup1")`

VB Syntax	CopySetup <*SetupName*>
VB Example	`oModule.CopySetup "OptimizationSetup1"`

DeleteSetups [Optimetrics]

Deletes the specified Optimetrics setups.

UI Access	Right-click the setup in the project tree, and then click **Delete** on the shortcut menu
Parameters	Name | Type | Description <NameArray> | Array of Strings | An Array of Setup Names
Return Value	None

Python Syntax	DeleteSetups (<NameArray>)
Python Example	`oModule.DeleteSetups (["OptimizationSetup1"])`

VB Syntax	DeleteSetups <NameArray>
VB Example	`oModule.DeleteSetups Array("OptimizationSetup1")`

DistributedAnalyzeSetup

Distributes all variable value instances within a parametric sweep to different machines already specified from within the user interface

UI Access	Right-click the parametric setup name in the project tree and select Distribute Analysis.
Parameters	<table><tr><th>Name</th><th>Type</th><th>Description</th></tr><tr><td><ParametricSetupName></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 `true`. If it's `false`, the command immediately returns while the analysis or analyses run in the background. The status of the analyses can be checked with the `AreThereSimulationsRunning` command.</td></tr></table>
Return Value	None

Python Syntax	DistributedAnalyzeSetup (<ParametricSetupName>)
Python Example	`oModule.DistributedAnalyzeSetup("ParametricSetup1")`

VB Syntax	DistributedAnalyzeSetup <ParametricSetupName>
VB Example	`oModule.DistributedAnalyzeSetup "ParametricSetup1"`

EditSetup

Modifies an existing solution setup.

<table>
<tr><td>UI Access</td><td>Double-click a solution setup in the project tree to modify its settings.</td></tr>
<tr><td>Parameters</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 solve setup being edited.</td></tr>
<tr><td><Attributes></td><td>Array</td><td>Structured array.
Array("NAME:<NewSetupName>", <NamedParameters>)
See the InsertSetup command for details and examples.</td></tr>
</table>
</td></tr>
<tr><td>Return Value</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("EddyCurrent",
[
  "NAME:EddyCurrent",
  "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 = oModule.GetModule("AnalysisSetup")
oModule.EditSetup("RFDischarge1",
  [
    "NAME:RFDischarge1",
```

```
"Enabled:="             , True,
[
  "NAME:MeshLink",
  "ImportMesh:="          , True,
  "Project:="             , "This Project*",
  "Product:="             , "HFSS",
  "Design:="              , "This Design*",
  "Soln:="                , "Setup1 : Sweep",
  [
    "NAME:Params",
    "bend_angle:="          , "bend_angle"
  ],
  "ForceSourceToSolve:="  , True,
  "PreservePartnerSoln:=" , False,
  "PathRelativeTo:="      , "SourceProduct",
  "ApplyMeshOp:="         , True
],
[
  "NAME:Excitations",
  [
    "NAME:1:1",
```

```
        "Magnitude:="           , "1",
        "Phase:="               , "0deg"
      ],
      [
        "NAME:2:1",
        "Magnitude:="           , "0",
        "Phase:="               , "0deg"
      ]
    ],
    [
      "NAME:Frequencies",
      "10GHz"
    ],
    "Minimum Power:="       , "0.01",
    "Maximum Power:="       , "1000000",
    "Minimum Pressure:="    , "100pascal",
    "Maximum Pressure:="    , "101325pascal",
    "Postproc Sampling:="   , 500,
    "Temperature:="         , "0cel",
    "BuiltInGas:="          , "Helium"
```

	])

VB Syntax	EditSetup <*SetupName*>, <*Attributes*>
VB Example	oModule.EditSetup "Setup1", Array("NAME:NewSetup", "AdaptiveFreq:=", "1GHz", "EnableDistribProbTypeOption:=", false, "SaveFields:=", "true", "Enabled:=", true, Array("NAME:Cap", "MaxPass:=", 10, "MinPass:=", 1, "MinConvPass:=", 2, "PerError:=", 1, "PerRefine:=", 30, "AutoIncreaseSolutionOrder:=", false, "SolutionOrder:=", "Normal"), Array("NAME:DC", "Residual:=", 1E-005, "SolveResOnly:=", false, Array("NAME:Cond",

```
                "MaxPass:=", 10,
                "MinPass:=", 1,
                "MinConvPass:=", 1,
                "PerError:=", 1,
                "PerRefine:=", 30),
        Array("NAME:Mult",
                "MaxPass:=", 1,
                "MinPass:=", 1,
                "MinConvPass:=", 1,
                "PerError:=", 1,
                "PerRefine:=", 30)),
Array("NAME:AC",
        "MaxPass:=", 10,
        "MinPass:=", 1,
        "MinConvPass:=", 2,
        "PerError:=", 1,
        "PerRefine:=", 30))
```

EditSetup [Parametric]

Modifies an existing parametric setup

UI Access	Right-click the setup in the project tree, and then click **Properties** on the shortcut menu.
Parameters	<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><ParametricParams></td><td>List</td><td>List that defines the parameters of the parametric setup; examples are listed below.</td></tr></table>
Return Value	None

Python Syntax	EditSetup (<SetupName>, <ParametricParams>)
Python Example	See EditSetup [Optimization]

VB Syntax	EditSetup <SetupName>, <ParametricParams>
VB Example	See EditSetup [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><SetupName></td><td>String</td><td>Name of the Setup</td></tr><tr><td><OptimizationParams></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
])
```

VB Syntax	EditSetup <SetupName>, <OptimizationParams>
VB Example	

```
oModule.EditSetup "OptimizationSetup1",
Array("NAME:OptimizationSetup1", "UseFastCalculationUpdateAlgo:=",  _
  false, "FastCalcOptCtrledByUser:=", false,
"IsEnabled:=", true, "SaveSolutions:=",  _
  false, Array("NAME:StartingPoint"), "Optimizer:=",
"Quasi Newton", Array("NAME:AnalysisStopOptions", "StopForNumIteration:=",  _
  true, "StopForElapsTime:=", false,
"StopForSlowImprovement:=", false, "StopForGrdTolerance:=",  _
  false, "MaxNumIteration:=", 1000, "MaxSolTimeInSec:=",
 3600, "RelGradientTolerance:=",  _
  0, "MinNumIteration:=", 10), "CostFuncNormType:=",
"L2", "PriorPSetup:=", "", "PreSolvePSetup:=",  _
```

	true, Array("NAME:Variables"), Array("NAME:LCS"), Array("NAME:Goals", Array("NAME:Goal", "ReportType:=", _ "Standard", "Solution:=", "TR", Array("NAME:SimValueContext", "SimValueContext:=", Array(_ 1, 0, 2, 0, false, false, -1, 1, 0, 1, 1, "", 0, 0)), "Calculation:=", "Time", "Name:=", _ "Time", Array("NAME:Ranges", "Range:=", Array("Var:=", "Time", "Type:=", "a")), "Condition:=", _ "==", Array("NAME:GoalValue", "GoalValueType:=", "Independent", "Format:=", _ "Real/Imag", "bG:=", Array("v:=", "[1;]")), "Weight:=", "[1;]")), "Acceptable_Cost:=", _ 0, "Noise:=", 0.0001, "UpdateDesign:=", false, "UpdateIteration:=", 5, "KeepReportAxis:=", _ true, "UpdateDesignWhenDone:=", true)

EditSetup [Sensitivity]

Modifies an existing sensitivity setup.

UI Access	Right-click the setup in the project tree, and then click Properties on the shortcut menu

Parameters	<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></td><td></td><td></td></tr></table>
Return Value	None

Python Syntax	EditSetup (<SetupName>, <SensitivityParams>)
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:=", 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])

VB Syntax	EditSetup <SetupName>, <SensitivityParams>
VB Example	oModule.EditSetup("OptimizationSetup1", Array("NAME:OptimizationSetup1", "UseFastCalculationUpdateAlgo:=", False, "FastCalcOptCtrledByUser:=", False, "IsEnabled:=", True, "SaveSolutions:=", False,

```
Array(
"NAME:StartingPoint"
),
"Optimizer:=", "Quasi Newton",
Array(
"NAME:AnalysisStopOptions",
"StopForNumIteration:="        , True,
"StopForElapsTime:=", False,
"StopForSlowImprovement:=", False,
"StopForGrdTolerance:=", False,
"MaxNumIteration:=", 1001,
"MaxSolTimeInSec:=", 3600,
"RelGradientTolerance:=", 0,
"MinNumIteration:="    , 10
),
"CostFuncNormType:=", "L2",
"PriorPSetup:=", "",
"PreSolvePSetup:=", True,
Array(
"NAME:Variables"
),
```

```
Array(
"NAME:LCS"
),
Array(
"NAME:Goals",
Array(
"NAME:Goal",
"ReportType:=", "Standard",
"Solution:=", "TR3",
Array(
"NAME:SimValueContext",
"SimValueContext:=",
Array(1,0,2,0,False,False,-1,1,0,1,1,"",0,0)
),
"Calculation:=", "mag(DIFF1.VAL)",
"Name:=", "DIFF1.VAL",
Array(
"NAME:Ranges",
"Range:=", Array("Var:=", "Time","Type:=", "a")
```

```
),
"Condition:=", "==",
Array(
"NAME:GoalValue",
"GoalValueType:=", "Independent",
"Format:=", "Real/Imag",
"bG:=", Array("v:=", "Array(1;]")
),
"Weight:=", "Array(1;]"
)
),
"Acceptable_Cost:=", 0,
"Noise:=", 0.0001,
"UpdateDesign:=", False,
"UpdateIteration:=", 5,
"KeepReportAxis:=", True,
"UpdateDesignWhenDone:=", True
))
```

EditSetup [Statistical]

Modifies an existing statistical setup.

UI Access	Right-click the setup in the project tree, and clickProperties on the shortcut menu.
Parameters	<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></td><td></td><td></td></tr></table>
Return Value	None

Python Syntax	EditSetup (<SetupName>, <StatisticalParams>)
Python Example	`See EditSetup [Optimization]`

VB Syntax	EditSetup <SetupName>, <StatisticalParams>
VB Example	`See EditSetup [Optimization]`

Example:

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

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

Set oDesktop = oAnsoftApp.GetAppDesktop()

oDesktop.RestoreWindow

Set oProject = oDesktop.SetActiveProject("optiguides")

Set oDesign = oProject.SetActiveDesign("HFSSModel1")

Set oModule = oDesign.GetModule("Optimetrics")

oModule.EditSetup "StatisticalSetup1", Array("NAME:StatisticalSetup1",
Array("NAME:ProdOptiSetupData",
"SaveFields:=", true, "CopyMesh:=", false),
Array("NAME:StartingPoint", "$length:=", "7.824547736mm",
"$width:=", "14.8570192mm"),
"MaxIterations:=", 50,
"PriorPSetup:=", "",

Array("NAME:Variables",
"$length:=", Array("i:=", true,
"int:=", false,
"Dist:=", "Uniform",
"Tol:=", "10%", "StdD:=", "0.2mm", "Min:=", "-3",
"Max:=", "3", "Shape:=", "1", "Scale:=", "0.04mm",
"Location:=", "0.4mm",
"Dataset:=", "", "LatinHypercube:=", "true", "VarMin:=", "0.2mm", "VarMax:=", "0.6mm", "Prob:=",
"0.01",
"Mean:=", "0.4mm"),

"$width:=", Array("i:=", true,
"int:=", false,
"Dist:=", "Gaussian",
"Tol:=", "10%",
"StdD:=", "0.2mm",
"Min:=", "-3", "Max:=", "3",
"Shape:=", "1",
```

```
"Scale:=", "0.04mm",
"Location:=", "0.4mm",
"Dataset:=", "",
"LatinHypercube:=", "true",
"VarMin:=", "0.2mm", "VarMax:=", "0.6mm",
"Prob:=", "0.02",
"Mean:=", "0.4mm")),

Array("NAME:Goals", Array("NAME:Goal",
"ReportType:=", "Modal Solution Data",
"Solution:=", "Setup1 : PortOnly",
Array("NAME:SimValueContext", "Domain:=", "Sweep"),
"Calculation:=", "returnloss",
"Name:=", "returnloss",

Array("NAME:Ranges",
"Range:=", Array("Var:=", "Freq",
"Type:=", "s",
"Start:=", "8.2GHz", "Stop:=", "0"))),

Array("NAME:Goal",
"ReportType:=", "Modal Solution Data",
"Solution:=", "Setup1 : PortOnly",
Array("NAME:SimValueContext",
"Domain:=", "Sweep"),
"Calculation:=", "reflect",
"Name:=", "reflect",
Array("NAME:Ranges",
"Range:=", Array("Var:=", "Freq",
"Type:=", "s",
"Start:=", "8.2GHz", "Stop:=", "0")))))
```

EnableSetup

Enables and disables a defined optimetrics analysis setup.

UI Access	Right-click on a setup in the project tree, select **Enable Setup** or **Disable Setup**
Parameters	<table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><*SetupName*></td><td>String</td><td>Name of specified setup.</td></tr><tr><td><*Enable*></td><td>Boolean</td><td>Determines whether enable or disable a setup. • **True** - enable setup. • **False** - disable setup.</td></tr></table>
Return Value	None.

Python Syntax	EnableSetup(<*SetupName*>, <*Enable*>)
Python Example	`oModule.EnableSetup("OptimizationSetup1", True)`

VB Syntax	EnableSetup <*SetupName*>, <*Enable*>
VB Example	`oModule.EnableSetup "OptimizationSetup1", true`

ExportDXConfigFile

Create an xml file with the setup information for Design Xplorer

UI Access	Right click on the Design Xplorer setup in the project tree and choose **Export External Connector Addin Configuration…**

Parameters	<table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><SetupName></td><td>String</td><td>Must be one of existing DesignExplorer setup names</td></tr><tr><td><FileName></td><td>String</td><td>Must be a valid file path and name</td></tr></table>
Return Value	None

Python Syntax	ExportDXConfigFile (<SetupName>, <FileName>)
Python Example	`oModule.ExportDXConfigFile ("DesignXplorerSetup1",` `"c:/exportdir/DXSetup1.xml")`

VB Syntax	ExportDXConfigFile <SetupName>, <FileName>
VB Example	`oModule.ExportDXConfigFile ("DesignXplorerSetup1",` `"c:/exportdir/DXSetup1.xml")`

ExportOptimetricsProfile

Export Optimetrics profile data

UI Access	Right click on the Optimetrics setup in the project tree and choose **View Analysis Result…** On the **Post Analysis Display** dialog box, click the **Profile** tab and click on the **Export** button.
Parameters	<table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><SetupName></td><td>String</td><td>Must be one of the existing Parametric, Optimization, Sensitivity, Statistical or DesignXplorer setup names</td></tr></table>

	<FileName>	String	Must be a valid file path and name with extension of csv, tab, dat, or txt
	[profileNum]	String	Must be a numeric string. Optional: defaulted to last profile number. It should be a zero indexed profile number.
Return Value	None		

Python Syntax	ExportOptimetricsProfile (<SetupName>, <FileName>, [profileNum]
Python Example	`oModule.ExportOptimetricsProfile ("StatisticalSetup1", "c:/exportdir/test.csv")`

VB Syntax	ExportOptimetricsProfile <SetupName>, <FileName>, [profileNum]
VB Example	`oModule.ExportOptimetricsProfile` `"StatisticalSetup1", "c:/exportdir/test.csv"`

ExportOptimetricsResult

Export an existing Optimization, Sensitivity, Statistical or DesignXplorer result. (Does not export Parametric results.)

UI Access	Right click on the desired Optimetrics setup in the project tree and choose **View Analysis Result…** On the **Post Analysis Display** dialog box, click the **Result** tab, then select **Table** view, and click on the **Export** button		
Parameters	Name	Type	Description
	<SetupName>	String	Must be one of the existing Optimization, Sensitivity, Statistical, or DesignXplorer setup names
	<FileName>	String	Must be a valid file path and name with extension of csv, tab, dat, or txt..
	[useFullOutputName]	Boolean	Optional: defaulted to false. If set to true values will be printed with units.

			This parameter is ignored for Optimization and Statistical results.
Return Value	None		

Python Syntax	ExportOptimetricsResult (<SetupName>, <FileName>, [useFullOutputName])
Python Example	`oModule.ExportOptimetricsResult ( "StatisticalSetup1", "c:/exportdir/test.csv",false)`

VB Syntax	ExportOptimetricsResult <SetupName>, <FileName>, [useFullOutputName]
VB Example	`oModule.ExportOptimetricsResult "StatisticalSetup1", "c:/exportdir/test.csv", false`

ExportParametricResults

Export existing Parametric results.

UI Access	Right click on the desired Parametric setup in the project tree and choose **View Analysis Result…** On the **Post Analysis Display** dialog box, click the **Result** tab, then select **Table** view, and click on the **Export** button.
Parameters	Name / Type / Description <SetupName> / String / Must be one of the existing Parametric setup names <FileName> / String / Must be a valid file path and name with extension of csv, tab, dat or txt <bOutputUnits> / Boolean / If set to true, values will be printed with units
Return Value	None

Python Syntax	ExportParametricResults (<SetupName>, <FileName>, <bOutputUnits>)
Python Example	`oModule.ExportParametricResults (` `"ParametricSetup1", "c:/exportdir/test.csv", False)`

VB Syntax	ExportParametricResults <SetupName>, <FileName>, <bOutputUnits>
VB Example	`oModule.ExportParametricResults` `"ParametricSetup1", "c:/exportdir/test.csv",false`

ExportParametricSetupTable

Exports the parametric setup table as a CSV file.

UI Access	Double-click parametric setup. Select **Table** tab. Click **Export**.
Parameters	<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>
Return Value	None

Python Syntax	ExportParametricSetupTable (<SetupName>, <filePath>)
Python Example	`oModule.ExportParametricSetupTable('ParametricSetup1', 'E:/Files/ParametricSetup1_Table.csv')`

VB Syntax	ExportParametricSetupTable <SetupName>, <filePath>
VB Example	`obj.ExportParametricSetupTable "ParametricSetup1", "E:/Files/ParametricSetup1_Table.csv"`

ExportRespSurfaceMinMaxTable

Exports min-max table from a response surface to a file

UI Access	Click on **Export...** From the **Response Surface** tab of the Design of Experiments **Post Analysis Display** dialog.
Parameters	<table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><*DOEName*></td><td>String</td><td>Name of the Design of Experiments (DOE) setup.</td></tr><tr><td><*FileName*></td><td>String</td><td>Output file name with path.</td></tr></table>
Return Value	None.

Python Syntax	ExportRespSurfaceMinMaxTable(<*DOEName*>, <*FileName*>)
Python Example	`oModule.ExportRespSurfaceMinMaxTable("DesignOfExperimentsSetup1", "C:/temp/DesignOfExperimentsSetup1_Min-Max_Search.csv")`

VB Syntax	ExportRespSurfaceMinMaxTable <*DOEName*>, <*FileName*>
VB Example	`oModule.ExportRespSurfaceMinMaxTable _` `"DesignOfExperimentsSetup1", _` `"C:/temp/DesignOfExperimentsSetup1_Min-Max_Search.csv"`

ExportRespSurfaceRefinePoints

Exports refinement points table to a file

UI Access	From the **Response Surface** tab of the Design of Experiments **Post Analysis Display** dialog box, select **Refinement Points** option under **View**, Click on **Export...**.
Parameters	Name / Type / Description <*DOEName*> / String / Name of the Design of Experiments (DOE) setup. <*FileName*> / String / Output file name with path.
Return Value	None.

Python Syntax	ExportRespSurfaceRefinePoints(<*DOEName*>, <*FileName*>)
Python Example	`oModule.ExportRespSurfaceRefinePoints("DesignOfExperimentsSetup1",` `"C:/temp/DesignOfExperimentsSetup1_Refine_Points.csv")`

VB Syntax	ExportRespSurfaceRefinePoints <*DOEName*>, <*FileName*>
VB Example	`oModule.ExportRespSurfaceRefinePoints_` `"DesignOfExperimentsSetup1", _`

	"C:/temp/DesignOfExperimentsSetup1_Refine_Points.csv"

ExportRespSurfaceResponsePoints

Exports response points table to a file

UI Access	From the **Response Surface** tab of the Design of Experiments **Post Analysis Display** dialog box, select **Response Points** option under **View**, Click on **Export...**.
Parameters	Name / Type / Description <*DOEName*> / String / Name of the Design of Experiments (DOE) setup. <*FileName*> / String / Output file name with path.
Return Value	None.

Python Syntax	ExportRespSurfaceResponsePoints (<*DOEName*>, <*FileName*>)
Python Example	oModule.ExportRespSurfaceResponsePoints("DesignOfExperimentsSetup1", "C:/temp/DesignOfExperimentsSetup1_Response_Points.csv")

VB Syntax	ExportRespSurfaceResponsePoints <*DOEName*>, <*FileName*>
VB 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")`

VB Syntax	ExportRespSurfaceVerificationPoints <*DOEName*>, <*FileName*>
VB 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.

VB Example:

```
oModule.GenerateVariationData "ParametricSetup1"
```

GetChildNames [Optimetrics]

If used without a specific optimization setup name, gets a list of all setups for all types. If a with a specific setup name, returns names for that optimization setup.

UI Access	NA
Parameters	Name: typeName; Type: text string; Description: 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

UI Access	NA
Parameters	Name | Type | Description Setup Name | text string | A optimetrics setup name, names returned by the GetChildNames().
Return Value	A script object for the setup See discussion of Optimetrics Setup Objects in Object Script Property Function Summary.

Python Syntax	GetChildObject()
Python Example	`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 ID and design name of the active design, in that order separated by a semicolon. If the active design is a sub-circuit, it also returns the parent ID and parent design name, listed before the active design information.

To get the *only* the design's name without the parent design name or design IDs, see oDesign.GetDesignName().

UI Access	N/A
Parameters	None.
Return Value	String indicating the name and ID of the active design and parent design (if the active design is a sub-circuit).

Python Syntax	GetName()
Python Example	oDesign.GetName( ) for a subcircuit *EMDesign2* in a parent design *EMDesign1* design_name = oDesign.GetName() 1;EMDesign1/2;EMDesign2 In the returned string: `1` is the parent design ID; `EMDesign1` is the parent design name. `2` is the active design ID; `EMDesign2` is the active design name.

VB Syntax	GetName
VB Example	`design_name = oDesign.GetName`

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()`

VB Syntax	GetObjPath
VB Example	`oEditor.GetObjPath`

GetOptimetricResult

Returns an Optimetric calculation. The specific calculation is determined by the setup.

UI Access	N/A
Parameters	Name | Type | Description

	<*SetupName*>	String	Optimetrics setup name.
	<*vars*>	Array	Array containing string variable names. Use the **Sweep Definitions** tab in the UI or the <SweepDefs> parameter in the InsertSetup script to determine appropriate inputs.
	<*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'])`

VB Syntax	GetOptimetricResult <*SetupName*>, <*vars*>, <*values*>
VB Example	`dim vars(1)` `vars(0) = "AR"` `vars(1) = "Re"` `dim values(1)` `values(0) = "4.64"` `values(1) = "6e+04"`

	`oModule.GetOptimetricResult "ParametricSetup1", vars, values`

GetPropEvaluatedValue

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

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

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

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

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)`

GetPropSIValue

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

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

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

Python Syntax	GetPropSIValue (<*PropName*>)
Python Example	`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 [Optimetrics]

Returns the property value for a setup property.

UI Access	NA		
Parameters	Name	Type	Description
	property-path		a child object's property path. See property path discussion here.
Return Value	Returns the value of an setup property.		

Python Syntax	GetPropValue(propPath)
Python Example	`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.`

VB Syntax	GetPropValue()
VB Example	`GetPropValue("Optimetrics/OptimizationSetup1/Enabled")` Returns True if Enabled, or False if disabled.

GetSetupNames [Optimetrics]

Gets a list of Optimetrics setup names

UI Access	NA
Parameters	Name | Type | Description None | |
Return Value	IAnsoftCollectionObj – a collection of Optimetrics setup names

Python Syntax	GetSetupNames()
Python Example	`oModule = oDesign.GetModule("Optimetrics")` `setupNames = oModule.GetSetupNames()`

VB Syntax	GetSetupNames()
VB Example	`Set oModule_opt = oDesign.GetModule("Optimetrics")`

	Set opt_setup_list = oModule.GetSetupNames() numsetups = setupnames.Count

GetSetupNamesByType [Optimetrics]

Gets a list of Optimetrics setup names by type.

UI Access	NA
Parameters	Name | Type | Description <Optimetrics type> | String | Examples: parametric, optimization, statistical, sensitivity
Return Value	Array of Optimetrics setup names of the given type.

Python Syntax	GetSetupNamesByType (<Optimetrics type>)
Python Example	for name in oModule.GetSetupNamesByType("optimization") AddInfoMessage(str(name))

VB Syntax	GetSetupNamesByType <Optimetrics type>
VB Example	For each name in oModule.GetSetupNamesByType("optimization") Msgbox name Next

ImportSetup

Import an Optimetric setup from a file.

<table>
<tr><td>UI Access</td><td>NA</td></tr>
<tr><td>Parameters</td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><SetupTypeName></td><td>String</td><td>Must be one of "OptiParametric" , "OptiOptimization", "OptiSensitivity", "OptiStatistical", or "OptiDesignExplorer".</td></tr>
<tr><td><SetupInfo></td><td>Array</td><td>Array("NAME:<SetupName>", "FilePath")
<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>Return Value</td><td>None</td></tr>
</table>

Python Syntax	ImportSetup (<SetupTypeName>, <SetupInfo>)
Python Example	oModule.ImportSetup ("OptiStatistical", ["NAME:StatisticalSetup1", "c:/importdir/mySetupInfoFile"])

VB Syntax	ImportSetup <SetupTypeName>, <SetupInfo>
VB Example	oModule.ImportSetup "OptiStatistical", Array("NAME:StatisticalSetup1", "c:/importdir/mySetupInfoFile")

InsertSetup

Adds a new solution setup.

UI Access	[product] > **Analysis Setup** > **Add Solution Setup**.
Parameters	See table below

Name	Type	Description
<*SetupType*>	String	Type of setup to be inserted.
<*Attributes*>	Array	Structured array. Array("NAME:<SetupName>", <NamedParameters>) <Named Parameters> "AdaptMultipleFreqs:=", <boolean>, "Frequency:=", <string>, "MaxDeltaS:=", <float>, "PortsOnly:=", <boolean>, "UseMatrixConv:=", <boolean>, ### If UseMatrixCov is True, ### [

		"NAME:Matrix Convergence", <Convergence Array>], MaximumPasses:=", <integer>, "MinimumPasses:=", <integer>, "MinimumConvergedPasses:=", <integer>, "PercentRefinement:=", <integer>, "IsEnabled:=", <boolean>, "BasisOrder:=", <integer>, "DoLambdaRefine:=", <boolean>, "DoMaterialLambda:=", <boolean>, "SetLambdaTarget:=", <boolean>, "Target:=", <float>, "PortAccuracy:=", <integer>, "SaveRadFieldsOnly:=", <boolean>, "SaveAnyFields:=", <boolean>, "ListsForFields:=", <array of string>, "IESolverType:=", <string>, "LambdaTargetForIESolver:=", <float> "UseDefaultLambdaTgtForIESolver:=", <boolean>

Return Value	None.

Python Syntax	InsertSetup(<*SetupType*>, <*Attributes*>)
Python Example	Maxwell 3D Eddy Current Setup `oModule = oDesign.GetModule("AnalysisSetup")` `oModule.InsertSetup("EddyCurrent",` `[` `"NAME:EddyCurrent_03",` `"Enabled:=" , True,` `[` `"NAME:MeshLink",` `"ImportMesh:=" , False` `],` `"MaximumPasses:=" , 4,` `"MinimumPasses:=" , 2,` `"MinimumConvergedPasses:=", 1,` `"PercentRefinement:=" , 30,` `"SolveFieldOnly:=" , False,` `"PercentError:=" , 0.1,` `"SolveMatrixAtLast:=" , True,` `"UseNonLinearIterNum:=" , False,`

```
[
  "NAME:ExpressionCache",
  [
    "NAME:CacheItem",
    "Title:="               , "eddy_loss1",
    "Expression:="          , "eddy_loss",
    "Intrinsics:="          , "Phase=\'0deg\'",
    "ReportType:="          , "Fields",
    [
      "NAME:ExpressionContext"
    ]
  ]
],
"UseCacheFor:="         , ["Pass"],
"UseIterativeSolver:="  , False,
"RelativeResidual:="    , 0.0001,
"NonLinearResidual:="   , 0.0001,
"SmoothBHCurve:="       , False,
"Frequency:="           , "200Hz",
"HasSweepSetup:="       , False,
```

```
  "UseHighOrderShapeFunc:=", False,
  "UseMuLink:="           , False,
  "LossAdaptiveCtrl:="    , "0.3"
])
```

HFSS simulation

```
oModule.InsertSetup("HfssDrivenAuto",
["NAME:Setup1",
  "Enabled:=", True,
  "AutoSolverSetting:=", "Balanced",
  ["NAME:Sweeps",
          ["NAME:Sweep",
            "RangeType:=", "LinearStep",
            "RangeStart:=", "1GHz",
            "RangeEnd:=", "10GHz",
            "RangeStep:=", "1GHz"
            ]
  ],
  "SaveRadFieldsOnly:=", False,
  "SaveAnyFields:=", True,
  "Type:=", "Discrete"
  ])
```

```
oModule.InsertSetup("HfssDrivenAuto",
        [
                "NAME:Setup2",
                "SolveType:="           , "Auto",
                "IsEnabled:="           , True,
        [
                "NAME:MeshLink",
                "ImportMesh:="          , False
        ],
        "AutoSolverSetting:="   , "Balanced",
        [
                "NAME:Sweeps",
                [
                        "NAME:Sweep",
                        "RangeType:="           , "LinearCount",
                        "RangeStart:="          , "0GHz",
                        "RangeEnd:="            , "10GHz",
                        "RangeCount:="          , 501
                ]
```

```
        ],
        "SaveRadFieldsOnly:="   , False,
        "SaveAnyFields:="       , True,
        "InfiniteSphereSetup:=" , "Infinite Sphere1",
        "ListsForFields:="      , ["Objectlist2"],
        "Type:="                , "Interpolating"
        ])

oModule.InsertSetup("HfssDriven",
["NAME:Setup3",
        "AdaptMultipleFreqs:=", False,
        "Frequency:=", "5GHz",
        "MaxDeltaS:=", 0.02,
        "PortsOnly:=", False,
        "UseMatrixConv:=", False,
        "MaximumPasses:=", 6,
        "MinimumPasses:=", 1,
        "MinimumConvergedPasses:=", 1,
        "PercentRefinement:=", 30,
        "IsEnabled:=", True,
        "BasisOrder:=", 1,
```

```
        "DoLambdaRefine:=", True,
        "DoMaterialLambda:=", True,
        "SetLambdaTarget:=", False,
        "Target:=", 0.3333,
        "UseMaxTetIncrease:=", False,
        "PortAccuracy:=", 2,
        "UseABCOnPort:=", False,
        "SetPortMinMaxTri:=", False,
        "UseDomains:=", True,
        "UseIterativeSolver:=", False,
        "IterativeResidual:=", 1E-06,
        "DDMSolverResidual:=", 0.0001,
        "EnhancedLowFreqAccuracy:=", True,
        "SaveRadFieldsOnly:=", False,
        "SaveAnyFields:=", True,
        "IESolverType:=", "Auto",
        "LambdaTargetForIESolver:=", 0.15,
        "UseDefaultLambdaTgtForIESolver:=", True,
        "SkipIERegionSolveDuringAdaptivePasses:=", True
        "RayDensityPerWavelength:=", 4,
```

```
        "MaxNumberOfBounces:="  , 5,
        "InfiniteSphereSetup:=" , "Infinite Sphere1",
        "SkipSBRSolveDuringAdaptivePasses:=", True,
        "PTDUTDSimulationSettings:=", "PTD Correction + UTD Rays",
        "PTDEdgeDensity:="      , 20
                                            ])
```

An HFSS with Hybrid and Arrays setup

```
import ScriptEnv
ScriptEnv.Initialize("Ansoft.ElectronicsDesktop")
oDesktop.RestoreWindow()
oProject = oDesktop.SetActiveProject("Project1")
oDesign = oProject.SetActiveDesign("HFSSDesign1")
oModule = oDesign.GetModule("AnalysisSetup")
oModule.InsertSetup("HfssDriven",
  [
    "NAME:Setup1",
    "SolveType:="           , "Single",
    "Frequency:="           , "5GHz",
    "MaxDeltaE:="           , 0.1,
    "MaximumPasses:="       , 6,
    "MinimumPasses:="       , 1,
    "MinimumConvergedPasses:=", 1,
```

```
    "PercentRefinement:="   , 30,
    "IsEnabled:="           , True,
    [
      "NAME:MeshLink",
      "ImportMesh:="          , False
    ],
    "BasisOrder:="          , 1,
    "DoLambdaRefine:="      , True,
    "DoMaterialLambda:="    , True,
    "SetLambdaTarget:="     , False,
    "Target:="              , 0.3333,
    "UseMaxTetIncrease:="   , False,
    "DrivenSolverType:="    , "Direct Solver",
    "EnhancedLowFreqAccuracy:=", False,
    "SaveRadFieldsOnly:="   , False,
    "SaveAnyFields:="       , True,
    "IESolverType:="        , "Auto",
    "LambdaTargetForIESolver:=", 0.15,
    "UseDefaultLambdaTgtForIESolver:=", True,
    "IE Solver Accuracy:="  , "Balanced",
```

```
    "RayDensityPerWavelength:=", 4,
    "MaxNumberOfBounces:="  , 5,
    "RadiationSetup:="      , "Infinite Sphere1",
    "PTDUTDSimulationSettings:=", "None",
    "EnableSBRSelfCoupling:=", False,
    "UseSBRAdvOptionsGOBlockage:=", False,
    "UseSBRAdvOptionsWedges:=", False,
    "SkipSBRSolveDuringAdaptivePasses:=", False,
    "UseSBREnhancedRadiatedPowerCalculation:=", True
  ])
```

An SBR+ Setup with Fast Frequency Looping

```
oModule.InsertSetup("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
        ])
```

SBR+ Setup with Enhanced Radiated Power Calculation

```
oModule.InsertSetup("HfssDriven",
  [
    "NAME:Setup2",
    "IsEnabled:="           , True,
    [
      "NAME:MeshLink",
      "ImportMesh:="          , False
    ],
    "IsSbrRangeDoppler:="   , False,
    "RayDensityPerWavelength:=", 4,
    "MaxNumberOfBounces:="  , 5,
    "EnableCWRays:="        , False,
    "RadiationSetup:="      , "Infinite Sphere1",
    "PTDUTDSimulationSettings:=", "None",
    "FastFrequencyLooping:=", False,
    "UseSBRAdvOptionsGOBlockage:=", False,
    "UseSBRAdvOptionsWedges:=", False,
    [
      "NAME:Sweeps",
      [
```

```
        "NAME:Sweep",
        "RangeType:="           , "LinearStep",
        "RangeStart:="          , "1GHz",
        "RangeEnd:="            , "10GHz",
        "RangeStep:="           , "1GHz"
      ]
    ],
    "ComputeFarFields:="    , True,
    "UseSBREnhancedRadiatedPowerCalculation:=", True,
    "IsGOBlockageEnabled:=" , False,
    "GOBlockageSurfaceSelfBlock:=", False
  ])
```

Insert 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.InsertSetup("HfssRFDischarge",
  [
    "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:="  , False,
      "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:="          , "Argon"
    "Save Electron Density:=", True,
    "Is Pulsed Signal:="    , True,
    "Duty Cycle:="          , 50
  ])
```

Setup with UseMatrixConv is True and Matrix Convergence Entry Specified as All

```
oModule.InsertSetup("Setup1",
  [
    "NAME:Setup1",
    "SolveType:="           , "Single",
    "Frequency:="           , "10GHz",
    "MaxDeltaS:="           , 0.02,
    "UseMatrixConv:="       , True,
    [
      "NAME:Matrix Convergence",
      "AllEntries:="          , True,
      "MagLimit:="            , "0.01",
      "PhaseLimit:="          , "2deg",
      "MagMinThreshold:="     , 0.01
```

```
    ],
    "MaximumPasses:="       , 12,
    "MinimumPasses:="       , 1,
    "MinimumConvergedPasses:=", 1,
    "PercentRefinement:="   , 30,
    "IsEnabled:="           , True,
    [
      "NAME:MeshLink",
      "ImportMesh:="          , False
    ],
    "BasisOrder:="          , 1,
    "DoLambdaRefine:="      , True,
    "DoMaterialLambda:="    , True,
    "SetLambdaTarget:="     , False,
    "Target:="              , 0.3333,
    "UseMaxTetIncrease:="   , False,
    "PortAccuracy:="        , 2,
    "UseABCOnPort:="        , False,
    "SetPortMinMaxTri:="    , False,
    "DrivenSolverType:="    , "Direct Solver",
```

```
    "EnhancedLowFreqAccuracy:=", False,
    "SaveRadFieldsOnly:="   , False,
    "SaveAnyFields:="       , True,
    "IESolverType:="        , "ACA",
    "LambdaTargetForIESolver:=", 0.15,
    "UseDefaultLambdaTgtForIESolver:=", True,
    "IE Solver Accuracy:="  , "Balanced",
    "InfiniteSphereSetup:=" , ""
  ])
```

Setup with UseMatrixConv as Entries and MeshLink

```
oModule = oDesign.GetModule("AnalysisSetup")
oModule.InsertSetup("Setup1",
  [
    "NAME:Setup1",
    "SolveType:="           , "Single",
    "Frequency:="           , "10GHz",
    "MaxDeltaS:="           , 0.02,
    "UseMatrixConv:="       , True,
    [
      "NAME:Matrix Convergence",
      [
```

```
        "NAME:Entries",
        [
            "NAME:Entry",
            "Port1:="          , "A[1,1]P1",
            "Port2:="          , "A[1,1]P1",
            "MagLimit:="               , "0.011",
            "PhaseLimit:="             , "5deg"
        ],
        [
            "NAME:Entry",
            "Port1:="          , "A[4,4]P2",
            "Port2:="          , "A[1,1]P2",
            "MagLimit:="               , "0.041",
            "PhaseLimit:="             , "5deg"
        ]
      ]
    ],
    "MaximumPasses:="       , 1,
    "MinimumPasses:="       , 1,
    "MinimumConvergedPasses:=", 1,
```

```
    "PercentRefinement:="   , 30,
    "IsEnabled:="           , True,
    [
      "NAME:MeshLink",
      "ImportMesh:="          , True,
      "Project:="             , "This Project*",
      "Product:="             , "",
      "Design:="              , "unit",
      "Soln:="                , "Setup1 : LastAdaptive",
      [
        "NAME:Params",
        "Airbox_dist:="         , "9.9931mm",
        "Scan_Theta:="          , "0deg",
        "Scan_phi:="            , "0deg",
        "VirtualObject_dist:="  , "2.9979mm",
        "coax_inner_rad:="      , "0.125mm",
        "coax_outer_rad:="      , "0.425mm",
        "cut_off:="             , "1.7mm",
        "feedX:="               , "-0.4mm",
        "feedY:="               , "2mm",
        "feed_length:="         , "1mm",
```

```
        "patchX:="              , "9.65mm",
        "rot:="                 , "45deg",
        "subH:="                , "0.8mm",
        "subX:="                , "15mm",
        "subY:="                , "15mm"
      ],
      "ForceSourceToSolve:="  , False,
      "PreservePartnerSoln:=" , False,
      "PathRelativeTo:="      , "TargetProject",
      "ApplyMeshOp:="         , True,
      "AdaptPort:="           , False
    ],
    "BasisOrder:="          , 1,
    "DoLambdaRefine:="      , False,
    "DoMaterialLambda:="    , True,
    "SetLambdaTarget:="     , False,
    "Target:="              , 0.3333,
    "UseMaxTetIncrease:="   , False,
    "PortAccuracy:="        , 2,
    "UseABCOnPort:="        , False,
```

```
    "SetPortMinMaxTri:="    , False,
    "DrivenSolverType:="    , "Domain Decomposition",
    "IterativeResidual:="   , 0.0001,
    "DDMSolverResidual:="   , 0.0001,
    "EnhancedLowFreqAccuracy:=", False,
    "SaveRadFieldsOnly:="   , False,
    "SaveAnyFields:="       , True,
    "IESolverType:="        , "Auto",
    "LambdaTargetForIESolver:=", 0.15,
    "UseDefaultLambdaTgtForIESolver:=", True,
    "IE Solver Accuracy:="  , "Balanced",
    "InfiniteSphereSetup:=" , ""
  ])
```

Example from design with UseMatrixConv True for AllDiagEntries

```
    "UseMatrixConv:="       , True,
    [
      "NAME:Matrix Convergence",
      "AllDiagEntries:="      , True,
      "DiagonalMag:="         , "0.03",
      "DiagonalPhase:="       , "4rad",
      "MagMinThreshold:="     , 0.01,
```

```
      "AllOffDiagEntries:="   , True,
      "OffDiagonalMag:="      , "0.05",
      "OffDiagonalPhase:="    , "6deg",
      "MagMinThreshold:="     , 0.01
    ],
```

Example from Modal design with UseMatrixConv True for Entries

```
    "UseMatrixConv:="       , True,
    [
      "NAME:Matrix Convergence",
      [
        "NAME:Entries",
        [
            "NAME:Entry",
            "Port1:="          , "Port1",
            "Port2:="          , "Port1",
            "MagLimit:="               , "0.02",
            "PhaseLimit:="             , "5deg"
        ],
        [
            "NAME:Entry",
```

```
            "Port1:="          , "Port3",
            "Port2:="          , "Port2",
            "MagLimit:="               , "0.03",
            "PhaseLimit:="             , "5deg"
        ]
      ]
    ],
```

Example from Design with Periodic Ports and UseMaxtrixConv is True

```
    "UseMatrixConv:="       , True,
    [
      "NAME:Matrix Convergence",
      [
        "NAME:Entries",
        [
          "NAME:Entry",
          "Port1:="              , "Incident_Port1",
          "Port2:="              , "Incident_Port1",
          "MagLimit:="           , "0.11",
          "PhaseLimit:="         , "5deg"
        ],
        [
```

```
            "NAME:Entry",
            "Port1:="               , "Incident_Port2",
            "Port2:="               , "Incident_Port2",
            "MagLimit:="            , "0.22",
            "PhaseLimit:="          , "5deg"
          ]
        ]
    ],
```

Example Terminal Design with UseMatrixConv = True and Expression Cache

```
oProject = oDesktop.SetActiveProject("term-2x2_1_parasolid")
oDesign = oProject.SetActiveDesign("unit")
oModule = oDesign.GetModule("AnalysisSetup")
oModule.InsertSetup("Setup1",
  [
    "NAME:Setup1",
    "SolveType:="           , "Single",
    "Frequency:="           , "10GHz",
    "MaxDeltaS:="           , 0.02,
    "UseMatrixConv:="       , True,
    [
```

```
      "NAME:Matrix Convergence",
      [
        "NAME:Entries",
        [
             "NAME:Entry",
             "Port1:="          , "P2",
             "Port2:="          , "P1",
             "MagLimit:="               , "0.05",
             "PhaseLimit:="             , "10deg"
        ]
      ]
    ],
    "MaximumPasses:="       , 15,
    "MinimumPasses:="       , 1,
    "MinimumConvergedPasses:=", 1,
    "PercentRefinement:="   , 30,
    "IsEnabled:="           , True,
    [
      "NAME:MeshLink",
      "ImportMesh:="          , False
    ],
```

```
    "BasisOrder:="          , 1,
    "DoLambdaRefine:="      , True,
    "DoMaterialLambda:="    , True,
    "SetLambdaTarget:="     , False,
    "Target:="              , 0.3333,
    "UseMaxTetIncrease:="   , False,
    "PortAccuracy:="        , 2,
    "UseABCOnPort:="        , False,
    "SetPortMinMaxTri:="    , False,
    "DrivenSolverType:="    , "Iterative Solver",
    "IterativeResidual:="   , 0.0001,
    "DDMSolverResidual:="   , 0.0001,
    "EnhancedLowFreqAccuracy:=", False,
    "SaveRadFieldsOnly:="   , False,
    "SaveAnyFields:="       , True,
    [
      "NAME:ExpressionCache",
      [
        "NAME:CacheItem",
        "Title:="               , "dB_AxialRatioValue_1",
```

```
        "Expression:="          , "dB(AxialRatioValue)",
        "Intrinsics:="          , "Phi=\'0deg\' Theta=\'0deg\'",
        "ReportType:="          , "Far Fields",
        [
          "NAME:ExpressionContext",
          "Context:="             , "Infinite Sphere1"
        ]
      ],
      [
        "NAME:CacheItem",
        "Title:="               , "dB_St_Diff1_Diff1__1",
        "Expression:="          , "dB(St(Diff1,Diff1))",
        "Intrinsics:="          , "",
        "ReportType:="          , "Terminal Solution Data",
        [
          "NAME:ExpressionContext",
          "Diff:="                , "Differential Pairs"
        ]
      ],
      [
        "NAME:CacheItem",
```

```
          "Title:="               , "dB_St_P1_P1__1",
          "Expression:="          , "dB(St(P1,P1))",
          "Intrinsics:="          , "",
          "ReportType:="          , "Terminal Solution Data",
          [
            "NAME:ExpressionContext",
            "Diff:="                , "Terminals"
          ]
]
     ],
     "CacheSaveKind:="       , "Delta",
     "ConstantDelta:="       , "0s",
     "IESolverType:="        , "Auto",
     "LambdaTargetForIESolver:=", 0.15,
     "UseDefaultLambdaTgtForIESolver:=", True,
     "IE Solver Accuracy:="  , "Balanced",
     "InfiniteSphereSetup:=" , ""
])
```

Example HFSS Hybrid and Arrays with Beta Parallel Component Mesh Adapt

```
oModule.InsertSetup("HfssDriven",
```

```
[
  "NAME:Setup2",
  "SolveType:="           , "Single",
  "Frequency:="           , "5GHz",
  "MaxDeltaS:="           , 0.02,
  "UseMatrixConv:="       , False,
  "MaximumPasses:="       , 6,
  "MinimumPasses:="       , 1,
  "MinimumConvergedPasses:=", 1,
  "PercentRefinement:="   , 30,
  "ComponentAdaptOption:=", "Fully Independent",
  "PerformFullDesignSolveAtLastPass:=", True,
  "IsEnabled:="           , True,
  [
    "NAME:MeshLink",
    "ImportMesh:="          , False
  ],
  "BasisOrder:="          , 1,
  "DoLambdaRefine:="      , True,
  "DoMaterialLambda:="    , True,
  "SetLambdaTarget:="     , False,
```

```
    "Target:="              , 0.3333,
    "UseMaxTetIncrease:="   , False,
    "PortAccuracy:="        , 2,
    "UseABCOnPort:="        , False,
    "SetPortMinMaxTri:="    , False,
    "DrivenSolverType:="    , "Direct Solver",
    "EnhancedLowFreqAccuracy:=", False,
    "SaveRadFieldsOnly:="   , False,
    "SaveAnyFields:="       , True,
    "IESolverType:="        , "Auto",
    "LambdaTargetForIESolver:=", 0.15,
    "UseDefaultLambdaTgtForIESolver:=", True,
    "IE Solver Accuracy:="  , "Balanced",
    "InfiniteSphereSetup:=" , "",
    "MaxPass:="             , 10,
    "MinPass:="             , 1,
    "MinConvPass:="         , 1,
    "PerError:="            , 1,
    "PerRefine:="           , 30
  ])
```

VB Syntax	InsertSetup <*SetupType*>, <*Attributes*>
VB Example	An HFSS Driven project solution with Auto Solution: oModule.InsertSetup "HfssDrivenAuto", Array("NAME:Setup1", "IsEnabled:=", true, _ "AutoSolverSetting:=", "Balanced", _ Array("NAME:Sweeps", Array("NAME:Sweep", "RangeType:=", "LinearCount", _ "RangeStart:=", "1GHz", "RangeEnd:=", "10GHz", "RangeCount:=", "451")), _ "SaveRadFieldsOnly:=", false, "SaveAnyFields:=", false, "Type:=", "Interpolating") A Driven solution type with a mesh link. References to dependent solve in old scripts are converted to mesh link form. oModule.InsertSetup "HfssDriven", Array("NAME:Setup1", "Frequency:=", "1GHz", "MaxDeltaE:=", 0.1, "MaximumPasses:=", 6, "MinimumPasses:=", 1, "MinimumConvergedPasses:=", 1, "PercentRefinement:=", 30, "IsEnabled:=", true, Array("NAME:MeshLink", "Project:=", "Tee.aedt",

```
"Design:=", "TeeModel",
"Soln:=", "Setup1 : LastAdaptive",
Array("NAME:Params", "offset:=", "0in"),
"ForceSourceToSolve:=", false,
"PreservePartnerSoln:=", false,
"PathRelativeTo:=", "SourceProduct",
"ApplyMeshOp:=", true),
"BasisOrder:=", 1,
"UseIterativeSolver:=", false,
"DoLambdaRefine:=", false,
"DoMaterialLambda:=", true,
"SetLambdaTarget:=", false,
"Target:=", 0.3333,
"UseMaxTetIncrease:=", false,
"MaxTetIncrease:=", 1000000,
"EnableSolverDomains:=", false,
"ThermalFeedback:=", false,
"UsingConstantDelta:=", 0,
"ConstantDelta:=", "0s",
"NumberSolveSteps:=", 1)
```

```
A Driven solution type with ports:
oModule.InsertSetup "HfssDriven",
Array("NAME:Setup2",
"Frequency:=", "1GHz",
"PortsOnly:=", false,
"MaxDeltaS:=", 0.02,
"UseMatrixConv:=", false,
"MaximumPasses:=", 6,
"MinimumPasses:=", 1,
"MinimumConvergedPasses:=", 1,
"PercentRefinement:=", 30,
"IsEnabled:=", true,
"BasisOrder:=", 1,
"UseIterativeSolver:=", true,
"IterativeResidual:=", 0.0001,
"DoLambdaRefine:=", true,
"DoMaterialLambda:=", false,
"SetLambdaTarget:=", false,
"Target:=", 0.3333,
"UseMaxTetIncrease:=", false,
"MaxTetIncrease:=", 1000000,
```

```
"PortAccuracy:=", 2,
"UseABCOnPort:=", true,
"SetPortMinMaxTri:=", false,
"EnableSolverDomains:=", false,
"ThermalFeedback:=", false,
"UsingConstantDelta:=", 0,
"ConstantDelta:=", "0s",
"NumberSolveSteps:=", 1)
An Eigenmode solution type:
oModule.InsertSetup "HfssEigen",
Array("NAME:Setup2",
"MinimumFrequency:=", "1.77347GHz",
"NumModes:=", 1,
"MaxDeltaFreq:=", 10,
"ConvergeOnRealFreq:=", true,
"MaximumPasses:=", 3,
"MinimumPasses:=", 1,
"MinimumConvergedPasses:=", 1,
"PercentRefinement:=", 30,
"IsEnabled:=", true,
```

```
"BasisOrder:=", 1,
"UseIterativeSolver:=", false,
"DoLambdaRefine:=", true,
"DoMaterialLambda:=", true,
"SetLambdaTarget:=", false,
"Target:=", 0.2,
"UseMaxTetIncrease:=", false,
"MaxTetIncrease:=", 1000000,
"UsingConstantDelta:=", 0,
"ConstantDelta:=", "0s",
"NumberSolveSteps:=", 1)
A Driven solution type with ports and matrix convergence:
oModule.InsertSetup "HfssDriven",
Array("NAME:Setup2",
"Frequency:=", "1GHz",
"PortsOnly:=", false,
"MaxDeltaS:=", 0.02,
"UseMatrixConv:=", true,
Array("NAME:ConvergenceMatrix",
"AllDiagEntries:=", true,
"MagMinThreshold:=", 0.01,
```

```
"Entry:=", Array("Port1:=", "abc", "ModeNum1:=", 1)),
"MaximumPasses:=", 6,
"MinimumPasses:=", 1,
"MinimumConvergedPasses:=", 1,
"PercentRefinement:=", 30,
"IsEnabled:=", true,
"BasisOrder:=", 1,
"UseIterativeSolver:=", false,
"DoLambdaRefine:=", true,
"DoMaterialLambda:=", true,
"SetLambdaTarget:=", false,
"Target:=", 0.3333,
"UseMaxTetIncrease:=", false,
"MaxTetIncrease:=", 1000000,
"PortAccuracy:=", 2,
"UseABCOnPort:=", true,
"SetPortMinMaxTri:=", false,
"EnableSolverDomains:=", false,
"ThermalFeedback:=", false,
"UsingConstantDelta:=", 0,
```

```
"ConstantDelta:=", "0s",
"NumberSolveSteps:=", 1)
A Driven solution type with Multi-Frequencies specified:
oModule.InsertSetup "HfssDriven", Array("NAME:Setup2", "AdaptMultipleFreqs:=", true, _
Array("NAME:MultipleAdaptiveFreqsSetup", "1.1GHz:=", Array( _
0.02), "10GHz:=", Array(0.02), "11GHz:=", Array(0.02), "12.5GHz:=", Array(0.02),
"15GHz:=", Array( _
0.02), "20GHz:=", Array(0.02)), "MaximumPasses:=", 6, "MinimumPasses:=", 1, "Min-
imumConvergedPasses:=",  _
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:=", false, "UseIterativeSolver:=",  _
false, "SaveRadFieldsOnly:=", false, "SaveAnyFields:=", true, "IESolverType:=",  _
"Auto", "LambdaTargetForIESolver:=", 0.15, "UseDefaultLambdaTgtForIESolver:=",  _
true, "RayDensityPerWavelength:=", 4, "MaxNumberOfBounces:=", 5)
An Driven Solution type with Broadband specified:
oModule.InsertSetup "HfssDriven", Array("NAME:Setup1", "AdaptMultipleFreqs:=", true, _
Array("NAME:MultipleAdaptiveFreqsSetup", Array("NAME:Broadband", "Low:=",  _
"2GHz", "High:=", "20GHz")), "MaximumPasses:=", 6, "MinimumPasses:=", 1, "Min-
imumConvergedPasses:=",  _
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:=", false, "UseIterativeSolver:=",  _
false, "SaveRadFieldsOnly:=", false, "SaveAnyFields:=", true, "IESolverType:=",  _
"Auto", "LambdaTargetForIESolver:=", 0.15, "UseDefaultLambdaTgtForIESolver:=",  _
true, "RayDensityPerWavelength:=", 4, "MaxNumberOfBounces:=", 5)
oModule.InsertFrequencySweep "Setup1", Array("NAME:Sweep", "IsEnabled:=", true,
"RangeType:=",  _
"LinearCount", "RangeStart:=", "2GHz", "RangeEnd:=", "20GHz", "RangeCount:=",  _
451, "Type:=", "Interpolating", "SaveFields:=", false, "SaveRadFields:=",  _
false, "InterpTolerance:=", 0.5, "InterpMaxSolns:=", 250, "InterpMinSolns:=",  _
0, "InterpMinSubranges:=", 1, "ExtrapToDC:=", false, "InterpUseS:=", true, "Inter-
pUsePortImped:=",  _
false, "InterpUsePropConst:=", true, "UseDerivativeConvergence:=", false, "Inter-
pDerivTolerance:=",  _
0.2, "UseFullBasis:=", true, "EnforcePassivity:=", true, "PassivityErrorTolerance:=",
_
0.0001)
```

InsertSetup [Parametric]

Inserts a new parametric setup.

<table>
<tr><td>UI Access</td><td>Right-click the Optimetrics folder in the project tree, and then click Add> Parametric on the shortcut menu.</td></tr>
<tr><td>Parameters</td><td>
<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td><SetupName></td><td>String</td><td>Name of the parametric setup.</td></tr>
<tr><td><SimSetups></td><td>Array of Strings</td><td>An array of Twin Builder solution setup names.</td></tr>
<tr><td><SweepDefs></td><td>Array</td><td>Array("NAME:Sweeps",
Array("NAME:SweepDefinition", "Variable:=",
<VarName>, "Data:=", <SweepData>,
"Synchronize:=", <SyncNum>), ...
Array("NAME:SweepDefinition", "Variable:=",
<VarName>, "Data:=", <SweepData>,
"Synchronize:=", <SyncNum>))</td></tr>
<tr><td><SweepData></td><td>String</td><td>"<SweepType>, <StartV>, <StopV>, <StepV>"</td></tr>
<tr><td><SweepType></td><td>String</td><td>The type of sweep data.</td></tr>
<tr><td><SyncNum></td><td>Integer</td><td><code>SweepDatas</code> with the same value are synchronized.</td></tr>
<tr><td><SweepOps></td><td></td><td>Array("NAME:Sweep Operations", "<OpType>:=,
Array(<VarValue>, …, <VarValue>), ...</td></tr>
</table>
</td></tr>
</table>

			<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"]]]]]])

VB Syntax	InsertSetup "OptiParametric", <ParametricParams>
VB Example	oModule.InsertSetup "OptiParametric", Array("NAME:ParametricSetup1", _ "SaveFields:=", true, _ Array("NAME:StartingPoint"), _ "Sim. Setups:=", Array("Setup1"),_ Array("NAME:Sweeps", _ Array("NAME:SweepDefinition", _ "Variable:=", "$width", _ "Data:=", "LIN 12mm 17mm 2.5mm", _

```
"OffsetF1:=", false, _
"Synchronize:=", 0),
Array("NAME:SweepDefinition", _
"Variable:=", "$length", _
"Data:=", "LIN 8mm 12mm 2mm", _
"OffsetF1:=", false, _
"Synchronize:=", 0)),
Array("NAME:Sweep Operations"), _
Array("NAME:Goals", _
Array("NAME:Goal", _
"Solution:=", "Setup1 : LastAdaptive", _
"Calculation:=", "returnloss", _
"Context:=", "",_
 Array("NAME:Ranges", _
"Range:=", Array("Var:=", "Freq", "Type:=","s", _
"Start:=", "8GHz", "Stop:=", "8GHz"))), _
Array("NAME:Goal", _
"Solution:=", "Setup1 : LastAdaptive",_
"Calculation:=", "reflect", _
```

	"Context:=", "", _ Array("NAME:Ranges", _ "Range:=", Array("Var:=", "Freq", "Type:=","s", _ "Start:=", "8GHz", "Stop:=", "8GHz")))))

InsertSetup [Optimization]

Use: Inserts a new optimization setup.

UI Access	Right-click the **Optimetrics** folder in the project tree, and then click **Add > Optimization** on the shortcut menu.
Parameters	See table below

Name	Type	Description
<OptimizationParams>	Array	Array("NAME:<SetupName>", "SaveFields:=", <SaveField>, <StartingPoint>, "Optimizer:=", <Optimizer>, "MaxIterations:=", <MaxIter>, "PriorPSetup:=", <PriorSetup>, "PreSolvePSetup:=", <Preceed>, <OptimizationVars>, <Constraint>, Array("NAME:Goals", Array("NAME:Goal", <OptiGoalSpec>, <OptimizationGoalSpec>), ... Array("NAME:Goal", <OptiGoalSpec>, <OptimizationGoalSpec>)), "Acceptable_Cost:=", <AcceptableCost>, "Noise:=", <Noise>, "UpdateDesignWhenDone:=", <UpdateDesign>
<OptimizationVars>	Array	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:=", "<=", ["NAME:GoalValue", "GoalValueType:=", "Independent", "Format:=", "Real/Imag", "bG:=", ["v:=", "[0.0001]"]], "Weight:=", "[1]"]], "Acceptable_Cost:=", 0.0002, "Noise:=", 0.0001, "UpdateDesign:=", true, "UpdateIteration:=", 5, "KeepReportAxis:=", true, "UpdateDesignWhenDone:=", true])

VB Syntax	InsertSetup "OptiOptimization", <OptimizationParams>
VB Example	oModule.InsertSetup "OptiOptimization",_ Array("NAME:OptimizationSetup1", _

```
"SaveFields:=", false, _
Array("NAME:StartingPoint", "$length:=", "8mm",_
"$width:=", "14.5mm"), _
"Optimizer:=", "Quasi Newton", _
"MaxIterations:=", 100, _
"PriorPSetup:=", "ParametricSetup1", _
"PreSolvePSetup:=", true, _
Array("NAME:Variables", _
"$length:=", Array("i:=", true, "Min:=", "6mm",_
"Max:=", "18mm", _
"MinStep:=", "0.001mm", "MaxStep:=", _
"1.2mm"), _
"$width:=", Array("i:=", true, "Min:=", _
"6.5mm", "Max:=", "19.5mm", _
"MinStep:=", "0.001mm", "MaxStep:=", _
"1.3mm")), _
Array("NAME:LCS"), _
Array("NAME:Goals", _
Array("NAME:Goal", _
"Solution:=", "Setup1 : LastAdaptive", _
```

```
"Calculation:=", "reflect", _
"Context:=", "", _
Array("NAME:Ranges", _
"Range:=", Array("Var:=", "Freq", _
"Type:=", "s", _
"Start:=", "8GHz", "Stop:=", "8GHz")), _
"Condition:=", "&lt;=", _
Array("NAME:GoalValue", _
"GoalValueType:=", "Independent", _
"Format:=", "Real/Imag", _
"bG:=", Array("v:=", "[0.0001]")), _
"Weight:=", "[1]")),
"Acceptable_Cost:=", 0.0002, _
"Noise:=", 0.0001, _
"UpdateDesign:=", true, _
"UpdateIteration:=", 5, _
"KeepReportAxis:=", true, _
"UpdateDesignWhenDone:=", true)
```

InsertSetup [Sensitivity]

Inserts a new sensitivity setup.

<table>
<tr><td>UI Access</td><td>Right-click Optimetrics in the project tree, and then click Add>Sensitivity on the shortcut menu.</td></tr>
<tr><td>Parameters</td><td>
<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td><SensitivityParams></td><td>Array</td><td>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>)</td></tr>
<tr><td><SensitivityVars></td><td>Array</td><td>Array("NAME:Variables",
"VarName:=", Array("i:=", <IncludeVar>,
"Min:=", <MinV>, "Max:=", <MaxV>,
"IDisp:=", <InitialDisp>),...
"VarName:=", Array("i:=", <IncludeVar>,
"Min:=", <MinV>, "Max:=", <MaxV>,
"IDisp:=", <InitialDisp>))</td></tr>
<tr><td><InitialDisp></td><td>VarValue</td><td>Index of the Primary goal. Index starts from zero.</td></tr>
<tr><td><PrimaryError></td><td>Double</td><td>Error associated with the Primary goal.</td></tr>
</table>
</td></tr>
</table>

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])
```

VB Syntax	InsertSetup "OptiSensitivity", <SensitivityParams>
VB Example	oModule.InsertSetup "OptiSensitivity", _ Array("NAME:SensitivitySetup1",_ "SaveFields:=", true,_ Array("NAME:StartingPoint"), _

```
"MaxIterations:=", 20,_
"PriorPSetup:=", "",_
"PreSolvePSetup:=", true, _
Array("NAME:Variables"), _
Array("NAME:LCS"),_
Array("NAME:Goals",_
Array("NAME:Goal", _
"Solution:=", "Setup1 : LastAdaptive",_
"Calculation:=", "returnloss",_
"Context:=", "",_
Array("NAME:Ranges",_
"Range:=", Array("Var:=", "Freq", "_
Type:=", "s",_
"Start:=", "8GHz", "Stop:=", "8GHz"))),_
Array("NAME:Goal",_
"Solution:=", "Setup1 : LastAdaptive",_
"Calculation:=", "reflect",_
"Context:=", "",_
Array("NAME:Ranges",_
```

	"Range:=", Array("Var:=","Freq",_ "Type:=", "s",_ "Start:=", "8GHz", "Stop:=", "8GHz")))),_ "Primary Goal:=", 1,_ "PrimaryError:=", 0.001)

InsertSetup [Statistical]

Inserts a new statistical setup.

UI Access	Right-click **Optimetrics** in the project tree, and then click **Add>Statistical** on the shortcut menu.
Parameters	

Name	Type	Description
<StatisticalParams>	Array	Array("NAME:<SetupName>", "SaveFields:=", <SaveField>, <StartingPoint>, "MaxIterations:=", <MaxIter>, "PriorPSetup:=", <PriorSetup>, "PreSolvePSetup:=", <Preceed>, <StatisticalVars>, Array("NAME:Goals", Array("NAME:Goal", <OptiGoalSpec>), …, Array("NAME:Goal", <OptiGoalSpec>)))
<StatisticalVars>	Array	Array("NAME:Variables", "VarName:=", Array("i:=", <IncludeVar>, "Dist:=", <DistType>, "Tol:=", <Tolerance>, "StdD:=", <StdD>, "Min:=", <MinCutoff>, "Max:=", <MaxCutoff>, ...

			"VarName:=", Array("i:=", <IncludeVar>, "Dist:=", <DistType>, "Tol:=", <Tolerance>, "StdD:=", <StdD>, "Min:=", <MinCutoff>, "Max:=", <MaxCutoff>))
	<DistType>	String	Distrbution can be "Gaussian" or "Uniform".
	<Tolerance>	VarValue	The tolerance for the variable when distribution is Uniform
	<StdD>	VarValue	The standard deviation for the variable when distribution is Gaussian.
	<MinCutoff>	Double	The minimum cut-off for the variable when distribution is Gaussian.
	<MaxCutoff>	Double	The maximum cut-off for the variable when distribution is Gaussian.
Return Value	None		

Python Syntax	InsertSetup ("OptiStatistical", <StatisticalParams>)
Python Example	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"]]]])

VB Syntax	InsertSetup "OptiStatistical", <StatisticalParams>
VB Example	oModule.InsertSetup "OptiStatistical", _

```
Array("NAME:StatisticalSetup1", _
"SaveFields:=", true, _
Array("NAME:StartingPoint"),_
"MaxIterations:=", 50,_
"PriorPSetup:=", "", _
Array("NAME:Variables"), _
Array("NAME:Goals", _
Array("NAME:Goal", _
"Solution:=", "Setup1 : LastAdaptive", _
"Calculation:=", "returnloss", _
"Context:=", "", _
Array("NAME:Ranges", _
"Range:=", Array("Var:=", "Freq", _
"Type:=", "s",_
"Start:=", "8GHz", "Stop:=", "8GHz"))),_
Array("NAME:Goal",_
"Solution:=", "Setup1 : LastAdaptive",_
"Calculation:=", "reflect",_
"Context:=", "", _
```

	Array("NAME:Ranges",_ "Range:=", Array("Var:=", "Freq", "Type:=",_ "s", "Start:=", "8GHz", "Stop:=", "8GHz"))))

Example:

```
Dim oAnsoftApp

Dim oDesktop

Dim oProject

Dim oDesign

Dim oEditor

Dim oModule

Set oAnsoftApp = CreateObject("Ansoft.ElectronicsDesktop")

Set oDesktop = oAnsoftApp.GetAppDesktop()

oDesktop.RestoreWindow

Set oProject = oDesktop.SetActiveProject("OptimTee")

Set oDesign = oProject.SetActiveDesign("TeeModel")

oDesign.ChangeProperty Array("NAME:AllTabs",
Array("NAME:LocalVariableTab",
Array("NAME:PropServers", "LocalVariables"),
Array("NAME:ChangedProps",
Array("NAME:offset",
Array("NAME:Statistical", "Included:=", true)))))
```

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

oModule.InsertSetup "OptiStatistical", Array("NAME:StatisticalSetup1",
Array("NAME:ProdOptiSetupData",
"SaveFields:=", false, "CopyMesh:=", false),
Array("NAME:StartingPoint", "offset:=", "0in"),
"MaxIterations:=", 50, "PriorPSetup:=", "",

Array("NAME:Variables",

"offset:=", Array("i:=", true,
"int:=", false,
"Dist:=", "Gaussian",
"Tol:=", "10%",
"StdD:=", ".5in",
"Min:=", "-3",
"Max:=", "3",
"Shape:=", "1",
"Scale:=", "0in",
"Location:=", "0in",
"Dataset:=", "",
"LatinHypercube:=", "true",
"VarMin:=", "-1in",
"VarMax:=", "1in",
"Prob:=", "0.01",
"Mean:=", "0in")),

Array("NAME:Goals", Array("NAME:Goal",
"ReportType:=", "Modal Solution Data",
"Solution:=", "Setup1 : LastAdaptive", Array("NAME:SimValueContext"),

"Calculation:=", "Power11",
"Name:=", "Power11",
Array("NAME:Ranges",
```

```
"Range:=", Array("Var:=", "Freq", "Type:=", "d",
"DiscreteValues:=", "10GHz")))))
```

For Q3D Extractor and Circuit the command details are as follows:

Inserts a new statistical setup.

Command: Right-click **Optimetrics** in the project tree, and then click **Add>Statistical** on the shortcut menu.

Syntax: InsertSetup "OptiStatistical", <StatisticalParams>

Return Value: None

Parameters: <StatisticalParams>

```
  Array("NAME:<SetupName>", "SaveFields:=",
  <SaveField>, <StartingPoint>, "MaxIterations:=",
  <MaxIter>, "PriorPSetup:=", <PriorSetup>,
  "PreSolvePSetup:=", <Preceed>, <StatisticalVars>,
  Array("NAME:Goals", Array("NAME:Goal",
  <OptiGoalSpec>), …, Array("NAME:Goal",
  <OptiGoalSpec>))),
<StatisticalVars>
  Array("NAME:Variables",
    "VarName:=", Array("i:=", <IncludeVar>, "Dist:=",
      <DistType>, "Tol:=", <Tolerance>,
      "StdD:=", <StdD>, "Min:=", <MinCutoff>, "Max:=",
      <MaxCutoff>, ...
```

```
  "VarName:=", Array("i:=", <IncludeVar>, "Dist:=",
    <DistType>, "Tol:=", <Tolerance>, "StdD:=",
    <StdD>, "Min:=", <MinCutoff>, "Max:=",
    <MaxCutoff>))
```

Parameters:

<DistType>

Type : <string>

Distrbution can be "Gaussian" or "Uniform".

<Tolerance>

Type: <VarValue>

The tolerance for the variable when distribution is Uniform.

<StdD>

Type: <VarValue>

The standard deviation for the variable when distribution is Gaussian.

<MinCutoff>

Type: <double>

The minimum cut-off for the variable when distribution is Gaussian.

<MaxCutoff>

Type: <double>

The maximum cut-off for the variable when distribution is Gaussian.

```
Example: oModule.InsertSetup "OptiStatistical", _
  Array("NAME:StatisticalSetup1", _
    "SaveFields:=", true, _
    Array("NAME:StartingPoint"),_
    "MaxIterations:=", 50,_
    "PriorPSetup:=", "", _
    Array("NAME:Variables"), _
    Array("NAME:Goals", _
      Array("NAME:Goal", _
"Solution:=", "Setup1 : LastAdaptive", _
"Calculation:=", "returnloss", _
"Context:=", "", _
Array("NAME:Ranges", _
"Range:=", Array("Var:=", "Freq", _
"Type:=", "s",_
"Start:=", "8GHz", "Stop:=", "8GHz"))),_
      Array("NAME:Goal",_
"Solution:=", "Setup1 : LastAdaptive",_
"Calculation:=", "reflect",_
"Context:=", "", _
Array("NAME:Ranges",_
```

"Range:=", Array("Var:=", "Freq", "Type:=", _

"s", "Start:=", "8GHz", "Stop:=", "8GHz"))))

PasteSetup [Optimetrics]

Pastes the specified Optimetrics setup.

UI Access	NA
Parameters	<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>
Return Value	None

Python Syntax	PasteSetup (<SetupName>)
Python Example	`oModule.PasteSetup ("OptimizationSetup1")`

VB Syntax	PasteSetup <SetupName>
VB 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")`

VB Syntax	RenameSetup <OldName> <NewName>
VB Example	`oModule.RenameSetup "OptimizationSetup1" "MyOptimization"`

SetPropValue [Optimetrics]

Sets the property value for the active Optimetrics setup.

UI Access	Set Property value on Optimetrics objects
Parameters	Name | Type | Description Property path | text string | Setup property path. See discussion of Property Path new Value | Text String, Number, or Boolean | New value data type is depending on the property type,
Return Value	True if the property is found and the new value is valid. Otherwise return False.

Python Syntax	SetPropValue(propPath, newValue)
Python Example	`oOptModule.SetPropValue("ParametricSetup1\Enabled", False) //disable ParametricSetup1` `oOptModule.SetPropValue("OptimizationSetup1/Optimizer","Quasi Newton")`

VB Syntax	SetPropValue(propPath, newValue)
VB Example	`SetPropValue("ParametricSetup1\Enabled", False)`

SolveAllSetup

Solves all Optimetrics setups

UI Access	Right-click on **Optimetrics** in Project Manager and select **Analyze>All** from context menu
Parameters	Name: isBlocking; Type: Boolean; Description: An optional arg that defaults to `true`. If it's `false`, the command immediately returns while the analysis or analyses run in the background. The status of the analyses can be checked with the `AreThereSimulationsRunning` command.
Return Value	None

Python Syntax	SolveAllSetup()

Python Example	`oModule.SolveAllSetup()`

VB Syntax	SolveAllSetup
VB Example	`oModule.SolveAllSetup`

SolveSetup [Optimetrics]

Solves the specified Optimetrics setup.

UI Access	Right-click the setup in the project tree, and then click **Analyze** on the shortcut menu.
Parameters	<table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><*SetupName*></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 `true`. If it's `false`, the command immediately returns while the analysis or analyses run in the background. The status of the analyses can be checked with the `AreThereSimulationsRunning` command.</td></tr></table>
Return Value	None

Python Syntax	SolveSetup (<*SetupName*>)
Python Example	`oModule.SolveSetup ("OptimizationSetup1")`

VB Syntax	SolveSetup <*SetupName*>

VB Example	`oModule.SolveSetup "OptimizationSetup1"`

General Commands Recognized by the Optimetrics Module

Following are general script commands recognized by the **Optimetrics** module:

CopySetup

DeleteSetups [Optimetrics]

DistrbutedAnalyzeSetup

EditSetup

EnableSetup

ExportDXConfigFile

ExportOptimetricsProfile

ExportOptimetricsResult

ExportParametricResults

ExportRespSurfaceMinMaxTable

ExportRespSurfaceRefinePoints

ExportRespSurfaceResponsePoints

ExportRespSurfaceVerificationPoints

ExportDOEResponseCurve

ExportDOEResponseCurveSlices

ExportDOEResponseSurface

ExportDOELocalSensitivity

CopySetup

Copy the specified Optimetrics setup.

UI Access	NA
Parameters	Name | Type | Description <*SetupName*> | String | Name of the setup.
Return Value	None.

Python Syntax	CopySetup (<*SetupName*>)
Python Example	`oModule.CopySetup ("OptimizationSetup1")`

VB Syntax	CopySetup <*SetupName*>
VB Example	`oModule.CopySetup "OptimizationSetup1"`

DeleteSetups [Optimetrics]

Deletes the specified Optimetrics setups.

UI Access	Right-click the setup in the project tree, and then click **Delete** on the shortcut menu
Parameters	Name | Type | Description <NameArray> | Array of Strings | An Array of Setup Names
Return Value	None

Python Syntax	DeleteSetups (<NameArray>)
Python Example	`oModule.DeleteSetups (["OptimizationSetup1"])`

VB Syntax	DeleteSetups <NameArray>
VB Example	`oModule.DeleteSetups Array("OptimizationSetup1")`

DistributedAnalyzeSetup

Distributes all variable value instances within a parametric sweep to different machines already specified from within the user interface

UI Access	Right-click the parametric setup name in the project tree and select Distribute Analysis.
Parameters	<table><tr><th>Name</th><th>Type</th><th>Description</th></tr><tr><td><ParametricSetupName></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 `true`. If it's `false`, the command immediately returns while the analysis or analyses run in the background. The status of the analyses can be checked with the `AreThereSimulationsRunning` command.</td></tr></table>
Return Value	None

Python Syntax	DistributedAnalyzeSetup (<ParametricSetupName>)
Python Example	`oModule.DistributedAnalyzeSetup("ParametricSetup1")`

VB Syntax	DistributedAnalyzeSetup <ParametricSetupName>
VB 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.
Parameters	<table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><*SetupName*></td><td>String</td><td>Name of the solve setup being edited.</td></tr><tr><td><*Attributes*></td><td>Array</td><td>Structured array. `Array("NAME:<NewSetupName>", <NamedParameters>)` `See the InsertSetup command for details and examples.`</td></tr></table>
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("EddyCurrent",
[
  "NAME:EddyCurrent",
  "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"
  ])
```

VB Syntax	EditSetup <*SetupName*>, <*Attributes*>

VB Example

```
oModule.EditSetup "Setup1",
Array("NAME:NewSetup",
"AdaptiveFreq:=", "1GHz",
"EnableDistribProbTypeOption:=", false,
"SaveFields:=", "true",
"Enabled:=", true,
Array("NAME:Cap",
        "MaxPass:=", 10,
        "MinPass:=", 1,
        "MinConvPass:=", 2,
        "PerError:=", 1,
        "PerRefine:=", 30,
        "AutoIncreaseSolutionOrder:=", false,
```

```
        "SolutionOrder:=", "Normal"),
Array("NAME:DC",
        "Residual:=", 1E-005,
        "SolveResOnly:=", false,
        Array("NAME:Cond",
                "MaxPass:=", 10,
                "MinPass:=", 1,
                "MinConvPass:=", 1,
                "PerError:=", 1,
                "PerRefine:=", 30),
        Array("NAME:Mult",
                "MaxPass:=", 1,
                "MinPass:=", 1,
                "MinConvPass:=", 1,
                "PerError:=", 1,
                "PerRefine:=", 30)),
Array("NAME:AC",
        "MaxPass:=", 10,
        "MinPass:=", 1,
        "MinConvPass:=", 2,
```

	"PerError:=", 1, "PerRefine:=", 30))

EnableSetup

Enables and disables a defined optimetrics analysis setup.

UI Access	Right-click on a setup in the project tree, select **Enable Setup** or **Disable Setup**
Parameters	<table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><*SetupName*></td><td>String</td><td>Name of specified setup.</td></tr><tr><td><*Enable*></td><td>Boolean</td><td>Determines whether enable or disable a setup. • **True** - enable setup. • **False** - disable setup.</td></tr></table>
Return Value	None.

Python Syntax	EnableSetup(<*SetupName*>, <*Enable*>)
Python Example	`oModule.EnableSetup("OptimizationSetup1", True)`

VB Syntax	EnableSetup <*SetupName*>, <*Enable*>
VB Example	`oModule.EnableSetup "OptimizationSetup1", true`

ExportDXConfigFile

Create an xml file with the setup information for Design Xplorer

UI Access	Right click on the Design Xplorer setup in the project tree and choose **Export External Connector Addin Configuration…**
Parameters	<table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><SetupName></td><td>String</td><td>Must be one of existing DesignExplorer setup names</td></tr><tr><td><FileName></td><td>String</td><td>Must be a valid file path and name</td></tr></table>
Return Value	None

Python Syntax	ExportDXConfigFile (<SetupName>, <FileName>)
Python Example	`oModule.ExportDXConfigFile ("DesignXplorerSetup1",` `"c:/exportdir/DXSetup1.xml")`

VB Syntax	ExportDXConfigFile <SetupName>, <FileName>
VB Example	`oModule.ExportDXConfigFile ("DesignXplorerSetup1",` `"c:/exportdir/DXSetup1.xml")`

ExportOptimetricsProfile

Export Optimetrics profile data

UI Access	Right click on the Optimetrics setup in the project tree and choose **View Analysis Result…** On the **Post Analysis Display** dialog box, click the **Profile** tab and click on the **Export** button.

Parameters	<table><tr><th>Name</th><th>Type</th><th>Description</th></tr><tr><td><SetupName></td><td>String</td><td>Must be one of the existing Parametric, Optimization, Sensitivity, Statistical or DesignXplorer setup names</td></tr><tr><td><FileName></td><td>String</td><td>Must be a valid file path and name with extension of csv, tab, dat, or txt</td></tr><tr><td>[profileNum]</td><td>String</td><td>Must be a numeric string. Optional: defaulted to last profile number. It should be a zero indexed profile number.</td></tr></table>
Return Value	None

Python Syntax	ExportOptimetricsProfile (<SetupName>, <FileName>, [profileNum]
Python Example	`oModule.ExportOptimetricsProfile ("StatisticalSetup1", "c:/exportdir/test.csv")`

VB Syntax	ExportOptimetricsProfile <SetupName>, <FileName>, [profileNum]
VB Example	`oModule.ExportOptimetricsProfile` `"StatisticalSetup1", "c:/exportdir/test.csv"`

ExportOptimetricsResult

Export an existing Optimization, Sensitivity, Statistical or DesignXplorer result. (Does not export Parametric results.)

UI Access	Right click on the desired Optimetrics setup in the project tree and choose **View Analysis Result…** On the **Post Analysis Display** dialog box, click the **Result** tab, then select **Table** view, and click on the **Export** button
Parameters	<table><tr><th>Name</th><th>Type</th><th>Description</th></tr><tr><td><SetupName></td><td>String</td><td>Must be one of the existing Optimization, Sensitivity, Statistical, or</td></tr></table>

			DesignXplorer setup names
	<FileName>	String	Must be a valid file path and name with extension of csv, tab, dat, or txt..
	[useFullOutputName]	Boolean	Optional: defaulted to false. If set to true values will be printed with units. This parameter is ignored for Optimization and Statistical results.
Return Value	None		

Python Syntax	ExportOptimetricsResult (<SetupName>, <FileName>, [useFullOutputName])
Python Example	`oModule.ExportOptimetricsResult ( "StatisticalSetup1", "c:/exportdir/test.csv",false)`

VB Syntax	ExportOptimetricsResult <SetupName>, <FileName>, [useFullOutputName]
VB Example	`oModule.ExportOptimetricsResult "StatisticalSetup1", "c:/exportdir/test.csv", false`

ExportParametricResults

Export existing Parametric results.

UI Access	Right click on the desired Parametric setup in the project tree and choose **View Analysis Result...** On the **Post Analysis Display** dialog box, click the **Result** tab, then select **Table** view, and click on the **Export** button.		
Parameters	Name	Type	Description
	<SetupName>	String	Must be one of the existing Parametric setup names
	<FileName>	String	Must be a valid file path and name with extension of csv, tab, dat or txt

	<bOutputUnits>	Boolean	If set to true, values will be printed with units
Return Value	None		

Python Syntax	ExportParametricResults (<SetupName>, <FileName>, <bOutputUnits>)
Python Example	`oModule.ExportParametricResults (` `"ParametricSetup1", "c:/exportdir/test.csv", False)`

VB Syntax	ExportParametricResults <SetupName>, <FileName>, <bOutputUnits>
VB Example	`oModule.ExportParametricResults` `"ParametricSetup1", "c:/exportdir/test.csv",false`

ExportParametricSetupTable

Exports the parametric setup table as a CSV file.

UI Access	Double-click parametric setup. Select **Table** tab. Click **Export**.		
Parameters	Name	Type	Description
	<SetupName>	String	Name of the setup.
	<filePath>	String	Full path for file export.
Return Value	None		

Python	ExportParametricSetupTable (<SetupName>, <filePath>)

Syntax	
Python Example	`oModule.ExportParametricSetupTable('ParametricSetup1', 'E:/Files/ParametricSetup1_Table.csv')`

VB Syntax	ExportParametricSetupTable <SetupName>, <filePath>
VB Example	`obj.ExportParametricSetupTable "ParametricSetup1", "E:/Files/ParametricSetup1_Table.csv"`

ExportRespSurfaceMinMaxTable

Exports min-max table from a response surface to a file

UI Access	Click on **Export...** From the **Response Surface** tab of the Design of Experiments **Post Analysis Display** dialog.
Parameters	Name / Type / Description <*DOEName*> / String / Name of the Design of Experiments (DOE) setup. <*FileName*> / String / Output file name with path.
Return Value	None.

Python Syntax	ExportRespSurfaceMinMaxTable(<*DOEName*>, <*FileName*>)
Python Example	`oModule.ExportRespSurfaceMinMaxTable("DesignOfExperimentsSetup1",`

	"C:/temp/DesignOfExperimentsSetup1_Min-Max_Search.csv")

VB Syntax	ExportRespSurfaceMinMaxTable <*DOEName*>, <*FileName*>
VB Example	oModule.ExportRespSurfaceMinMaxTable _ "DesignOfExperimentsSetup1", _ "C:/temp/DesignOfExperimentsSetup1_Min-Max_Search.csv"

ExportRespSurfaceRefinePoints

Exports refinement points table to a file

UI Access	From the **Response Surface** tab of the Design of Experiments **Post Analysis Display** dialog box, select **Refinement Points** option under **View**, Click on **Export...**.
Parameters	Name / Type / Description <*DOEName*> / String / Name of the Design of Experiments (DOE) setup. <*FileName*> / String / Output file name with path.
Return Value	None.

Python Syntax	ExportRespSurfaceRefinePoints(<*DOEName*>, <*FileName*>)
Python Example	oModule.ExportRespSurfaceRefinePoints("DesignOfExperimentsSetup1", "C:/temp/DesignOfExperimentsSetup1_Refine_Points.csv")

VB Syntax	ExportRespSurfaceRefinePoints <*DOEName*>, <*FileName*>

VB Example	oModule.ExportRespSurfaceRefinePoints_ "DesignOfExperimentsSetup1", _ "C:/temp/DesignOfExperimentsSetup1_Refine_Points.csv"

ExportRespSurfaceResponsePoints

Exports response points table to a file

UI Access	From the **Response Surface** tab of the Design of Experiments **Post Analysis Display** dialog box, select **Response Points** option under **View**, Click on **Export...**.
Parameters	Name / Type / Description <*DOEName*> / String / Name of the Design of Experiments (DOE) setup. <*FileName*> / String / Output file name with path.
Return Value	None.

Python Syntax	ExportRespSurfaceResponsePoints (<*DOEName*>, <*FileName*>)
Python Example	oModule.ExportRespSurfaceResponsePoints("DesignOfExperimentsSetup1", "C:/temp/DesignOfExperimentsSetup1_Response_Points.csv")

VB Syntax	ExportRespSurfaceResponsePoints <*DOEName*>, <*FileName*>
VB 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")

VB Syntax	ExportRespSurfaceVerificationPoints <*DOEName*>, <*FileName*>
VB 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")`

VB Syntax	*DOEName* (*<FileName>*)
VB Example	`oModule.ExportDOEResponseCurve "DesignOfExperimentsSetup1", Array( _ "$CPU_Power=25W", "$Fan_X=2m_per_sec", "$Memory_Power=5W"), "$CPU_Power", 50_ "CPU_Monitor1.Temperature", _C:/Users/AEDT/R24.1/D/DesignOfExperimentsSetup1_responsecurve.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** dia-

	log.
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")`

VB Syntax	*DOEName* (*<FileName>*)
VB Example	`oModule.ExportDOEResponseCurveSlices "DesignOfExperimentsSetup1", Array( "$CPU_Power=25W", "$Fan_X=2m_per_sec", "$Memory_Power=5W"), "$CPU_Power", 50, _"$Fan_X", 50, "CPU_Monitor1.Temperature", _C:/Users/AEDT/R24.1/D/DesignOfExperimentsSetup1.csv"`

ExportDOEResponseSurface

Exports 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.ExportDOEResponseSurface("DesignOfExperimentsSetup1", "C:/Users/AEDT/R24.1/D/DesignOfExperimentsSetup1_Response_Points.csv")`

VB Syntax	*DOEName* (*<FileName>*)
VB Example	`oModule.ExportDOEResponseCurveSurface "DesignOfExperimentsSetup1", Array( "$CPU_Power=25W", "$Fan_X=2m_per_sec", "$Memory_Power=5W"), "$CPU_Power", 50, _"$Fan_X", 50, "CPU_Monitor1.Temperature", _C:/Users/AEDT/R24.1/D/DesignOfExperimentsSetup1.csv"`

ExportDOELocalSensitivity

Exports local sensitivity 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.ExportDOELocalSensitivity("DesignOfExperimentsSetup1", "C:/Users/AEDT/R24.1/D/DesignOfExperimentsSetup1_Response_Points.csv")`

VB Syntax	*DOEName* (*<FileName>*)
VB Example	`oModule.ExportDOELocalSensitivity "DesignOfExperimentsSetup1", Array( _ "$CPU_Power=25W", "$Fan_X=2m_per_sec", "$Memory_Power=5W"), Array("$CPU_Power", _"$Fan_X", "$Memory_Power"), "CPU_Monitor1.Temperature", true, _"C:/Users/AEDT/R24.1/D/DesignOfExperimentsSetup1_localsensitivity.csv"`

ExportDOELocalSensitivityCurve

Exports local sensitivity curves 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.ExportDOELocalSensitivityCurve("DesignOfExperimentsSetup1", "C:/Users/AEDT/R24.1/D/DesignOfExperimentsSetup1_Response_Points.csv")`

VB Syntax	*DOEName* (<*FileName*>)
VB Example	`oModule.ExportDOELocalSensitivityCurve "DesignOfExperimentsSetup1", Array(_ "$CPU_Power=25W", "$Fan_X=2m_per_sec", "$Memory_Power=5W"), Array("$CPU_Power", _"$Fan_X", "$Memory_Power"), "Input Parameters", 50, "CPU_Monitor1.Temperature", _"C:/Users/AEDT/R24.1/D/DesignOfExperimentsSetup1_localsensitivitycurves.csv"`

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

UI Access	NA
Parameters	Name: Setup Name; Type: text string; Description: A optimetrics setup name, names returned by the GetChildNames().
Return Value	A script object for the setup See discussion of Optimetrics Setup Objects in Object Script Property Function Summary.

Python Syntax	GetChildObject()
Python Example	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 ID and design name of the active design, in that order separated by a semicolon. If the active design is a sub-circuit, it also returns the parent ID and parent design name, listed before the active design information.

To get the *only* the design's name without the parent design name or design IDs, see oDesign.GetDesignName().

UI Access	N/A
Parameters	None.
Return Value	String indicating the name and ID of the active design and parent design (if the active design is a sub-circuit).

Python Syntax	GetName()
Python Example	oDesign.GetName( ) for a subcircuit *EMDesign2* in a parent design *EMDesign1* design_name = oDesign.GetName()

	1;EMDesign1/2;EMDesign2 In the returned string: `1` is the parent design ID; `EMDesign1` is the parent design name. `2` is the active design ID; `EMDesign2` is the active design name.

VB Syntax	GetName
VB Example	`design_name = oDesign.GetName`

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()`

VB Syntax	GetObjPath

VB Example	`oEditor.GetObjPath`

GetOptimetricResult

Returns an Optimetric calculation. The specific calculation is determined by the setup.

UI Access	N/A
Parameters	<table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><*SetupName*></td><td>String</td><td>Optimetrics setup name.</td></tr><tr><td><*vars*></td><td>Array</td><td>Array containing string variable names. Use the **Sweep Definitions** tab in the UI or the <SweepDefs> parameter in the InsertSetup script to determine appropriate inputs.</td></tr><tr><td><*values*></td><td>Array</td><td>*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.</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'])`

VB Syntax	GetOptimetricResult <*SetupName*>, <*vars*>, <*values*>
VB Example	`dim vars(1)`

	vars(0) = "AR" vars(1) = "Re" dim values(1) values(0) = "4.64" values(1) = "6e+04" oModule.GetOptimetricResult "ParametricSetup1", vars, values

GetPropEvaluatedValue

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

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

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

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

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)`

GetPropSIValue

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

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

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

Return Value	Property value as a double floating value, or NAN if the property value cannot be converted to double floating point.

Python Syntax	GetPropSIValue (<*PropName*>)
Python Example	`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 [Optimetrics]

Returns the property value for a setup property.

UI Access	NA
Parameters	Name | Type | Description property-path | | a child object's property path. See property path discussion here.
Return Value	Returns the value of an setup property.

Python Syntax	GetPropValue(propPath)
Python Example	`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.

VB Syntax	GetPropValue()
VB Example	GetPropValue("Optimetrics/OptimizationSetup1/Enabled") Returns True if Enabled, or False if disabled.

GetSetupNames [Optimetrics]

Gets a list of Optimetrics setup names

UI Access	NA
Parameters	Name | Type | Description None | |
Return Value	IAnsoftCollectionObj – a collection of Optimetrics setup names

Python Syntax	GetSetupNames()
Python Example	oModule = oDesign.GetModule("Optimetrics") setupNames = oModule.GetSetupNames()

VB Syntax	GetSetupNames()
VB Example	`Set oModule_opt = oDesign.GetModule("Optimetrics")` `Set opt_setup_list = oModule.GetSetupNames()` `numsetups = setupnames.Count`

GetSetupNamesByType [Optimetrics]

Gets a list of Optimetrics setup names by type.

UI Access	NA
Parameters	Name | Type | Description <Optimetrics type> | String | Examples: parametric, optimization, statistical, sensitivity
Return Value	Array of Optimetrics setup names of the given type.

Python Syntax	GetSetupNamesByType (<Optimetrics type>)
Python Example	`for name in oModule.GetSetupNamesByType("optimization")` `  AddInfoMessage(str(name))`

VB Syntax	GetSetupNamesByType <Optimetrics type>
VB Example	`For each name in oModule.GetSetupNamesByType("optimization")` `  Msgbox name` `Next`

ImportSetup

Import an Optimetric setup from a file.

<table>
<tr><td>UI Access</td><td colspan="3">NA</td></tr>
<tr><td>Parameters</td><td colspan="3"><table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><SetupTypeName></td><td>String</td><td>Must be one of "OptiParametric" , "OptiOptimization", "OptiSensitivity", "OptiStatistical", or "OptiDesignExplorer".</td></tr>
<tr><td><SetupInfo></td><td>Array</td><td>Array("NAME:<SetupName>", "FilePath")
<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>Return Value</td><td colspan="3">None</td></tr>
</table>

Python Syntax	ImportSetup (<SetupTypeName>, <SetupInfo>)
Python Example	oModule.ImportSetup ("OptiStatistical", ["NAME:StatisticalSetup1", "c:/importdir/mySetupInfoFile"])

VB Syntax	ImportSetup <SetupTypeName>, <SetupInfo>
VB Example	oModule.ImportSetup "OptiStatistical", Array("NAME:StatisticalSetup1", "c:/importdir/mySetupInfoFile")

InsertSetup

Adds a new solution setup.

UI Access	[product] > **Analysis Setup** > **Add Solution Setup**.

Parameters

Name	Type	Description
<*SetupType*>	String	Type of setup to be inserted.
<*Attributes*>	Array	Structured array. Array("NAME:<SetupName>", <NamedParameters>) <Named Parameters> "AdaptMultipleFreqs:=", <boolean>, "Frequency:=", <string>, "MaxDeltaS:=", <float>, "PortsOnly:=", <boolean>, "UseMatrixConv:=", <boolean>, ### If UseMatrixCov is True, ### [

		"NAME:Matrix Convergence", <Convergence Array>], MaximumPasses:=", <integer>, "MinimumPasses:=", <integer>, "MinimumConvergedPasses:=", <integer>, "PercentRefinement:=", <integer>, "IsEnabled:=", <boolean>, "BasisOrder:=", <integer>, "DoLambdaRefine:=", <boolean>, "DoMaterialLambda:=", <boolean>, "SetLambdaTarget:=", <boolean>, "Target:=", <float>, "PortAccuracy:=", <integer>, "SaveRadFieldsOnly:=", <boolean>, "SaveAnyFields:=", <boolean>, "ListsForFields:=", <array of string>, "IESolverType:=", <string>, "LambdaTargetForIESolver:=", <float> "UseDefaultLambdaTgtForIESolver:=", <boolean>

Return Value	None.

Python Syntax	InsertSetup(<*SetupType*>, <*Attributes*>)
Python Example	Maxwell 3D Eddy Current Setup `oModule = oDesign.GetModule("AnalysisSetup")` `oModule.InsertSetup("EddyCurrent",` `[` `"NAME:EddyCurrent_03",` `"Enabled:=" , True,` `[` `"NAME:MeshLink",` `"ImportMesh:=" , False` `],` `"MaximumPasses:=" , 4,` `"MinimumPasses:=" , 2,` `"MinimumConvergedPasses:=", 1,` `"PercentRefinement:=" , 30,` `"SolveFieldOnly:=" , False,` `"PercentError:=" , 0.1,` `"SolveMatrixAtLast:=" , True,` `"UseNonLinearIterNum:=" , False,`

```
[
  "NAME:ExpressionCache",
  [
    "NAME:CacheItem",
    "Title:="               , "eddy_loss1",
    "Expression:="          , "eddy_loss",
    "Intrinsics:="          , "Phase=\'0deg\'",
    "ReportType:="          , "Fields",
    [
      "NAME:ExpressionContext"
    ]
  ]
],
"UseCacheFor:="         , ["Pass"],
"UseIterativeSolver:="  , False,
"RelativeResidual:="    , 0.0001,
"NonLinearResidual:="   , 0.0001,
"SmoothBHCurve:="       , False,
"Frequency:="           , "200Hz",
"HasSweepSetup:="       , False,
```

```
  "UseHighOrderShapeFunc:=", False,
  "UseMuLink:="           , False,
  "LossAdaptiveCtrl:="    , "0.3"
])
```

HFSS simulation

```
oModule.InsertSetup("HfssDrivenAuto",
["NAME:Setup1",
  "Enabled:=", True,
  "AutoSolverSetting:=", "Balanced",
  ["NAME:Sweeps",
          ["NAME:Sweep",
            "RangeType:=", "LinearStep",
            "RangeStart:=", "1GHz",
            "RangeEnd:=", "10GHz",
            "RangeStep:=", "1GHz"
            ]
  ],
  "SaveRadFieldsOnly:=", False,
  "SaveAnyFields:=", True,
  "Type:=", "Discrete"
  ])
```

```
oModule.InsertSetup("HfssDrivenAuto",
        [
                "NAME:Setup2",
                "SolveType:="           , "Auto",
                "IsEnabled:="           , True,
        [
                "NAME:MeshLink",
                "ImportMesh:="          , False
        ],
        "AutoSolverSetting:="   , "Balanced",
        [
                "NAME:Sweeps",
                [
                        "NAME:Sweep",
                        "RangeType:="           , "LinearCount",
                        "RangeStart:="          , "0GHz",
                        "RangeEnd:="            , "10GHz",
                        "RangeCount:="          , 501
                ]
```

```
        ],
        "SaveRadFieldsOnly:="   , False,
        "SaveAnyFields:="       , True,
        "InfiniteSphereSetup:=" , "Infinite Sphere1",
        "ListsForFields:="      , ["Objectlist2"],
        "Type:="                , "Interpolating"
        ])

oModule.InsertSetup("HfssDriven",
["NAME:Setup3",
        "AdaptMultipleFreqs:=", False,
        "Frequency:=", "5GHz",
        "MaxDeltaS:=", 0.02,
        "PortsOnly:=", False,
        "UseMatrixConv:=", False,
        "MaximumPasses:=", 6,
        "MinimumPasses:=", 1,
        "MinimumConvergedPasses:=", 1,
        "PercentRefinement:=", 30,
        "IsEnabled:=", True,
        "BasisOrder:=", 1,
```

```
        "DoLambdaRefine:=", True,
        "DoMaterialLambda:=", True,
        "SetLambdaTarget:=", False,
        "Target:=", 0.3333,
        "UseMaxTetIncrease:=", False,
        "PortAccuracy:=", 2,
        "UseABCOnPort:=", False,
        "SetPortMinMaxTri:=", False,
        "UseDomains:=", True,
        "UseIterativeSolver:=", False,
        "IterativeResidual:=", 1E-06,
        "DDMSolverResidual:=", 0.0001,
        "EnhancedLowFreqAccuracy:=", True,
        "SaveRadFieldsOnly:=", False,
        "SaveAnyFields:=", True,
        "IESolverType:=", "Auto",
        "LambdaTargetForIESolver:=", 0.15,
        "UseDefaultLambdaTgtForIESolver:=", True,
        "SkipIERegionSolveDuringAdaptivePasses:=", True
        "RayDensityPerWavelength:=", 4,
```

```
        "MaxNumberOfBounces:="  , 5,
        "InfiniteSphereSetup:=" , "Infinite Sphere1",
        "SkipSBRSolveDuringAdaptivePasses:=", True,
        "PTDUTDSimulationSettings:=", "PTD Correction + UTD Rays",
        "PTDEdgeDensity:="      , 20
                                            ])
```

An HFSS with Hybrid and Arrays setup

```
import ScriptEnv
ScriptEnv.Initialize("Ansoft.ElectronicsDesktop")
oDesktop.RestoreWindow()
oProject = oDesktop.SetActiveProject("Project1")
oDesign = oProject.SetActiveDesign("HFSSDesign1")
oModule = oDesign.GetModule("AnalysisSetup")
oModule.InsertSetup("HfssDriven",
  [
    "NAME:Setup1",
    "SolveType:="           , "Single",
    "Frequency:="           , "5GHz",
    "MaxDeltaE:="           , 0.1,
    "MaximumPasses:="       , 6,
    "MinimumPasses:="       , 1,
    "MinimumConvergedPasses:=", 1,
```

```
    "PercentRefinement:="   , 30,
    "IsEnabled:="           , True,
    [
      "NAME:MeshLink",
      "ImportMesh:="          , False
    ],
    "BasisOrder:="          , 1,
    "DoLambdaRefine:="      , True,
    "DoMaterialLambda:="    , True,
    "SetLambdaTarget:="     , False,
    "Target:="              , 0.3333,
    "UseMaxTetIncrease:="   , False,
    "DrivenSolverType:="    , "Direct Solver",
    "EnhancedLowFreqAccuracy:=", False,
    "SaveRadFieldsOnly:="   , False,
    "SaveAnyFields:="       , True,
    "IESolverType:="        , "Auto",
    "LambdaTargetForIESolver:=", 0.15,
    "UseDefaultLambdaTgtForIESolver:=", True,
    "IE Solver Accuracy:="  , "Balanced",
```

```
    "RayDensityPerWavelength:=", 4,
    "MaxNumberOfBounces:="  , 5,
    "RadiationSetup:="      , "Infinite Sphere1",
    "PTDUTDSimulationSettings:=", "None",
    "EnableSBRSelfCoupling:=", False,
    "UseSBRAdvOptionsGOBlockage:=", False,
    "UseSBRAdvOptionsWedges:=", False,
    "SkipSBRSolveDuringAdaptivePasses:=", False,
    "UseSBREnhancedRadiatedPowerCalculation:=", True
  ])
```

An SBR+ Setup with Fast Frequency Looping

```
oModule.InsertSetup("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
        ])
```

SBR+ Setup with Enhanced Radiated Power Calculation

```
oModule.InsertSetup("HfssDriven",
  [
    "NAME:Setup2",
    "IsEnabled:="           , True,
    [
      "NAME:MeshLink",
      "ImportMesh:="          , False
    ],
    "IsSbrRangeDoppler:="   , False,
    "RayDensityPerWavelength:=", 4,
    "MaxNumberOfBounces:="  , 5,
    "EnableCWRays:="        , False,
    "RadiationSetup:="      , "Infinite Sphere1",
    "PTDUTDSimulationSettings:=", "None",
    "FastFrequencyLooping:=", False,
    "UseSBRAdvOptionsGOBlockage:=", False,
    "UseSBRAdvOptionsWedges:=", False,
    [
      "NAME:Sweeps",
      [
```

```
        "NAME:Sweep",
        "RangeType:="           , "LinearStep",
        "RangeStart:="          , "1GHz",
        "RangeEnd:="            , "10GHz",
        "RangeStep:="           , "1GHz"
      ]
    ],
    "ComputeFarFields:="    , True,
    "UseSBREnhancedRadiatedPowerCalculation:=", True,
    "IsGOBlockageEnabled:=" , False,
    "GOBlockageSurfaceSelfBlock:=", False
  ])
```

Insert 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.InsertSetup("HfssRFDischarge",
  [
    "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:="  , False,
      "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:="          , "Argon"
    "Save Electron Density:=", True,
    "Is Pulsed Signal:="    , True,
    "Duty Cycle:="          , 50
  ])
```

Setup with UseMatrixConv is True and Matrix Convergence Entry Specified as All

```
oModule.InsertSetup("Setup1",
  [
    "NAME:Setup1",
    "SolveType:="           , "Single",
    "Frequency:="           , "10GHz",
    "MaxDeltaS:="           , 0.02,
    "UseMatrixConv:="       , True,
    [
      "NAME:Matrix Convergence",
      "AllEntries:="          , True,
      "MagLimit:="            , "0.01",
      "PhaseLimit:="          , "2deg",
      "MagMinThreshold:="     , 0.01
```

```
    ],
    "MaximumPasses:="       , 12,
    "MinimumPasses:="       , 1,
    "MinimumConvergedPasses:=", 1,
    "PercentRefinement:="   , 30,
    "IsEnabled:="           , True,
    [
      "NAME:MeshLink",
      "ImportMesh:="          , False
    ],
    "BasisOrder:="          , 1,
    "DoLambdaRefine:="      , True,
    "DoMaterialLambda:="    , True,
    "SetLambdaTarget:="     , False,
    "Target:="              , 0.3333,
    "UseMaxTetIncrease:="   , False,
    "PortAccuracy:="        , 2,
    "UseABCOnPort:="        , False,
    "SetPortMinMaxTri:="    , False,
    "DrivenSolverType:="    , "Direct Solver",
```

```
    "EnhancedLowFreqAccuracy:=", False,
    "SaveRadFieldsOnly:="   , False,
    "SaveAnyFields:="       , True,
    "IESolverType:="        , "ACA",
    "LambdaTargetForIESolver:=", 0.15,
    "UseDefaultLambdaTgtForIESolver:=", True,
    "IE Solver Accuracy:="  , "Balanced",
    "InfiniteSphereSetup:=" , ""
  ])
```

Setup with UseMatrixConv as Entries and MeshLink

```
oModule = oDesign.GetModule("AnalysisSetup")
oModule.InsertSetup("Setup1",
  [
    "NAME:Setup1",
    "SolveType:="           , "Single",
    "Frequency:="           , "10GHz",
    "MaxDeltaS:="           , 0.02,
    "UseMatrixConv:="       , True,
    [
      "NAME:Matrix Convergence",
      [
```

```
        "NAME:Entries",
        [
            "NAME:Entry",
            "Port1:="          , "A[1,1]P1",
            "Port2:="          , "A[1,1]P1",
            "MagLimit:="               , "0.011",
            "PhaseLimit:="             , "5deg"
        ],
        [
            "NAME:Entry",
            "Port1:="          , "A[4,4]P2",
            "Port2:="          , "A[1,1]P2",
            "MagLimit:="               , "0.041",
            "PhaseLimit:="             , "5deg"
        ]
      ]
    ],
    "MaximumPasses:="       , 1,
    "MinimumPasses:="       , 1,
    "MinimumConvergedPasses:=", 1,
```

```
    "PercentRefinement:="   , 30,
    "IsEnabled:="           , True,
    [
      "NAME:MeshLink",
      "ImportMesh:="          , True,
      "Project:="             , "This Project*",
      "Product:="             , "",
      "Design:="              , "unit",
      "Soln:="                , "Setup1 : LastAdaptive",
      [
        "NAME:Params",
        "Airbox_dist:="         , "9.9931mm",
        "Scan_Theta:="          , "0deg",
        "Scan_phi:="            , "0deg",
        "VirtualObject_dist:="  , "2.9979mm",
        "coax_inner_rad:="      , "0.125mm",
        "coax_outer_rad:="      , "0.425mm",
        "cut_off:="             , "1.7mm",
        "feedX:="               , "-0.4mm",
        "feedY:="               , "2mm",
        "feed_length:="         , "1mm",
```

```
        "patchX:="              , "9.65mm",
        "rot:="                 , "45deg",
        "subH:="                , "0.8mm",
        "subX:="                , "15mm",
        "subY:="                , "15mm"
      ],
      "ForceSourceToSolve:="  , False,
      "PreservePartnerSoln:=" , False,
      "PathRelativeTo:="      , "TargetProject",
      "ApplyMeshOp:="         , True,
      "AdaptPort:="           , False
    ],
    "BasisOrder:="          , 1,
    "DoLambdaRefine:="      , False,
    "DoMaterialLambda:="    , True,
    "SetLambdaTarget:="     , False,
    "Target:="              , 0.3333,
    "UseMaxTetIncrease:="   , False,
    "PortAccuracy:="        , 2,
    "UseABCOnPort:="        , False,
```

```
    "SetPortMinMaxTri:="    , False,
    "DrivenSolverType:="    , "Domain Decomposition",
    "IterativeResidual:="   , 0.0001,
    "DDMSolverResidual:="   , 0.0001,
    "EnhancedLowFreqAccuracy:=", False,
    "SaveRadFieldsOnly:="   , False,
    "SaveAnyFields:="       , True,
    "IESolverType:="        , "Auto",
    "LambdaTargetForIESolver:=", 0.15,
    "UseDefaultLambdaTgtForIESolver:=", True,
    "IE Solver Accuracy:="  , "Balanced",
    "InfiniteSphereSetup:=" , ""
  ])
```

Example from design with UseMatrixConv True for AllDiagEntries

```
    "UseMatrixConv:="       , True,
    [
      "NAME:Matrix Convergence",
      "AllDiagEntries:="      , True,
      "DiagonalMag:="         , "0.03",
      "DiagonalPhase:="       , "4rad",
      "MagMinThreshold:="     , 0.01,
```

```
      "AllOffDiagEntries:="   , True,
      "OffDiagonalMag:="      , "0.05",
      "OffDiagonalPhase:="    , "6deg",
      "MagMinThreshold:="     , 0.01
    ],
```

Example from Modal design with UseMatrixConv True for Entries

```
    "UseMatrixConv:="       , True,
    [
      "NAME:Matrix Convergence",
      [
        "NAME:Entries",
        [
            "NAME:Entry",
            "Port1:="          , "Port1",
            "Port2:="          , "Port1",
            "MagLimit:="               , "0.02",
            "PhaseLimit:="             , "5deg"
        ],
        [
            "NAME:Entry",
```

```
            "Port1:="          , "Port3",
            "Port2:="          , "Port2",
            "MagLimit:="               , "0.03",
            "PhaseLimit:="             , "5deg"
        ]
      ]
    ],
```

Example from Design with Periodic Ports and UseMaxtrixConv is True

```
    "UseMatrixConv:="       , True,
    [
      "NAME:Matrix Convergence",
      [
        "NAME:Entries",
        [
          "NAME:Entry",
          "Port1:="              , "Incident_Port1",
          "Port2:="              , "Incident_Port1",
          "MagLimit:="           , "0.11",
          "PhaseLimit:="         , "5deg"
        ],
        [
```

```
            "NAME:Entry",
            "Port1:="               , "Incident_Port2",
            "Port2:="               , "Incident_Port2",
            "MagLimit:="            , "0.22",
            "PhaseLimit:="          , "5deg"
          ]
        ]
    ],
```

Example Terminal Design with UseMatrixConv = True and Expression Cache

```
oProject = oDesktop.SetActiveProject("term-2x2_1_parasolid")
oDesign = oProject.SetActiveDesign("unit")
oModule = oDesign.GetModule("AnalysisSetup")
oModule.InsertSetup("Setup1",
  [
    "NAME:Setup1",
    "SolveType:="           , "Single",
    "Frequency:="           , "10GHz",
    "MaxDeltaS:="           , 0.02,
    "UseMatrixConv:="       , True,
    [
```

```
      "NAME:Matrix Convergence",
      [
        "NAME:Entries",
        [
             "NAME:Entry",
             "Port1:="          , "P2",
             "Port2:="          , "P1",
             "MagLimit:="               , "0.05",
             "PhaseLimit:="             , "10deg"
        ]
      ]
    ],
    "MaximumPasses:="       , 15,
    "MinimumPasses:="       , 1,
    "MinimumConvergedPasses:=", 1,
    "PercentRefinement:="   , 30,
    "IsEnabled:="           , True,
    [
      "NAME:MeshLink",
      "ImportMesh:="          , False
    ],
```

```
    "BasisOrder:="          , 1,
    "DoLambdaRefine:="      , True,
    "DoMaterialLambda:="    , True,
    "SetLambdaTarget:="     , False,
    "Target:="              , 0.3333,
    "UseMaxTetIncrease:="   , False,
    "PortAccuracy:="        , 2,
    "UseABCOnPort:="        , False,
    "SetPortMinMaxTri:="    , False,
    "DrivenSolverType:="    , "Iterative Solver",
    "IterativeResidual:="   , 0.0001,
    "DDMSolverResidual:="   , 0.0001,
    "EnhancedLowFreqAccuracy:=", False,
    "SaveRadFieldsOnly:="   , False,
    "SaveAnyFields:="       , True,
    [
      "NAME:ExpressionCache",
      [
        "NAME:CacheItem",
        "Title:="               , "dB_AxialRatioValue_1",
```

```
        "Expression:="          , "dB(AxialRatioValue)",
        "Intrinsics:="          , "Phi=\'0deg\' Theta=\'0deg\'",
        "ReportType:="          , "Far Fields",
        [
          "NAME:ExpressionContext",
          "Context:="             , "Infinite Sphere1"
        ]
      ],
      [
        "NAME:CacheItem",
        "Title:="               , "dB_St_Diff1_Diff1__1",
        "Expression:="          , "dB(St(Diff1,Diff1))",
        "Intrinsics:="          , "",
        "ReportType:="          , "Terminal Solution Data",
        [
          "NAME:ExpressionContext",
          "Diff:="                , "Differential Pairs"
        ]
      ],
      [
        "NAME:CacheItem",
```

```
          "Title:="               , "dB_St_P1_P1__1",
          "Expression:="          , "dB(St(P1,P1))",
          "Intrinsics:="          , "",
          "ReportType:="          , "Terminal Solution Data",
          [
            "NAME:ExpressionContext",
            "Diff:="                , "Terminals"
          ]
]
     ],
     "CacheSaveKind:="       , "Delta",
     "ConstantDelta:="       , "0s",
     "IESolverType:="        , "Auto",
     "LambdaTargetForIESolver:=", 0.15,
     "UseDefaultLambdaTgtForIESolver:=", True,
     "IE Solver Accuracy:="  , "Balanced",
     "InfiniteSphereSetup:=" , ""
])
```

Example HFSS Hybrid and Arrays with Beta Parallel Component Mesh Adapt

```
oModule.InsertSetup("HfssDriven",
```

```
[
  "NAME:Setup2",
  "SolveType:="           , "Single",
  "Frequency:="           , "5GHz",
  "MaxDeltaS:="           , 0.02,
  "UseMatrixConv:="       , False,
  "MaximumPasses:="       , 6,
  "MinimumPasses:="       , 1,
  "MinimumConvergedPasses:=", 1,
  "PercentRefinement:="   , 30,
  "ComponentAdaptOption:=", "Fully Independent",
  "PerformFullDesignSolveAtLastPass:=", True,
  "IsEnabled:="           , True,
  [
    "NAME:MeshLink",
    "ImportMesh:="          , False
  ],
  "BasisOrder:="          , 1,
  "DoLambdaRefine:="      , True,
  "DoMaterialLambda:="    , True,
  "SetLambdaTarget:="     , False,
```

```
    "Target:="              , 0.3333,
    "UseMaxTetIncrease:="   , False,
    "PortAccuracy:="        , 2,
    "UseABCOnPort:="        , False,
    "SetPortMinMaxTri:="    , False,
    "DrivenSolverType:="    , "Direct Solver",
    "EnhancedLowFreqAccuracy:=", False,
    "SaveRadFieldsOnly:="   , False,
    "SaveAnyFields:="       , True,
    "IESolverType:="        , "Auto",
    "LambdaTargetForIESolver:=", 0.15,
    "UseDefaultLambdaTgtForIESolver:=", True,
    "IE Solver Accuracy:="  , "Balanced",
    "InfiniteSphereSetup:=" , "",
    "MaxPass:="             , 10,
    "MinPass:="             , 1,
    "MinConvPass:="         , 1,
    "PerError:="            , 1,
    "PerRefine:="           , 30
  ])
```

VB Syntax	InsertSetup <*SetupType*>, <*Attributes*>
VB Example	An HFSS Driven project solution with Auto Solution: oModule.InsertSetup "HfssDrivenAuto", Array("NAME:Setup1", "IsEnabled:=", true, _ "AutoSolverSetting:=", "Balanced", _ Array("NAME:Sweeps", Array("NAME:Sweep", "RangeType:=", "LinearCount", _ "RangeStart:=", "1GHz", "RangeEnd:=", "10GHz", "RangeCount:=", "451")), _ "SaveRadFieldsOnly:=", false, "SaveAnyFields:=", false, "Type:=", "Interpolating") A Driven solution type with a mesh link. References to dependent solve in old scripts are converted to mesh link form. oModule.InsertSetup "HfssDriven", Array("NAME:Setup1", "Frequency:=", "1GHz", "MaxDeltaE:=", 0.1, "MaximumPasses:=", 6, "MinimumPasses:=", 1, "MinimumConvergedPasses:=", 1, "PercentRefinement:=", 30, "IsEnabled:=", true, Array("NAME:MeshLink", "Project:=", "Tee.aedt",

```
"Design:=", "TeeModel",
"Soln:=", "Setup1 : LastAdaptive",
Array("NAME:Params", "offset:=", "0in"),
"ForceSourceToSolve:=", false,
"PreservePartnerSoln:=", false,
"PathRelativeTo:=", "SourceProduct",
"ApplyMeshOp:=", true),
"BasisOrder:=", 1,
"UseIterativeSolver:=", false,
"DoLambdaRefine:=", false,
"DoMaterialLambda:=", true,
"SetLambdaTarget:=", false,
"Target:=", 0.3333,
"UseMaxTetIncrease:=", false,
"MaxTetIncrease:=", 1000000,
"EnableSolverDomains:=", false,
"ThermalFeedback:=", false,
"UsingConstantDelta:=", 0,
"ConstantDelta:=", "0s",
"NumberSolveSteps:=", 1)
```

```
A Driven solution type with ports:
oModule.InsertSetup "HfssDriven",
Array("NAME:Setup2",
"Frequency:=", "1GHz",
"PortsOnly:=", false,
"MaxDeltaS:=", 0.02,
"UseMatrixConv:=", false,
"MaximumPasses:=", 6,
"MinimumPasses:=", 1,
"MinimumConvergedPasses:=", 1,
"PercentRefinement:=", 30,
"IsEnabled:=", true,
"BasisOrder:=", 1,
"UseIterativeSolver:=", true,
"IterativeResidual:=", 0.0001,
"DoLambdaRefine:=", true,
"DoMaterialLambda:=", false,
"SetLambdaTarget:=", false,
"Target:=", 0.3333,
"UseMaxTetIncrease:=", false,
"MaxTetIncrease:=", 1000000,
```

```
"PortAccuracy:=", 2,
"UseABCOnPort:=", true,
"SetPortMinMaxTri:=", false,
"EnableSolverDomains:=", false,
"ThermalFeedback:=", false,
"UsingConstantDelta:=", 0,
"ConstantDelta:=", "0s",
"NumberSolveSteps:=", 1)
An Eigenmode solution type:
oModule.InsertSetup "HfssEigen",
Array("NAME:Setup2",
"MinimumFrequency:=", "1.77347GHz",
"NumModes:=", 1,
"MaxDeltaFreq:=", 10,
"ConvergeOnRealFreq:=", true,
"MaximumPasses:=", 3,
"MinimumPasses:=", 1,
"MinimumConvergedPasses:=", 1,
"PercentRefinement:=", 30,
"IsEnabled:=", true,
```

```
"BasisOrder:=", 1,
"UseIterativeSolver:=", false,
"DoLambdaRefine:=", true,
"DoMaterialLambda:=", true,
"SetLambdaTarget:=", false,
"Target:=", 0.2,
"UseMaxTetIncrease:=", false,
"MaxTetIncrease:=", 1000000,
"UsingConstantDelta:=", 0,
"ConstantDelta:=", "0s",
"NumberSolveSteps:=", 1)
A Driven solution type with ports and matrix convergence:
oModule.InsertSetup "HfssDriven",
Array("NAME:Setup2",
"Frequency:=", "1GHz",
"PortsOnly:=", false,
"MaxDeltaS:=", 0.02,
"UseMatrixConv:=", true,
Array("NAME:ConvergenceMatrix",
"AllDiagEntries:=", true,
"MagMinThreshold:=", 0.01,
```

```
"Entry:=", Array("Port1:=", "abc", "ModeNum1:=", 1)),
"MaximumPasses:=", 6,
"MinimumPasses:=", 1,
"MinimumConvergedPasses:=", 1,
"PercentRefinement:=", 30,
"IsEnabled:=", true,
"BasisOrder:=", 1,
"UseIterativeSolver:=", false,
"DoLambdaRefine:=", true,
"DoMaterialLambda:=", true,
"SetLambdaTarget:=", false,
"Target:=", 0.3333,
"UseMaxTetIncrease:=", false,
"MaxTetIncrease:=", 1000000,
"PortAccuracy:=", 2,
"UseABCOnPort:=", true,
"SetPortMinMaxTri:=", false,
"EnableSolverDomains:=", false,
"ThermalFeedback:=", false,
"UsingConstantDelta:=", 0,
```

```
"ConstantDelta:=", "0s",
"NumberSolveSteps:=", 1)
A Driven solution type with Multi-Frequencies specified:
oModule.InsertSetup "HfssDriven", Array("NAME:Setup2", "AdaptMultipleFreqs:=", true, _
Array("NAME:MultipleAdaptiveFreqsSetup", "1.1GHz:=", Array( _
0.02), "10GHz:=", Array(0.02), "11GHz:=", Array(0.02), "12.5GHz:=", Array(0.02),
"15GHz:=", Array( _
0.02), "20GHz:=", Array(0.02)), "MaximumPasses:=", 6, "MinimumPasses:=", 1, "Min-
imumConvergedPasses:=",  _
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:=", false, "UseIterativeSolver:=",  _
false, "SaveRadFieldsOnly:=", false, "SaveAnyFields:=", true, "IESolverType:=",  _
"Auto", "LambdaTargetForIESolver:=", 0.15, "UseDefaultLambdaTgtForIESolver:=",  _
true, "RayDensityPerWavelength:=", 4, "MaxNumberOfBounces:=", 5)
An Driven Solution type with Broadband specified:
oModule.InsertSetup "HfssDriven", Array("NAME:Setup1", "AdaptMultipleFreqs:=", true, _
Array("NAME:MultipleAdaptiveFreqsSetup", Array("NAME:Broadband", "Low:=",  _
"2GHz", "High:=", "20GHz")), "MaximumPasses:=", 6, "MinimumPasses:=", 1, "Min-
imumConvergedPasses:=",  _
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:=", false, "UseIterativeSolver:=",  _
false, "SaveRadFieldsOnly:=", false, "SaveAnyFields:=", true, "IESolverType:=",  _
"Auto", "LambdaTargetForIESolver:=", 0.15, "UseDefaultLambdaTgtForIESolver:=",  _
true, "RayDensityPerWavelength:=", 4, "MaxNumberOfBounces:=", 5)
oModule.InsertFrequencySweep "Setup1", Array("NAME:Sweep", "IsEnabled:=", true,
"RangeType:=",  _
"LinearCount", "RangeStart:=", "2GHz", "RangeEnd:=", "20GHz", "RangeCount:=",  _
451, "Type:=", "Interpolating", "SaveFields:=", false, "SaveRadFields:=",  _
false, "InterpTolerance:=", 0.5, "InterpMaxSolns:=", 250, "InterpMinSolns:=",  _
0, "InterpMinSubranges:=", 1, "ExtrapToDC:=", false, "InterpUseS:=", true, "Inter-
pUsePortImped:=",  _
false, "InterpUsePropConst:=", true, "UseDerivativeConvergence:=", false, "Inter-
pDerivTolerance:=",  _
0.2, "UseFullBasis:=", true, "EnforcePassivity:=", true, "PassivityErrorTolerance:=",
_
0.0001)
```

PasteSetup [Optimetrics]

Pastes the specified Optimetrics setup.

UI Access	NA
Parameters	<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>
Return Value	None

Python Syntax	PasteSetup (<SetupName>)
Python Example	`oModule.PasteSetup ("OptimizationSetup1")`

VB Syntax	PasteSetup <SetupName>
VB 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	<table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><OldName></td><td>String</td><td>The name that needs to be replaced</td></tr><tr><td><NewName></td><td>String</td><td>Replacement name</td></tr></table>
Return Value	None

Python Syntax	RenameSetup (<OldName> <NewName>)
Python Example	`oModule.RenameSetup ("OptimizationSetup1" "MyOptimization")`

VB Syntax	RenameSetup <OldName> <NewName>
VB Example	`oModule.RenameSetup "OptimizationSetup1" "MyOptimization"`

SetPropValue [Optimetrics]

Sets the property value for the active Optimetrics setup.

UI Access	Set Property value on Optimetrics objects
Parameters	<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>
Return Value	True if the property is found and the new value is valid. Otherwise return False.

Python Syntax	SetPropValue(propPath, newValue)
Python Example	`oOptModule.SetPropValue("ParametricSetup1\Enabled", False) //disable ParametricSetup1` `oOptModule.SetPropValue("OptimizationSetup1/Optimizer","Quasi Newton")`

VB Syntax	SetPropValue(propPath, newValue)
VB Example	`SetPropValue("ParametricSetup1\Enabled", False)`

SolveAllSetup

Solves all Optimetrics setups

UI Access	Right-click on **Optimetrics** in Project Manager and select **Analyze>All** from context menu
Parameters	Name: isBlocking; Type: Boolean; Description: An optional arg that defaults to `true`. If it's `false`, the command immediately returns while the analysis or analyses run in the background. The status of the analyses can be checked with the `AreThereSimulationsRunning` command.
Return Value	None

Python Syntax	SolveAllSetup()
Python Example	`oModule.SolveAllSetup()`

VB Syntax	SolveAllSetup
VB Example	`oModule.SolveAllSetup`

SolveSetup [Optimetrics]

Solves the specified Optimetrics setup.

UI Access	Right-click the setup in the project tree, and then click **Analyze** on the shortcut menu.
Parameters	<table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><*SetupName*></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 `true`. If it's `false`, the command immediately returns while the analysis or analyses run in the background. The status of the analyses can be checked with the `AreThereSimulationsRunning` command.</td></tr></table>
Return Value	None

Python Syntax	SolveSetup (<*SetupName*>)
Python Example	`oModule.SolveSetup ("OptimizationSetup1")`

VB Syntax	SolveSetup <*SetupName*>
VB Example	`oModule.SolveSetup "OptimizationSetup1"`

Parametric Script Commands

EditSetup [Parametric]

ExportParametricSetupTable

GenerateVariationData [Parametric]

InsertSetup [Parametric]

EditSetup [Parametric]

Modifies an existing parametric setup

UI Access	Right-click the setup in the project tree, and then click **Properties** on the shortcut menu.
Parameters	<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><ParametricParams></td><td>List</td><td>List that defines the parameters of the parametric setup; examples are listed below.</td></tr></table>
Return Value	None

Python Syntax	EditSetup (<SetupName>, <ParametricParams>)
Python Example	See EditSetup [Optimization]

VB Syntax	EditSetup <SetupName>, <ParametricParams>
VB Example	See EditSetup [Optimization]

ExportParametricSetupTable

Exports the parametric setup table as a CSV file.

UI Access	Double-click parametric setup. Select **Table** tab. Click **Export**.

Parameters	<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>
Return Value	None

Python Syntax	ExportParametricSetupTable (<SetupName>, <filePath>)
Python Example	`oModule.ExportParametricSetupTable('ParametricSetup1', 'E:/Files/ParametricSetup1_Table.csv')`

VB Syntax	ExportParametricSetupTable <SetupName>, <filePath>
VB Example	`obj.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.

VB Example:

```
oModule.GenerateVariationData "ParametricSetup1"
```

InsertSetup [Parametric]

Inserts a new parametric setup.

UI Access	Right-click the **Optimetrics** folder in the project tree, and then click **Add> Parametric** on the shortcut menu.		
Parameters	Name	Type	Description
	<Parametric Params>	Array	Array("NAME:<SetupName>", "SaveFields:=", <SaveField>, <StartingPoint>, "Sim. Setups:=", <SimSetups>, <SweepDefs>, <SweepOps>, Array("NAME:Goals", Array("NAME:Goal", <OptiGoalSpec>), ... Array("NAME:Goal", <OptiGoalSpec>))
	<SetupName>	String	Name of the parametric setup.
	<SimSetups>	Array of Strings	An array of Twin Builder solution setup names.
	<SweepDefs>	Array	Array("NAME:Sweeps", Array("NAME:SweepDefinition", "Variable:=", <VarName>, "Data:=", <SweepData>, "Synchronize:=", <SyncNum>), ... Array("NAME:SweepDefinition", "Variable:=",

			<VarName>, "Data:=", <SweepData>, "Synchronize:=", <SyncNum>))
	<SweepData>	String	"<SweepType>, <StartV>, <StopV>, <StepV>"
	<SweepType>	String	The type of sweep data.
	<SyncNum>	Integer	`SweepDatas` with the same value are synchronized.
	<SweepOps>		Array("NAME:Sweep Operations", "<OpType>:=, Array(<VarValue>, …, <VarValue>), ... <OpType>:=, Array(<VarValue>, …, <VarValue>))
	<OpType>	String	The sweep operation type.
Return Value	None		

Python Syntax	InsertSetup ("OptiParametric", <ParametricParams>)
Python Example	`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"
            ]
          ]
        ]
      ]
    ]
])
```

VB Syntax	InsertSetup "OptiParametric", <ParametricParams>
VB Example	oModule.InsertSetup "OptiParametric", Array("NAME:ParametricSetup1", _ "SaveFields:=", true, _ Array("NAME:StartingPoint"), _

```
"Sim. Setups:=", Array("Setup1"),_
Array("NAME:Sweeps", _
Array("NAME:SweepDefinition", _
"Variable:=", "$width", _
"Data:=", "LIN 12mm 17mm 2.5mm", _
"OffsetF1:=", false, _
"Synchronize:=", 0),
Array("NAME:SweepDefinition", _
"Variable:=", "$length", _
"Data:=", "LIN 8mm 12mm 2mm", _
"OffsetF1:=", false, _
"Synchronize:=", 0)),
Array("NAME:Sweep Operations"), _
Array("NAME:Goals", _
Array("NAME:Goal", _
"Solution:=", "Setup1 : LastAdaptive", _
"Calculation:=", "returnloss", _
"Context:=", "",_
 Array("NAME:Ranges", _
```

```
"Range:=", Array("Var:=", "Freq", "Type:=","s", _
"Start:=", "8GHz", "Stop:=", "8GHz"))), _
Array("NAME:Goal", _
"Solution:=", "Setup1 : LastAdaptive",_
"Calculation:=", "reflect", _
"Context:=", "", _
Array("NAME:Ranges", _
"Range:=", Array("Var:=", "Freq", "Type:=","s", _
"Start:=", "8GHz", "Stop:=", "8GHz")))))
```

Optimization Script Commands

EditSetup [Optimization]

InsertSetup [Optimization]

EditSetup [Optimization]

Modifies an existing optimization setup.

UI Access	Right-click the setup in the Project 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><SetupName></td><td>String</td><td>Name of the Setup</td></tr><tr><td><OptimizationParams></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
])
```

VB Syntax	EditSetup <SetupName>, <OptimizationParams>
VB Example	

```
oModule.EditSetup "OptimizationSetup1",
Array("NAME:OptimizationSetup1", "UseFastCalculationUpdateAlgo:=",  _
  false, "FastCalcOptCtrledByUser:=", false,
"IsEnabled:=", true, "SaveSolutions:=",  _
  false, Array("NAME:StartingPoint"), "Optimizer:=",
"Quasi Newton", Array("NAME:AnalysisStopOptions", "StopForNumIteration:=",  _
  true, "StopForElapsTime:=", false,
"StopForSlowImprovement:=", false, "StopForGrdTolerance:=",  _
  false, "MaxNumIteration:=", 1000, "MaxSolTimeInSec:=",
 3600, "RelGradientTolerance:=",  _
  0, "MinNumIteration:=", 10), "CostFuncNormType:=",
"L2", "PriorPSetup:=", "", "PreSolvePSetup:=",  _
```

```
  true, Array("NAME:Variables"), Array("NAME:LCS"),
Array("NAME:Goals", Array("NAME:Goal", "ReportType:=",  _
  "Standard", "Solution:=", "TR", Array("NAME:SimValueContext",
 "SimValueContext:=", Array( _
  1, 0, 2, 0, false, false, -1, 1, 0, 1, 1, "", 0, 0)),
 "Calculation:=", "Time", "Name:=",  _
  "Time", Array("NAME:Ranges", "Range:=",
Array("Var:=", "Time", "Type:=", "a")), "Condition:=",  _
  "==", Array("NAME:GoalValue", "GoalValueType:=",
"Independent", "Format:=",  _
  "Real/Imag", "bG:=", Array("v:=", "[1;]")),
"Weight:=", "[1;]")), "Acceptable_Cost:=",  _
  0, "Noise:=", 0.0001, "UpdateDesign:=", false,
"UpdateIteration:=", 5, "KeepReportAxis:=",  _
  true, "UpdateDesignWhenDone:=", true)
```

InsertSetup [Optimization]

Use: Inserts a new optimization setup.

UI Access	Right-click the **Optimetrics** folder in the project tree, and then click **Add > Optimization** on the shortcut menu.
Parameters	Name | Type | Description

	<OptimizationParams>	Array	Array("NAME:<SetupName>", "SaveFields:=", <SaveField>, <StartingPoint>, "Optimizer:=", <Optimizer>, "MaxIterations:=", <MaxIter>, "PriorPSetup:=", <PriorSetup>, "PreSolvePSetup:=", <Preceed>, <OptimizationVars>, <Constraint>, Array("NAME:Goals", Array("NAME:Goal", <OptiGoalSpec>, <OptimizationGoalSpec>), ... Array("NAME:Goal", <OptiGoalSpec>, <OptimizationGoalSpec>)), "Acceptable_Cost:=", <AcceptableCost>, "Noise:=", <Noise>, "UpdateDesignWhenDone:=", <UpdateDesign>
	<OptimizationVars>	Array	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])

VB Syntax	InsertSetup "OptiOptimization", <OptimizationParams>
VB Example	oModule.InsertSetup "OptiOptimization",_ Array("NAME:OptimizationSetup1", _ "SaveFields:=", false, _ Array("NAME:StartingPoint", "$length:=", "8mm",_ "$width:=", "14.5mm"), _ "Optimizer:=", "Quasi Newton", _ "MaxIterations:=", 100, _ "PriorPSetup:=", "ParametricSetup1", _ "PreSolvePSetup:=", true, _ Array("NAME:Variables", _

```
"$length:=", Array("i:=", true, "Min:=", "6mm",_
"Max:=", "18mm", _
"MinStep:=", "0.001mm", "MaxStep:=", _
"1.2mm"), _
"$width:=", Array("i:=", true, "Min:=", _
"6.5mm", "Max:=", "19.5mm", _
"MinStep:=", "0.001mm", "MaxStep:=", _
"1.3mm")), _
Array("NAME:LCS"), _
Array("NAME:Goals", _
Array("NAME:Goal", _
"Solution:=", "Setup1 : LastAdaptive", _
"Calculation:=", "reflect", _
"Context:=", "", _
Array("NAME:Ranges", _
"Range:=", Array("Var:=", "Freq", _
"Type:=", "s", _
"Start:=", "8GHz", "Stop:=", "8GHz")), _
"Condition:=", "&lt;=", _
Array("NAME:GoalValue", _
```

```
"GoalValueType:=", "Independent", _
"Format:=", "Real/Imag", _
"bG:=", Array("v:=", "[0.0001]")), _
"Weight:=", "[1]")),
"Acceptable_Cost:=", 0.0002, _
"Noise:=", 0.0001, _
"UpdateDesign:=", true, _
"UpdateIteration:=", 5, _
"KeepReportAxis:=", true, _
"UpdateDesignWhenDone:=", true)
```

Sensitivity Script Commands

EditSetup [Sensitivity]

InsertSetup [Sensitivity]

EditSetup [Sensitivity]

Modifies an existing sensitivity setup.

UI Access	Right-click the setup in the project tree, and then click Properties on the shortcut menu
Parameters	

Name	Type	Description
<SetupName>	String	Name of the Setup

Return Value	None

Python Syntax	EditSetup (<SetupName>, <SensitivityParams>)
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:=", 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])

VB Syntax	EditSetup <SetupName>, <SensitivityParams>
VB Example	oModule.EditSetup("OptimizationSetup1", Array("NAME:OptimizationSetup1", "UseFastCalculationUpdateAlgo:=", False, "FastCalcOptCtrledByUser:=", False, "IsEnabled:=", True, "SaveSolutions:=", False, Array("NAME:StartingPoint"

```
),
"Optimizer:=", "Quasi Newton",
Array(
"NAME:AnalysisStopOptions",
"StopForNumIteration:="        , True,
"StopForElapsTime:=", False,
"StopForSlowImprovement:=", False,
"StopForGrdTolerance:=", False,
"MaxNumIteration:=", 1001,
"MaxSolTimeInSec:=", 3600,
"RelGradientTolerance:=", 0,
"MinNumIteration:="    , 10
),
"CostFuncNormType:=", "L2",
"PriorPSetup:=", "",
"PreSolvePSetup:=", True,
Array(
"NAME:Variables"
),
Array(
"NAME:LCS"
```

```
),
Array(
"NAME:Goals",
Array(
"NAME:Goal",
"ReportType:=", "Standard",
"Solution:=", "TR3",
Array(
"NAME:SimValueContext",
"SimValueContext:=",
Array(1,0,2,0,False,False,-1,1,0,1,1,"",0,0)
),
"Calculation:=", "mag(DIFF1.VAL)",
"Name:=", "DIFF1.VAL",
Array(
"NAME:Ranges",
"Range:=", Array("Var:=", "Time","Type:=", "a")
),
"Condition:=", "==",
```

	Array("NAME:GoalValue", "GoalValueType:=", "Independent", "Format:=", "Real/Imag", "bG:=", Array("v:=", "Array(1;]")), "Weight:=", "Array(1;]")), "Acceptable_Cost:=", 0, "Noise:=", 0.0001, "UpdateDesign:=", False, "UpdateIteration:=", 5, "KeepReportAxis:=", True, "UpdateDesignWhenDone:=", True))

InsertSetup [Sensitivity]

Inserts a new sensitivity setup.

UI Access	Right-click **Optimetrics** in the project tree, and then click **Add>Sensitivity** on the shortcut menu.
Parameters	Name | Type | Description

	<SensitivityParams>	Array	Array("NAME:<SetupName>", "SaveFields:=", <SaveField>, <StartingPoint>, "MaxIterations:=", <MaxIter>, "PriorPSetup:=", <PriorSetup>, "PreSolvePSetup:=", <Preceed>, <SensitivityVars>, <Constraint>, Array("NAME:Goals", Array("NAME:Goal", <OptiGoalSpec>), ..., Array("NAME:Goal", <OptiGoalSpec>)), "Primary Goal:=". <PrimaryGoalID>, "PrimaryError:=", <PrimaryError>)
	<SensitivityVars>	Array	Array("NAME:Variables", "VarName:=", Array("i:=", <IncludeVar>, "Min:=", <MinV>, "Max:=", <MaxV>, "IDisp:=", <InitialDisp>),... "VarName:=", Array("i:=", <IncludeVar>, "Min:=", <MinV>, "Max:=", <MaxV>, "IDisp:=", <InitialDisp>))
	<InitialDisp>	VarValue	Index of the Primary goal. Index starts from zero.
	<PrimaryError>	Double	Error associated with the Primary goal.
Return Value	None		

Python Syntax	InsertSetup ("OptiSensitivity", <SensitivityParams>)
Python Example	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])

VB Syntax	InsertSetup "OptiSensitivity", <SensitivityParams>
VB Example	oModule.InsertSetup "OptiSensitivity", _ Array("NAME:SensitivitySetup1",_ "SaveFields:=", true,_ Array("NAME:StartingPoint"), _ "MaxIterations:=", 20,_ "PriorPSetup:=", "",_

```
"PreSolvePSetup:=", true, _
Array("NAME:Variables"), _
Array("NAME:LCS"),_
Array("NAME:Goals",_
Array("NAME:Goal", _
"Solution:=", "Setup1 : LastAdaptive",_
"Calculation:=", "returnloss",_
"Context:=", "",_
Array("NAME:Ranges",_
"Range:=", Array("Var:=", "Freq", "_
Type:=", "s",_
"Start:=", "8GHz", "Stop:=", "8GHz"))),_
Array("NAME:Goal",_
"Solution:=", "Setup1 : LastAdaptive",_
"Calculation:=", "reflect",_
"Context:=", "",_
Array("NAME:Ranges",_
"Range:=", Array("Var:=","Freq",_
"Type:=", "s",_
"Start:=", "8GHz", "Stop:=", "8GHz")))),_
```

	"Primary Goal:=", 1,_ "PrimaryError:=", 0.001)

Statistical Script Commands

EditSetup [Statistical]

Modifies an existing statistical setup.

UI Access	Right-click the setup in the project tree, and clickProperties on the shortcut menu.
Parameters	Name | Type | Description <SetupName> | String | Name of the Setup | |
Return Value	None

Python Syntax	EditSetup (<SetupName>, <StatisticalParams>)
Python Example	See EditSetup [Optimization]

VB Syntax	EditSetup <SetupName>, <StatisticalParams>
VB Example	See EditSetup [Optimization]

Example:

```
Dim oAnsoftApp

Dim oDesktop

Dim oProject

Dim oDesign

Dim oEditor

Dim oModule

Set oAnsoftApp = CreateObject("Ansoft.ElectronicsDesktop")

Set oDesktop = oAnsoftApp.GetAppDesktop()

oDesktop.RestoreWindow

Set oProject = oDesktop.SetActiveProject("optiguides")

Set oDesign = oProject.SetActiveDesign("HFSSModel1")

Set oModule = oDesign.GetModule("Optimetrics")

oModule.EditSetup "StatisticalSetup1", Array("NAME:StatisticalSetup1",
Array("NAME:ProdOptiSetupData",
"SaveFields:=", true, "CopyMesh:=", false),
Array("NAME:StartingPoint", "$length:=", "7.824547736mm",
"$width:=", "14.8570192mm"),
"MaxIterations:=", 50,
"PriorPSetup:=", "",

Array("NAME:Variables",
"$length:=", Array("i:=", true,
"int:=", false,
"Dist:=", "Uniform",
```

```
"Tol:=", "10%", "StdD:=", "0.2mm", "Min:=", "-3",
"Max:=", "3", "Shape:=", "1", "Scale:=", "0.04mm",
"Location:=", "0.4mm",
"Dataset:=", "", "LatinHypercube:=", "true", "VarMin:=", "0.2mm", "VarMax:=", "0.6mm", "Prob:=",
"0.01",
"Mean:=", "0.4mm"),

"$width:=", Array("i:=", true,
"int:=", false,
"Dist:=", "Gaussian",
"Tol:=", "10%",
"StdD:=", "0.2mm",
"Min:=", "-3", "Max:=", "3",
"Shape:=", "1",
"Scale:=", "0.04mm",
"Location:=", "0.4mm",
"Dataset:=", "",
"LatinHypercube:=", "true",
"VarMin:=", "0.2mm", "VarMax:=", "0.6mm",
"Prob:=", "0.02",
"Mean:=", "0.4mm")),

Array("NAME:Goals", Array("NAME:Goal",
"ReportType:=", "Modal Solution Data",
"Solution:=", "Setup1 : PortOnly",
Array("NAME:SimValueContext", "Domain:=", "Sweep"),
"Calculation:=", "returnloss",
"Name:=", "returnloss",

Array("NAME:Ranges",
"Range:=", Array("Var:=", "Freq",
"Type:=", "s",
"Start:=", "8.2GHz", "Stop:=", "0"))),
```

```
Array("NAME:Goal",
"ReportType:=", "Modal Solution Data",
"Solution:=", "Setup1 : PortOnly",
Array("NAME:SimValueContext",
"Domain:=", "Sweep"),
"Calculation:=", "reflect",
"Name:=", "reflect",
Array("NAME:Ranges",
"Range:=", Array("Var:=", "Freq",
"Type:=", "s",
"Start:=", "8.2GHz", "Stop:=", "0")))))
```

InsertSetup [Statistical]

Inserts a new statistical setup.

<table>
<tr><td>UI Access</td><td>Right-click Optimetrics in the project tree, and then click Add>Statistical on the shortcut menu.</td></tr>
<tr><td>Parameters</td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><StatisticalParams></td><td>Array</td><td>Array("NAME:<SetupName>", "SaveFields:=",
<SaveField>, <StartingPoint>, "MaxIterations:=",
<MaxIter>, "PriorPSetup:=", <PriorSetup>,
"PreSolvePSetup:=", <Preceed>, <StatisticalVars>,
Array("NAME:Goals", Array("NAME:Goal",
<OptiGoalSpec>), …, Array("NAME:Goal",
<OptiGoalSpec>)))</td></tr>
<tr><td><StatisticalVars></td><td>Array</td><td>Array("NAME:Variables",
"VarName:=", Array("i:=", <IncludeVar>, "Dist:=",
<DistType>, "Tol:=", <Tolerance>,</td></tr>
</table>
</td></tr>
</table>

			"StdD:=", <StdD>, "Min:=", <MinCutoff>, "Max:=", <MaxCutoff>, ... "VarName:=", Array("i:=", <IncludeVar>, "Dist:=", <DistType>, "Tol:=", <Tolerance>, "StdD:=", <StdD>, "Min:=", <MinCutoff>, "Max:=", <MaxCutoff>))
	<DistType>	String	Distrbution can be "Gaussian" or "Uniform".
	<Tolerance>	VarValue	The tolerance for the variable when distribution is Uniform
	<StdD>	VarValue	The standard deviation for the variable when distribution is Gaussian.
	<MinCutoff>	Double	The minimum cut-off for the variable when distribution is Gaussian.
	<MaxCutoff>	Double	The maximum cut-off for the variable when distribution is Gaussian.
Return Value	None		

Python Syntax	InsertSetup ("OptiStatistical", <StatisticalParams>)
Python Example	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"]]]])

VB Syntax	InsertSetup "OptiStatistical", <StatisticalParams>
VB Example	oModule.InsertSetup

```
"OptiStatistical", _
Array("NAME:StatisticalSetup1", _
"SaveFields:=", true, _
Array("NAME:StartingPoint"),_
"MaxIterations:=", 50,_
"PriorPSetup:=", "", _
Array("NAME:Variables"), _
Array("NAME:Goals", _
Array("NAME:Goal", _
"Solution:=", "Setup1 : LastAdaptive", _
"Calculation:=", "returnloss", _
"Context:=", "", _
Array("NAME:Ranges", _
"Range:=", Array("Var:=", "Freq", _
"Type:=", "s",_
"Start:=", "8GHz", "Stop:=", "8GHz"))),_
Array("NAME:Goal",_
"Solution:=", "Setup1 : LastAdaptive",_
"Calculation:=", "reflect",_
```

	"Context:=", "", _ Array("NAME:Ranges",_ "Range:=", Array("Var:=", "Freq", "Type:=",_ "s", "Start:=", "8GHz", "Stop:=", "8GHz"))))

Example:

```
Dim oAnsoftApp
Dim oDesktop
Dim oProject
Dim oDesign
Dim oEditor
Dim oModule
Set oAnsoftApp = CreateObject("Ansoft.ElectronicsDesktop")
Set oDesktop = oAnsoftApp.GetAppDesktop()
oDesktop.RestoreWindow
Set oProject = oDesktop.SetActiveProject("OptimTee")
Set oDesign = oProject.SetActiveDesign("TeeModel")
oDesign.ChangeProperty Array("NAME:AllTabs",
Array("NAME:LocalVariableTab",
Array("NAME:PropServers", "LocalVariables"),
Array("NAME:ChangedProps",
```

```
Array("NAME:offset",
Array("NAME:Statistical", "Included:=", true)))))

Set oModule = oDesign.GetModule("Optimetrics")

oModule.InsertSetup "OptiStatistical", Array("NAME:StatisticalSetup1",
Array("NAME:ProdOptiSetupData",
"SaveFields:=", false, "CopyMesh:=", false),
Array("NAME:StartingPoint", "offset:=", "0in"),
"MaxIterations:=", 50, "PriorPSetup:=", "",

Array("NAME:Variables",

"offset:=", Array("i:=", true,
"int:=", false,
"Dist:=", "Gaussian",
"Tol:=", "10%",
"StdD:=", ".5in",
"Min:=", "-3",
"Max:=", "3",
"Shape:=", "1",
"Scale:=", "0in",
"Location:=", "0in",
"Dataset:=", "",
"LatinHypercube:=", "true",
"VarMin:=", "-1in",
"VarMax:=", "1in",
"Prob:=", "0.01",
"Mean:=", "0in")),

Array("NAME:Goals", Array("NAME:Goal",
"ReportType:=", "Modal Solution Data",
"Solution:=", "Setup1 : LastAdaptive", Array("NAME:SimValueContext"),
```

```
"Calculation:=", "Power11",
"Name:=", "Power11",
Array("NAME:Ranges",
"Range:=", Array("Var:=", "Freq", "Type:=", "d",
"DiscreteValues:=", "10GHz")))))
```

For Q3D Extractor and Circuit the command details are as follows:

Inserts a new statistical setup.

Command: Right-click **Optimetrics** in the project tree, and then click **Add>Statistical** on the shortcut menu.

Syntax: InsertSetup "OptiStatistical", <StatisticalParams>

Return Value: None

Parameters: <StatisticalParams>

```
  Array("NAME:<SetupName>", "SaveFields:=",
  <SaveField>, <StartingPoint>, "MaxIterations:=",
  <MaxIter>, "PriorPSetup:=", <PriorSetup>,
  "PreSolvePSetup:=", <Preceed>, <StatisticalVars>,
  Array("NAME:Goals", Array("NAME:Goal",
  <OptiGoalSpec>), …, Array("NAME:Goal",
  <OptiGoalSpec>))),
<StatisticalVars>
  Array("NAME:Variables",
    "VarName:=", Array("i:=", <IncludeVar>, "Dist:=",
      <DistType>, "Tol:=", <Tolerance>,
```

```
      "StdD:=", <StdD>, "Min:=", <MinCutoff>, "Max:=",
      <MaxCutoff>, ...
   "VarName:=", Array("i:=", <IncludeVar>, "Dist:=",
      <DistType>, "Tol:=", <Tolerance>, "StdD:=",
      <StdD>, "Min:=", <MinCutoff>, "Max:=",
      <MaxCutoff>))
```

Parameters:

<DistType>

Type : <string>

Distrbution can be "Gaussian" or "Uniform".

<Tolerance>

Type: <VarValue>

The tolerance for the variable when distribution is Uniform.

<StdD>

Type: <VarValue>

The standard deviation for the variable when distribution is Gaussian.

<MinCutoff>

Type: <double>

The minimum cut-off for the variable when distribution is Gaussian.

<MaxCutoff>

```
    Type: <double>
    The maximum cut-off for the variable when distribution is Gaussian.
Example: oModule.InsertSetup "OptiStatistical", _
  Array("NAME:StatisticalSetup1", _
    "SaveFields:=", true, _
    Array("NAME:StartingPoint"),_
    "MaxIterations:=", 50,_
    "PriorPSetup:=", "", _
    Array("NAME:Variables"), _
    Array("NAME:Goals", _
      Array("NAME:Goal", _
"Solution:=", "Setup1 : LastAdaptive", _
"Calculation:=", "returnloss", _
"Context:=", "", _
Array("NAME:Ranges", _
"Range:=", Array("Var:=", "Freq", _
"Type:=", "s",_
"Start:=", "8GHz", "Stop:=", "8GHz"))),_
      Array("NAME:Goal",_
"Solution:=", "Setup1 : LastAdaptive",_
"Calculation:=", "reflect",_
```

```
"Context:=", "", _
Array("NAME:Ranges",_
"Range:=", Array("Var:=", "Freq", "Type:=", _
"s", "Start:=", "8GHz", "Stop:=", "8GHz"))))
```

12 - User Defined Document Script Commands

The product has to implement the GetModule call to create the UserDefinedDocument scripting object (e.g., Check AltraSimDesign.cpp (function GetMgrIDispatch()). To access the UserDefinedDocuments scripting object, use:

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

Once you have the scripting object, you can use the following methods:

- AddDocument
- EditDocument
- RenameDocument
- DeleteDocument
- UpdateDocument
- ViewHtmlDocument
- ViewPdfDocument
- SaveHtmlDocumentAs
- SavePdfDocumentAs
- GetDocumentDefinitionNames
- DeleteAllDocuments
- UpdateAllDocuments

You can find examples of how to use these methods on each of the methods' pages. A complete example of a Python script is also available, as is an example with a line by line explanation.

AddDocument

Creates a new document based on provided data and traces. The document names come from UserDefinedDocument folder under syslib, userlib, and personallib. This creates a document and places it in the Project Manager under **Results** > **Documents**.

UI Access	Right click on **Results** > **Create Document**. Choose a document name.
Parameters	<table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><data></td><td>Array</td><td>Data the defines the document.</td></tr><tr><td><traces></td><td>Array</td><td>trace data for the inputs in the document.</td></tr></table>
Return Value	None

Python Syntax	AddDocument (<*data*>, <*traces*>)
Python Example	`oModule.AddDocument(` `[` `  "NAME:Design Summary",` `  "",` `  "SysLib",` `  "DesignSummary",` `  ["NAME:Inputs"]` `],` `["NAME:DocTraces"]` `)`

VB Syntax	AddDocument <*data*>, <*traces*>

VB Example

In this example, the document names come from the UserDefinedDocument folder in the syslib, userlib, and personallib folders. This creates a document and places it in the Documents folder under results.

```
oModule.AddDocument _
Array("NAME:Design Summary", _
"", "SysLib", "DesignSummary", _
Array("NAME:Inputs")), _
Array("NAME:DocTraces")
```

The following example is explained in Explication of a Sample UDD Script.

```
oModule.AddDocument Array("NAME:Test Report1", "Test Report", "SysLib", _
"Examples/TestUDDInputs", Array("NAME:Inputs", Array("NAME:DLMetrics", "Solution", _
"Data Line Metrics", -1, -1), Array("NAME:DQ0", "Trace", "DQ0", -1, -1), _
Array("NAME:DQS", "Trace", "DQS", -1, -1), _
Array("NAME:Name", "Text", "User Name", Array("Sita Ramesh")), _
Array("NAME:Summary", "Bool", "Display Summary", Array(true)), _
Array("NAME:Version", "Number", "Script Version"))), _
Array("NAME:DocTraces", Array("NAME:DLMetrics", _
Array("User Defined", "", "DDR3 AC-Timing 4-DQ1", Array("Context:=", ""), _
Array("Index:=", Array("All"), "Trise:=", Array( "Nominal"), "Tfall:=", _
Array("Nominal"), "Pulse_Width:=", Array("Nominal"), _
"Data_Rate:=", Array( "Nominal"),"Length:=", Array("Nominal")), _
Array("Probe Component:=", Array("")), Array())), _
Array("NAME:DQ0", Array( _
```

	"Standard", "DQ0", "NexximTransient", Array("NAME:Context", "SimValueContext:=", Array(_ 1, 0, 2, 0, false, false, -1, 1, 0, 1, 1, "", 0, 0, _ "DE", false, "0", "DP", false, "20000000", "DT", false, "0.001", "WE", _ false, "100ns", "WM", false, "100ns", "WN", false, "0ps", "WS", false, "0ps")), _ Array("Time:=", Array("All"), "Trise:=", Array("Nominal"), "Tfall:=", Array("Nominal"), _ "Pulse_Width:=", Array("Nominal"), "Data_Rate:=", Array("Nominal"), _ "Length:=", Array("Nominal")), Array("Probe Component:=", Array("DQ0")), Array()))

DeleteAllDocuments

Deletes all documents in the object.

UI Access	Right click on **Documents** in the Project Manager and click **Delete All Documents**.
Parameters	None.
Return Value	None.

Python Syntax	DeleteAllDocuments()
Python Example	`oModule.DeleteAllDocuments()`

VB Syntax	DeleteAllDocuments

VB Example	`oModule.DeleteAllDocuments`

DeleteDocument

Deletes a specified document.

UI Access	Right click on the created document in the Project Manager under **Results** > **Documents** and click **Delete**.
Parameters	Name | Type | Description <name> | String | Name of the document to be deleted.
Return Value	None.

Python Syntax	DeleteDocument(<*names*>)
Python Example	`oModule.DeleteDocument("Project Design Summary")`

VB Syntax	DeleteDocument <*names*>
VB 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.

UI Access	Right click on the created document in the Project Manager under **Results** > **Documents** and click **Modify document**.

Parameters	<table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><originalName></td><td>String</td><td>Name of the original document</td></tr><tr><td><modifiedData></td><td>Array</td><td>New data to add to or modify the document</td></tr><tr><td><modifiedraces></td><td>Array</td><td>Trace data for the inputs of the document</td></tr></table>
Return Value	None.

Python Syntax	EditDocument(<*name*>,<*data*>,<*traces*>)
Python Example	`oModule.EditDocument("Design Summary",` `[` `  "NAME:Design Summary",` `  "",` `  "SysLib",` `  "DesignSummary",` `  ["NAME:Inputs"]` `],` `["NAME:DocTraces"]` `)`

VB Syntax	EditDocument <*name*>, <*data*>, <*traces*>

VB Example	`oModule.EditDocument "Design Summary", _` `Array("NAME:Design Summary", "", "SysLib", _` `"DesignSummary", Array("NAME:Inputs")), Array("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
- personallib/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("/")`

VB Syntax	GetDocumentDefinitionNames <*separator*>
VB 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()`

VB Syntax	GetDocumentNames
VB Example	`oModule.GetDocumentNames`

RenameDocument

Changes the name of a document.

UI Access	Right click on the created document in the Project Manager under **Results**> **Documents** and click **Rename**.
Parameters	Name | Type | Description <oldName> | String | Current name of the document <newName> | String | New name of the document

Return Value	None.

Python Syntax	RenameDocument(<*oldName*>, <*newName*>)
Python Example	`oModule.RenameDocument("Design Summary", "Project Design Summary")`

VB Syntax	RenameDocument <*oldName*>, <*newName*>
VB Example	`oModule.RenameDocument "Design Summary", "Project Design Summary"`

SaveHtmlDocumentAs

Saves a pre-existing HTML file to a different name and/or location.

UI Access	Right click on the created document in the Project Manager under **Results** > **Documents** and click **Save As** > **HTML**.
Parameters	Name / Type / Description <*name*> / String / Name of the document to be saved. <*saveTo*> / String / File path to save the document to.
Return Value	none

Python Syntax	SaveHtmlDocumentAs(<*name*>, <*saveTo*>)
Python Example	`oModule.SaveHtmlDocumentAs("Design Summary 1", "DS1.html")`

VB Syntax	SaveHtmlDocumentAs <*name*> <*saveTo*>
VB Example	`oModule.SaveHtmlDocumentAs "Design Summary 1" "DS1.html"`

SavePdfDocumentAs

Saves a pre-existing PDF file to a different name and/or location.

UI Access	Right click on the created document in the Project Manager under **Results** > **Documents** and click **Save As** > **PDF**.
Parameters	Name / Type / Description <*name*> / String / Name of the document to be saved. <*saveTo*> / String / File path to save the document at.
Return Value	None.

Python Syntax	SavePdfDocumentAs(<*name*>, <*saveTo*>)
Python Example	`oModule.SavePdfDocumentAs("Design Summary 1", "DS1.pdf")`

VB Syntax	SavePdfDocumentAs <*name*>, <*saveTo*>
VB Example	`oModule.SavePdfDocumentAs "Design Summary 1", "DS1.pdf"`

UpdateAllDocuments

Refreshes the contents of all created documents. This action is made on the folder rather than the individual document.

UI Access	Right click on **Results** > **Documents** in the Project Manager and click **Update All Documents**.
Parameters	None.
Return Value	None.

Python Syntax	UpdateAllDocuments()
Python Example	`oModule.UpdateAllDocuments()`

VB Syntax	UpdateAllDocuments
VB 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")`

VB Syntax	UpdateDocument <*name*>
VB 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	<table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><*name*></td><td>String</td><td>Name of the document to be viewed as a HTML</td></tr></table>
Return Value	None

Python Syntax	ViewHtmlDocument(<*name*>)
Python Example	`oModule.ViewHtmlDocument("Design Summary 1")`

VB Syntax	ViewHtmlDocument <*name*>
VB Example	`oModule.ViewHtmlDocument "Design Summary 1"`

ViewPdfDocument

Displays a pre-existing document as a PDF file.

UI Access	Right click on the created document in the Project Manager under **Results** > **Documents** and click **View PDF Document**.
Parameters	Name | Type | Description *<name>* | String | Name of the document to be viewed as a PDF
Return Value	None.

Python Syntax	ViewPdfDocument(*<name>*)
Python Example	`oModule.ViewPdfDocument("Design Summary 1")`

VB Syntax	ViewPdfDocument *<name>*
VB Example	`oModule.ViewPdfDocument "Design Summary 1"`

Explication of a Sample UDD Script

This VB script defines a document. It is a portion of one of the UDD example VB scripts.

```
Array("NAME:Test Report",              ' Name of the document
"Test Report",                         ' Description of the document
"SysLib",                              ' Location of the python script: Syslib, Userlib, PeronalLib, etc.
"TestUDDReport",                       ' Relative path of the script in the UserDefinedDocuments folder

' This array is the start of the input definition.
```

```
Array("NAME:Inputs",                    ' Document Inputs keyword

' This array contains the Solution input.
Array("NAME:DLMetrics",                 ' Input name
"Solution",                             ' Solution Input Type
"Data Line Metrics",                    ' Input Description
-1,                                     ' Solution ID
-1),                                    ' Report ID

' This array contains the trace input.
Array("NAME:DQ0",                       ' Input name
"Trace",                                ' Trace Input Type
"DQ0",                                  ' Input Description
-1,                                     ' Solution ID
-1),                                    ' Report ID

' This array contains the text input.
Array("NAME:Name",                      ' Input name
"Text",                                 ' Text Input Type
"User Name",                            ' Input Description
Array("Sita Ramesh")),                  ' Default Value

' This array contains the Bool input.
Array("NAME:Summary",                   ' Input name
"Bool",                                 ' Boolean Input Type
"Display Summary",                      ' Input Description
Array(true)),                           ' Default Value

' This array contains the number input.
```

```
Array("NAME:Version",                ' Input name
"Number",                            ' Number Input Type
"Script Version",                    ' Input Description
Array(1021)))),                      ' Default Value

' This array contains trace selection for the solution and trace inputs.
Array("NAME:DocTraces",              ' Document traces keyword

' This array has input for "DLMetrics".
Array("NAME:DLMetrics",              ' Input name

' This array defines a trace similar to the UDO. This trace definition is a User defined solution
Array("User Defined", "", "DDR3 AC-Timing 4-DQ1", Array("Context:=", ""), Array("Index:=", Array
("All"), "Trise:=", Array("Nominal"), "Tfall:=", Array("Nominal"), "Pulse_Width:=", Array("Nom-
inal"), "Data_Rate:=", Array("Nominal"), "Length:=", Array("Nominal")), Array("Probe Com-
ponent:=", Array("")), Array())),

' This array is for input "DQ0".
Array("NAME:DQ0",

' This array defines a trace similar to the UDO. This trace definition is a Standard solution.
Array("Standard", "DQ0", "NexximTransient", Array("NAME:Context", "SimValueContext:=", Array(1,
0, 2, 0, false, false, -1, 1, 0, 1, 1, "", 0, 0, "DE", false, "0", "DP", _
false, "20000000", "DT", false, "0.001", "WE", false, "100ns", "WM", false, _
"100ns", "WN", false, "0ps", "WS", false, "0ps")), Array("Time:=", Array("All"), "Trise:=",
Array( _
"Nominal"), "Tfall:=", Array("Nominal"), "Pulse_Width:=", Array("Nominal"), "Data_Rate:=", Array
("Nominal"), "Length:=", Array("Nominal")), Array("Probe Component:=", Array( _
"DQ0")), Array())))
```

Example Python Script: Defining a Document

This script creates a user-defied solution and a document.

```
import ScriptEnv
ScriptEnv.Initialize("Ansoft.ElectronicsDesktop")
oDesktop.RestoreWindow()
oProject = oDesktop.SetActiveProject("BJT_Inverter")
oDesign = oProject.SetActiveDesign("Nexxim1")
oModule = oDesign.GetModule("UserDefinedSolutionModule")

oModule.CreateUserDefinedSolution("UDS Distance Trace Arithmetic Result1", "SysLib", "TraceArith-
metic/Distance Sweep Trace Arithmetic",
       [
              "Offset 1:=", "0",
              "Scale 1:=", "1",
              "Offset 2:=", "0",
              "Scale 2:=", "1",
              "Operation:=", "Add"
       ],
       [
              [
                     "Standard",
                     "probe1",
                     "Transient",
                     [
                            style="font-family: monospace;">"NAME:Context",
                            style="font-family: monospace;">"SimValueContext:=", [1,0,2,0,False,False,-
1,1,0,1,1,"",0,0,"DE",False,"0","DP",False,"500000000","DT",False,"0.001","NUMLEVELS",False,"0","-
WE",False,"10us","WM",False,"10us","WN",False,"0ns","WS",False,"0ns"]
                     ],
                     [
                            "Time:=", ["All"]
                     ],
                     [
                            "Probe Component:=", ["V(Port1)"]
```

```
                        ],
                        []
                ],
                [
                        "Standard",
                        "probe2",
                        "Transient",
                        [
                                "NAME:Context",
                                "SimValueContext:=", [1,0,2,0,False,False,-
1,1,0,1,1,"",0,0,"DE",False,"0","DP",False,"500000000","DT",False,"0.001","NUMLEVELS",False,"0",-
"WE",False,"10us","WM",False,"10us","WN",False,"0ns","WS",False,"0ns"]
                        ],
                        [
                                "Time:=", ["All"]
                        ],
                        [
                                "Probe Component:=", ["V(Port1)"]
                        ],
                        []
                ]
        ],
        [])
oModule = oDesign.GetModule("UserDefinedDocuments")
oModule.AddDocument(
        [
                "NAME:Test Report",
                "Test Report",
                "SysLib",
                "TestUDDInputs",
                [
                        "NAME:Inputs",
                        [
                                "NAME:UDS1",
```

```
                        "Solution",
                        "UDS Distance Trace Arithmetic Result1",
                        1000000,
                        0
                ],
                [
                        "NAME:UDS2",
                        "Solution",
                        "UDS Distance Trace Arithmetic Result2",
                        1000000,
                        2
                ]
        ]
],
[
        "NAME:DocTraces",
        [
                "NAME:UDS1",
                [
                        "User Defined",
                        "",
                        "UDS Distance Trace Arithmetic Result1",
                        [
                                "Context:=", ""
                        ],
                        [
                                "Distance:=", ["All"]
                        ],
                        [
                                "Probe Component:=", [""]
                        ],
                        []
```

```
                        ]
                ],
                [
                        "NAME:UDS2",
                        [
                                "User Defined",
                                "",
                                "UDS Distance Trace Arithmetic Result1",
                                [
                                        "Context:=", ""
                                ],
                                [
                                        "Distance:=", ["All"]
                                ],
                                [
                                        "Probe Component:=", [""]
                                ],
                        []
                ]
        ]
])
```

This page intentionally
left blank.

13 - User Defined Solutions Commands

User Defined Solution commands should be executed by the "UserDefinedSolutionModule" module.

```
Set oDesign = oProject.SetActiveDesign("TestDesign1")
Set oModule =
oDesign.GetModule("UserDefinedSolutionModule")
```

The list of commands is as follows:

CreateUserDefinedSolution

Creates a new user defined solution.

UI Access	Right-click on **Results** > **Create User Defined Solution**.		
Parameters	Name	Type	Description
	<SoluName>	String	Name of user defined solution.
	<LibType>	String	Indicates the library where the UDS plugin file is located. This parameter must be one of the following values: "SysLib", "UserLib", "PersonalLib".
	<RelativePath>	String	The path of the UDS plugin file relative to the "UserDefinedOutputs" sub-directory of the library specified by *<LibType>*.
	<PropList>	Array	Strings specify name-value pairs corresponding to the UDS properties specified in the plugin file.

			For example: Array("multiply_factor:=", "2.0", "component_name:=", "resistor1")
	<*ProbeSelections*>	Array	Name of the probe being specified. Note: this must match a probe name specified in the UDS plugin file.
	<DynamicProbes>	Array	Array of <ProbeSelection>'s, representing the probes that are used by dynamic-probes.
Return Value	String name of created user defined solution.		

Python Syntax	CreateUserDefinedSolution(<*SoluName*>, <*LibType*>, <*RelativePath*>, <*PropList*>, <*ProbeSelections*>, <*DynamicProbes*>)
Python Example	oModule.CreateUserDefinedSolution("ConstantTimestep1", "SysLib", "ConstantTimestep", [], [], [])

VB Syntax	CreateUserDefinedSolution <*SoluName*>, <*LibType*>, <*RelativePath*>, <*PropList*>, <*ProbeSelections*>, <*DynamicProbes*>
VB Example	oModule.CreateUserDefinedSolution "User Defined Solution 1", _ "SysLib", "Example", _ Array("multiplication_factor:=", "1.2"), _ Array(Array("Modal Solution Data", "Probe 1", "Setup1 : LastAdaptive", _

	Array(), Array("Freq:=", Array("All")), _ Array("Probe Component:=", Array("dB(S(1,1))")), Array()), _ Array("Modal Solution Data", "Probe 2", "Setup1 : LastAdaptive", Array(), _ Array("Freq:=", Array("All")), _ Array("Probe Component:=", Array("mag(S(1,1))")), Array())), _ Array(Array("Modal Solution Data", _ "Dynamic Probe 1", "Setup1 : LastAdaptive", Array(), _ Array("Freq:=", Array("All")), _ Array("Probe Component:=", Array("Freq")), Array()))

DeleteUserDefinedSolutions

Deletes one or more user defined solutions.

UI Access	**Delete** button from the **User Defined Solutions** dialog.
Parameters	Name | Type | Description <*SoluNames*> | Array | Array of strings containing names of User Defined Solutions to be deleted.
Return Value	None.

Python Syntax	DeleteUserDefinedSolutions(<*SoluNames*>)
Python Example	`oModule.DeleteUserDefinedSolutions(["Solution1", "Solution2"])`

VB Syntax	DeleteUserDefinedSolutions <*SoluNames*>
VB Example	`oModule.DeleteUserDefinedSolutions _` `Array("Solution1", "Solution2")`

EditUserDefinedSolution

Modifies an existing user defined solution.

UI Access	**Edit** button from the **User Defined Solutions** dialog box.
Parameters	See table below
Return Value	String name of update user defined solution.

Name	Type	Description
<*SoluName*>	String	Name of user defined solution to be edited.
<*NewSoluName*>	String	New name for the specified user defined solution.
<*LibType*>	String	Indicates the library where the UDS plugin file is located. This parameter must be one of the following values: "SysLib", "UserLib", "PersonalLib".
<*RelativePath*>	String	The path of the UDS plugin file relative to the "UserDefinedOutputs" sub-directory of the library specified by <*LibType*>.
<*PropList*>	Array	Strings specify name-value pairs corresponding to the UDS properties specified in the plugin file. `For example:` `Array("multiply_factor:=", "2.0", "component_name:=", "resistor1")`
<*ProbeSelections*>	Array	Name of the probe being specified. Note: this must match a probe name specified in the UDS plugin file.
<DynamicProbes>	Array	Array of <ProbeSelection>'s, representing the probes that are used by dynamic-probes.

Python Syntax	EditUserDefinedSolution(<*SoluName*>, <*NewSoluName*>, <*LibType*>, <*RelativePath*>, <*PropList*>, <*ProbeSelections*>, <*DynamicProbes*>)
Python Example	`oModule.EditUserDefinedSolution("ConstantTimestep1"` `"ConstantTimestep1After", "SysLib",` `"ConstantTimestep", [], [], [])`

VB Syntax	EditUserDefinedSolution <*SoluName*>, <*NewSoluName*>, <*LibType*>, <*RelativePath*>, <*PropList*>, <*ProbeSelections*>, <*DynamicProbes*>
VB Example	`oModule.CreateUserDefinedSolution "User Defined Solution 1", _` `"User Defined Solution 1", "SysLib", "Example", _` `Array("multiplication_factor:=", "1.2"), _` `Array(Array("Modal Solution Data",` `"Probe 1", "Setup1 : LastAdaptive", _` `Array(), Array("Freq:=", Array( "All")), _` `Array("Probe Component:=", Array("dB(S(1,1))")), Array()), _` `Array( "Modal Solution Data", "Probe 2",` `"Setup1 : LastAdaptive", Array(), _` `Array("Freq:=", Array( "All")), _` `Array("Probe Component:=", Array("mag(S(1,1))")), Array())), _` `Array(Array("Modal Solution Data", _` `"Dynamic Probe 1", "Setup1 : LastAdaptive", Array(), _`

	`Array("Freq:=", Array( "All")), _` `Array("Probe Component:=", Array("Freq")), Array()))`

GetUserDefinedSolutionNames

Retrieves user defined solution names.

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

Python Syntax	GetUserDefinedSolutionNames()
Python Example	`oModule.GetUserDefinedSolutionNames()`

VB Syntax	GetUserDefinedSolutionNames
VB Example	`oModule.GetUserDefinedSolutionNames`

GetUserDefinedSolutionProperties

Obtains properties for a specified user defined solution.

UI Access	N/A

Parameters	<table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><*SoluName*></td><td>String</td><td>Name of a specified user defined solution.</td></tr></table>
Return Value	Array of strings containing properties values.

Python Syntax	GetUserDefinedSolutionProperties(<*SoluName*>)
Python Example	`oModule.GetUserDefinedSolutionProperties("ConstantTimestep1")`

VB Syntax	GetUserDefinedSolutionProperties <*SoluName*>
VB Example	`oModule.GetUserDefinedSolutionProperties "ConstantTimestep1"`

This page intentionally
left blank.

14 - 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:

> **Warning: The following commands are preliminary. Their syntax may change. Be prepared to make changes to legacy scripts.**

> **Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.**

AddDiffPair

Specifies a differential pair from two terminal ports.

UI Access	From the **Network Data Explorer** tab, select **Differential Pairs** in the **NDE** ribbon to open the **Differential Pairs** window. Then select differential pairs from the **Pairs** group box and click **Add Pairs >>**.
Parameters	**Name** / **Type** / **Description** *<positiveTerminal>* / Integer / The portNumber of the positive terminal. *<negativeTerminal>* / Integer / The portNumber of the negative terminal. *<diffName>* / String / The new name of the differential pin (e.g., Diff1). *<commName>* / String / The new name of the common pin (e.g., Comm1). *<diffImpedance>* / Double / The differential pin characteristic impedance. *<commImpedance>* / Double / The common pin characteristic impedance. *<matched>* / Boolean / Specify whether to use matched pair. **Note:** The default value is **False**.
Return Value	Boolean.

Python Syntax	AddDiffPair()
Python Example	`success = oPostProc.AddDiffPair(2, 1, "Diff1", "Comm1", 100, 25)`

VB Syntax	AddDiffPair

VB Example	`Set 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	**Name** / **Type** / **Description** *<dataArray>* / Array / These network data objects are cascaded together to create the new network data.
Return Value	Network IDispatch. **Note:** Cascade returns **None** if the specified network data does not have compatible terminal ports defined.

Python Syntax	Cascade()
Python Example	`oNDE = oDesktop.GetTool("ndExplorer")` `oData2 = oNDE.Cascade([oData1, oData2, oData3])`

VB Syntax	Cascade
VB Example	`Set oNDE = oDesktop.GetTool("ndExplorer")` `Set oData2 = oNDE.Cascade Array(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()`

VB Syntax	ClearDiffPairs
VB Example	`Set 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()`

VB Syntax	Clone
VB Example	`Set oNDE = oDesktop.GetTool("ndExplorer")` `Set 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()`

VB Syntax	Close
VB Example	`Set 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.

UI Access	From the **Network Data Explorer** tab, select **Transforms** in the **NDE** ribbon. Then select **Combine** from the drop-down menu.
Parameters	**Name** / **Type** / **Description** *<dataArray>* / Array / These network data objects are combined to create the new network data. *<freqTol>* / Double / If abs(f1 – f2) < freqTol, f1 and f2 are considered the same frequency. **Note:** The default value is **100**.
Return Value	Network IDispatch. **Note:** Combine returns **None** if the network data does not have the same number of ports.

Python Syntax	Combine()
Python Example	`oNDE = oDesktop.GetTool("ndExplorer")`

	`oData4 = oNDE.Combine([oData1, oData2, oData3], 1000)`

VB Syntax	Combine
VB Example	`Set oNDE = oDesktop.GetTool("ndExplorer")` `Set oData4 = oNDE.Combine Array(oData1, oData2, oData3), 1000`

Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.

Deembed (SPISim)

Creates a new network data, dembedding the frontData and the backData from the front and back of the originalData.

Note: Deembed opens and interacts with **SPISim** to complete.

UI Access	From the **Network Data Explorer** tab, select **Transforms** in the **NDE** ribbon. Then select **Deembed** from the drop-down menu to open the **Deembed...** window, and choose **Given 3 S-params in order of: Total, Front, Back** from the **Settings** group box.
Parameters	**Name** / **Type** / **Description** *<originalData>* / Object / Original network data that is deembeded. *<frontData>* / Object / Network data that is deembeded from the front of the original. *<backData>* / Object / Network data that is deembeded from the back of the original.
Return Value	Network IDispatch.

	Note: Deembed returns **None** if the specified network data does not have compatible terminal ports defined.

Python Syntax	Deembed()
Python Example	`oNDE = oDesktop.GetTool("ndExplorer")` `oData2 = oNDE.Deembed(oData1, oDataFront, oDataBack)`

VB Syntax	Deembed
VB Example	`Set oNDE = oDesktop.GetTool("ndExplorer")` `Set 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.

UI Access	From the **Network Data Explorer** tab, select **Transforms** in the **NDE** ribbon. Then select **Deembed** from the drop-down menu to open the **Deembed...** window, and choose **Given 3 S-params in order of: Total, Back** from the **Settings** group box.

Parameters	<table><tr><th>Name</th><th>Type</th><th>Description</th></tr><tr><td><originalData></td><td>Object</td><td>Original network data that is deembeded.</td></tr><tr><td><backData></td><td>Object</td><td>Network data that is deembeded from the back of the original.</td></tr></table>
Return Value	Network IDispatch. **Note:** DeembedBack returns **None** if the specified network data does not have compatible terminal ports defined.

Python Syntax	DeembedBack()
Python Example	`oNDE = oDesktop.GetTool("ndExplorer")` `oData2 = oNDE.DeembedBack(oData1, oDataBack)`

VB Syntax	DeembedBack
VB Example	`Set oNDE = oDesktop.GetTool("ndExplorer")` `Set 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	**Name** / **Type** / **Description** *<originalData>* / Object / Original network data that is deembeded. *<frontData>* / Object / Network Data that is deembeded from the front of the original.
Return Value	Network IDispatch. **Note:** DeembedFront returns **None** if the specified network data does not have compatible terminal ports defined.

Python Syntax	DeembedFront()
Python Example	`oNDE = oDesktop.GetTool("ndExplorer")` `oData2 = oNDE.DeembedFront(oData1, oDataFront)`

VB Syntax	DeembedFront
VB Example	`Set oNDE = oDesktop.GetTool("ndExplorer")`

	`Set 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()`

VB Syntax	DisableDiffPairs
VB Example	`Set 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()`

VB Syntax	EnableDiffPairs
VB Example	`Set oNDE = oDesktop.GetTool("ndExplorer")` `Set 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).

<table>
<tr><td>UI Access</td><td>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.</td></tr>
<tr><td>Parameters</td><td>
<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td><filePath></td><td>String</td><td>The full path to the file.</td></tr>
<tr><td><matrixType></td><td>Integer</td><td>One of the following:
0 – S-Parameters
1 – Y-Parameters
2 – Z-Parameters
3 – Gamma
4 – Impedance
Note: The default value is 0.</td></tr>
<tr><td><formalType></td><td>Integer</td><td>One of the following:
0 – Magnitude/Angle (degrees)
1 – Real/Imaginary
2 – dB/Angle (degrees)</td></tr>
</table>
</td></tr>
</table>

			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, [])`

VB Syntax	ExportCitiFile
VB Example	`Set oNDE = oDesktop.GetTool("ndExplorer")` `Set success = oData.ExportCitiFile "D:\folder\ne133.cit", 0, 1, 12, Array()`

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>UI Access</td><td>From the Network Data Explorer tab, select File > Save As to open a Save As explorer window. Then select MATLAB (*.m) from the Save as type: drop-down menu.</td></tr>
<tr><td>Parameters</td><td>
<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td><filePath></td><td>String</td><td>The full path to the file.</td></tr>
<tr><td><matrixType></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**.
	<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	ExportMatlab()
Python	`oNDE = oDesktop.GetTool("ndExplorer")`

Example	`success = oData.ExportMatlab("D:\folder\ne133.m", 0, 1, 12, [])`

VB Syntax	ExportMatlab
VB Example	`Set oNDE = oDesktop.GetTool "ndExplorer"` `Set success = oData.ExportMatlab "D:\folder\ne133.m", 0, 1, 12, Array()`

ExportNMFData

*Use:*Exports s-parameters in neutral file format.

*Command:*In the **matrix** tab of the **Solution** dialog box, click **Export->S-Parameter**. Then select the **Neutral Model** file format.

Syntax: ExportNetworkData <SolutionName> <FileName> <ReduceMatrix><Reference Impedance> <FrequencyArray> <DesignVariation> <Format> <Length> <PassNumber>

*Return Value:*None

*Parameters:*SolutionName>

    Type: <String>

    Format: <SetupName>:<SolutionName>

    Solution that is exported.

<FileName>

    Type:<String>

    The name of the file. The file extension will determine the file format.

<ReduceMatrix>

    Type: <string>

```
     Either "Original" or one of the reduce matrix setup name
<Reference Impedance>
     Type: <Double>
     Reference impedance
<FrequencyArray>
     Type: <Array of doubles>
     Value: Array(<double>,<double>….)
     Frequency points in the sweep that is exported.
<DesignVariation>
   Type: <String>
   Design variation at which the solution is exported.
<Format>
     Type: <String>
     Values: "MagPhase", "RealImag", "DbPhase".
     The format in which the sparameters will be exported.
<Length>
     Type:<String>
     Length for exporting sparameters.
<PassNumber>
     Type:<integer>
```

```
    Pass number.
```

VB Example:

```
oDesign.ExportNMFData "Setup7 : LastAdaptive", "C:/temp/neu.nmf", "Original", _ 50, Array
(10000000, 20000000, 30000000, 40000000, 50000000, 60000000, 70000000, _ 80000000, 90000000,
100000000), "", "MagPhase","7meter"
```

Python Syntax	ExportNMFData("", # Empty string. 1, #The number 1. "Name", # This is the full path of the file from which the solution is loaded "ExportFile", # full path of file to export to ["NAME:Frequencies"], #optional, if none defined all frequencies are used ["NAME:NMFOptions", #Export NMF options object "DataTypes:=", ["S"], #DataTypes can be "S", "Y","Z", "G", and "Z0", for S , Y, Z matrix, Gamma and Z0 (zero) "DisplayFormat:=", "MA",

```
#DisplayFormat "MA", "RI", "DB"
        "FileType:=", "",
# Export File Type
2 - Spreadsheet(*.tab)
3 - Touchstone(*.sNp)
4 - Citifile(*.cit)
6 - Neutral format(*.nmf)
7 - Matlab format(*.m)
        "Renormalize:=", false,
#Renormalize true/false
        "RefImpedance:=",50,
# Reference Impedance
        "Precision:=", 8,
# Number of digits Precision
        "Variables:=", ["FF", "cap", "Rs"]
# Array of variables
        "Variations:=", ["", "", ""]
# Array of variations to export solutions for
        ["NAME:ConstantVars"]
#Array of variables that are constant, can be empty
```

	["NAME: DependentVars"] #Array of variables that are dependent, can be empty "MatrixSize:=", 2, #Matrix size, optional (used in nmf file header) "CreateNPortModel:=", true #Create a model based on the exported file true/false])
Python Example	oTool = oDesktop.GetTool("NdExplorer") oTool.ExportNetworkData("", True, "C:/.../100OHM.S2P", "$SYSLIB/Test.s2p", "", ["NAME:Frequencies", 500000000, 1000000000, 10000000000, 25000000000, 50000000000, 75000000000, 100000000000],

```
        [
                "NAME:TSOptions",
                "DataTypes:="           , ["S"],
                "DisplayFormat:="       , "DB",
                "FileType:="            , 3,
                "Renormalize:="         , True,
                "RefImpedance:="        , 50,
                "Precision:="           , 6,
                "UseMultipleCores:="    , False,
                "NumberOfCores:="       , 1,
                "Comments:="            , True,
                "Noise:="               , False
        ])
```

ExportNetworkData

Exports matrix solution data to a file.

UI Access	N/A		
Parameters	Name	Type	Description
	<*DesignVariationKey*>	String	Design variation key. Pass empty string for the current nominal variation.
	<*SolnSelectionArray*>	Array	Array of selected solutions.

		`Array(<SolnSelector>, <SolnSelector>, ...)` If more than one array entry, this indicates a combined Interpolating sweep.
<SolnSelector>	String	Solution setup name and solution name, separated by a colon.
<FileFormat>	Integer	File format value. 2 : Tab delimited spreadsheet format (.tab) 3 : Touchstone (.sNp) 4 : CitiFile (.cit) 7 : Matlab (.m) 8 : Terminal Z0 spreadsheet
<OutFile>	String	Full path to the file to write out.
<FreqsArray>	Array	The frequencies to export. The <FreqsArray> argument contains a vector (e.g. "1GHz", "2GHz", ...) to use, or "all". To export all frequencies, use Array("all"). If no frequencies are specified, all frequencies are used.
<DoRenorm>	Boolean	Specifies whether to renormalize the data before export.
<RenormImped>	Double	Real impedance value in ohms, for renormalization. Required in syntax, but ignored if DoRenorm is false.
<DataType>	String	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	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	`oModule.ExportNetworkData("", ["Setup1:Sweep1"], 3, "C://Documents/package_HFSSDesign1.s4p",` `['all'], True, 50, "S", -1, 0, 15, True, True, True`

VB Syntax	ExportNetworkData <*DesignVariationKey*>, <*SolnSelectionArray*>, <*SolnSelector*>, <*FileFormat*>, <*OutFile*>, <*FreqsArray*>, <*DoRenorm*>, <*RenormImped*>, [Optional <*DataType*>], [Optional <*pass*>], [Optional <*ComplexFormat*>], [Optional <*Precision*>], [Optional <*UseExportFreqs*>], [Optional <*IncludeGammaComments*>], [Optional <*SupportNonStdExport*>]
VB Example	`oModule.ExportNetworkData "width='2in'", _` `  Array("Setup1:Sweep1"),2, "c:\mydir\out.tab", _` `  Array("all"), false, 0`

	oModule.ExportNetworkData "width='2in'", _ Array("Setup1:Sweep1", "Setup1:Sweep2"), 3, _ "c:\mydir\out.s2p", Array(1.0e9, 1.5e9, 2.0e9), _ true, 50.0

Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.

ExportSpreadsheet

Writes the S-parameters to disk in spreadsheet format (*.tab text file).

UI Access	From the **Network Data Explorer** tab, select **File** > **Save As** to open a **Save As** explorer window. Then select **Data Table (spreadsheet) (*.tab)** from the **Save as type:** drop-down menu.
Parameters	See table below

Name	Type	Description
<filePath>	String	The full path to the file.
<matrixType>	Integer	One of the following: • 0 – S-Parameters • 1 – Y-Parameters • 2 – Z-Parameters • 3 – Gamma • 4 – Impedance **Note:** The default value is **0**.
<formalType>	Integer	One of the following: • 0 – Magnitude/Angle (degrees)

			• 1 – Real/Imaginary • 2 – dB/Angle (degrees) **Note:** The default value is **0**.
	<precision>	Integer	The number of significant figures. **Note:** The default value is **10**.
	<frequencies>	Array of Doubles	A subset of the calculated frequencies. Any array values that don't have calculated data will be ignored. An empty array will cause all frequencies to be written. **Note:** The default value is an empty array.
	<renormalize>	Boolean	If **True**, will renormalize the data to the value of renormImpedance before writing. **Note:** The default value is **False**.
	<renormImpedance>	Double	Data will be renormalized to this impedance before writing only if renormalize is **True**. **Note:** The default value is **50 ohms**.
Return Value	Boolean True if file was successfully written; else False.		

Python Syn-	ExportSpreadsheet()

tax	
Python Example	`oNDE = oDesktop.GetTool("ndExplorer")` `oData = oNDE.Open(<model path>)` `success = oData.ExportSpreadsheet("D:\folder\ne133.tab", 0, 1, 12, [], True, 23)`

VB Syntax	ExportSpreadsheet
VB Example	`Set oData=oDesktop.GetTool("ndExplorer")` `success = oData.ExportSpreadsheet "D:\folder\ne133.tab", 0, 1, 12, Array(), 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).

UI Access	From the **Network Data Explorer** tab, select **File** > **Save As** to open a **Save As** explorer window. Then select **Touchstone Format 1.0 (*.s*n*p)** from the **Save as type:** drop-down menu.

Parameters

Name	Type	Description
<filePath>	String	The full path to the file.
<matrixType>	Integer	One of the following: • 0 – S-Parameters • 1 – Y-Parameters • 2 – Z-Parameters

		• 3 – Gamma • 4 – Impedance **Note:** The default value is **0**.
<formalType>	Integer	One of the following: • 0 – Magnitude/Angle (degrees) • 1 – Real/Imaginary • 2 – dB/Angle (degrees) **Note:** The default value is **0**.
<precision>	Integer	The number of significant figures. **Note:** The default value is **10**.
<frequencies>	Array of Doubles	A subset of the calculated frequencies. Any array values that don't have calculated data will be ignored. An empty array will cause all frequencies to be written. **Note:** The default value is an empty array.
<renormalize>	Boolean	If **True**, will renormalize the data to the value of renormImpedance before writing. **Note:** The default value is **False**.
<renormImpedance>	Double	Data will be renormalized to this impedance before writing only if renormalize is **True**.

			Note: The default value is **50 ohms**.
	<freqUnits>	String	One of the following: "Hz", "KHz", "MHz", "GHz". **Note:** The default value is **"Hz"**.
Return Value	Boolean True if file was successfully written; else False.		

Python Syntax	ExportTouchstone()
Python Example	`oNDE = oDesktop.GetTool("ndExplorer")` `success = oData.ExportTouchstone("D:\folder\ne133.s2p", 0, 1, 12, [], True, 23, "GHz")`

VB Syntax	ExportTouchstone
VB Example	`Set oNDE = oDesktop.GetTool("ndExplorer")` `success = oData.ExportTouchstone "D:\folder\ne133.s2p", 0, 1, 12, Array(), 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>UI Access</td><td>From the Network Data Explorer tab, select File > Save As to open a Save As explorer window. Then select Touchstone 2 Format (*.ts) from the Save as type: drop-down menu.</td></tr>
<tr><td>Parameters</td><td>
<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td><filePath></td><td>String</td><td>The full path to the file.</td></tr>
<tr><td><matrixType></td><td>Integer</td><td>One of the following:
0 – S-Parameters
1 – Y-Parameters
2 – Z-Parameters
3 – Gamma
4 – Impedance
Note: The default value is 0.</td></tr>
<tr><td><formalType></td><td>Integer</td><td>One of the following:
0 – Magnitude/Angle (degrees)
1 – Real/Imaginary
2 – dB/Angle (degrees)
Note: The default value is 0.</td></tr>
<tr><td><precision></td><td>Integer</td><td>The number of significant figures.</td></tr>
</table>
</td></tr>
</table>

			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")`

VB Syntax	ExportTouchstone2
VB Example	`Set oNDE = oDesktop.GetTool("ndExplorer")` `Set success = oData.ExportTouchstone2 "D:\folder\ne133.ts", 0, 1, 12, Array(), 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	**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*).
Return Value	Network IDispatch.

Python Syntax	Extract()
Python Example	`oNDE = oDesktop.GetTool("ndExplorer")` `oData2 = oNDE.Extract(oData1, [3, 2])`

VB Syntax	Extract
VB Example	`Set oNDE = oDesktop.GetTool "ndExplorer"` `Set oData2 = oNDE.Extract oData1, Array(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()`

VB Syntax	GetFrequencies
VB Example	`Set oNDE = oDesktop.GetTool("ndExplorer")` `Set 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()`

VB Syntax	GetFrequencyCount
VB Example	`Set oData = oDesktop.GetTool("ndExplorer")` `Set 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()`

VB Syntax	GetName
VB Example	`Set oNDE = oDesktop.GetTool("ndExplorer")` `strName = oNDE.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()`

VB Syntax	GetPortCount
VB Example	`Set 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.

UI Access	From the **Network Data Explorer** tab, ensure the **Full Port Names** check box is activated. Port numbers will be represented in the object portNames.
Parameters	**Name** / **Type** / **Description** *<portName>* / String / The designation of the port number.
Return Value	Integer.

Python Syntax	GetPortNumber()
Python Example	`oNDE = oDesktop.GetTool("ndExplorer")` `index = oPostProc.GetPortNumber("Input35")`

VB Syntax	GetPortNumber
VB Example	`Set 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()`

VB Syntax	GetPostProcSettings

VB Example	`Set oNDE = oDesktop.GetTool("ndExplorer")` `Set 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.

UI Access	None.
Parameters	**Name** / **Type** / **Description** *<solutionID>* / Integer / ID of the solution (obtained with LoadSolution). *<variation>* / String / The variation key.
Return Value	Network IDispatch.

Python Syntax	GetSolutionVariation()
Python Example	`oNDE = oDesktop.GetTool("ndExplorer")` `oData = oNDE.GetSolutionVariation("O, "offset='0.1'")`

VB Syntax	GetSolutionVariation
VB Example	`Set oNDE = oDesktop.GetTool("ndExplorer")` `oData = oNDE.GetSolutionVariation "O, "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()`

VB Syntax	GetVariation
VB Example	`Set oNDE = oDesktop.GetTool("ndExplorer")` `Set 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.

UI Access	None.

Parameters

Name	**Type**	**Description**
<NetworkData>	Object	The second network data to compare against.
<matrixType>	Integer	One of the following: • 0 – S-Parameters • 1 – Y-Parameters • 2 – Z-Parameters • 3 – Gamma • 4 – Impedance **Note:** The default value is **0**.
<comparePortNames>	Boolean	If **True**, port names must be identical. **Note:** The default value is **True**.
<compareNoise>	Boolean	If **True**, and both network data have noise data, the comparison will fail unless the noise values also match. **Note:** The default value is **10**.

	<relativeTolerance>	Double	If the absoluteTolerance test fails, then 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)`

VB Syntax	HasSameData
VB Example	`Set oNDE = oDesktop.GetTool("ndExplorer")` `Set 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.

UI Access	After analyzing a design, from the **Project Manager** window, expand the **Project Tree** and Analysis folders. Then right-click on the completed analysis icon and select **Network Data Explorer** to open the solution in the **Network Data Explorer** tab.
Parameters	**Name** / **Type** / **Description** *<projectName>* / String / Full path to the Electronics Desktop file. *<designName>* / String / Name of the design. *<solutionName>* / String / Name of the solution.
Return Value	Integer ID (identifying the solution); < 0 if the solution is not loaded.

Python Syntax	LoadSolution()
Python Example	`oNDE = oDesktop.GetTool("ndExplorer")` `id = oNDE.LoadSolution("D:\folder\Tee.aedt", "HFSS_Test", "Setup1:Sweep1")`

VB Syntax	LoadSolution
VB Example	`Set 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.

UI Access	From the **Network Data Explorer** tab, select **Open** in the **NDE** ribbon, or select **File** > **Open** to open an explorer window. Then navigate to the required S-parameter file.
Parameters	**Name** / **Type** / **Description** *<fileName>* / String / Full path to the file containing S-parameters.
Return Value	Network IDispatch.

Python Syntax	Open()
Python Example	`oNDE = oDesktop.GetTool("ndExplorer")` `oData = oNDE.Open("D:\folder\test.s2p")`

VB Syntax	Open
VB Example	`Set oNDE = oDesktop.GetTool("ndExplorer")` `Set 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	**Name** / **Type** / **Description** *<oData>* / Object / Data from this object is copied to a new network data and the copy is transformed. *<portNames>* / Array of Strings / The new names (one for each port).
Return Value	Network IDispatch.

Python Syntax	Rename()
Python Example	`oNDE = oDesktop.GetTool("ndExplorer")` `oData2 = oNDE.Rename(oData1, ["input", "control", "output"])`

VB Syntax	Rename

VB Example	`Set oNDE = oDesktop.GetTool "ndExplorer"` `Set oData2 = oNDE.Rename oData1, Array("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	**Name** / **Type** / **Description** *<oData>* / Object / Data from this object is copied to a new network data and the copy is transformed. *<portImpedances>* / Array of Doubles / The new impedance values to be applied to each port.
Return Value	Network IDispatch. **Note:** Renormalize returns **None** if there is not an impedance value for each port.

Python Syntax	Renormalize()
Python Example	`oNDE = oDesktop.GetTool("ndExplorer")` `oData2 = oNDE.Renormalize(oData1, [3, 2])`

VB Syntax	Renormalize
VB Example	`Set oNDE = oDesktop.GetTool("ndExplorer")` `oData2 = oNDE.Renormalize oData1, Array(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).

UI Access	From the **Network Data Explorer** tab, select **Edit Ports** in the **NDE** ribbon to open the **Port properties** window. **Click+drag** the terminal port rows to reorder them. Make changes, as necessary, then click **OK**.
Parameters	**Name** / **Type** / **Description** *<newOrder>* / Array of Integers / Must have one entry for each port; all numbers 1 to *n* must be present.
Return Value	None.

Python Syntax	Reorder()
Python Example	`oNDE = oDesktop.GetTool("ndExplorer")` `oPostProc.Reorder([3, 2, 1])`

VB Syntax	Reorder
VB Example	`Set oNDE = oDesktop.GetTool("ndExplorer")` `oPostProc.Reorder Array(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

Name	Type	Description
<oData>	Object	Data from this object is copied to a new network data and the copy is transformed.

	<portNumbers>	Array of Integers	The port numbers in new order (integers between 1 and *n*).
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])`

VB Syntax	Reorder
VB Example	`Set oNDE = oDesktop.GetTool("ndExplorer")` `Set oData2 = oNDE.Reorder Array(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.

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()`

VB Syntax	Reset
VB Example	`Set 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	**Name** / **Type** / **Description** *<impedances>* / Array of Doubles or String / Must have one entry for each port or a single complex (or real) value which will be applied to all ports.
Return Value	None.

Python Syntax	SetAllPortImpedances()
Python Example	`oNDE = oDesktop.GetTool("ndExplorer")` `oPostProc.SetAllPortImpedances([23, "2+3i", 50])` **-or-** `oData.SetAllPortImpedances(50)`

VB Syntax	SetAllPortImpedances
VB Example	`Set oData=oDesktop.GetTool("ndExplorer")` `oPostProc.SetAllPortImpedances Array(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** window. Make changes, then click **OK**.
Parameters	**Name** / **Type** / **Description** *<portNumber>* / Integer / The port number of the changed terminal port (integers between 1 and **n**) . *<distance>* / String / Impedance with units (e.g., “20mm”); if no units, meters are assumed
Return Value	None.

Python Syntax	SetPortDeembedDistance()
Python Example	`oNDE = oDesktop.GetTool("ndExplorer")` `oPostProc.SetPortDeembedDistance(2, "20mm")`

VB Syntax	SetPortDeembedDistance
VB Example	`Set 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	**Name** / **Type** / **Description** *<portNumber>* / Integer / The port number of the changed terminal port (integers between 1 and *n*). *<impedance>* / Double or String / Double if impedance value is real; string format (e.g., 24+10i) for complex impedance.
Return Value	None.

Python Syntax	SetPortImpedance()
Python Example	`oNDE = oDesktop.GetTool("ndExplorer")` `oPostProc.SetPortImpedance(2, "25+3i")`

VB Syntax	SetPortImpedance
VB Example	`Set 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.

UI Access	None.
Parameters	**Name** / **Type** / **Description** *<PostProcSettings>* / Object / The settings that will be applied to the data.
Return Value	None.

Python Syntax	SetPostProcSettings()
Python Example	`oNDE = oDesktop.GetTool("ndExplorer")` `oData.SetPostProcSettings(oPostProc)`

VB Syntax	SetPostProcSettings

VB Example	`Set 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.

UI Access	From the **Network Data Explorer** tab, select **Transforms** in the **NDE** ribbon. Then select **Smooth** from the drop-down menu.
Parameters	**Name** / **Type** / **Description** *<oData>* / Object / Data from this object is copied to a new network data and the copy is transformed. *<order>* / Integer / The number of adjacent points that are smoothed. **Note:** The default value is **10**. *<enforceCausality>* / Boolean / Casual data is calculated before smoothing. **Note:** The default value is **False**.
Return Value	Network IDispatch.

Python Syntax	Smooth()
Python Example	`oNDE = oDesktop.GetTool("ndExplorer")`

	`oData2 = oNDE.Smooth(oData1, 3, True)`

VB Syntax	Smooth
VB Example	`Set oNDE = oDesktop.GetTool("ndExplorer")` `Set 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	**Name** / **Type** / **Description** *<oData>* / Object / Data from this object is copied to a new network data and the copy is transformed. *<factor>* / Double / The transmission line lengths are multiplied by this number.
Return Value	Network IDispatch.

	Note: Stretch returns **None** if the network data argument does not represent a matrix of transmission lines.

Python Syntax	Stretch()
Python Example	`oNDE = oDesktop.GetTool("ndExplorer")` `oData2 = oNDE.Stretch(oData1, 3.1)`

VB Syntax	Stretch
VB Example	`Set oNDE = oDesktop.GetTool("ndExplorer")` `Set 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.

UI Access	From the **Network Data Explorer** tab, select **Transforms** in the **NDE** ribbon. Then select **Terminate** from the drop-down menu.
Parameters	**Name** / **Type** / **Description** *<oData>* / Object / Data from this object is copied to a new network data and the copy is transformed.

	<portNumbers>	Array of Integers	The port numbers that are terminated (integers between 1 and *n*).
	<termImpedances>	Array of Complex Numbers	The impedance values used to terminate the specified ports.
Return Value	Network IDispatch. **Note:** Terminate returns **None** if the port numbers are not valid or there are no impedance value for the port numbers.		

Python Syntax	Terminate()
Python Example	`oNDE = oDesktop.GetTool("ndExplorer")` `oData2 = oNDE.Terminate(oData1, [2, 3], [23, "10+2i"])`

VB Syntax	Terminate
VB Example	`Set oNDE = oDesktop.GetTool("ndExplorer")` `Set oData2 = oNDE.Terminate oData1, Array(2, 3), Array(23, "10+2i")`

15 - 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

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

The topics for this section include:

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()`

VB Syntax	GetComponentName()
VB Example	`name = CompInstance.GetComponentName()`

GetEditor [Component Instance]

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

Command: None

Syntax: GetEditor(<editorName>)

Return Value: Returns interface to editor requested.

VB Example:

```
Set oLayout2 = CompInstance.GetEditor("Layout");
Returns the interface to the layout containing a selected component.
This interface can be used to call Layout Scripting functions.
```

Python Syntax	GetEditor(<editorName>)
Python Example	`oLayout2 = CompInstance.GetEditor("Layout")`

GetInstanceID [Component Instance]

Use: Returns the instanceID of the CompInstance.

Command: None

Syntax: GetInstanceID()

Return Value: String

VB Example: `id = CompInstance.GetInstanceID();`

```
Returns id of compInstance (e.g. 7) and stores it in "id".
Note that this is not the same number as the one used in the RefDes.
```

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

VB Example: `name = CompInstance.GetInstanceName();`

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.

Command: None

Syntax: GetParentDesign()

Return Value: Returns interface to design.

Example: `Set oDesign2 = CompInstance.GetParentDesign();`

Returns the interface to the design containing the compInstance.

This interface can be used to call Design functions. See: Design Object Script Commands.

Python Syntax	GetParentDesign()
Python Example	`oDesign2 = CompInstance.GetParentDesign()`

GetPropHost

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

Command: None

Syntax: GetPropHost()

Return Value: Returns interface to PropHost.

VB Example: `Set oPropHost2 = CompInstance.GetPropHost();`

Returns the interface to the properties of the compInstance.

This interface can be used to call PropHost functions; for more information see Callback Scripting Using PropHost Object.

Python Syntax	GetPropHost()
Python Example	`oPropHost2 = CompInstance.GetPropHost()`

GetPropServerName

Use: Returns the PropServerName of the Component corresponding to this CompInstance.

Command: None

Syntax: GetPropServerName()

Return Value: String

VB Example:

```
name = CompInstance.GetPropServerName();
Returns propserver name of compInstance (e.g., CompInst@MS_TRL;7) and stores it in "name".
```

Python Syntax	GetPropServerName()
Python Example	`name = CompInstance.GetPropServerName()`

16 - Schematic Scripting

The Schematic scripting interface is a set of commands that match the data changing methods available in the UI of the Schematic, plus some selection and query methods. Examples of the commands available through the scripting interface are adding items, removing items, and modifying items on the Schematic. Identifying objects on the Schematic is done by a unique ID that is generated and returned by all methods that add objects, and by the FindElements and GetSelections methods. This ID can then be passed into commands to modify or remove Schematic objects.

- Since most Schematic objects can have sub-components (such as property displays or segments on a wire, etc), IDs can also be in the format of "TopLevelID:SubComponentIndex".
- The SubComponentIndex is a one-based index into the sub components of the item thus making the second segment of component CAP1 identified by "CAP1:2".
- A SubComponentIndex of zero will refer to the TopLevel item only.
- When an ID is mentioned later in this document, it will refer to either a simple string ID representing a top-level Schematic item or a "TopLevelID:SubComponentIndex" pair unless otherwise stated.
- If a specified ID or SubComponentIndex does not exist, then the function will ignore that entry and proceed with other specified IDs. Even if no valid IDs are specified, the functions will still return success.

The topics for this section include:

Method Format

In the following formats, [Opt=x] appearing after a parameter indicates that the parameter is optional and the default value is x.

When calling object-creation methods as functions, parenthesis are required in order to retrieve the name of the created object. Parenthesis are also required in JavaScript. When using VB/VBScript to call a method as a subroutine, however, parenthesis are not required.

All methods to create Schematic objects have the following form:

```
Create[type](VARIANT parameters,
// Array of type specific parameters
VARIANT attributes,
// attributes is in the format of:
Array("NAME:Attributes", _
"Page:=", int, _ // [Opt=1] Page number (one-based)
// Note: Page 1 always exists
"X:=", double, _ // X position of the object
"Y:=", double, _ // Y position of the object
"Angle:=", double, _ // Rotation angle (radians)
"Flip:=", bool) // True if mirrored
[out,retval] string id)
```

The algorithm used in the create methods is:

1) Create SchAdd*[type]*Command object with parameters passed in

2) Execute the command

All methods to modify Schematic objects have the following form:

```
[modification](Array("NAME:Selections",
"Page:=", Page number [optional: Default=1]
"Selections:=", IDs to modify see format below),
VARIANT params
// Array of modification specific parameters)
```

The algorithms in these types of methods will be:

1) Go through ids and get the SelectableObjs they correspond to

2) Select the SelectableObjs found in (1)

3) Select any sub components specified

4) Create the Sch*[modification]*Command

5) Execute the command

ID Format

When IDs are passed to a function it can be in one of the following formats:

- Array of integers of top-level items only
- String of IDs separated by spaces or commas
- Array of strings with each string element being an ID

Point Format

When a method takes an array of points, each element in the array is a string in the format of: "x y", "x,y", or "(x,y)".

Editor Scripting IDs

Objects in the schematic are identified by text IDs. These IDs are used in scripts to perform actions on objects. These IDs are returned by all methods that add objects, and by the FindElements and GetSelections methods. The IDs are then passed into commands to modify or remove Schematic objects.

Format of IDs for different schematic objects:

Components: CompInst@<CompName>;<compInstID>;<optional schematicID>

Geometric primitive: SchObj@<schematicID>

Global Port/ground :: GPort@<portName>;<schematicID>

Interface Port :: IPort@<portName>;<schematicID>

Page Port: PagePort@<portName>;<schematicID>

Wire : Wire@<netName>;<schematicID>;<segment index list>

Find this information in the Properties Window for the selected object as follows:

<CompName> General tab/CompName

<compInstID> General tab/ID

<schematicID> Symbol tab/SchematicID

<portName> Param Values tab/PortName

<netName> General tab/NetName

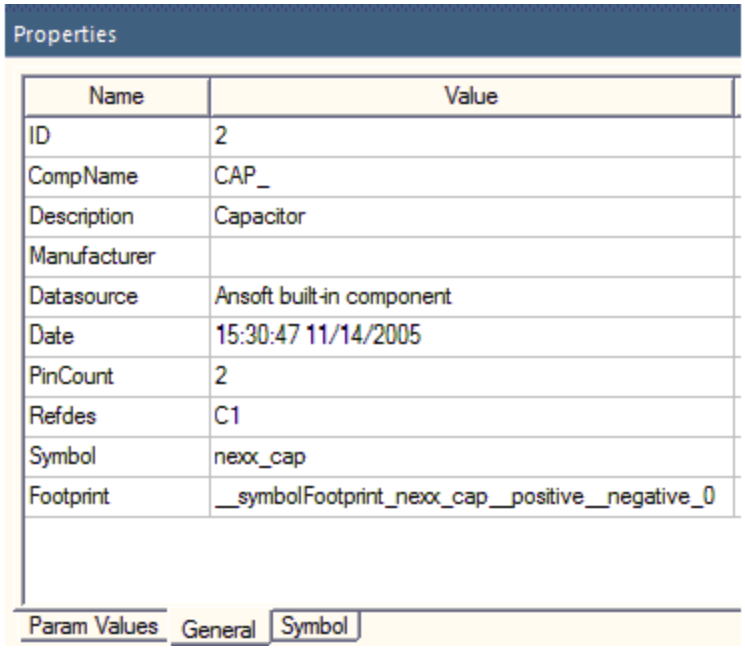
<segment index list> numbers separated by commas; Symbol tab/Segmentn - e.g. Segment0 refers to index "0"

In the Layout, the ID for a selected object is always the Name or Net property in the Footprint tab of the Properties Window

Format for Components

```
Array("CompInst@CAP_;2;4"))
```

The format for Components is CompInst@<CompName>;<compInstID>;<optional schematicID>. In the documented example, <CompName> stands for the component name CAP_, <compInstID> stands for its ID 2, and <optional schematicID> is the schematic ID 4. If you select the component, the **Properties** window gets updated and displays its details. For example, the selected object's component name and ID appear in the **General** tab of the **Properties** window as shown below.



The SchematicID is shown in the **Symbol** tab.

Create Method List

This section lists the scripts that are available in the Schematic scripting interface to create and change data.

Script Position Parameters

In Create Method scripts, X and Y position parameters are expressed in meters.

- If your layout grid units are metric, you may enter the X and Y positions directly into a script as parameters.
- If your layout grid is expressed in mils, you must first convert the component's positional coordinates to meters.

To convert an X or Y position to meters based on the minor grid size:

X or Y position in meters = desired_position x (0.00254 / minor_grid_size)

where desired_position and minor_grid_size have the same units. For example, to position an object at 500mm with a minor grid size of 20mm:

X or Y position in meters = 500 x (0.00254 / 20)

This section lists the following commands:

CreateText (Schematic Editor)

CreateWire (Schematic Editor)

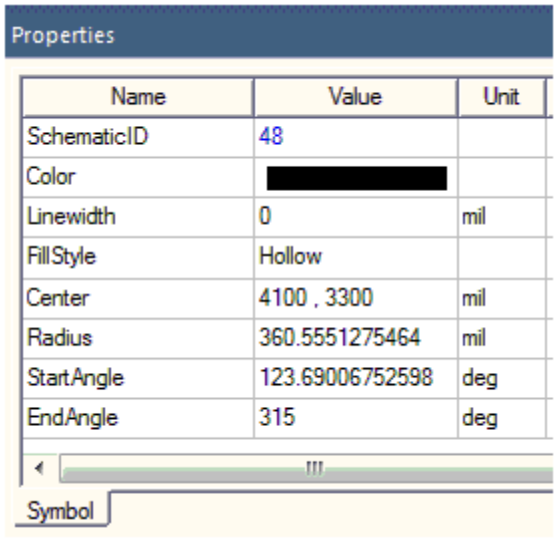
CreateArc (Schematic Editor)

Create an arc

<table>
<tr><th>UI Access</th><td colspan="3">Draw>Primitive>Arc</td></tr>
<tr><td>Parameters</td><td colspan="3">
<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td><ArcData></td><td>Array</td><td>Array("NAME:ArcData" _
"x:=", double, _ // X position of the object
"y:=", double, _ // Y position of the object
"Radius:=", double, _ // Radius of the circle
"StartAng:=", double,_ // Start angle of the arc (radians)
"EndAng:=", double, _ // End angle of the arc (radians)
"LineWidth:=", double, // the width of the line, in meters
"Color:=", int, // the RBG value of the arc color
"Id:=", int), // [Opt=New id] Id for this item</td></tr>
<tr><td><Attributes></td><td>Array</td><td>Array("NAME:Attributes", _
"Page:=", int) _ // [Opt=1] Page number (one-based)
[out,retval] string id)</td></tr>
</table>
</td></tr>
<tr><td>Return Value</td><td colspan="3">String
Unique id</td></tr>
</table>

Arc is created with the format SchObj@<schematicID>

The Schematic ID can be found on the **Symbols** tab of the **Properties** window when you select the arc.



Python Syntax	CreateArc()
Python Example	`oEditor.CreateArc` `(["NAME:ArcData", _` `"X:=", -0.004318, "Y:=", -0.00127, _` `"Radius:=", 0.00297299377732279, _`

	"StartAng:=", 1.9195673303788, _ "EndAng:=", 3.32144615338227, _ "Id:=", 10], _ ["NAME:Attributes", "Page:=", 1])

VB Syntax	CreateArc()
VB Example	oEditor.CreateArc Array("NAME:ArcData", _ "X:=", -0.004318, "Y:=", -0.00127, _ "Radius:=", 0.00297299377732279, _ "StartAng:=", 1.9195673303788, _ "EndAng:=", 3.32144615338227, _ "Id:=", 10), _ Array("NAME:Attributes", "Page:=", 1)

VB Example:

```
Dim oAnsoftApp
Dim oDesktop
Dim oProject
Dim oDesign
```

```
Dim oEditor
Dim oModule
Set oAnsoftApp = CreateObject("Ansoft.ElectronicsDesktop")
Set oDesktop = oAnsoftApp.GetAppDesktop()
oDesktop.RestoreWindow
Set oProject = oDesktop.SetActiveProject("Project4")
Set oDesign = oProject.SetActiveDesign("Circuit1")
Set oEditor = oDesign.SetActiveEditor("SchematicEditor")
Dim ArcID
ArcID = oEditor.CreateArc(Array("NAME:ArcData", "X:=", 0.10414, "Y:=", 0.08382, "Radius:=", _
0.00915810023967853, "StartAng:=", 2.15879893034246, "EndAng:=", _
5.49778714378214, "LineWidth:=", 0, "Color:=", 0, "Id:=", 48), Array("NAME:Attributes", "Page:=",
_
1))
MsgBox "Arc ID = " & ArcID
```

CreateCircle (Schematic Editor)

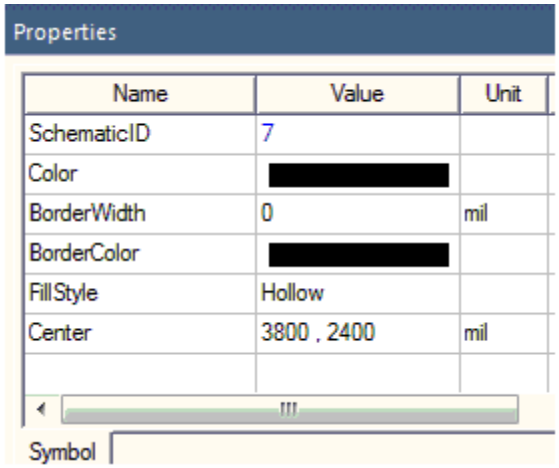
Create a circle

UI Access	**Draw>Primitive>Circle**		
Parameters	Name	Type	Description
	<CircleData>	Array	`Array("NAME:CircleData" _`

			"x:=", double, _ // X position of the object "y:=", double, _ // Y position of the object "Radius:=", double, _ // Radius of the circle "LineWidth:=", double, // the width of the circle border, in meters "Bordercolor:=", int, // the RBG value of the border color "Fill:=", int, // the fill pattern id, 0 = hollow, 1 = solid, 2 = NEDiagonal, 3 = OrthoCross, 4 = DiagCross, 5 = NWDiagonal, 6 = Horizontal, 7 = Vertical "Color:=", int, // the RBG value of the circle fill color "Id:=", int), // [Opt=New id] Id for this item
	<Attributes>	Array	`Array("NAME:Attributes", _` "Page:=", int) _ // [Opt=1] Page number (one-based) [out,retval] string id)
Return Value	String Unique id		

Circle is created and the returned value has the format SchObj@<schematicID>

The schematic ID of the circle can be located in the **Properties** window on the **Symbols** tab.



Python Syntax	CreateCircle()
Python Example	`oEditor.CreateCircle(["NAME:CircleData", _` `"X:=", -0.004572, "Y:=", -0.000508, _` `"Radius:=", 0.001778, "Id:=", 12], _` `["NAME:Attributes", "Page:=", 1])`

VB Syntax	CreateCircle()

VB Example	oEditor.CreateCircle Array("NAME:CircleData", _ "X:=", -0.004572, "Y:=", -0.000508, _ "Radius:=", 0.001778, "Id:=", 12), _ Array("NAME:Attributes", "Page:=", 1)

CreateComponent (Schematic Editor)

Creates a component of a given type.

UI Access	N/A.
Parameters	See table below.

Name	Type	Description
<params>	Array	Structured array. Array("NAME:ComponentProps", "Name:=", <String path to component in library>, "Id:=", <String Component ID>)
<attributes>	Array	Structured array. Array("NAME:Attributes", "Page:=", <Integer: page number>, **Note:** For the page number, page 1 always exists. "X:=", <Double: X position of object>, "Y:=", <Double: Y position of the object>,

			"Angle:=", <Double: rotation angle in degrees>, "Flip:=", <boolean: True if mirrored; else False>, "PinNumber:=", <Integer: index of pin in symbol 0 to n-1> **Note:** PinNumber is optional and is set to -1 by default. PinNumber affects where the symbol is placed relative to the X and Y parameters. If PinNumber is set to -1, the symbol is placed with its center at (X,Y). If it is a pin index, the symbol is placed so that the pin is located at (X,Y). The PinNumber is not recorded.
Return Value	String containing unique ID in the format: `CompInst@<CompName>;<compInstID>;<optional schematicID>` For example, the return value `CompInst@CAP;4;35` indicates that the component name is `CAP`, its ID is `4`, and its Schematic ID is `35`. You can also see these details in the **Properties** window when you select the created component.		

Python Syntax	CreateComponent(<*params*>, <*attributes*>)
Python Example	`oEditor.CreateComponent(` `  ["NAME:ComponentProps",` `  "Name:=", "Maxwell Circuit Elements\Passive Elements:Res",`

	"Id:=", "1"], ["NAME:Attributes", "Page:=", 1, "X:=", -0.01524, "Y:=", 0.00508, "Angle:=", 0, "Flip:=", false])

VB Syntax	CreateComponent <*params*>, <*attributes*>
VB Example	Set oProject = oDesktop.SetActiveProject("Project1") Set oDesign = oProject.SetActiveDesign("Circuit1") Set oEditor = oDesign.SetActiveEditor("SchematicEditor") Dim CompID CompID = oEditor.CreateComponent Array("NAME:ComponentProps", "Name:=", "Maxwell Circuit Elements\Passive Elements:Res", "Id:=", "1"), Array("NAME:Attributes", "Page:=", 1, "X:=", -0.01524, "Y:=", 0.00508, "Angle:=", 0, "Flip:=", false)

CreateCurve [Schematic]

Creates a Bezier curve.

<table>
<tr><td>UI Access</td><td>Draw > Primitive > Curve</td></tr>
<tr><td>Parameters</td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><*curve data*></td><td>Array</td><td>("NAME:CurveData",
"Points:=", Array(), //Control Points of curve
"LineWidth:=", 0, //Linewidth of curve
"Color:=", 0, //Color
"Id:=", 1
)</td></tr>
<tr><td><*attributes*></td><td>Array</td><td>Array("NAME:Attributes", _
"Page:=", int) _ // [Opt=1] Page number
(one-based)</td></tr>
</table>
</td></tr>
<tr><td>Return Value</td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><id></td><td>String</td><td>Unique ID of this object. Use it to perform actions (such as select and move) on object.</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><td>Python Syntax</td><td>oEditor.CreateCurve([<*CurveData*>], [<*attributes*>])</td></tr>
<tr><td>Python Example</td><td>

```
curveObjId = oEditor.CreateCurve(
  [
    "NAME:CurveData",
    "Points:=" , ["(-0.149860, 0.233680)","(-0.139700,
0.254000)","(-0.121920, 0.238760)","(-0.147320, 0.226060)","(-0.139700,
0.218440)","(-0.121920, 0.220980)"],
    "LineWidth:=" , 0,
```

</td></tr>
</table>

	"Color:=" , 0, "Id:=" , 7], ["NAME:Attributes", "Page:=" , 1])

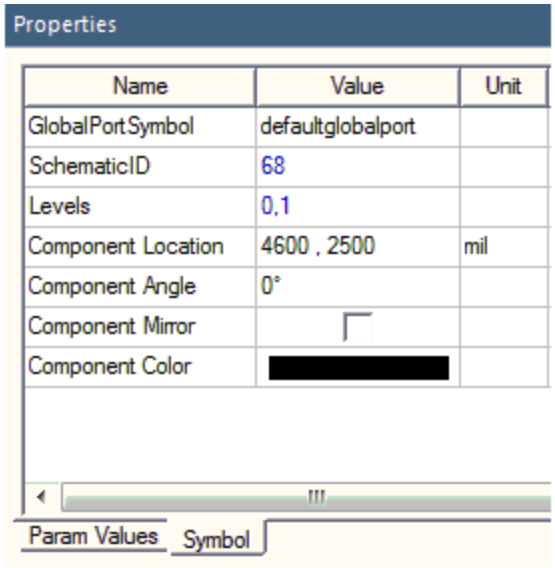
VB Syntax	oEditor.CreateCurve Array(<CurveData>),Array(<attributes>)
VB Example	oEditor.CreateCurve Array("NAME:CurveData", "Points:=", Array("(0.048260, 0.050800)", _ "(0.076200, 0.073660)", "(0.091440, 0.033020)", "(0.060960, 0.038100)"), "LineWidth:=", _ 0, "Color:=", 0, "Id:=", 1), Array("NAME:Attributes", "Page:=", 1) or curveObjId = oEditor.CreateCurve(Array("NAME:CurveData", "Points:=", Array("(0.048260, 0.050800)", _ "(0.076200, 0.073660)", "(0.091440, 0.033020)", "(0.060960, 0.038100)"), "LineWidth:=", _ 0, "Color:=", 0, "Id:=", 1), Array("NAME:Attributes", "Page:=", 1))

CreateGlobalPort (Schematic Editor)

Create a global port and enable placement on the schematic

UI Access	Draw>Global Port		
Parameters	Name	Type	Description
	<portname>	String	Name for this port - leave empty to use default unique name
	<id>	Integer	ID number for the port - leave empty to use default (recommended)
	<pagenumber>	Integer	The (1-based) page number of the schematic page this port should be added to - leave empty to use default (1)
	<xval>	Double	X position of the object
	<yval>	Double	Y position of the object
	<angle>	Double	Rotation of the object, in radians. Rotations of multiples of 90 degrees are valid. Other values will be adjusted to the nearest valid angle.
	<degrees>	Double	Rotation of the object, in degrees. Rotations of multiples of 90 degrees are valid. Other values will be adjusted to the nearest valid angle
	<flip>	Boolean	True if mirrored
Return Value	String PortID, the identifier that can be used for this port in script operations involving this schematic		

Global port is created and the return value has the format GPort@<portName>;<schematicID>
The Schematic ID can be found in the **Properties** window on the **Symbols** tab when you select the global port.



Python Syntax	CreateGlobalPort(["NAME:GlobalPortProps", "Name:=", <*portname*>, "Id:=", <*id*>], ["NAME:Attributes", "Page:=", <*pagenumber*>, "X:=", <*xval*>, "Y:=", <*yval*>, "Angle:=", <*angle*>, [or "Degrees:=", <*degrees*> ,] "Flip:=", <*flip*>])
Python Example	`oEditor.CreateGlobalPort` `(["NAME:GlobalPortProps", "Name:=", "G_0", "Id:=", 1],_` `["NAME:Attributes", "Page:=", 1, "X:=", -0.02286, "Y:=",` `0.00254, "Angle:=", 0, "Flip:=", false)`

VB Syntax	CreateGlobalPort(Array("NAME:GlobalPortProps", "Name:=", <*portname*>, "Id:=", <*id*>), Array("NAME:Attributes", "Page:=", <*pagenumber*>, "X:=", <*xval*>, "Y:=", <*yval*>, "Angle:=", <*angle*>, [or "Degrees:=", <*degrees*>,] "Flip:=", <*flip*>)
VB Example	`oEditor.CreateGlobalPort` `Array("NAME:GlobalPortProps", "Name:=", "G_0", "Id:=", 1),_` ` Array("NAME:Attributes","Page:=", 1, "X:=", -0.02286,` `"Y:=", 0.00254, "Angle:=", 0, "Flip:=", false)`

VB Example:

```
Dim oAnsoftApp
Dim oDesktop
Dim oProject
Dim oDesign
Dim oEditor
Dim oModule
Set oAnsoftApp = CreateObject("Ansoft.ElectronicsDesktop")
Set oDesktop = oAnsoftApp.GetAppDesktop()
oDesktop.RestoreWindow
Set oProject = oDesktop.SetActiveProject("Project4")
Set oDesign = oProject.SetActiveDesign("Circuit1")
Set oEditor = oDesign.SetActiveEditor("SchematicEditor")
```

```
Dim GPort

GPort = oEditor.CreateGlobalPort(Array("NAME:GlobalPortProps", "Name:=", "G_1", "Id:=", 68),
Array("NAME:Attributes", "Page:=", _

1, "X:=", 0.11684, "Y:=", 0.0635, "Angle:=", 0, "Flip:=", false))

MsgBox "Port = " & GPort
```

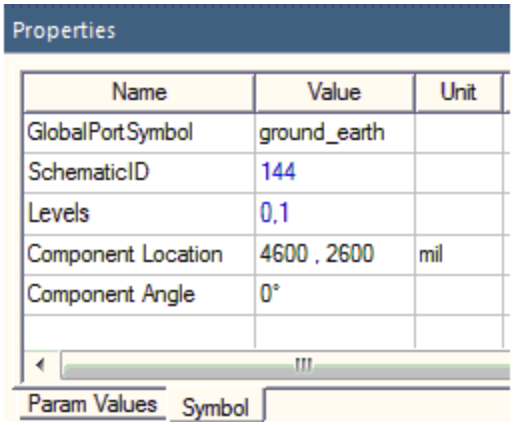
CreateGround (Schematic Editor)

Create a ground

<table>
<tr><td>UI Access</td><td colspan="3">Draw>Ground</td></tr>
<tr><td rowspan="3">Parameters</td><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><GroundProps></td><td>Array</td><td>Array("NAME:GroundProps",_
"Id:=", int), // [Opt=New id] Id for this item</td></tr>
<tr><td><Attributes></td><td>Array</td><td>Array("NAME:Attributes", _
"Page:=", int, _ // [Opt=1] Page number (one-based)
Note: Page 1 always exists
"X:=", double, _ // X position of the object
"Y:=", double, _ // Y position of the object
"Angle:=", double, _ // Rotation angle (radians)
"Flip:=", bool), // True if mirrored
[out,retval] string id)</td></tr>
<tr><td>Return Value</td><td colspan="3">String
Unique id</td></tr>
</table>

Ground is created and the returned value has the format GPort@<portname>;<schematicID>

The Schematic ID for ground can be found on the **Symbols** tab of the **Properties** window when you select the object.



Python Syntax	CreateGround()
Python Example	`oEditor.CreateGround (["NAME:GroundProps", "Id:=", 8), ["NAME:Attributes", "Page:=", 1, "X:=", -0.04064, "Y:=", -0.00254, "Angle:=", 0, "Flip:=", false])`

VB Syntax	CreateGround()
VB Example	oEditor.CreateGround Array("NAME:GroundProps", "Id:=", 8), Array("NAME:Attributes", "Page:=", 1, "X:=", -0.04064, "Y:=", -0.00254, "Angle:=", 0, "Flip:=", false)

VB Example:

```
Dim oAnsoftApp
Dim oDesktop
Dim oProject
Dim oDesign
Dim oEditor
Dim oModule
Set oAnsoftApp = CreateObject("Ansoft.ElectronicsDesktop")
Set oDesktop = oAnsoftApp.GetAppDesktop()
oDesktop.RestoreWindow
Set oProject = oDesktop.SetActiveProject("Project4")
Set oDesign = oProject.SetActiveDesign("Circuit1")
Set oEditor = oDesign.SetActiveEditor("SchematicEditor")
```

```
Gnd = oEditor.CreateGround(Array("NAME:GroundProps", "Id:=", 144), Array("NAME:Attributes",
"Page:=", _
1, "X:=", 0.11684, "Y:=", 0.06604, "Angle:=", 0, "Flip:=", false))


MsgBox "GndIndex =" & Gnd
```

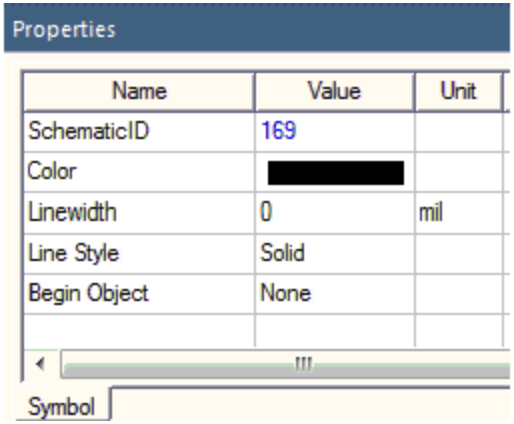
CreateLine (Schematic Editor)

Creates a line.

<table>
<tr><td>UI Access</td><td colspan="3">Draw>Primitive>Line</td></tr>
<tr><td>Parameters</td><td colspan="3"><table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><LineData></td><td>Array</td><td>Array("NAME:LineData" _
"Points:=", Array of points, _
"LineWidth:=", double, // the width of the line, in meters
"Color:=", int, // the RBG value of the line color
"Id:=", int), // [Opt=New id] Id for this item</td></tr>
<tr><td><Attributes></td><td>Array</td><td>Array("NAME:Attributes", _
"Page:=", int) _ // [Opt=1] Page number (one-based)
[out,retval] string id)</td></tr>
</table></td></tr>
<tr><td>Return Value</td><td colspan="3">String
Unique id</td></tr>
</table>

Line is created and the returned value has the format SchObj@<schematicID>

The Schematic ID can be found on the **Symbol** tab of the **Properties** window when you select the line.



Python Syntax	CreateLine()
Python Example	`oEditor.CreateLine (["NAME:LineData",_` `"Points:=", ["(-0.055652, 0.020669)", _` `"(-0.036923, 0.011257)", "(-0.023144, 0.018049)"],_` `"LineWidth:=", 0, "Color:=", 0, "Id:=", 22)], _` `["NAME:Attributes", "Page:=", 1])`

VB Syntax	CreateLine()
VB Example	oEditor.CreateLine Array("NAME:LineData",_ "Points:=", Array("(-0.055652, 0.020669)", _ "(-0.036923, 0.011257)", "(-0.023144, 0.018049)"),_ "LineWidth:=", 0, "Color:=", 0, "Id:=", 22), _ Array("NAME:Attributes", "Page:=", 1)

VB Example:

```
Dim oAnsoftApp
Dim oDesktop
Dim oProject
Dim oDesign
Dim oEditor
Dim oModule
Set oAnsoftApp = CreateObject("Ansoft.ElectronicsDesktop")
Set oDesktop = oAnsoftApp.GetAppDesktop()
oDesktop.RestoreWindow
Set oProject = oDesktop.SetActiveProject("Project4")
Set oDesign = oProject.SetActiveDesign("Circuit1")
Set oEditor = oDesign.SetActiveEditor("SchematicEditor")
LineID = oEditor.CreateLine(Array("NAME:LineData", "Points:=", Array("(0.050800, 0.106680)", _
```

```
"(0.124460, 0.106680)", "(0.124460, 0.106680)"), "LineWidth:=", 0, "Color:=", _
0, "Id:=", 169), Array("NAME:Attributes", "Page:=", 1))
MsgBox "LineID = " & LineID
```

CreateNPort (Schematic Editor)

Use: Creates an N-Port definition and component and adds them to the current project, layout, and schematic.

Command: CreateNPort

Syntax: **CreateNPort**

```
Array("NAME:Contents",
"definition_name:=",<nport_definition_name>,
"placement:=", <component_placement>,
"layer:=", <placement_layer_name>,
<nport_data_definition>)
```

<nport_definition_name>:

```
quoted string (name of the component definition)
```

<component_placement>:

```
Array("x:=", <value>, // x coordinate
"y:=", <value>, // y coordinate
"scaling:=", <value>) // double
```

Return Value: Returns the name of the newly created component.

<nport_data_definition**>** - see the I/O format in TODO

VB Example: **Example:**

```
oEditor.CreateNPort
Array("NAME:Contents",
"definition_name:=", "NetworkData3",
"placement:=",
Array("x:=", "-9mm",
"y:=", "-5mm",
"scaling:=", "2"),
"layer:=", "Symbols",
Array("NAME:NPortData",
Array("NAME:NetworkData2",
"filelocation:=", "UsePath",
"filename:=", "",
"domain:=", "frequency",
"numberofports:=", 2,
"datamode:=", "Import",
"devicename:=", "", "
ImpedanceTab:=", 1,
```

```
"NoiseDataTab:=", 1,
"DCBehaviorTab:=", 1,
"SolutionName:=", "",
"dcbehavior:=", "DCOpen",
"displayformat:=", "MagnitudePhase",
"datatype:=", "SMatrix",
"interptype:=", "Linear",
"extraptype:=", "Same as interpolation",
"ShowRefPin:=", 0,
"RefNodeCheckbox:=", 1)))
```

CreatePagePort (Schematic Editor)

Create a PagePort

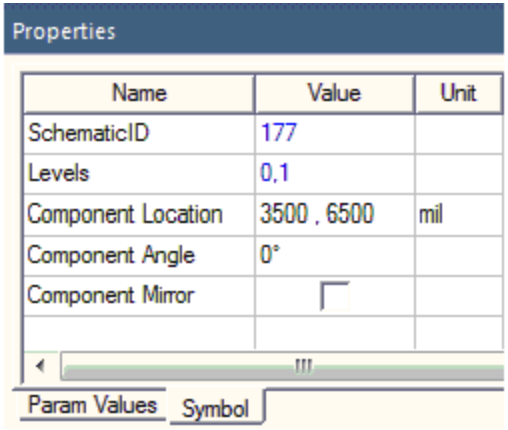
<table>
<tr><td>UI Access</td><td colspan="3">Draw>Page Connector</td></tr>
<tr><td>Parameters</td><td colspan="3"><table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><PagePortProps></td><td>Array</td><td>Array("NAME:PagePortProps"),
Array("NAME:Attributes", _
"Page:=", int, _ // [Opt=1] Page number (one-based)
Note: Page 1 always exists
"X:=", double, _ // X position of the object
"Y:=", double, _ // Y position of the object</td></tr>
</table></td></tr>
</table>

			"Angle:=", double, _ // Rotation angle (radians) "Flip:=", bool), // True if mirrored [out,retval] string id)
Return Value	String Unique id		

The pageport is created and it has the following format:

`PagePort@<portName>;<schematicID>`

The schematic ID can be found in the **Properties** window on the **Symbols** tab when you select the pageport.



Python Syntax	CreatePagePort()
Python Example	oEditor.CreatePagePort (["NAME:PagePortProps", "Name:=", "pageport_0", "Id:=", _ 12], ["NAME:Attributes", "Page:=",_ 1, "X:=", -0.0381, "Y:=", -0.0127, "Angle:=", _ 0, "Flip:=", false])

VB Syntax	CreatePagePort()
VB Example	oEditor.CreatePagePort Array("NAME:PagePortProps", "Name:=", "pageport_0", "Id:=", _ 12), Array("NAME:Attributes", "Page:=",_ 1, "X:=", -0.0381, "Y:=", -0.0127, "Angle:=", _ 0, "Flip:=", false)

VB Example:

```
Dim oAnsoftApp
Dim oDesktop
Dim oProject
```

```
Dim oDesign
Dim oEditor
Dim oModule
Set oAnsoftApp = CreateObject("Ansoft.ElectronicsDesktop")
Set oDesktop = oAnsoftApp.GetAppDesktop()
oDesktop.RestoreWindow
Set oProject = oDesktop.SetActiveProject("Project4")
Set oDesign = oProject.SetActiveDesign("Circuit1")
Set oEditor = oDesign.SetActiveEditor("SchematicEditor")
PagePortID = oEditor.CreatePagePort(Array("NAME:PagePortProps", "Name:=", "pageport_0", "Id:=", _
177), Array("NAME:Attributes", "Page:=", 1, "X:=", 0.0889, "Y:=", 0.1651, "Angle:=", _
0, "Flip:=", false))
MsgBox "PagePort = " & PagePortID
```

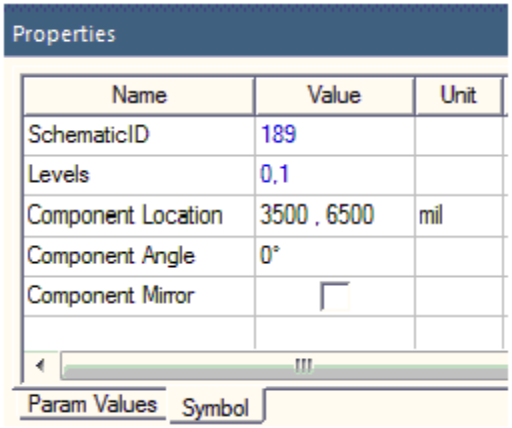
CreateIPort (Schematic Editor)

Create an interface port

UI Access	Draw>Interface Port		
Parameters	Name	Type	Description
	<IPortProps>	Array	`Array("NAME:IPortProps"),` "Name:=", string), // [Opt=default name] Name for this port "Id:=", int), // Port id number

	<Attributes>	Array	`Array("NAME:Attributes", _` "Page:=", int, _ // [Opt=1] Page number (one-based) **Note:** Page 1 always exists "X:=", double, _ // X position of the object "Y:=", double, _ // Y position of the object "Angle:=", double, _ // Rotation angle (radians) "Flip:=", bool), // True if mirrored [out,retval] string id)
Return Value	String Unique id		

Interface port is created and the returned value has the following format:
IPort@<portName>;<schematicID>
The interface port schematic ID can be found in the **Properties** window on the **Symbols** tab.



Python Syntax	CreateIPort()
Python Example	`oEditor.CreateIPort (["NAME:IPortProps", "Name:=", _` `"Port1", "Id:=", 4], ["NAME:Attributes","Page:=", _` `1, "X:=", -0.0381, "Y:=", 0.0127, "Angle:=",_` `0, "Flip:=", false])`

VB Syntax	CreateIPort()
VB Example	`oEditor.CreateIPort Array("NAME:IPortProps", "Name:=",_`

	"Port1", "Id:=", 4), Array("NAME:Attributes","Page:=", _ 1, "X:=", -0.0381, "Y:=", 0.0127, "Angle:=",_ 0, "Flip:=", false)

VB Example:

```
Dim oAnsoftApp

Dim oDesktop

Dim oProject

Dim oDesign

Dim oEditor

Dim oModule

Set oAnsoftApp = CreateObject("Ansoft.ElectronicsDesktop")

Set oDesktop = oAnsoftApp.GetAppDesktop()

oDesktop.RestoreWindow

Set oProject = oDesktop.SetActiveProject("Project4")

Set oDesign = oProject.SetActiveDesign("Circuit1")

Set oEditor = oDesign.SetActiveEditor("SchematicEditor")

IPortID = oEditor.CreateIPort(Array("NAME:IPortProps", "Name:=", "Port1", "Id:=", 189), Array
("NAME:Attributes", "Page:=", _

1, "X:=", 0.0889, "Y:=", 0.1651, "Angle:=", 0, "Flip:=", false))

MsgBox "IPort ID = " & IPortID
```

CreatePolygon (Schematic Editor)

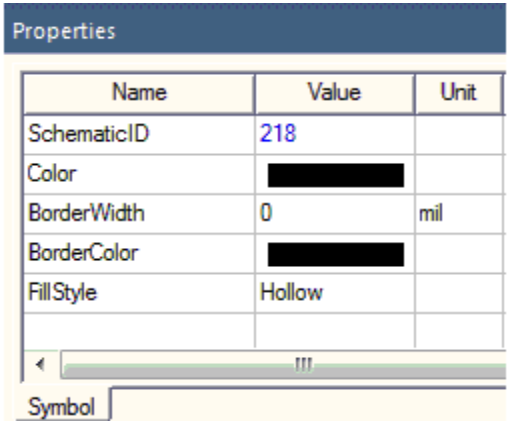
Create a polygon

<table>
<tr><td>UI Access</td><td colspan="3">Draw>Primitive>Polygon</td></tr>
<tr><td>Parameters</td><td colspan="3"><table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><PolygonData></td><td>Array</td><td>Array("NAME:PolygonData" _
"Points:=", Array of points, _
"LineWidth:=", double, // the width of the Polygon border, in meters
"Bordercolor:=", int, // the RBG value of the border color
"Fill:=", int, // the fill pattern id, 0 = hollow, 1 = solid, 2 = NEDiagonal,
3 = OrthoCross, 4 = DiagCross, 5 = NWDiagonal, 6 = Horizontal, 7 = Vertical
"Color:=", int, // the RBG value of the Polygon fill color
"Id:=", int), // [Opt=New id] Id for this item</td></tr>
<tr><td><Attributes></td><td>Array</td><td>Array("NAME:Attributes", _
"Page:=", int) _ // [Opt=1] Page number (one-based)
[out,retval] string id)</td></tr>
</table></td></tr>
<tr><td>Return Value</td><td colspan="3">String
Unique id</td></tr>
</table>

Polygon is created and the return value has the format

SchObj@<schematicID>

The Schematic ID can be found in the **Properties** window on the **Symbol** tab when you select the polygon.



Python Syntax	CreatePolygon()
Python Example	`oEditor.CreatePolygon(` `["NAME:PolygonData", "Points:=", [ _` `"(-0.058951, 0.006308)", "(-0.046142, 0.010480)",_` `"(-0.046239, 0.001067)"), "LineWidth:=", _` `0, "BorderColor:=", 0, "Fill:=", 0, "Color:=",_` `0, "Id:=", 27], ["NAME:Attributes", "Page:=", 1]])`

VB Syntax	CreatePolygon()

VB Example	oEditor.CreatePolygon Array("NAME:PolygonData", "Points:=", Array(_ "(-0.058951, 0.006308)", "(-0.046142, 0.010480)",_ "(-0.046239, 0.001067)"), "LineWidth:=", _ 0, "BorderColor:=", 0, "Fill:=", 0, "Color:=",_ 0, "Id:=", 27), Array("NAME:Attributes", "Page:=", 1)

VB Example:

```
Dim oAnsoftApp
Dim oDesktop
Dim oProject
Dim oDesign
Dim oEditor
Dim oModule
Set oAnsoftApp = CreateObject("Ansoft.ElectronicsDesktop")
Set oDesktop = oAnsoftApp.GetAppDesktop()
oDesktop.RestoreWindow
Set oProject = oDesktop.SetActiveProject("Project4")
Set oDesign = oProject.SetActiveDesign("Circuit1")
```

```
Set oEditor = oDesign.SetActiveEditor("SchematicEditor")
polygonID = oEditor.CreatePolygon(Array("NAME:PolygonData", "Points:=", Array( _
"(0.101600, 0.058420)", "(0.101600, 0.060960)", "(0.101600, 0.058420)", _
"(0.099060, 0.060960)", "(0.101600, 0.060960)", "(0.101600, 0.058420)", _
"(0.101600, 0.060960)", "(0.101600, 0.060960)"), "LineWidth:=", 0, "BorderColor:=", _
0, "Fill:=", 0, "Color:=", 0, "Id:=", 218), Array("NAME:Attributes", "Page:=", 1))
MsgBox "Polygon ID = " & polygonID
```

CreateRectangle (Schematic Editor)

Create a rectangle

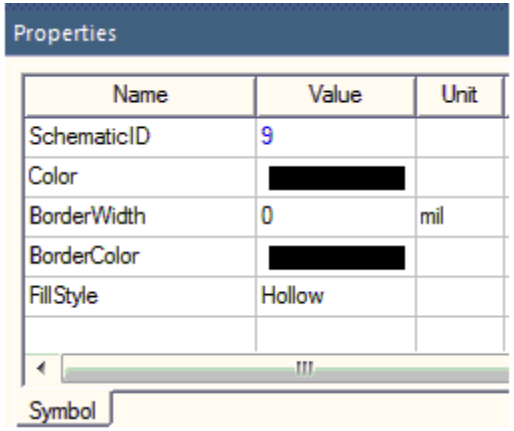
UI Access	Draw>Primitive>Rectangle		
Parameters	Name	Type	Description
	<RectData>	Array	`Array("NAME:RectData" _` "x1:=", double, _ // X position of the upper left "y1:=", double, _ // Y position of the upper left "x2:=", double, _ // X position of the lower right "y2:=", double",_ // Y position of the lower right "LineWidth:=", double, // the width of the Rectangle border, in meters "Bordercolor:=", int, // the RBG value of the border color "Fill:=", int, // the fill pattern id, 0 = hollow, 1 = solid, 2 = NEDiagonal, 3 = OrthoCross, 4 = DiagCross, 5 = NWDiagonal, 6 = Horizontal, 7 = Vertical "Color:=", int, // the RBG value of the Rectangle fill color

			"Id:=", int), // [Opt=New id] Id for this item
	<Attributes>	Array	`Array("NAME:Attributes", _` "Page:=", int) _ // [Opt=1] Page number (one-based) [out,retval] string id)
Return Value	String Unique id		

Rectangle is created and the return value has the format

SchObj@<schematicID>

The Schematic ID can be found in the **Properties** window on the **Symbol** tab when you select the rectangle.



Python Syntax	CreateRectangle()
Python Example	oEditor.CreateRectangle (["NAME:RectData", "X1:=", -0.0755449856733525, "Y1:=", _ 0.0335755491881566, "X2:=", -0.0611831900668577, "Y2:=", _ 0.0254242597898758, "LineWidth:=", _ 0, "BorderColor:=", 0, "Fill:=", 0, "Color:=", 0, "Id:=", 31], _ ["NAME:Attributes", "Page:=", 1])

VB Syntax	CreateRectangle()
VB Example	oEditor.CreateRectangle Array("NAME:RectData", "X1:=", -0.0755449856733525, "Y1:=", _ 0.0335755491881566, "X2:=", -0.0611831900668577, "Y2:=", _ 0.0254242597898758, "LineWidth:=", _ 0, "BorderColor:=", 0, "Fill:=", 0, "Color:=", 0, "Id:=", 31), _ Array("NAME:Attributes", "Page:=", 1)

VB Example:

```
Dim oAnsoftApp
Dim oDesktop
Dim oProject
```

```
Dim oDesign

Dim oEditor

Dim oModule

Set oAnsoftApp = CreateObject("Ansoft.ElectronicsDesktop")

Set oDesktop = oAnsoftApp.GetAppDesktop()

oDesktop.RestoreWindow

Set oProject = oDesktop.SetActiveProject("Project4")

Set oDesign = oProject.SetActiveDesign("Circuit1")

Set oEditor = oDesign.SetActiveEditor("SchematicEditor")

rectangleID = oEditor.CreateRectangle(Array("NAME:RectData", "X1:=", 0.1016, "Y1:=", 0.0635,
"X2:=", _

0.10414, "Y2:=", 0.06096, "LineWidth:=", 0, "BorderColor:=", 0, "Fill:=", 0, "Color:=", _

0, "Id:=", 9), Array("NAME:Attributes", "Page:=", 1))

MsgBox "Rectangle ID = " & rectangleID
```

CreateText (Schematic Editor)

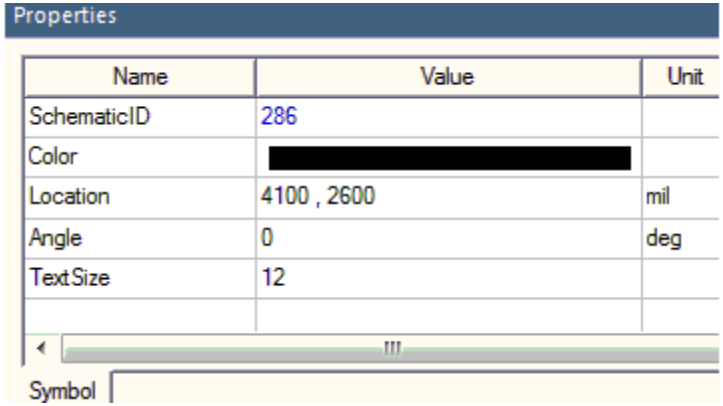
Create text

<table>
<tr><td>UI Access</td><td>Draw>Primitive>Text</td></tr>
<tr><td>Parameters</td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><TextData></td><td>Array</td><td>Array("NAME:TextData" _
"x:=", double, _ // X position of the object</td></tr>
</table>
</td></tr>
</table>

			"y:=", double, _ // Y position of the object "Text:=", string, _ // Text to display "Color:=", int, // the RBG value of the text color "Id:=", int), // Id for this item "ShowRect:=", bool, // true or false, whether to show the highlight rectangle for the text "X1:=", double, // the text rectangle left X value, in meters "Y1:=", double, // the text rectangle upper Y value, in meters "X2:=", double, // the text rectangle right X value, in meters "Y2:=", double, // the text rectangle lower Y value, in meters "RectLineWidth:=", double, // the width of the rectangle border, in meters "RectBordercolor:=", int, // the RBG value of the rectangle border color "RectFill:=", int, // the rectangle fill pattern id, 0 = hollow, 1 = solid, 2 = NEDiagonal, 3 = OrthoCross, 4 = DiagCross, 5 = NWDiagonal, 6 = Horizontal, 7 = Vertical "RectColor:=", int, // the RBG value of the rectangle fill color
	<Attributes>	Array	`Array("NAME:Attributes", _` "Page:=", int) _ // [Opt=1] Page number (one-based) [out,retval] string id)
Return Value	String Unique id		

Text is created and the return value has the format: SchObj@<SchematicID>

The Schematic ID can be found in the **Properties** window on the **Symbols** tab when you select the text.



Python Syntax	CreateText()
Python Example	`oEditor.CreateText(` `["NAME:TextData", "X:=", -0.0764183381088826, "Y:=", _` `0.040174212034384, "Size:=", 12, "Angle:=",_` `0, "Text:=", "Control Circuit", "Color:=", _` `0, "Id:=", 34, "ShowRect:=", false, "X1:=", _` `-0.0793287547755609, "Y1:=", _`

	0.0407033787010528, "X2:=", -0.0502245881087778,_ "Y2:=", 0.035411712034365, "RectLineWidth:=", _ 0, "RectBorderColor:=", 0, "RectFill:=", 0, "RectColor:=", 0], ["NAME:Attributes", "Page:=", 1])

VB Syntax	CreateText()
VB Example	oEditor.CreateText Array("NAME:TextData", "X:=", -0.0764183381088826, "Y:=", _ 0.040174212034384, "Size:=", 12, "Angle:=",_ 0, "Text:=", "Control Circuit", "Color:=", _ 0, "Id:=", 34, "ShowRect:=", false, "X1:=", _ -0.0793287547755609, "Y1:=", _ 0.0407033787010528, "X2:=", -0.0502245881087778,_ "Y2:=", 0.035411712034365, "RectLineWidth:=", _ 0, "RectBorderColor:=", 0, "RectFill:=", 0, "RectColor:=", 0), Array("NAME:Attributes", "Page:=", 1)

VB Example:

```
Dim oAnsoftApp
Dim oDesktop
```

```
Dim oProject
Dim oDesign
Dim oEditor
Dim oModule
Set oAnsoftApp = CreateObject("Ansoft.ElectronicsDesktop")
Set oDesktop = oAnsoftApp.GetAppDesktop()
oDesktop.RestoreWindow
Set oProject = oDesktop.SetActiveProject("Project4")
Set oDesign = oProject.SetActiveDesign("Circuit1")
Set oEditor = oDesign.SetActiveEditor("SchematicEditor")
textobj = oEditor.CreateText(Array("NAME:TextData", "X:=", 0.10414, "Y:=", 0.06604, "Size:=", _
12, "Angle:=", 0, "Text:=", "CreateDefaultTectxt" & Chr(13) & Chr(10) & "", "Color:=", _
0, "Id:=", 286, "ShowRect:=", false, "X1:=", 0.100141851851836, "Y1:=", _
0.0670983333333376, "X2:=", 0.140123333333477, "Y2:=", 0.0565149999999619, "RectLineWidth:=", _
0, "RectBorderColor:=", 0, "RectFill:=", 0, "RectColor:=", 0), Array("NAME:Attributes", "Page:=",
_
1))
MsgBox "text = " & textobj
```

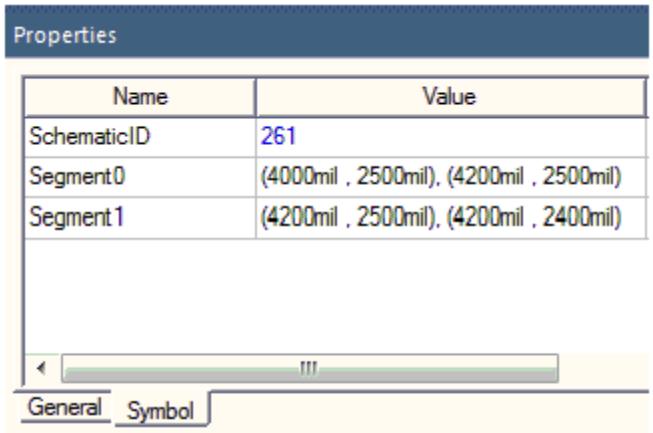
CreateWire (Schematic Editor)

Create a wire

<table>
<tr><td>UI Access</td><td>Draw>Wire</td></tr>
<tr><td>Parameters</td><td>
<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td><WireData></td><td>Array</td><td>Array("NAME:WireData" _
"Name:=", string), // [Opt=default name] Name for this wire
"Id:=", int), // Wire idnum
"Points:=", Points on wire)</td></tr>
<tr><td><Attributes></td><td>Array</td><td>Array("NAME:Attributes", _
"Page:=", int, _ // [Opt=1] Page number (one-based)
Note: Page 1 always exists
[out,retval] string id)</td></tr>
</table>
</td></tr>
<tr><td>Return Value</td><td>String
Unique id</td></tr>
</table>

Wire is created and its return value has the format Wire@<netName>;<schematicID>;<segment index list>

The Schematic ID can be found in the **Symbol** tab of the **Properties** window when you select the wire. The wire net name appears in the **General** tab. The parameter segment index list indicates the number of segments i.e. the segment count.



Python Syntax	CreateWire()
Python Example	`oEditor.CreateWire(` `["NAME:WireData", "Name:=", "", "Id:=", 41, "Points:=", [ _` `"(-0.033020, 0.015240)", "(-0.017780, 0.015240)",` `"(-0.017780, 0.012700)"]], ["NAME:Attributes", "Page:=", 1])`

VB Syntax	CreateWire()
VB Example	`oEditor.CreateWire`

	Array("NAME:WireData", "Name:=", "", "Id:=", 41, "Points:=", Array(_ "(-0.033020, 0.015240)", "(-0.017780, 0.015240)", "(-0.017780, 0.012700)")), Array("NAME:Attributes", "Page:=", 1)

VB Example:

```
Dim oAnsoftApp
Dim oDesktop
Dim oProject
Dim oDesign
Dim oEditor
Dim oModule
Set oAnsoftApp = CreateObject("Ansoft.ElectronicsDesktop")
Set oDesktop = oAnsoftApp.GetAppDesktop()
oDesktop.RestoreWindow
Set oProject = oDesktop.SetActiveProject("Project4")
Set oDesign = oProject.SetActiveDesign("Circuit1")
Set oEditor = oDesign.SetActiveEditor("SchematicEditor")
WireID = oEditor.CreateWire(Array("NAME:WireData", "Name:=", "", "Id:=", 261, "Points:=", Array(
_
"(0.101600, 0.063500)", "(0.106680, 0.063500)", "(0.106680, 0.060960)")), Array("NAME:At-
tributes", "Page:=", _
1))
MsgBox "Wire ID = " & WireID
```

General Method List

This section presents the general script methods that are available.

The general method script commands are listed here.

Activate (Schematic Editor)

`Enable items previously disabled`

UI Access	Edit>Activate		
Parameters	Name	Type	Description
	Name	Type	Description
Return Value	Value		

> **Note:**
>
> For more information about the component name, schematic IDs, and formats, see the section Editor Scripting IDs.

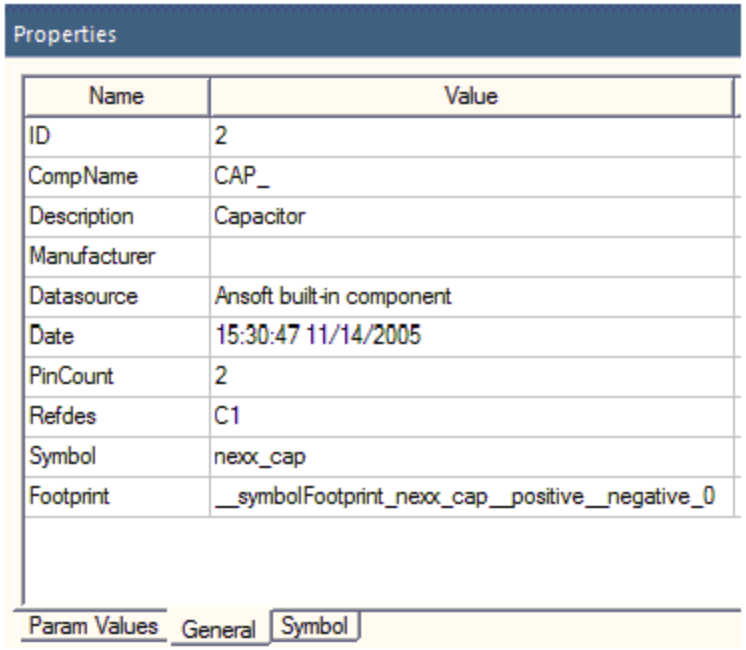
Python Syntax	Activate(["NAME:Selections","Selections:=", [<components>]])
Python Example	oEditor.Activate(["NAME:Selections","Selections:=", ["CompInst@STEP;1;1"]])

VB Syntax	Activate Array("NAME:Selections", "Selections:=", Array("<components>"))
VB Example	`oEditor.Activate Array("NAME:Selections",` `"Selections:=", Array("CompInst@STEP;1;1"))`

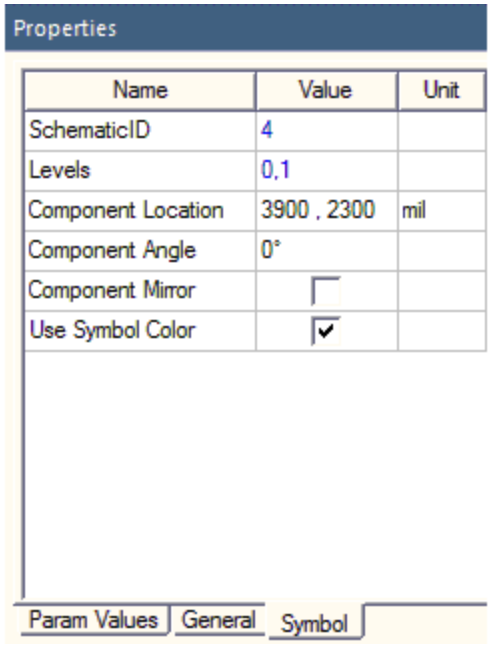
AddPinGrounds (Schematic Editor)

```
Adds grounds at all unconnected pins
AddPinGrounds (
Array("NAME:Selections", _ // PagePort Name
"Selections:=", _ // Net Name
Array("CompInst@CAP_;2;4"))
```

> **Note:**
>
> The format for Components is CompInst@<CompName>;<compInstID>;<optional schematicID>. In the documented example, <CompName> stands for the component name CAP_, <compInstID> stands for its ID 2, and <optional schematicID> is the schematic ID 4. If you select the component, the **Properties** window gets updated and displays its details. For example, the selected object's component name and ID appear in the **General** tab of the **Properties** window as shown below.



To view the SchematicID click the **Symbol** tab.



Python Syntax	AddPinPageConnectors ()
Python Example	`oEditor.AddPinPageconnectors["NAME:Selections",` `"Selections:=", [ _` `"CompInst@C;1;1"]]`

AddPinlPorts (Schematic Editor)

Adds Interface Ports at unconnected pins of all selected components. Each port has a unique name.

UI Access	Draw>Add at unconnected pins>Interface Ports		
Parameters	Name	Type	Description
	<IntPortName,NetName>	Array	`Array("NAME:Selections", _ // Interface Port Name` "Selections:=", _ // Net Name
Return Value	None		

Note:

For more information about the component name, schematic IDs, and formats, see the section Editor Scripting IDs.

Python Syntax	AddPinIPorts ()
Python Example	`oEditor.AddPinIPorts` `["NAME:Selections", "Selections:=", ["CompInst@R;2;37"]]`

VB Syntax	AddPinIPorts ()
VB Example	`oEditor.AddPinIPorts` `Array("NAME:Selections", "Selections:=", Array("CompInst@R;2;37"))`

AddPinPageConnectors (Schematic Editor)

Adds Page Ports at unconnected pins of all selected components

UI Access	Draw>Add at unconnected pins>Grounds
Parameters	Name | Type | Description Name | Type | Description
Return Value	None

> **Note:**
>
> For more information about the component name, schematic IDs, and formats, see the section Editor Scripting IDs.

Python Syntax	AddPinPageConnectors ()
Python Example	`oEditor.AddPinPageconnectors` `["NAME:Selections", "Selections:=", [ _` `"CompInst@C;1;1"]]`

VB Syntax	AddPinPageConnectors ()
VB Example	`oEditor.AddPinPageconnectors` `Array("NAME:Selections", "Selections:=", Array( _` `"CompInst@C;1;1"))`

AddPinCustomTerminations (Schematic Editor)

Adds Custom Terminations at unconnected pins of all selected components.

UI Access	Draw > Add at unconnected pins> Custom Terminations.
Parameters	**Name** / **Type** / **Description** ComponentInstanceName / <string> / Instance Name of the unconnected pin that needs to be terminated. PinName / <string> / Pin name of the unconnected pin. TerminationComponentName / <string> / Optional parameter. Definition Name of the custom component that is being terminated at the unconnected pin. TermName / <string> / "Interface Ports" "Page Ports" "Grounds" "<PinName of the termination component>" Terminationtype / <INT> / 0 (Interface Port), 1 (Page Port), 2 (Ground Port), 3 (Custom Termination)
Return Value	None

Python Syn-	AddPinCustomTerminations ()

tax	
Python Example	oEditor.AddPinCustomTerminations(["NAME:Connections", "Connection1:=", ["InstanceName:=" , "CompInst@etch_sline_s20_z70_1;1;1" , 'PinName:=" "Port1" , "TermName:=" , "Interface Ports" , "TermType:=", 1], "Connection2:=", ["InstanceName:=" , "CompInst@etch_sline_s20_z70_1;1;1" , 'PinName:=", "Port3" , "TermName:=" , "Page Ports" , "TermType:=", 2], "Connection3:=", ["InstanceName:=" , "CompInst@etch_sline_s20_z70_1;1;1" , 'PinName:=" "Port7" , "TermComponent:=" , "test" , "TermName:=" , "Port1" , "TermType:=" , 4], "Connection4:=", ["InstanceName:=" , "CompInst@etch_sline_s20_z70_1;1;1" , 'PinName:=" "Port10","TermComponent:=" , "oneport" , "TermName:=" , "Port1" , "TermType:=" , 4]]

VB Syntax	AddPinCustomTerminations()
VB Example	oEditor.AddPinCustomTerminationsArray("NAME:Connections", "Connection1:=", Array("InstanceName:=" , _"CompInst@etch_sline_s20_z70_1;1;1" , "PinName:=", "Port1" , "TermName:=", _"Grounds" , "TermType:=", 3), "Connection2:=", Array("InstanceName:=" , _"CompInst@etch_sline_s20_z70_1;1;1" , "PinName:=" , "Port7" , "TermComponent:=" , _ "oneport" , "TermName:=" , "Port1" , "TermType:=" , 4))

AlignHorizontal (Schematic Editor)

Align items horizontally

UI Access	Draw>Align Horizontal
Parameters	<table><tr><th>Name</th><th>Type</th><th>Description</th></tr><tr><td><Selections></td><td>Array</td><td>Array("NAME:Selections", _ "Page:=", page number, _ // [Opt=1] Page number "Selections:=", IDs to modify)</td></tr><tr><td><AlignParameters></td><td>Array</td><td>Array("NAME:AlignParameters", _ "Disconnect:=", bool _ // [Opt=0] Should wires disconnect "Rubberband:=", bool) _ // [Opt=1] Should wires staircase Note: Alignment occurs relative to the first item in ids</td></tr></table>
Return Value	Value

Note:

For more information about the component name, schematic IDs, and formats, see the section Editor Scripting IDs.

Python Syntax	AlignHorizontal()
Python Example	oEditor.AlignHorizontal (["NAME:Selections", "Selections:=", [_ "CompInst@R;2;4", "CompInst@C;1;1"]], ["NAME:AlignParameters", "Disconnect:=", _

	false, "Rubberband:=", false])

VB Syntax	AlignHorizontal()
VB Example	oEditor.AlignHorizontal Array("NAME:Selections", "Selections:=", Array(_ "CompInst@R;2;4", "CompInst@C;1;1")), Array("NAME:AlignParameters", "Disconnect:=", _ false, "Rubberband:=", false)

AlignVertical (Schematic Editor)

Align items vertically

UI Access	Draw>Align Vertical		
Parameters	Name	Type	Description
	<Selections>	Array	Array("NAME:Selections", _ "Page:=", page number, _ // [Opt=1] Page number "Selections:=", IDs to modify)
	<Align Parameters>	Array	Array("NAME:AlignParameters", _ "Disconnect:=", bool _ // [Opt=0] Should wires disconnect "Rubberband:=", bool) _ // [Opt=1] Should wires staircase Note: Alignment occurs relative to the first item in ids
Return Value	None		

> **Note:**
>
> For more information about the component name, schematic IDs, and formats, see the section Editor Scripting IDs.

Python Syntax	AlignVertical()
Python Example	`oEditor.AlignVertical (["NAME:Selections", "Selections:=", ["CompInst@R;2;4", _ "CompInst@C;1;1"]], ["NAME:AlignParameters", "Disconnect:=", false, "Rubberband:=", _ false])`

VB Syntax	AlignVertical()
VB Example	`oEditor.AlignVertical Array("NAME:Selections", "Selections:=", Array("CompInst@R;2;4", _ "CompInst@C;1;1")), Array("NAME:AlignParameters", "Disconnect:=", false, "Rubberband:=", _ false)`

BringToFront (Schematic Editor)

Bring the selected object to the front of the view

<table>
<tr><td>UI Access</td><td>Draw>Bring to Front</td></tr>
<tr><td>Parameters</td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><Selections></td><td>Array</td><td>["NAME: Selections", "Selections:=",[<Selected Components>]]

<Selected Components>

CompInst@DefName; ID, schematic ID , CompInst@DefName; ID, schematic ID

CompInst@R;1,2</td></tr>
</table>
</td></tr>
<tr><td>Return Value</td><td>None</td></tr>
</table>

Note:

For more information about the component name, schematic IDs, and formats, see the section Editor Scripting IDs.

Python Syntax	BringToFront( )
Python Example	`oEditor.BringToFront(` `["NAME:Selections", "Selections:=", ["SchObj@1"]])`

VB Syntax	BringToFront( )
VB Example	`oEditor.BringToFront` `Array("NAME:Selections", "Selections:=", Array("SchObj@1"))`

CloseEditor (Schematic Editor)

`Close an Editor`

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

> **Note:**
>
> For more information about the component name, schematic IDs, and formats, see the section Editor Scripting IDs.

Python Syntax	CloseEditor()
Python Example	`oDefinitionEditor.CloseEditor()`

VB Syntax	CloseEditor
VB Example	`oDefinitionEditor.CloseEditor`

Copy (Schematic Editor)

`GeneralCopy`

```
// Copy items for pasting
Copy(
Array("NAME:Selections", _
"Page:=", page number, _ // [Opt=1] Page number
"Selections:=", IDs to modify))
```

> **Note:**
>
> For more information about the component name, schematic IDs, and formats, see the section Editor Scripting IDs.

Python Syntax	Copy( )
Python Example	`oEditor.Copy (["NAME:Selections", "Selections:=", ["CompInst@R;2;37", _ "CompInst@C;1;1"]])`

CopyData [Schematic Editor]

Use: Copy graphic object or component properties for pasting.

Command: **Edit> Copy Data** (or Right-click on graphic object or component and select **Copy Data**)

Syntax: CopyData Array("NAME:Selections", "Selections:=", Array(<selections>))

Return Value: None.

Parameters: <selections>

Type: Comma-separated list of strings

Identifiers for each selected item to be copied, in the form CompInst@<id>, where id is the compname;ID;schematicID for each element.

Example: `Set oEditor = oDesign.SetActiveEditor("SchematicEditor")`

```
oEditor.CopyData Array("NAME:Selections", "Selections:=", Array("CompInst@Cap;1;1"))
```

CopySubdesign [Schematic Editor]

Copy components that represent subdesigns, as well as their represented designs, for pasting. The designs, when pasted, will be independent, with unique names based on the original design.

UI Access	(Right-click on hierarchical component) **Copy as New Design**
Parameters	<table><tr><th>Name</th><th>Type</th><th>Description</th></tr><tr><td><selections></td><td>Comma-separated list of strings</td><td>Identifiers for each selected item to be copied, in the form CompInst@<id>, where id is the compname;ID;schematicID for each element.</td></tr></table>
Return Value	None

Python Syntax	CopySubdesign (["NAME:Selections", "Selections:=", [<selections>]])
Python Example	`oEditor.CopySubdesign ([` `"NAME:Selections", "Selections:=", [` `"CompInst@Simplorer2;23;17"]])`

VB Syntax	CopySubdesign Array("NAME:Selections", "Selections:=", Array(<selections>))
VB Example	`oEditor.CopySubdesign Array(` `"NAME:Selections", "Selections:=",` `Array("CompInst@Simplorer2;23;17"))`

CreatePage (Schematic Editor)

Create a page at the end of the existing pages

UI Access	Schematic>New Page
Parameters	Name / Type / Description <PageName> / String / Name of the page that has been created <PageNum> / Integer / Page Number
Return Value	None

> **Note:**
>
> For more information about the component name, schematic IDs, and formats, see the section Editor Scripting IDs.

Python Syntax	CreatePage(string name, // Page name [out,retval] int num) // Page number
Python Example	`oEditor.CreatePage ("TestPage1")`

VB Syntax	CreatePage(string name, // Page name [out,retval] int num) // Page number
VB Example	`oEditor.CreatePage "TestPage1"`

Cut (Schematic Editor)

Cut Page Selection

UI Access	Edit>Cut
Parameters	<table><tr><th>Name</th><th>Type</th><th>Description</th></tr><tr><td><Selections></td><td>Array</td><td>`Array("NAME:Selections", _` "Page:=", page number, _ // [Opt=1] Page number "Selections:=", IDs to modify))</td></tr></table>
Return Value	None

> **Note:**
>
> For more information about the component name, schematic IDs, and formats, see the section Editor Scripting IDs.

Python Syntax	Cut()
Python Example	`oEditor.Cut (["NAME:Selections", "Selections:=", ["CompInst@R;10;93"]])`

VB Syntax	Cut()
VB Example	oEditor.Cut Array("NAME:Selections", "Selections:=", Array("CompInst@R;10;93"))

DeactivateOpen (Schematic Editor)

Disable items and replace with an open circuit

UI Access	Edit>Deactivate (Open)		
Parameters	Name	Type	Description
	<Selections>	Array	Array("NAME:Selections", _ "Page:=", page number, _ // [Opt=1] Page number "Selections:=", IDs to modify))
Return Value	None		

Note:

For more information about the component name, schematic IDs, and formats, see the section Editor Scripting IDs.

Python Syntax	DeactivateOpen()
Python Example	oEditor.DeactivateOpen (["NAME:Selections", "Selections:=", ["CompInst@R;2;37"]])

VB Syntax	DeactivateOpen()
VB Example	oEditor.DeactivateOpen Array("NAME:Selections", "Selections:=", Array("CompInst@R;2;37"))

DeactivateShort (Schematic Editor)

Disable items and replace with a closed circuit

UI Access	Edit>Deactivate (Short)		
Parameters	Name	Type	Description
	<Selections>	Array	Array("NAME:Selections", _ "Page:=", page number, _ // [Opt=1] Page number "Selections:=", IDs to modify))
Return Value	None		

Note:

For more information about the component name, schematic IDs, and formats, see the section Editor Scripting IDs.

Python Syntax	DeactivateShort ()
Python Example	oEditor.DeactivateShort(["NAME:Selections", "Selections:=", [_ "CompInst@R;2;37"])

VB Syntax	DeactivateShort ()
VB Example	`oEditor.DeactivateShort Array("NAME:Selections", "Selections:=", Array( _` `"CompInst@R;2;37"))`

Delete (Schematic Editor)

Delete items

UI Access	Edit>Delete		
Parameters	Name	Type	Description
	<Selections>	Array	`Array("NAME:Selections", _` "Page:=", page number, _ // [Opt=1] Page number "Selections:=", IDs to modify))
Return Value	None		

Note:

For more information about the component name, schematic IDs, and formats, see the section Editor Scripting IDs.

Python Syntax	Delete()
Python Example	`oEditor.Delete (["NAME:Selections", "Selections:=",` `["CompInst@R;10;93"]])`

VB Syntax	Delete()
VB Example	`oEditor.Delete Array("NAME:Selections", "Selections:=", Array("CompInst@R;10;93"))`

DeletePage (Schematic Editor)

To reduce the number of pages in the schematic

UI Access	Schematic>Remove Page
Parameters	Name | Type | Description None | |
Return Value	None

> **Note:**
>
> For more information about the component name, schematic IDs, and formats, see the section Editor Scripting IDs.

Python Syntax	DeletePage(<*pagenum*>, <*pagenum*>*)
Python Example	`num = oEditor.DeletePage([1, 4])`

VB Syntax	DeletePage Array(<*pagenum*>[, <*pagenum*>]*)
VB Example	`Dim oEditor`

	`Set num = oEditor.DeletePage Array(1, 4)`

ElectricRuleCheck (Schematic Editor)

Check Electric Rule

UI Access	Schematic>Electric Rule Check
Parameters	<table><tr><th>Name</th><th>Type</th><th>Description</th></tr><tr><td><ElectricRules></td><td>Array</td><td>Array(string, ...), bool) Array Array of strings containing names of electric rules to perform. Valid names are: "PinRule": Checks for components with unconnected pins "OutputPinRule": Checks for nets with more than one output pin bool True if this should check all subcircuits False if this should just check this circuit</td></tr></table>
Return Value	None

Note:

For more information about the component name, schematic IDs, and formats, see the section Editor Scripting IDs.

Python Syntax	ElectricRuleCheck()
Python Example	`oEditor.ElectricRuleCheck (["PinRule", "OutputPinRule", "OverlapCompRule"], True)`

VB Syntax	ElectricRuleCheck()
VB Example	`oEditor.ElectricRuleCheck Array("PinRule", "OutputPinRule", "OverlapCompRule"), True`

ExportImage (Schematic Editor)

To export a picture for a specified page of the current design to a file. The image size can also be specified. The filename extension determines the type of image exported.

UI Access	None.
Parameters	

Name	Type	Description
<filename>	String	The name of the file, with format-specific extension. Extensions supported are: bmp, gif, jpg, jpeg, png, tif, tiff.
<pagenum>	Integer	The page number of the current schematic. Page numbers start at 1. If the number is out of range, the current schematic page will be printed.
<dx>	Integer	The width of the image. If `dx` is less than 160, 160 will be used for the width.
<dy>	Integer	The height of the image. If `dy` is less than 160, 160 will be used for the height.

Return Value	None

Note:

For more information about the component name, schematic IDs, and formats, see the section Editor Scripting IDs.

Python Syntax	ExportImage (<filename>, <pagenum>, <dx>, <dy>)
Python Example	`oEditor.ExportImage ("c:\mysch.png", 1, 800, 400)`

VB Syntax	ExportImage (<filename>, <pagenum>, <dx>, <dy>)
VB Example	`oEditor.ExportImage "c:\mysch.png", 1, 800, 400`

FindElements (Schematic Editor)

Select elements based on properties

UI Access	Edit>Find Elements

Parameters

Name	Type	Description
<propname>	String	The name of the property, or "*" to search through all properties
<propvalue>	String	The value of the property, or a substring
<criterion>	Integer	0 = Contains 1 = Does not contain 2 = Exact match
<types>	Integer	2 - components only 4 - graphic objects only

			8 - wires only 16 - ports only Add the numbers to look through more types - 30 is a common number, searching through everything
	<match>	Boolean	True = match all of the properties specified False = match any of the properties specified
	<case>	Boolean	True = case-sensitive False = not case-sensitive
	<sub>	Boolean	True = search subcircuits as well as current schematic False = only search current schematic
	<only>	Boolean	True = search within current selections False = search schematic(s)
Return Value	Array of strings, the names of the elements selected		

Note:

For more information about the component name, schematic IDs, and formats, see the section Editor Scripting IDs.

Python Syntax	FindElements(["NAME:SearchProps", "Prop:=", [<propname>, <propvalue>, <criterion>)[, [<propname>, <propvalue>, <criterion>]]*], ["NAME:Parameters", "Filter:=", <types>, "MatchAll:=", <match>, "MatchCase:=", <case>, "SearchSubCkt:=", <sub>, "SearchSelectionOnly:=", <only>])
Python Example	`oEditor.FindElements(` `[`

	"NAME:SearchProps", "Prop:=", ["CompName","",0]], ["NAME:Parameters", "Filter:=" , 30, "MatchAll:=" , False, "MatchCase:=" , False, "SearchSubCkt:=" , True, "SearchSelectionOnly:=" , False])

VB Syntax	FindElements(Array("NAME:SearchProps", "Prop:=", Array(<propname>, <propvalue>, <criterion>)[, Array (<propname>, <propvalue>, <criterion>)]*), Array("NAME:Parameters", "Filter:=", <types>, "MatchAll:=", <match>, "MatchCase:=", <case>, "SearchSubCkt:=", <sub>, "SearchSelectionOnly:=", <only>)
VB Example	eltarray = oEditor.FindElements Array("NAME:SearchProps", "Prop:=", Array("InstanceName", "R", 0)), Array("NAME:Parameters", "Filter:=",_2, "MatchAll:=", true, "MatchCase:=", false,

	`"SearchSubCkt:=", false, "SearchSelectionOnly:=", false)`

FitToBorder [Schematic Editor]

Use: To reset the schematic view to just include the page border, if any.

UI Access	View>Fit Border
Parameters	Name | Type | Description None | |
Return Value	None

Python Syntax	FitToBorder()
Python Example	`oEditor.FitToBorder()`

VB Syntax	FitToBorder
VB Example	`oEditor.FitToBorder`

FlipHorizontal (Schematic Editor)

Flip items about the horizontal (x) axis

UI Access	NA

Parameters	Name	Type	Description
	<Selections>	Array	`Array("NAME:Selections", _` "Page:=", page number, _ // [Opt=1] Page number "Selections:=", IDs to modify))
	<FlipParameters>	Array	`Array("NAME: FlipParameters", _` "Disconnect:=", bool _ // [Opt=0] Should wires disconnect "Rubberband:=", bool) _ // [Opt=1] Should wires staircase
Return Value	None		

Note:

For more information about the component name, schematic IDs, and formats, see the section Editor Scripting IDs.

Python Syntax	FlipHorizontal(
Python Example	`oEditor.FlipHorizontal(` `["NAME:Selections", "Selections:=", [ _` `"CompInst@R;2;4", "CompInst@C;1;1"]],` `["NAME:FlipParameters",` `"Disconnect:=", true, "Rubberband:=", false])`

VB Syntax	FlipHorizontal(

VB Example	oEditor.FlipHorizontal Array("NAME:Selections", "Selections:=", Array(_ "CompInst@R;2;4", "CompInst@C;1;1")), Array("NAME:FlipParameters", "Disconnect:=", true, "Rubberband:=", false)

FlipVertical (Schematic Editor)

Flip items about the vertical (y) axis

UI Access	NA
Parameters	<table><tr><th>Name</th><th>Type</th><th>Description</th></tr><tr><td><Selections></td><td>Array</td><td>Array("NAME:Selections", _ "Page:=", page number, _ // [Opt=1] Page number "Selections:=", IDs to modify))</td></tr><tr><td><FlipParameters></td><td>Array</td><td>Array("NAME:FlipParameters", _ "Disconnect:=", bool _ // [Opt=0] Should wires disconnect "Rubberband:=", bool) _ // [Opt=1] Should wires staircase</td></tr></table>
Return Value	None

> **Note:**
>
> For more information about the component name, schematic IDs, and formats, see the section Editor Scripting IDs.

Python Syntax	FlipVertical(
Python Example	oEditor.FlipVertical (["NAME:Selections", "Selections:=", ["CompInst@R;2;4", _ "CompInst@C;1;1"]], ["NAME:FlipParameters", "Disconnect:=", true, "Rubberband:=", false])

VB Syntax	FlipVertical(
VB Example	oEditor.FlipVertical Array("NAME:Selections", "Selections:=", Array("CompInst@R;2;4", _ "CompInst@C;1;1")), Array("NAME:FlipParameters", "Disconnect:=", true, "Rubberband:=", false)

GridSetup (Schematic Editor)

Changes the settings on the schematic grid

UI Access	Schematic>Grid Setup

Parameters

Name	Type	Description
<Options>	Array	Array("NAME:Options", _ "MajorGrid:=", string, _ // Major grid size with units "Divisions:=", int, _ // Number of minor grid divisions "MajorColor:=", int, _ // RGB Color of major grid lines

			"MinorColor:=", int, _ // RGB Color of minor grid lines "ShowGrid:=", bool, _ // Should the grid be shown? "SnapToGrid:=", bool, _ // Should objects snap to grid? "BackgroundColor:=", int, _ // RGB Color of the background "SaveAsDefault:=", bool))
Return Value	None		

Note:

For more information about the component name, schematic IDs, and formats, see the section Editor Scripting IDs.

Python Syntax	GridSetup()
Python Example	`oEditor.GridSetup([` `"NAME:Options", _` `"MajorGrid:=", "10mm", _` `"Divisions:=", 10, _` `"MajorColor:=", 0x00ff0000, _ # Red` `"MinorColor:=", 0x0000ff00, _ # Green` `"ShowGrid:=", true, _`

	"SnapToGrid:=", true, _ "BackgroundColor:=", 0x00ffffff, _ # White "SaveAsDefault:=", false])

VB Syntax	GridSetup()
VB Example	oEditor.GridSetup Array("NAME:Options", _ "MajorGrid:=", "10mm", _ "Divisions:=", 10, _ "MajorColor:=", 0x00ff0000, _ // Red "MinorColor:=", 0x0000ff00, _ // Green "ShowGrid:=", true, _ "SnapToGrid:=", true, _ "BackgroundColor:=", 0x00ffffff, _ // White "SaveAsDefault:=", false)

For example:

```
oEditor.GridSetup Array(
"NAME:Options", _
"MajorGrid:=", "10mm", _
```

```
"Divisions:=", 10, _
"MajorColor:=", 0x00ff0000, _ // Red
"MinorColor:=", 0x0000ff00, _ // Green
"ShowGrid:=", true, _
"SnapToGrid:=", true, _
"BackgroundColor:=", 0x00ffffff, _ // White
"SaveAsDefault:=", false)
```

Move (Schematic Editor)

Move items

<table>
<tr><td>UI Access</td><td>NA</td></tr>
<tr><td>Parameters</td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><Selections></td><td>Array</td><td>Array("NAME:Selections", _
"Page:=", page number, _ // [Opt=1] Page number
"Selections:=", IDs to modify))</td></tr>
<tr><td><MoveParameters></td><td>Array</td><td>Array("NAME:MoveParameters", _
"xdelta:=", double _ // X distance to move
"ydelta:=", double _ // Y distance to move
"Disconnect:=", bool _ // [Opt=0] Should wires disconnect
"Rubberband:=", bool) _ // [Opt=1] Should wires staircase</td></tr>
</table>
</td></tr>
<tr><td>Return Value</td><td>None</td></tr>
</table>

Note:

For more information about the component name, schematic IDs, and formats, see the section Editor Scripting IDs.

Python Syntax	Move()
Python Example	`oEditor.Move (["NAME:elements", "circle_0", "rect_2"], [0.0165613577023499, -0.001])`

VB Syntax	Move()
VB Example	`oEditor.Move Array("NAME:elements", "circle_0", "rect_2"), Array(0.0165613577023499, -0.001)`

NameNets (Schematic Editor)

Change names of nets on the schematic

UI Access	Schematic>Auto-Name Wires		
Parameters	Name	Type	Description
	None		
Return Value	None		

Note:

For more information about the component name, schematic IDs, and formats, see the section Editor Scripting IDs.

Python Syntax	NameNets(bool display, // Show Net names or not ["NAME:NetNames", _ "string:=", _ #Old name of the net (e.g. net_1) string, _ #New name of the net (e.g. my_net) ...]) # Repeat for additional nets to change
Python Example	`oEditor.NameNets (True, ["NAME:NetNames",` `"Wire@net_2;41:1:=", "A[0]"])`

VB Syntax	NameNets(bool display, // Show Net names or not Array("NAME:NetNames", _ "string:=", _ // Old name of the net (e.g. net_1) string, _ // New name of the net (e.g. my_net) ...) // Repeat for additional nets to change

VB Example	`oEditor.NameNets true, Array("NAME:NetNames",` `"Wire@net_2;41:1:=", "A[0]")`

PageBorders (Schematic Editor)

> **Note:**
>
> This command has been replaced with the SetPageData command. Legacy scripts using this command will play back for the first page of a schematic only.

Sets up visible page borders on the schematic

```
PageBorders(
Array("NAME:Options", _
"PageSize:=", _
Array("x:=", string, _ // Width of page border with units
"y:=", string), _ // Height of page border with units
"PageMargins:=", _
Array("x:=", string, _ // Margin Width with units
"y:=", string), _ // Margin Height with units
"ZonesHoriz:=", int, _ // Number of Horizontal zones
"ZonesVert:=", int)) // Number of Vertical zones
```

> **Note:**
>
> For more information about the component name, schematic IDs, and formats, see the section Editor Scripting IDs.

Pan (Schematic Editor)

Pan Display

UI Access	View>Pan
Parameters	Name | Type | Description None | |
Return Value	None

> **Note:**
>
> For more information about the component name, schematic IDs, and formats, see the section Editor Scripting IDs.

Python Syntax	Pan([double, _ // Move the visible area by X meters double]) // Move the visible area by Y meters
Python Example	`oDefinitionEditor.Pan` `([-0.00922653869663931, 0.0068683990422214 7])`

VB Syntax	Pan(Array(double, _ // Move the visible area by X meters double)) // Move the visible area by Y meters
VB Example	`oDefinitionEditor.Pan` `Array(-0.00922653869663931, 0.00686839904222147)`

Paste (Schematic Editor)

Paste copied items

UI Access	Edit>Paste
Parameters	Name | Type | Description None | |
Return Value	None

> **Note:**
>
> For more information about the component name, schematic IDs, and formats, see the section Editor Scripting IDs.

Python Syntax	Paste(VARIANT attrs)# specified attributes
Python Example	`oEditor.Paste (["NAME:Attributes",` `"Page:=", 1, "X:=", -0.059131200000001, "Y:=",_` ` 0.030784799999999, "Angle:=", 0, "Flip:=",false])`

VB Syntax	Paste(VARIANT attrs); // specified attributes
VB Example	oEditor.Paste Array("NAME:Attributes", "Page:=", 1, "X:=", -0.059131200000001, "Y:=",_ 0.030784799999999, "Angle:=", 0, "Flip:=",false)

PasteData [Schematic Editor]

Use: Paste graphic object or component properties to another object or component.

Command: **Edit> Paste Data** (or Right-click on graphic object or component and select **Paste Data**)

Syntax: PasteData Array("NAME:Selections", "Selections:=", Array(<selections>), "SelectList:=", Array(<select_prop_display>, <select_parameters>), "Exclude:=", Array(<ExcludeList>))

Return Value: None.

Parameters: Selections

Type: Comma separated component instance list where user wants to paste data

Identifiers for each selected item to be copied, in the form CompInst@<id>, where id is the compname;ID;schematicID for each element.

SelectList

Type: Comma separated Boolean list where:

<select_prop_display> true to paste property displays

<select_parameters> true to paste parameters

<ExcludeList>

Type: Comma separated list of property names to exclude from being pasted.

Example: `Set oEditor = oDesign.SetActiveEditor("SchematicEditor")`

```
oEditor.PasteData Array("NAME:Selections", "Selections:=", Array("CompInst@Cap;2;3"),
"SelectList:=", Array(true, true), "Exclude:=", Array("IC"))
```

PasteDesign (Schematic Editor)

Paste a design that has already been copied to the clipboard into schematic as a subdesign.

Syntax:

PasteDesign(

pasteOption , // see below for values

Array("NAME:Attributes", _

"Page:=", int, _ // [Opt=1] Page number (one-based)

// Note: Page 1 always exists

"X:=", double, _ // X position of the object

"Y:=", double, _ // Y position of the object

"Angle:=", double, _ // Rotation angle (radians)

"Flip:=", bool) // True if mirrored

)

Return Value: Component is created and the returned value has the format CompInst@<CompName>;<compInstID>;<optional schematicID>

For instance a return value CompInst@CAP_;4;35 indicates that the component name is CAP_, its ID is 5, and its Schematic ID is 35. These details can be found in the Properties window when you select the created component. The component name and ID can be found in the Gen-eral tab and the Schematic ID can be found in the Symbols tab.

Parameters: pasteOption should be one of the following:

0 to link to the existing design that was copied

1 to create a new copy of the design that was copied but keep the original layers of the design being copied

2 to create a new copy of the design and merge the layers of the design being copied into the design receiving the copy

VB Example:

```
Dim oAnsoftApp
Dim oDesktop
Dim oProject
Dim oDesign
Dim oEditor
Dim oModule
Set oAnsoftApp = CreateObject("Ansoft.ElectronicsDesktop")
Set oDesktop = oAnsoftApp.GetAppDesktop()
oDesktop.RestoreWindow
Set oProject = oDesktop.SetActiveProject("SimpleCircuitPaste")
oProject.CopyDesign "D"
Set oDesign = oProject.SetActiveDesign("A")
Set oEditor = oDesign.SetActiveEditor("SchematicEditor")
oEditor.PasteDesign 1, Array("NAME:Attributes", "Page:=", 1, "X:=", 0.0762, "Y:=", _
0.0635, "Angle:=", 0, "Flip:=", false)
```

PushExcitations

Allows access to computed excitations for transient and linear frequency solutions. The script command can be accessed from three locations.

<table>
<tr><td>UI Access</td><td>

Layout Editor
Schematic Editor
Select a Nexxim solution in a 3D Layout design, right click, and choose Push Excitations.

</td></tr>
<tr><td>Parameters</td><td>
<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td><Reference Designation></td><td>String</td><td>"<refdes>"

This argument is empty when excitations are pushed from a Nexxim solution.</td></tr>
<tr><td><PushExcitations Parameters></td><td>Array</td><td>This parameter changes depending on whether the excitations comes from a transient or linear frequency solution. The keyword <code>"transient:="</code> indicates to AEDT which solution generated the excitations. If <code>"transient:="</code> is present, <code>"CalcThevenin ="</code> and its value are ignored.

For a linear frequency solution, use this array:

<code>Array("NAME:options",
"CalcThevenin =" , <true|false>,
"Sol:=", "<solution name>")</code>

If <code>CalcThevenin</code> is true, Thevenin’s equivalent is calculated. Parameters for the linear frequency solution do not include a Freq argument, so all frequencies from the solution are used.

For a transient solution, use:

<code>Array("NAME:options",
"transient:=", Array(</code></td></tr>
</table>
</td></tr>
</table>

			"start:=", <start time>, "stop:=", <stop time>, "maxHarmonics:=", <max harmonics>, "winType:=", <window>, ["widthPct:=", <width percentage>,] ["kaiser:=", <Kaiser value>,] ["correctCoherentGain:=", true]), "Sol:=", "<solution name>") winType can have the following values: • Rectangular • Bartlett • Blackman • Hamming • Hanning • Kaiser • Welch • Weber • Lanzcos
Return Value	None		

Note:

For more information about the component name, schematic IDs, and formats, see the section Editor Scripting IDs.

VB Syntax	PushExcitations "<refdes>", Array("NAME:options", "transient:=", ["CalcThevenin =" , <true>], Array("start:=", <start time>, "stop:=", <stop time>, "maxHarmonics:=", <max harmonics>, "winType:=", <window>, ["widthPct:=", <width percentage>,] ["kaiser:=", <Kaier value>,] ["correctCoherentGain:=", true]), "Sol:=", "<solution name>")
VB Example	For a transient solution: `Set oEditor = oDesign.SetActiveEditor("Layout")` `oEditor.PushExcitations "U3", Array("NAME:options", _` `"transient:=", Array("start:=", 0, "stop:=", 5E-005, _` `"maxHarmonics:=", 100, "winType:=", "Rectangular", _` `"widthPct:=", 100, "kaiser:=", 0, "correctCoherentGain:=",_` `true), "Sol:=", "Transient")` `Set oDesign = oProject.SetActiveEditor("Design1")` `oDesign.PushExcitations "", Array("NAME:options", _` `"transient:=", Array("start:=", 0, "stop:=", 1E-08, _` `"maxHarmonics:=", 100, "winType:=", "Hamming", _` `"widthPct:=", 100, "kaiser:=", 0, "correctCoherentGain:=",_`

	true), "Sol:=", "Transient Setup 1") For a linear frequency solution: Set oEditor = oDesign.SetActiveEditor("SchematicEditor") oEditor.PushExcitations "S1", Array("NAME:options", _ "CalcThevenin:=", false, "Sol:=", "LinearFrequency")

Rotate (Schematic Editor)

Rotate items

UI Access	Draw>Rotate		
Parameters	Name	Type	Description
	<Selections>	Array	Array("NAME:Selections", _ "Page:=", page number, _ // [Opt=1] Page number "Selections:=", IDs to modify))
	<RotateParameters>	Array	Array("NAME:RotateParameters", _ "Disconnect:=", bool _ // [Opt=0] Should wires disconnect "Rubberband:=", bool _ // [Opt=1] Should wires staircase "Angle:=", double) // [Opt=/2] Angle to rotate Note: Rotation occurs around center of ids
Return Value	None		

Note:

When you record a script, the schematic editor's Rotate command shows the rotation angle for some older scripts in radians, while newer scripts show rotation angle in degrees.

```
oEditor.Rotate Array("NAME:Selections", "Selections:=", _
Array("CompInst@RES_;1;1:1")), Array("NAME:RotateParameters", _
"Angle:=", 1.5707963267949, "Disconnect:=", false, "Rubberband:=", false)
```

If "Degrees=xxx" is not shown, the rotation must be scaled.

Note:

For more information about the component name, schematic IDs, and formats, see the section Editor Scripting IDs.

Python Syntax	Rotate()
Python Example	`oEditor.Rotate (["NAME:Selections", "Selections:=", ["CompInst@R;2;4", "CompInst@C;1;1"]], ["NAME:RotateParameters","Degrees:=", 90, "Disconnect:=", _ false, "Rubberband:=", false])`

VB Syntax	Rotate()
VB Example	`oEditor.Rotate Array("NAME:Selections", "Selections:=",` `Array("CompInst@R;2;4", "CompInst@C;1;1")),` `Array("NAME:RotateParameters","Degrees:=", 90, "Disconnect:=", _` `false, "Rubberband:=", false)`

SelectAll (Schematic Editor)

Select all elements on the given page.

UI Access	Edit>Select All		
Parameters	Name	Type	Description
	None		
Return Value	None		

Note:

For more information about the component name, schematic IDs, and formats, see the section Editor Scripting IDs.

Python Syntax	SelectAll(int pageNum) # Page number The first page number is 1. If an invalid page number is passed in (i.e. 0, -1, etc), `SelectAll` will use the active page on the currently active view. If there is no active view, the first page, page 1, is used.
Python Example	`oEditor.SelectAll(1)`

VB Syntax	SelectAll(int pageNum) // Page number The first page number is 1. If an invalid page number is passed in (i.e. 0, -1, etc), `SelectAll` will use the active page on the currently active view. If there is no active view, the first page, page 1, is used.
VB Example	`oEditor.SelectAll 1`

SelectPage (Schematic Editor)

Select a page in the UI

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

> **Note:**
>
> For more information about the component name, schematic IDs, and formats, see the section Editor Scripting IDs.

Python Syntax	SelectPage(int page) # Page number
Python Example	`oEditor.SelectPage(1)`

VB Syntax	SelectPage(int page) // Page number

VB Example	`oEditor.SelectPage 1`

SendToBack

Send the selected object to the back of the view

UI Access	Draw>Send To Back
Parameters	Name | Type | Description <Object> | String | Object to send to the back.
Return Value	None

> **Note:**
>
> For more information about the component name, schematic IDs, and formats, see the section Editor Scripting IDs

Python Syntax	SendToBack (["NAME:Selections", "Selections:=", [<Object>, <Object>, ...]])
Python Example	`oDefinitionEditor.SendToBack(["NAME:Selections",` `"Selections:=", ["SchObj@10"]])`

VB Syntax	SendToBack Array("NAME:Selections", "Selections:=", Array (<Object>, <Object>, ...))
VB Example	`oDefinitionEditor.SendToBack Array("NAME:Selections",` `"Selections:=", Array( "SchObj@10"))`

SetPageData [Schematic Editor]

> **Note:**
>
> This command replaces the **PageBorders** command.

To add or remove page border, title block and page property data from the specified page. Title blocks will not be shown if the page border type is "None". Any number of title blocks may be specified, as well as any number of page properties. Some of the page properties are special and their values will be updated automatically as appropriate, such as Project, ProjectPath, Design, Number and Date.

UI Access	Schematic>Page Borders and Title Blocks...		
Parameters	Name	Type	Description
	<bordertype>	String	One of: "None", "Outline", "ANSI", "ISO", "DIN"
	<sizeX>	Double (with units)	Size of X axis
	<sizeY>	Double (with units)	Size of Y axis
	<marginX>	Double (with units)	Margin of X axis
	<marginY>	Double (with units)	Margin of Y axis
	<zonesH>	Integer	
	<zonesV>	Integer	
	<symbolName>	String	The name of the title block symbol.
	<libName>	String	The name of the library (with path from library directory root) that contains the title block symbol
	<Hsnap>	String	One of: "Left", "Right", "Center", "None"
	<Vsnap>	String	One of "Top", "Bottom", "Center", "None"
	<rotation>	String	One of "0", "90", "270"

	<propName>	String	The name of a page property followed by ":="
	<propVal>	String	The value of a page property
Return Value	None		

Python Syntax	SetPageData (*<pagenum>*, `["NAME:Border", "BorderType:=", <bordertype>, "PageSize:=", ["x:=", <sizeX>, "y:=", <sizeY>],` `"PageMargins:=", ["x:=", <marginX>,` `"y:=", <marginX>],"ZonesHoriz:=", <zonesH>,` `"ZonesVert:=", <zonesV>],` `["NAME:TitleBlocks"[, "TitleBlock:=",` `[<symbolName>, <libName>, <Hsnap>, <Vsnap>,` `<rotation>, -1)]*],` `["NAME:PageProps"[, <propName>, <propVal>]*])`
Python Example	`oEditor.SetPageData (1, ["NAME:Border",` `"BorderType:=", "ANSI", "PageSize:=", ["x:=", "11in", "y:=", "8.5in"],` `"PageMargins:=", ["x:=", "0.38in", "y:=", "0.25in"], "ZonesHoriz:=",` `2, "ZonesVert:=", 2], ["NAME:TitleBlocks", "TitleBlock:=", ["TitleBlk1",` `"TitleBlocks", "Right", "Bottom", "0", -1]], ["NAME:PageProps",`

	"Title:=", "Design Title", "Author:=", "I. M. User", "Number:=", "1", "Project:=", "Project8", "ProjectPath:=", "e:/projects/", "Design:=", "TwinBuilder1", "Date:=", "9:53:11 PM Jan 27, 2016"])

VB Syntax	SetPageData (<*pagenum*>, Array("NAME:Border", "BorderType:=", <*bordertype*>, "PageSize:=", Array("x:=", <*sizeX*>, "y:=", <*sizeY*>), "PageMargins:=", Array("x:=", <*marginX*>, "y:=", <*marginX*>),"ZonesHoriz:=", <*zonesH*>, "ZonesVert:=", <*zonesV*>), Array("NAME:TitleBlocks"[, "TitleBlock:=", Array(<*symbolName*>, <*libName*>, <*Hsnap*>, <*Vsnap*>, <*rotation*>, -1)]*), Array("NAME:PageProps"[, <*propName*>, <*propVal*>]*)
VB Example	Dim oEditor oEditor.SetPageData 1, Array("NAME:Border", "BorderType:=", "ANSI", "PageSize:=", Array("x:=", "11in", "y:=", "8.5in"), "PageMargins:=", Array("x:=", "0.38in", "y:=", "0.25in"), "ZonesHoriz:=", 2, "ZonesVert:=", 2), Array("NAME:TitleBlocks", "TitleBlock:=", Array("TitleBlk1", "TitleBlocks", "Right", "Bottom", "0", -1)), Array("NAME:PageProps",

	"Title:=", "Design Title", "Author:=", "I. M. User", "Number:=", "1", "Project:=", "Project8", "ProjectPath:=", "e:/projects/", "Design:=", "TwinBuilder1", "Date:=", "9:53:11 PM Jan 27, 2016")

ShowVariableBlock (Schematic Editor)

Use: Control display of Variable Text Block in Schematic Editor.

Command: Schematic Ribbon > Settings > Design Variables

Syntax: ShowVariableBlock("True") // Display Text Block

ShowVariableBlock("False") // Hide Text Block

Return Value: None.

Parameters: True // Display Text Block

False // Hide Text Block

Python Syntax	ShowVariableBlock ()
Python Example	oEditor ShowVariableBlock [("True")]

SortComponents (Schematic Editor)

Sorts all components, or a block of components, using the specified method

<table>
<tr><td>UI Access</td><td>Schematic>Sort Components</td></tr>
<tr><td>Parameters</td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><Type></td><td>Array</td><td>Array(type, _ // string specifying sort type
method)) // string specifying sort method type
// string argument specifying sort type: "All Components" or "Blocks"
// An empty string corresponds to "All Components" method
// string argument specifying sort method: "By Name", "Left to Right", or "Signal Flow"</td></tr>
</table>
</td></tr>
<tr><td>Return Value</td><td>None</td></tr>
</table>

Note:

For more information about the component name, schematic IDs, and formats, see the section Editor Scripting IDs.

Python Syntax	SortComponents ()
Python Example	`oEditor.SortComponents()`

VB Syntax	SortComponents ()
VB Example	`oEditor.SortComponents`

Wire (Schematic Editor)

Use: Wire specified pairs of pins in two selected component instances.

Command: Select two component instances in the Schematic Editor, then right click on one and choose **Wire**.

Syntax:

Wire (string firstComponent, _

string firstListOfPins, _// list of pins separated by space

string secondComponent, _

string secondListOfPins) //second list of pins separated by space

Return Value: None

VB Example: `oEditor.Wire "CompInst@etch_sline_s20_z70_1;10;76", "A1 A2", "Com-pInst@CX0805MRNPO9BB220;1;1", "B1 B2"`

ZoomArea (Schematic Editor)

Zoom Area

<table>
<tr><td>UI Access</td><td colspan="3">View>Zoom Area</td></tr>
<tr><td>Parameters</td><td colspan="3"><table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td><Size></td><td>Array</td><td>Array(double, _
//Base X value of the area to zoom into double),
_ // Base Y value of the area to zoom into
Array(double, _ // X size of the area to zoom
double)) // Y Size of the area to zoom
NOTE: All values in meters</td></tr>
</table></td></tr>
<tr><td>Return Value</td><td colspan="3">None</td></tr>
</table>

Note:

For more information about the component name, schematic IDs, and formats, see the section Editor Scripting IDs.

Python Syntax	ZoomArea()
Python Example	`oEditor.ZoomArea ([0.00307569800375911, 0.021101611633739], [-0.050433530091516, -0.0220260723256825])`

Python Syntax	ZoomArea()
Python Example	`oEditor.ZoomArea ([0.00307569800375911, 0.021101611633739], [-0.050433530091516, -0.0220260723256825])`

ZoomIn (Schematic Editor)

Zoom into the schematic

UI Access	View>Zoom In
Parameters	Name | Type | Description None | |
Return Value	None

Note:

For more information about the component name, schematic IDs, and formats, see the section Editor Scripting IDs.

Python Syntax	ZoomIn()
Python Example	`oEditor.ZoomIn()`

VB Syntax	ZoomIn()
VB Example	`oEditor.ZoomIn`

ZoomOut (Schematic Editor)

Zoom out from the schematic

UI Access	View>Zoom Out		
Parameters	Name	Type	Description
	None		
Return Value	None		

Note:

For more information about the component name, schematic IDs, and formats, see the section Editor Scripting IDs.

Python Syntax	ZoomOut()
Python Example	`oEditor.ZoomOut()`

VB Syntax	ZoomOut()
VB Example	`oEditor.ZoomOut`

ZoomPrevious (Schematic Editor)

Restore the zoom to the previous settings

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

> **Note:**
>
> For more information about the component name, schematic IDs, and formats, see the section Editor Scripting IDs.

Python Syntax	ZoomPrevious()
Python Example	`oEditor.ZoomPrevious()`

VB Syntax	ZoomPrevious()
VB Example	`oEditor.ZoomPrevious`

ZoomToFit (Schematic Editor)

Set the Zoom to Fit

UI Access	View>Fit Drawing
Parameters	Name | Type | Description None | |
Return Value	None

> **Note:**
>
> For more information about the component name, schematic IDs, and formats, see the section Editor Scripting IDs.

Python Syntax	ZoomToFit()
Python Example	`oEditor.ZoomToFit()`

VB Syntax	ZoomToFit()
VB Example	`oEditor.ZoomToFit`

Property Method List

This section presents the property script methods that are available.

ChangeProperty (Schematic Editor)

Change a property

UI Access	NA
Parameters	Name | Type | Description Name | Type | Description
Return Value	Value

Python Syntax	ChangeProperty()
Python Example	`oEditor.ChangeProperty(["NAME:AllTabs",` `["NAME:ComponentTab", ["NAME:PropServers", _` `"Wire@net_1;14:1"], ["NAME:ChangedProps",` `["NAME:NetName", "ExtraText:=", _`

	"net_10", "Name:=", "net_10", "SplitWires:=", false]]])

VB Syntax	ChangeProperty()
VB Example	oEditor.ChangeProperty Array("NAME:AllTabs", Array("NAME:ComponentTab", Array("NAME:PropServers", _ "Wire@net_1;14:1"), Array("NAME:ChangedProps", Array("NAME:NetName", "ExtraText:=", _ "net_10", "Name:=", "net_10", "SplitWires:=", false))))

GetEvaluatedPropertyValue (Schematic Editor)

Get value.

Command: None.

Syntax: GetEvaluatedPropertyValue<tabDescription, componentID, propName>

Return Value: Evaluated value of variable property in double format.

Parameters: tabDescription = name of property tab where property is found

componentID = id of component instance where property is found,

in the format: "CompInst@<name of component type>;<id of compInstance>"

propName = name of the variable property

VB Example:

```
dim info
```

```
evalValue = oEditor.GetEvaluatedPropertyValue("PassedParameterTab", "CompInst@CAP_;3", "C")
```

Notes:

1. This function is only available with the schematic editor.

2. Calling this function on non-numeric properties (e.g. Text properties) returns 0.

GetProperties (Schematic Editor)

To get the names of properties for a specified tab and element.

UI Access	NA		
Parameters	Name	Type	Description
	<tab>	String	One of six choices, "PassedParameterTab", "ComponentTab", "Component", "BaseElementTab", "Quantities", "Signals"
	<objectId>	String	Instance ID of object with the property
Return Value	Array of strings, the property names		

Python Syntax	GetProperties(<tab>, <objectId>)
Python Example	`comparray = oEditor.GetAllComponents()` `for i in comparray:` `  instval = oEditor.GetPropertyValue` `  ("PassedParameterTab",i,"InstanceName")` `  proparray = oEditor.GetProperties ("Quantities",i)` `  for j in proparray:`

	direction = oEditor.GetPropertyAttribute ("Quantities",i, j, "Direction") AddInfoMessage(str(j) + " = (direction) " + str(direction))

VB Syntax	GetProperties(<tab>, <objectId>)
VB Example	comparray = oEditor.GetAllComponents for each i in comparray instval = oEditor.GetPropertyValue ("PassedParameterTab",i,"InstanceName") proparray = oEditor.GetProperties ("Quantities",i) for each j in proparray direction = oEditor.GetPropertyAttribute ("Quantities",i, j, "Direction") MsgBox j + " = (direction) " + direction next next

GetPropertyAttribute [Schematic Editor]

To find the value of an attribute of a specified property.

UI Access	NA
Parameters	<table><tr><th>Name</th><th>Type</th><th>Description</th></tr><tr><td><tab></td><td>String</td><td>One of six choices, "PassedParameterTab", "ComponentTab", "Component", "BaseElementTab", "Quantities", "Signals"</td></tr><tr><td><objectId></td><td>String</td><td>Instance ID of object with the property</td></tr><tr><td><propName></td><td>String</td><td>The name of the property</td></tr><tr><td><attribute></td><td>String</td><td>Currently, only the "Direction" attribute is supported. Return values will be one of "In", "Out" or "InOut"</td></tr></table>
Return Value	A string, the attribute's value.

Python Syntax	GetPropertyAttribute(<tab>, <objectId>, <propName>, <attribute>)
Python Example	`comparray = oEditor.GetAllComponents()` `for i in comparray:` `instval = oEditor.GetPropertyValue` `("PassedParameterTab",i,"InstanceName")` `proparray = oEditor.GetProperties ("Quantities",i)` `for j in proparray:` `direction = oEditor.GetPropertyAttribute` `("Quantities",i, j, "Direction")` `AddInfoMessage(str(j)+ " = (direction) " +`

	str(direction))

VB Syntax	GetPropertyAttribute(<tab>, <objectId>, <propName>, <attribute>)
VB Example	comparray = oEditor.GetAllComponents for each i in comparray instval = oEditor.GetPropertyValue ("PassedParameterTab",i,"InstanceName") proparray = oEditor.GetProperties ("Quantities",i) for each j in proparray direction = oEditor.GetPropertyAttribute("Quantities",i, j, "Direction") MsgBox j + " = (direction) " + direction next next

GetPropertyValue (Schematic Editor)

Get a property

UI Access	NA		
Parameters	Name	Type	Description

	<tab>	String	Tab with the property
	<item>	String	Name of the object
	<propname>	String	Name of the property
	<value>	String	Value of the property
Return Value	None		

Python Syntax	GetPropertyValue(<tab>,<item>,<propname>,<value>)
Python Example	None

VB Syntax	GetPropertyValue(<tab>,<item>,<propname>,<value>)
VB Example	Please refer to the examples that can be found in C:\Program Files\...[Ansys Installation]...\Examples\Twin Builder\Applications\Scripting\Ex6_MathMod

SetPropertyValue (Schematic Editor)

Set a property.

UI Access	NA		
Parameters	Name	Type	Description
	<tab>	String	Tab with the property
	<item>	String	Name of the objects
	<propname>	String	Name of the property
	<value>	String	The new value of the property
Return Value	None		

Python Syntax	SetPropertyValue(<params>)
Python Example	None

VB Syntax	SetPropertyValue(<params>)
VB Example	Please refer to the examples that can be found in C:\Program Files\...[Ansys Installation]...\Examples\Twin Builder\Applications\Scripting\Ex6_MathMod

Information Method List

This section presents the information methods that are available.

GetCompInstanceFromRefDes (Schematic Editor)

GetComponentInfo (Schematic Editor)

GetComponentPins (Schematic Editor)

GetComponentPinInfo (Schematic Editor)

GetComponentPinLocation [Schematic Editor]

GetEditorName (Schematic Editor)

GetNetConnections (Schematic Editor)

GetNumPages [Schematic Editor]

GetPortInfo (Schematic Editor)

GetSelections (Schematic Editor)

GetSignals (Schematic Editor)

GetAllPorts

Get all ports in a design.

UI Access	NA
Parameters	Name | Type | Description None | |
Return Value	Ports in the design

Python Syntax	GetAllPorts()
Python Example	`oEditor.GetAllPorts()`

VB Syntax	GetAllPorts
VB Example	`Set oEditor = oDesign.SetActiveEditor("SchematicEditor")` `oEditor.GetAllPorts()`

GetCompInstanceFromRefDes (Schematic Editor)

Informational

UI Access	NA
Parameters	Name | Type | Description

	Component	String	Name of the Component
Return Value	Returns IDispatch for CompInstance		

Python Syntax	GetCompInstanceFromRefDes(string) # string is refDes for the component
Python Example	`GetCompInstanceFromRefDes("R1")`

VB Syntax	GetCompInstanceFromRefDes(string) // string is refDes for the component
VB Example	`GetCompInstanceFromRefDes("R1")`

GetComponentInfo (Schematic Editor)

Informational

UI Access	NA		
Parameters	Name	Type	Description
	<compID>	String	The Schematic component's ID, obtained from GetSelections or through FindElements
Return Value	Array of Strings, as follows: "Page=*number*"		

	"ComponentName=*string*" "GateCount=*number*" "LocationX=*number*" "LocationY=*number*" "BBoxLLx=*number*" "BBoxLLy=*number*" "BBoxURx=*number*" "BBoxURy=*number*" "Angle=*number*" (in degrees) "Flip=true *or* false"

Python Syntax	GetComponentInfo(<compID>)
Python Example	`For i in selectionArray:` `  compArray = oEditor.GetComponentInfo(i)` `  For Each k in compArray:` `  AddInfoMessage(str(k) +` `  "Got with GetComponentInfo")`

VB Syntax	GetComponentInfo<compID>

VB Example	`selectionArray = oEditor.GetSelections` `For Each i in selectionArray` `  compArray = oEditor.GetComponentInfo(i)` `  For Each k in compArray` `  MsgBox k, 0, "GetComponentInfo"` `  Next` `Next`

GetComponentPins (Schematic Editor)

Informational

UI Access	NA
Parameters	Name | Type | Description <compID> | String | The Schematic component's ID, obtained from GetSelections or through FindElements.
Return Value	Array of strings, which are the names of all the component's pins

Python Syntax	GetComponentPins(<compID>)
Python Example	`selectionArray = oEditor.GetSelections()` `for i in selectionArray` `  pinArray = oEditor.GetComponentPins(i)`

	for j in pinArray AddInfoMessage("Pin: " + str(j))

VB Syntax	GetComponentPins<compID>
VB Example	selectionArray = oEditor.GetSelections for Each i in selectionArray pinArray = oEditor.GetComponentPins(i) for each j in pinArray MsgBox j, 0, "GetComponentPins" Next Next

GetComponentPinInfo (Schematic Editor)

Informational - gets details about a pin on a schematic component.

UI Access	NA
Parameters	Name / Type / Description <compID> / String / The Schematic component's ID <pinName> / String / The pin name
Return Value	String values, as follows: "X=number" "Y=number"

	"Angle=number" (in degrees) "Flip=true or false" "WireId=netID or empty string" (netID is the identifier of the schematic net, and can be used in other informational methods)

Python Syntax	GetComponentPinInfo(<compID>, <pinName>)
Python Example	`info = oEditor.GetComponentPinInfo` `("1","N1")`

VB Syntax	GetComponentPinInfo <compID>, <pinName>
VB Example	`info = oEditor.GetComponentPinInfo` `("1","N1")`

GetComponentPinLocation [Schematic Editor]

Informational - gets the X or Y location of a pin on a schematic component. Invoke once with TRUE for "XorY" to get the X value of the pin's location in the schematic, and once with FALSE to get the Y value.

UI Access	NA
Parameters	

Name	Type	Description
<compID>	String	The Schematic component’s ID

	<pinName>	String	The pin name
	<XorY>	Boolean	TRUE gets the X value; FALSE gets the Y value.
Return Value	String values, as follows: "X=*number*" "Y=*number*"		

Python Syntax	GetComponentPinLocation <compID>, <pinname>, <XorY>
Python Example	`varx = oEditor.GetComponentPinLocation` `(compid, "N1", TRUE)`

VB Syntax	GetComponentPinLocation <compID>, <pinname>, <XorY>
VB Example	`varx = oEditor.GetComponentPinLocation` `(compid,"N1", TRUE)`

GetEditorName (Schematic Editor)

Get the name of the schematic editor.

UI Access	N/A
Parameters	None
Return Value	String containing the name of the schematic editor. “SchematicEditor”

Python Syntax	GetEditorName()
Python Example	`oEditor.GetEditorName()`

VB Syntax	GetEditorName([out, retval] string) // name of the editor
VB Example	`dim info` `info = oEditor.GetEditorName`

GetNetConnections (Schematic Editor)

Use: Informational.

Command: None.

Syntax: GetNetConnections(<netName>)

Return Value: Array of strings containing the connections for the net identified by the

netName argument. Strings in the array are in one of four formats:

"IPort portID x y pinName"

"GPort portID x y pinName"

"PagePort portID x y pinName"

"Component componentName compID x y pinName"

where *IPort, GPort, PagePort, and Component* are key words

that indicate what the rest of the string specifies:

IPort indicates that *portID* specifies the name of the interface port,

GPort indicates that *portID* specifies the name of the global port,

PagePort indicates that *portID* specifies the name of the page port,

Component indicates that *compID* specifies the component instance identifier,

x and *y* are the connection point,

pinName is the name of the connected pin,

componentName is the name of the component.

Parameters: <netName>

Type: String

The name of the net. Specifying "0" as the netName will return all objects connected to ground including all ground ports.

VB Example:

```
netArray = oEditor.GetNetConnections("net_1")
```

GetNumPages [Schematic Editor]

To determine how many pages there are in the schematic.

UI Access	NA
Parameters	Name | Type | Description None | |
Return Value	Integer, the number of pages in the schematic.

Python Syntax	GetNumPages()
Python Example	`num = oEditor. GetNumPages()`

VB Syntax	GetNumPages
VB Example	`Set num = oEditor. GetNumPages`

GetPortInfo (Schematic Editor)

Informational – get details about a port on a schematic component.

UI Access	NA
Parameters	<table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><portID></td><td>String</td><td>The Schematic port's ID, obtained from GetSelections or through FindElements.</td></tr></table>
Return Value	Array of strings, as follows: "Name=string" // Name of the port. "X=val" // Connection point X location. "Y=val" // Connection point Y location. "Angle=val" // Component angle. "Flip=true or false" // Whether the component is flipped or not. "WireId=string" // The identifier of the schematic net, can be used in other informational methods.

Python Syntax	GetPortInfo(<portID>)
Python Example	`selectionArray = oEditor.GetSelections()` `for i in selectionArray:` `info = oEditor.GetPortInfo(i)` `for j in info:` `AddInfoMessage(str("GetPortInfo: " + j))`

VB Syntax	GetPortInfo<portID>
VB Example	`selectionArray = oEditor.GetSelections` `For Each i in selectionArray` `info = oEditor.GetPortInfo(i)` `for each j in info` `MsgBox j, mbokonly, "GetPortInfo"` `Next` `Next`

GetSelections (Schematic Editor)

Returns an array of currently selected objects.

UI Access	N/A
Parameters	None.

Return Value	Array containing object IDs

Python Syntax	GetSelections()
Python Example	`oEditor.GetSelections()`

VB Syntax	GetSelections
VB Example	`oEditor.GetSelections`

GetSignals (Schematic Editor)

Informational

UI Access	NA
Parameters	<table><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><wirename></td><td>String</td><td>The wire's name, as seen in the UI, obtained in script through parsing return data from `GetWireInfo`.</td></tr></table>
Return Value	Array of strings which contains the signal names in the *wirename* argument. The *wirename* "Net1" would return an array with a single string "Net1". The *wirename* "WW[0-3],A,B" would return an array with six strings "WW[0]", "WW[1]", "WW[2]", "WW[3]", "A", :B".

Python Syntax	GetSignals(<wirename>)
Python Example	`selectionArray = oEditor.GetSelections()`

	for i in selectionArray: netArray3 = oEditor.GetWireInfo(i) for n in netArray3: wname = n[0:9] if (wname == "WireName:"): nn = len(n) wn = n[nn-10:] AddInfoMessage("Wirename, parsed from ID :" + str(wn)) sigs = oEditor.GetSignals(wn) for ll in sigs: AddInfoMessage("Element in array from GetSignals :" + str(ll))

VB Syntax	GetSignals(<wirename>)
VB Example	selectionArray = oEditor.GetSelections For Each i in selectionArray netArray3 = oEditor.GetWireInfo(i)

```
for each n in netArray3

wname = Left(n, 9)

If (wname = "WireName:") Then

nn = Len(n)

wn = Right(n, nn - 10)

MsgBox wn, 0, "Wirename, parsed from ID"

sigs = oEditor.GetSignals(wn)

for each ll in sigs
MsgBox ll, 0, "Element in array from GetSignals"

Next
End If
Next
Next
```

GetWireConnections (Schematic Editor)

Informational.

UI Access	NA
Parameters	Name | Type | Description None | |
Return Value	Array of strings containing the connections for the wire identified by the wireID argument. The strings in the array are in one of four formats: "InterfacePort=portname portID x,y" "GlobalPort=portname portID x,y" "PagePort=portname portID x,y" "ComponentPin=compId pinname x,y" where x,y is the connection point.

Python Syntax	GetWireConnections(<wireID>)
Python Example	`selectionArray = oEditor.GetSelections()` `for i in selectionArray:` `  AddInfoMessage("Selected element" + i)` `  netArray = oEditor.GetWireConnections(i)` `  for m in netArray:` `  AddInfoMessage("GetWireConnections" + str(m))`

VB Syntax	GetWireConnections(<wireID>)

VB Example	`selectionArray = oEditor.GetSelections` `For Each i in selectionArray` `MsgBox i, 0, "Selected element"` `netArray = oEditor.GetWireConnections(i)` `for each m in netArray` `MsgBox m, mbokonly, "GetWireConnections"` `Next` `Next`

GetWireInfo (Schematic Editor)

Informational.

UI Access	NA		
Parameters	Name	Type	Description
	<wire id>	String	The Schematic wire's ID, obtained from `GetSelections` or through `FindElements`.

Return Value	Array of strings containing the wire info in the wireID argument. The strings in the array are: "Page=number" "WireName=string" "SegmentCount=number" "IPortCount=number" "GPortCount=number" "PagePortCount=number" "PinCount=number" "BBoxLLx=val" "BBoxLLy=val" "BBoxURx=val" "BBoxURy=val"

Python Syntax	GetWireInfo(<wireID>)
Python Example	`selectionArray = oEditor.GetSelections()` `for i in selectionArray:` `  AddInfoMessage("Selected element" + str(i))` `  netArray2 = oEditor.GetWireInfo(i)` `  for n in netArray2:`

	AddInfoMessage("GetWireInfo" + str(n))

VB Syntax	GetWireInfo(<wireID>)
VB Example	selectionArray = oEditor.GetSelections For Each i in selectionArray MsgBox i, 0, "Selected element" netArray3 = oEditor.GetWireInfo(i) for each n in netArray3 MsgBox n, 0, "GetWireInfo" Next Next

GetWireSegments (Schematic Editor)

Informational.

UI Access	NA
Parameters	Name: <wireID> — Type: String — Description: The Schematic wire's ID, obtained from GetSelections or through FindElements

Return Value	Array of strings containing the wire segments in the wireID argument. The strings in the array are: "Segment=val,val val,val number" the "val" quantities are segment endpoints x1,y1 x2,y2 and the "number" is a schematic segment ID

Python Syntax	GetWireSegments (<wireID>)
Python Example	`selectionArray = oEditor.GetSelections()` `for i in selectionArray:` `  AddInfoMessage("Selected element" + str(i))` `  netArray2 = oEditor.GetWireSegments(i)` `  for n in netArray2:` `  AddInfoMessage("GetWireSegments" + str(n))`

VB Syntax	GetWireSegments (<wireID>)
VB Example	`selectionArray = oEditor.GetSelections` `For Each i in selectionArray` `  MsgBox i, 0, "Selected element"` `  netArray2 = oEditor.GetWireSegments(i)` `  for each n in netArray2`

	MsgBox n, 0, "GetWireSegments" Next Next

17 - Design Verification Script Commands

The Design Verification (DV) module controls the use of DV rule sets and runs in a Circuit project. The DV module is accessed via the design script object.

```
Set oDesign = oProject.GetActiveDesign()

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

The topics for this section include:

AddRuleSet

Use: Adds a rule set to the design

Command: Right-click **Design Verification** in the **Project Tree** and choose **Add Rule Set**

Syntax: AddRuleSet Array("NAME:<RuleSetName>",

"ScriptNames:=", <ScriptInfo>,

"ScriptActiveFlags:=", <ScriptFlags>)

Return Value: None

Parameters: <RuleSetName>:

<string> // name of the rule set to create

<ScriptInfo>:

Array(<string>, <string>,...) // names of scripts to be in the rule set

<ScriptFlags>:

sequence of "t" and "f" characters

t indicates a script that is active (used in when the rule set is run)

f indicates a script is not currently active (used when the rule set is run)

applied to the scripts as ordered in <ScriptInfo>

<string>

VB Example:

```
oModule.AddRuleSet Array("NAME:Rule Set 9", _
"ScriptNames:=", Array("And", _
"Find Shorts"), _
"ScriptActiveFlags:=", "tf")
```

AddRun

Use: Adds a run to a rule set already in the design

Command: Right-click on a rule set item under **Design Verification** in the **Project Tree** and choose **Add Run**

Syntax: AddRun <RuleSetName>,

Array("NAME:<RunName>",

"TargetType:=", <TargetTypeInfo>, // optional

"TargetObjects:=", <TargetObjectsInfo>, // optional

"IgnoredObjects:=", <IgnoredObjectsInfo>, // optional

"ArcTolerance:=", <ToleranceString>) // optional

Return Value: None

Parameters: **<RuleSetName>:**

<string> // name of an existing rule set

<RunName>:

<string> // name of the run to create

<TargetTypeInfo>:

<int> // 0 for entire layout (default if not specified)

// 1 for specified portion of layout

<TargetObjectsInfo>:

objects on which to perform the check

may specify if TargetTypeInfo is 1 (specified portion of layout)

"<ObjectName>, <ObjectName>, ..."

<ObjectName>:

<string> // name of a layout object

<IgnoredObjectsInfo>:

objects which to ignore when performing the check

may specify if TargetTypeInfo is 0 (entire layout)

"<ObjectName>, <ObjectName>, ..."

<ToleranceString>:

<string> // real number and units

// if not specified, the current layout arc tolerance is used

VB Example:

```
oModule.AddRun "Rule Set 10", _
Array("NAME:All", _
"TargetType:=", 0)
oModule.AddRun "Rule Set 10", _
Array("NAME:Selected Objs2", _
"TargetType:=", 1, _
"TargetObjects:=", "line_975,line_1015")
oModule.AddRun "Rule Set 10", _
Array("NAME:Objects to Ignore2", _
"TargetType:=", 0, _
"IgnoredObjects:=", "via_209,line_736")
```

DeleteRuleSet

Use: Deletes a rule set from the design

Command: Right-click on a rule set item under **Design Verification** in the **Project Tree** and choose **Delete**

Syntax: DeleteRuleSet <RuleSetName>

Return Value: None

Parameters: <RuleSetName>:

<string> // name of the rule set to delete

VB Example:

```
oModule.DeleteRuleSet "Rule Set 9"
```

DeleteRun

Use: Deletes a run from an existing a rule set

Command: Right-click on a run item under **Design Verification** in the **Project Tree** and choose **Delete**

Syntax: DeleteRun <RuleSetName>, <RunName>

Return Value: None

Parameters: <RuleSetName>:

<string> // name of an existing rule set

<RunName>:

<string> // name of the run to delete

VB Example:

```
oModule.DeleteRun "Rule Set 10", "All
```

EditRuleSet

Use: Edits a existing rule set

Command: Right-click on a rule set item under **Design Verification** in the **Project Tree** and choose **Properties**.

Syntax: EditRuleSet <ExistingRuleSetName>,

Array("NAME:<RuleSetName>",

"ScriptNames:=", <ScriptInfo>,

"ScriptActiveFlags:=", <ScriptFlags>)

Return Value: None

Parameters: <ExistingRuleSetName>:

<string> // name of the rule set to change

<RuleSetName>:

<string> // name of the rule set after change

<ScriptInfo>:

Array(<string>, <string>,...) // names of scripts to be in the rule set

<ScriptFlags>:

sequence of "t" and "f" characters

t indicates a script that is active (used in when the rule set is run)

f indicates a script is not currently active (used when the rule set is run)

applied to the scripts as ordered in <ScriptInfo>

<string>

VB Example:

```
oModule.EditRuleSet "Rule Set 9", _
Array("NAME:Rule Set 10", _
```

```
"ScriptNames:=", Array("And", _
"Find Shorts"), _
"ScriptActiveFlags:=", "tt")
```

EditRun

Use: Edits an existing run.

Command: Right-click on a run item under **Design Verification** in the **Project Tree** and choose **Properties**.

Syntax: EditRun <RuleSetName>,

<ExistingRunName>,

Array("NAME:<RunName>",

"TargetType:=", <TargetTypeInfo>, // optional

"TargetObjects:=", <TargetObjectsInfo>, // optional

"IgnoredObjects:=", <IgnoredObjectsInfo>, // optional

"ArcTolerance:=", <ToleranceString>) // optional

Return Value: None

Parameters: <RuleSetName>:

<string> // name of an existing rule set

<ExistingRunName>:

<string> // name of the run to change

<RunName>:

<string> // name of the run after the changes

<TargetTypeInfo>:

<int> // 0 for entire layout (default if not specified)

// 1 for specified portion of layout

<TargetObjectsInfo>:

objects on which to perform the check

may specify if TargetTypeInfo is 1 (specified portion of layout)

"<ObjectName>, <ObjectName>, ..."

<ObjectName):

<string> // name of a layout object

<IgnoredObjectsInfo>:

objects which to ignore when performing the check

may specify if TargetTypeInfo is 0 (entire layout)

"<ObjectName>, <ObjectName>, ..."

<ToleranceString>:

<string> // real number and units

// if not specified, the current layout arc tolerance is used

VB Example:

```
oModule.EditRun "Rule Set 10", _
"All", _
Array("NAME:New All", _
"TargetType:=", 0)
```

```
oModule.EditRun "Rule Set 10", _
"New All", _
Array("NAME:Selected Objs2", _
"TargetType:=", 1, _
"TargetObjects:=", "line_975,line_1015")
oModule.EditRun "Rule Set 10", _
"Selected Objs2", _
Array("NAME:Objects to Ignore2", _
"TargetType:=", 0, _
"IgnoredObjects:=", "via_209,line_736")
```

RenameRuleSet

Use: Renames a existing rule set

Command: Right-click on a rule set item under **Design Verification** in the **Project Tree** and choose **Rename**

Syntax: RenameRuleSet <ExistingRuleSetName>, <NewRuleSetName>

Return Value: None

Parameters: <ExistingRuleSetName>:

<string> // name of the rule set to change

<NewRuleSetName>:

<string> // new name for the rule set

VB Example: `oModule.RenameRuleSet "Rule Set 12", "Rule Set 30"`

RenameRun

Use: Renames an existing run.

Command: Right-click on a run item under **Design Verification** in the **Project Tree** and choose **Rename**

Syntax: RenameRun <RuleSetName>, <ExistingRunName>, <NewRunName>

Return Value: None

Parameters: <RuleSetName>:

<string> // name of an existing rule set

<ExistingRunName>:

<string> // name of the run to change

<NewRunName>:

<string> // new name for the run

VB Example:
```
oModule.RenameRun "Rule Set 30", "New All", "Sel Objects"
```

RunAllDV

Use: Executes all runs in the Design Verification Project Tree item.

Command: Right-click on **Design Verification** in the **Project Tree** and choose **Run All**

Syntax: RunAllDV

Return Value: None

Parameters: None

VB Example:
```
oModule.RunAllDV
```

RunAllRuleSetDV

Use: Executes all runs in an existing rule set

Command: Right-click on a rule set item under **Design Verification** in the **Project Tree** and choose **Run All**

Syntax: RunAllRuleSetDV <RuleSetName>

Return Value: None

Parameters: <RuleSetName>:

<string> // name of an existing rule set

VB Example: `oModule.RunAllRuleSetDV "Rule Set 30"`

RunDV

Use: Executes the specified run in an existing rule set

Command: Right-click on a run item under **Design Verification** in the **Project Tree** and choose **Run**

Syntax: RunDV <RuleSetName>, <RunName>

Return Value: None

Parameters: <RuleSetName>:

<string> // name of an existing rule set

<RunName>:

<string> // name of the run to execute

VB Example: `oModule.RunDV "Rule Set 30", "Run 2"`

This page intentionally
left blank.

18 - Nexxim Scripting

Nexxim scripting commands should be executed by the **oDesign** object.

For example,

oDesign.GetModule("DataBlock")

oModule.CommandName <args>

The topics for this section include:

Nexxim Netlist Scripting

The following sections present the Nexxim design level scripting commands that are available.

Analyze [Nexxim]

Simulates the given setup

Syntax: Analyze(STRING setup1) // setup name

VB Example:

```
oModule. Analyze <setup1>
```

Python Syntax	Analyze(<setupName>)
Python Example	`oDesign.Analyze("QuickEyeAnalysis")`

CopyEyeItemAsCommand

Use: Make a QuickEye copy of a VerifEye analysis, or a VerifEye copy of a QuickEye analysis.

Command: Select an analysis in the Project tree, then right-click and select **Copy As QuickEye** or **Copy As VerifEye**.

Syntax: CopyEyeItemAsCommand <itemPathList>

Return Value: Copy of the analysis.

Parameters: <itemPathList>

Type: Array of strings // Type of analysis from which to copy.

VB Example:

```
oDesign.CopyEyeItemAsCommand Array("eye_diagram_sch|Nexxim1|Analysis|VerifEyeAnalysis")
oDesign.CopyEyeItemAsCommand Array("eye_diagram_sch|Nexxim1|Analysis|QuickEyeAnalysis")
```

Python Syntax	CopyEyeItemAsCommand <itemPathList>
Python	

Example	oDesign.CopyEyeItemAsCommand(["eye_diagram_sch|Nexxim1|Analysis|VerifEyeAnalysis"]) oDesign.CopyEyeItemAsCommand(["eye_diagram_sch|Nexxim1|Analysis|QuickEyeAnalysis"])

DeleteDesignInstance

Use: Delete the design instance

Command: None

Syntax: DeleteDesignInstance <instance_name>

Return Value: None

Parameters: <instance_name>

Type: string

VB Example: oDesign.DeleteDesignInstance <instance_name>

Python Syntax	DeleteDesignInstance(<instanceName>)
Python Example	oProject.DeleteDesignInstance("Circuit1")

EditImportData

Use: Edit an imported solution

Command: None

Syntax: EditImportData <oldname> <newlink> <newpath> <newname> <data> <file>

Return Value: None

Parameters: <oldname>

Type: string

<newlink>

Type: int

<newpath>

Type: string

<newname>

Type: string

<data>

Type: array

<file>

Type: string

VB Example: oDesign.EditImportData <oldname> <newlink> <newpath> <newname> <data> <file>

EditNotes

Updates the notes for the design.

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

Python Syntax	EditNotes(*<DesignNotes>*)

Python Example	`oDesign.EditNotes("My design notes.")`

VB Syntax	EditNotes <*DesignNotes*>
VB Example	`oDesign.EditNotes "My design notes."`

ExportForSpice [Nexxim]

Export matrix solution data to a file in the given spice format.

Command: None

Syntax: ExportForSpice <theVariationKey> <sourceNames> <theType> <bandwidth> <fullwaveSpiceFilename> <lumpedElementFilename> <Unused> <Unused> <partialFractionFilename> <fittingError> <maximumOrder> <useCommonGround> <doPassivityCheck>

Return Value: None

Parameters: <theVariationKey>

Type: string

<sourceNames>

Type: array

<theType: 0=PSpice, 2=MaxwellSpice, 3=Spectre>

Type: long

<bandwidth: 0=Low Bandwidth, 1=Broad Bandwidth>

Type: long

<fullwaveSpiceFilename>

Type: string

<lumpedElementFilename>

Type: string

<Unused>

Type: string

<Unused>

Type: string

<partialFractionFilename>

Type: string

<fittingError [Default:0.5]>

Type: double

<maximumOrder [Default:200]>

Type: int

<useCommonGround [Default:FALSE]>

Type: boolean

<doPassivityCheck [Default:FALSE]>

Type: boolean

VB Example:

```
oDesign.ExportForSpice "", Array("LNA:LNA"), 0, 0, _
"C:/Documents and Settings/Marcia.BMT/My Documents/Code Modifications/Test" & _
" Designs/s-params_fws_fws.lib", _
```

```
"C:/Documents and Settings/Marcia.BMT/My Documents/Code Modifications/Test" & _
" Designs/s-params_fws_lfws.lib", "", "", "", 0.52, 210, 1, 1
```

Python Syntax	ExportForSpice(<theVariationKey> <sourceNames> <theType> <bandwidth> <fullwaveSpiceFilename> <lumpedElementFilename> <Unused> <Unused> <partialFractionFilename> <fittingError> <maximumOrder> <useCommonGround> <doPassivityCheck>)
Python Example	oDesign.ExportForSpice("", ["LNA:LNA"] , 0, 0, "C:/Documents and Settings/Marcia.BMT/MyDocuments/Code Modifications/Test" & " Designs/s-params_fws_fws.lib", "C:/Documents and Settings/Marcia.BMT/MyDocuments/Code Modifications/Test" & " Designs/s-params_fws_lfws.lib", "", "", "", 0.52, 210,

	1, 1)

ExportForHSpice [Nexxim]

Export matrix solution data to an HSpice subcircuit.

Command: None

Syntax: ExportForHSpice <theVariationKey> <sourceNames> <theType> <bandwidth> <fullwaveSpiceFilename> <lumpedElementFilename> <Unused> <Unused> <partialFractionFilename> <fitting Error> <maximumOrder> <useCommonGround> <doPassivityCheck>

Return Value: None

Parameters: <theVariationKey>

Type: string

<sourceNames>

Type: array

<theType: 1=HSPICE>

Type:long

<bandwidth: 0=Low Bandwidth, 1=Broad Bandwidth>

Type: long

<fullwaveSpiceFilename>

Type: string

<lumpedElementFilename>

Type: string

<Unused>

Type: string

<Unused>

Type: string

<partialFractionFilename>

Type: string

<fittingError [Default:0.5]>

Type: double

<maximumOrder [Default: 200]>

Type: int

<useCommonGround [Default: FALSE]>

Type: boolean

<doPassivityCheck [Default: FALSE]>

Type: boolean

VB Example:

```
oDesign.ExportForHSpice "", Array("Setup 1:Sweep 1"), 1, 0, "C:/Projects/MyTRL_fws.sp", "", "",
"", "", 0.5, 0, 200, 0, 0
```

Python Syntax	ExportForHSpice(<theVariationKey> <sourceNames> <theType> <bandwidth> <fullwaveSpiceFilename> <lumpedElementFilename> <Unused> <Unused> <partialFractionFilename> <fitting Error> <maximumOrder> <useCommonGround> <doPassivityCheck>)
Python	

Example	`oDesign.ExportForHSpice( "", ["Setup1:Sweep 1"], 1, 0, "C:/Projects/MyTRL_fws.sp", "", "", "", "", 0.5, 0, 200, 0, 0)`

ExportNetlist [Nexxim]

Exports a netlist solution.

UI Access	None
Parameters	<table><tr><th>Name</th><th>Type</th><th>Description</th></tr><tr><td><*solution*></td><td>String</td><td>The Circuit solution in the design. For Maxwell Circuit, the solution must be an empty string, "".</td></tr><tr><td><*fileName*></td><td>String</td><td>File and path to the netlist file. For Maxwell Circuit, the file name must be a SPH file.</td></tr></table>
Return Value	None.

Python Syntax	ExportNetlist(<setupName>,<fileName>)
Python Example	`oDesign.ExportNetlist("QuickEyeAnalysis", "SolutionData")`

VB Syntax	ExportNetlist <*solution*> <*fileName*>
VB Example	`oDesign.ExportNetlist "QuickEyeAnalysis" "SolutionData"`

GetModule [Nexxim]

Get the IDispatch for the specified module.

Command: None

Syntax: GetModule <"modulename"> <object_variable>

Return Value: Module IDispatch

Parameters: <"modulename">

Type: string

<object_variable>

Type: string

VB Example: `oDesign.GetModule "modulename" object_variable`

Python Syntax	GetModule(<moduleName>,<object_variable>)
Python Example	`oDesign.GetModule("modulename" object_variable)`

GetName [Nexxim]

Get the name of the active design.

Command: None

Syntax: GetName <namestring_variable>

Return Value: String

Parameters: <namestring_variable>

Type: string

VB Example: `oDesign.GetName namestring_variable`

Python Syntax	GetName(<namestring_variable>)
Python Example	`oDesign.GetName(namestring_variable)`

GetActiveEditor

Get the name of the active editor.

Command: None

Syntax: GetActiveEditor <object_variable>

Return Value: String

Parameters: <object_variable>

Type: string

VB Example: oDesign.GetActiveEditor <object_variable>

Note that GetActiveEditor returns NULL if the editor is open in a non-active window.

Python Syntax	GetActiveEditor(<object_variable>)
Python Example	`oDesign.GetActiveEditor(editor_string)` *

GetEditor [Nexxim]

Get the named editor without activating it.

Command: None

Syntax: GetEditor <editorName object_variable>

Return Value: String

Parameters: <editorName>

Type: string

<object_variable>

Type: string

VB Example: oDesign.GetEditor <editorName> <object_variable>

GetResultsDirectory [Nexxim]

Gives access to the directory containing the specified Nexxim solution files.

Command: None

Syntax: GetResultsDirectory ([in] BSTR solname, [in] BSTR variation, [in] BOOL finalDir, [out, retval] BSTR* dirname)

Return Value: <dirname>

Type: String // Directory name. (The returned directory path does not contain a trailing slash.).

Parameters: <solname>

Type: string

<variation>

Type: string

<finalDir>

Type: boolean // True if the final directory is desired; False if the working/temporary directory is desired.

VB Example: `Dim dirName`

Set oDesign = oProject.GetActiveDesign() dirName = oDesign.GetResultsDirectory ("Trans", "res='1000'", false)

Python Syntax	dirname = GetResultsDirectory (< solname>, <variation>,<finalDir>,)
Python Example	`dirName = oDesign.GetResultsDirectory("Trans", "res='1000'", False)`

ImportData [Nexxim]

Imports a network solution

Command: None

Syntax: ImportData <data> <filename> <linktofile>

Return Value: None

Parameters: `<data>`

Type: string // data to be imported

<filename>

Type: string // name of the file

<linktofile>

Type: int // link to the file

VB Example: `oDesign.ImportData <data> <filename> <linktofile>`

Python Syntax	ImportData(<data>,<file>, <customization>)

Python Example

```
oDesign.ImportData(
   [
      "NAME:NPortData",
      "Description:="           , "",
      "ImageFile:="             , "",
      "SymbolPinConfiguration:=", 0,
      [
         "NAME:PortInfoBlk"
      ],
      [
         "NAME:PortOrderBlk"
      ],
      "filename:="      , "E:/R18Examples/Signal Integrity/models/Serial_Chan-
nel/connector_model.s4p",
      "numberofports:=" , 4,
      "sssfilename:="           , "",
      "sssmodel:="              , False,
      "PortNames:="             , ["Port1","Port2","Port3","Port4"],
      "domain:="                , "frequency",
```

```
        "datamode:="              , "Link",
        "devicename:="            , "",
        "SolutionName:="  , "connector_model",
        "displayformat:=" , "MagnitudePhase",
        "datatype:="              , "SMatrix",
        [
           "NAME:DesignerCustomization",
           "DCOption:="           , 0,
           "InterpOption:="       , 0,
           "ExtrapOption:="       , 1,
           "Convolution:="        , 0,
           "Passivity:="          , 0,
           "Reciprocal:="         , False,
           "ModelOption:="        , "",
           "DataType:="           , 1
        ],
        [
           "NAME:NexximCustomization",
           "DCOption:="           , 3,
           "InterpOption:="       , 1,
           "ExtrapOption:="       , 3,
```

```
        "Convolution:="                , 0,
        "Passivity:="          , 0,
        "Reciprocal:="         , False,
        "ModelOption:="                , "",
        "DataType:="           , 2
    ],
    [
        "NAME:HSpiceCustomization",
        "DCOption:="           , 1,
        "InterpOption:="       , 2,
        "ExtrapOption:="       , 3,
        "Convolution:="        , 0,
        "Passivity:="          , 0,
        "Reciprocal:="         , False,
        "ModelOption:="        , "",
        "DataType:="           , 3
    ],
    "NoiseModelOption:="       , "External"
], "", False)
```

ImportDataFilePath [Nexxim]

Returns the file path of the imported solution given an imported solution name. If the solution is not found it returns an empty string.

Command: None

Syntax: ImportDataFilePath <solution_name>

Return Value: File path of imported solution, empty string if solution not found in the design.

Type: string

Parameters: `<solution_name>`

Type: string

VB Example: `oDesign.ImportDataFilePath <solution_name>`

Python Syntax	ImportDataFilePath(<solution_name>)
Python Example	`solution_path_variable = oDesign.ImportDataFilePath("MySolution")`

InsertDesign [Nexxim]

Insert a design

Command: None

Syntax: InsertDesign <type>, <name>, <stationerypath>, <parentname>

Return Value: None

Parameters: `<type>`

Type: string

<name>

Type: string

<stationerypath>

Type: string

<parentname>

Type: string

VB Example: `oDesign.InsertDesign <type>, <name>, <stationerypath>, <parentname>`

where, <type> is NULL for top-level design

Python Syntax	InsertDesign(<typeName> <designName> ,<pathName>,<parentName>)
Python Example	`oProject.InsertDesign("Circuit Design", "Circuit1", "", "")`

PasteItemCommand

Paste tree items, such as Altrasim Solution and Solve Setups pasted to the Analysis Tree, or Frequency Sweeps pasted to the Solve Setup Tree.

Command: Right-click on the Analysis item in the Project tree and select Paste.

Syntax: PasteItemCommand <ItemPath>

Return Value: None

Parameters: `<ItemPath>`

Type: string

VB Example: `oDesign.PasteItemCommand "Project1|Nexxim1|Analysis"`

Python Syntax	PasteItemCommand(<ItemPath>)
Python Example	`oDesign.PasteItemCommand("Project1|Nexxim1|Analysis")`

Redo [Nexxim]

Redo the last operation

Command: Edit>Redo

Syntax: Redo

Return Value: None

Parameters: None

VB Example: `oDesign.Redo`

Python Syntax	Redo()
Python Example	`oDesign.Redo()`

RenameDesignInstance [Nexxim]

Rename the design instance

Command: None

Syntax: RenameDesignInstance <oldname> <newname>

Return Value: None

Parameters: <oldname>

Type: string

<newname>

Type: string

VB Example: oDesign.RenameDesignInstance <oldname> <newname>

Python Syntax	RenameDesignInstance(<oldName>,<newName>)
Python Example	oDesign.RenameDesignInstance("Circuit1", "TestCircuit1")

RenameImportData [Nexxim]

Rename an imported solution

Command: None

Syntax: RenameImportData <oldname> <newname>

Return Value: None

Parameters: <oldname>

Type: string

<newname>

Type: string

VB Example: oDesign.RenameImportData <oldname> <newname>

Python Syntax	RenameImportData(<oldName>,<newName>)
Python Example	`oDesign.RenameImportData("Solution", "ConnectorData")`

SetActiveEditor [Nexxim]

Set the active editor to the named one. (Activates window or opens window if necessary).

Command: None

Syntax: SetActiveEditor <editorName> <object_variable>

Return Value: None

Parameters: <editorName>

Type: string

<object_variable>

Type: string

VB Example: oDesign.SetActiveEditor <editorName> <object_variable>

Python Syntax	SetActiveEditor(<editorName> <object_variable>)
Python Example	`oDesign.SetActiveEditor("Schematic" object_variable)`

StartAnalysis [Nexxim]

Simulates all setups

Command: None

Syntax: StartAnalysis

Return Value: None

Parameters: None

VB Example: oDesign.StartAnalysis

Python Syntax	StartAnalysis()
Python Example	oDesign.StartAnalysis()

Undo [Nexxim]

Undo the last operation

Command: **Edit>Undo**

Syntax: Undo

Return Value: None

Parameters: None

VB Example: oDesign.Undo

Python Syntax	Undo()
Python Example	

	`oDesign.Undo()`

UseCircuitSParameterDefinition

Selects whether the circuit or high-frequency definition of S-parameters is used.

Command: None

Syntax: UseCircuitSparameterDefinition <boolean_operator>

Return Value: None

Parameters: `<boolean_operator>`

Type: bool

VB Example: `oDesign.UseCircuitSparameterDefinition true`

```
oDesign.UseCircuitSparameterDefinition true
```

Python Syntax	UseCircuitSparameterDefinition(True)
Python Example	`oDesign.UseCircuitSparameterDefinition(True)`

Nexxim Data Block Commands

The following sections present the commands that are available in the "DataBlock" Module and can be used with the Module command interface:

For example,

```
oDesign.GetModule("DataBlock")
```

```
oModule.CommandName <args>
```

Nexxim data clock commands are listed below.

AddTemperatureDataBlock

Use: Add a Temperature Data Block

Command: None

Syntax: AddTemperatureDataBlock

Return Value: None

Parameters: data

Type: array

VB Example: oModule.AddTemperatureDataBlock data

Python Syntax	AddTemperatureDataBlock()
Python Example	oModule.AddTemperatureDataBlock(data)

AddLibRefDataBlock

Use: Add a LibRef Data Block

Command: None

Syntax: AddLibRefDataBlock

Return Value: None

Parameters: name

Type: string

data

Type: array

VB Example: oModule.AddLibRefDataBlock name data

Python Syntax	AddLibRefDataBlock(name data)
Python Example	`oModule.AddLibRefDataBlock("MyLibRefBlock" data)`

AddNetlistDataBlock

Use: Add a NetList Data Block

Command: None

Syntax: AddNetlistDataBlock

Return Value: None

Parameters: data

Type:

array

VB Example: oModule.AddNetlistDataBlock data

Python Syntax	AddNetlistDataBlock(data)
Python Example	`oModule.AddLibRefDataBlock(data)` *

AddPrintToAuditDataBlock

Use: Add a PrintToAudit Data Block

Command: None

Syntax: AddPrintToAuditDataBlock

Return Value: None

Parameters: None

VB Example: oModule.AddPrintToAuditDataBlock

AddStateVariableDataBlock

Use: Add a StateVariable Data Block

Command: None

Syntax: AddStateVariableDataBlock

Return Value: None

Parameters: None

VB Example: oModule.AddStateVariableDataBlock

AddSubstrateDataBlock

Use: Add a Substrate Data Block

Command: None

Syntax: AddSubstrateDataBlock

Return Value: None

Parameters: data

Type:

array

VB Example: oModule.AddSubstrateDataBlock

Python Syntax	AddSubstrateDataBlock(data)
Python Example	`oModule.AddSubstrateDataBlock(data)` *

AddDeviceNoiseDataBlock

Use: Add a DeviceNoise Data Block

Command: None

Syntax: AddDeviceNoiseDataBlock

Return Value: None

Parameters: None

VB Example: `oModule.AddDeviceNoiseDataBlock`

EditTemperatureDataBlock

Use: Edit a Temperature Data Block

Command: None

Syntax: EditTemperatureDataBlock

Return Value: None

Parameters: Name

Type: string

data

Type: array

VB Example: `oModule.EditTemperatureDataBlock`

Python Syntax	EditTemperatureDataBlock(name data)
Python Example	`oModule.EditTemperatureDataBlock("MyTemperatureBlock" data) *`

EditLibRefDataBlock

Use: Edit a LibRef Data Block

Command: None

Syntax: EditLibRefDataBlock

Return Value: None

Parameters: name

Type: string

data

Type: array

VB Example: oModule.EditLibRefDataBlock name data

Python Syntax	EditLibRefDataBlock(name data)
Python Example	`oModule.EditLibRefDataBlock("MyLibRefBlock" data)`

EditNetlistDataBlock

Use: Edit a Netlist Data Block

Command: None

Syntax: EditNetlistDataBlock

Return Value: None

Parameters: Name

Type:

string

data

Type:

array

VB Example: oModule.EditNetlistDataBlock name data

Python Syntax	EditNetlistDataBlock(name data)
Python Example	oModule.EditNetlistDataBlock("MyNetlistBlock" data) *

EditPrintToAuditDataBlock

Use: Edit a PrintToAudit Data Block

Command: None

Syntax: EditPrintToAuditDataBlock

Return Value: None

Parameters: None

VB Example: oModule.EditPrintToAuditDataBlock

EditStateVariableDataBlock

Use: Edit a StateVariable Data Block

Command: None

Syntax: EditStateVariableDataBlock

Return Value: None

Parameters: None

VB Example: oModule.EditStateVariableDataBlock

EditSubstrateDataBlock

Use: Edit a Substrate Data Block

Command: None

Syntax: EditSubstrateDataBlock

Return Value: None

Parameters: Name

Type:

string

data

Type: array

VB Example: oModule.EditSubstrateDataBlock name data

Python Syntax	EditSubstrateDataBlock(name data)
Python Example	oModule.EditSubstrateDataBlock("MySubstrateBlock" data) *

EditDeviceNoiseDataBlock

Use: Edit a DeviceNoise Data Block

Command: None

Syntax: EditDeviceNoiseDataBlock

Return Value: None

Parameters: None

VB Example: `oModule.EditDeviceNoiseDataBlock`

GetTemperatureDataBlock

Use: Returns the TemperatureDataBlock name

Syntax: GetTemperatureDataBlock

Return Value: String

Parameters: None

VB Example:

```
Dim tempStr
    tempStr = oModule.GetTemperatureDataBlock
```

Python Syntax	GetTemperatureDataBlock()
Python Example	`tempStr = oModule.GetTemperatureDataBlock()`

GetAllLibRefDataBlocks

Use: Returns a name list of all library reference data blocks in the Nexxim Design

Syntax: GetAllLibRefDataBlocks

Return Value: Array of Strings

Parameters: None

VB Example: Dim librefArr

librefArr = oModule.GetAllLibRefDataBlocks

Python Syntax	GetAllLibRefDataBlocks()
Python Example	`librefArr = oModule.GetAllLibRefDataBlocks()`

GetAllNetlistDataBlocks

Use: Returns a name list of all netlist data blocks in the Nexxim Design

Syntax: GetAllNetlistDataBlocks

Return Value: Array of Strings

Parameters: None

VB Example: Dim netlistArr

netlistArr = oModule.GetAllNetlistDataBlocks

Python Syntax	GetAllNetlistDataBlocks()
Python Example	`netlistArr = oModule.GetAllNetlistDataBlocks()`

GetAllSubstrateDataBlocks

Use: Returns a name list of all substrate data blocks in the Nexxim Design

Syntax: GetAllSubstrateDataBlocks

Return Value: Array of Strings

Parameters: None

VB Example: Dim subsArr

subsArr = oModule.GetAllSubstrateDataBlocks

Python Syntax	GetAllSubstrateDataBlocks()
Python Example	`subsArr = oModule.GetAllSubstrateDataBlocks()`

Remove [Nexxim]

Use: Removes a Data Block

Command: None

Syntax: Remove <Data Block>

Return Value: None

Parameters: Data Block

Type:

string

VB Example: oModule.Remove <Data Block>

Python Syntax	Remove(<Data Block>)
Python Example	`oModule.Remove("MyDataBlock") *`

Rename [Nexxim]

Use: Renames a Data Block

Command: None

Syntax: Rename <Data Block>

Return Value: None

Parameters: OldName

Type:

string

NewName

Type:

string

VB Example: oModule.Rename <OldName> <NewName>

Python Syntax	Rename(<OldName> <NewName>)
Python Example	`oModule.Rename("MyDataBlock" "SavedDataBlock") *`

Nexxim Simulation Setup Commands

The following sections present the commands that are available in the "SimSetup" module:

For example,

oDesign.GetModule("SimSetup")

oModule.CommandName <args>

Add

AddSweep

Add [Nexxim]

Use: Adds a new setup

Syntax: Add(

```
STRING newsetup) // new setup name
```

VB Example: `oModule. Add <newsetup>`

Add Alter Block [Nexxim]

Use: Adds a new Alter Block

Syntax: AddAlterBlock(

```
ARRAY alterdata) // Alter block data
```

VB Example:

```
oModule.AddAlterBlock Array("NAME:AlterData", "DataBlockID:=", 22, "Name:=", _
```

```
"AlterData1", "Alter:=", Array("Alter1", "V:=", _
".ALTER second_one" & Chr(13) & Chr(10) & _
".param myval=3000" & Chr(13) & Chr(10) & _
".LNA POI 3 1000 3000" & Chr(13) & Chr(10) & _
".PRINT LNA V(2)"))
```

Python Syntax	AddAlterBlock(<data>[])
Python Example	oModule.AddAlterBlock(["NAME:AlterData", "DataBlockID:=" , 22, "Name:=" , "AlterData", "Alter:=" , ["Alter1", "V:=" , '''.ALTER first_one .param myval=2000 .HB TONES=2000 MAXk=3 .PRINT HB V(2)''']])

Add AMI Analysis [Nexxim]

Use: Adds a new AMI Analysis setup

Syntax: AddAMIAnalysis(

```
ARRAY newsetup) // new setup data
```

VB Example:

```
oModule.AddAMIAnalysis Array("NAME:SimSetup", "DataBlockID:=", 29, "SimSetupID:=", _
3, "OptionName:=", "(Default Options)", "AdditionalOptions:=", "", "AlterBlockName:=", _
"", "FilterText:=", "", "AnalysisEnabled:=", 1, Array("NAME:OutputQuantities"), Array
("NAME:NoiseOutputQuantities"), "Name:=", _
"AMIAnalysis1", "InputType:=", 1, "AMIAnalysis:=", Array(32, false))
```

Python Syntax	AddAMIAnalysis(<setupData[]>)
Python Example	`oModule.AddAMIAnalysis(` `[` `  "NAME:SimSetup",` `  "DataBlockID:=" , 29,` `  "SimSetupID:=" , 1,` `  "OptionName:=" , "(Default Options)",` `  "AdditionalOptions:=" , "",` `  "AlterBlockName:=" , "",`

```
  "FilterText:=" , "",
  "AnalysisEnabled:=" , 1,
  [
    "NAME:OutputQuantities"
  ],
  [
    "NAME:NoiseOutputQuantities"
  ],
  "Name:=" , "AMIAnalysis",
  "InputType:=" , 1,
  "AMIAnalysis:=" , [32,False]
])
```

Add DC Analysis [Nexxim]

Use: Adds a new DC analysis setup

Syntax: AddDCAnalysis(

```
ARRAY newsetup) // new setup data
```

VB Example:

```
oModule.AddDCAnalysis Array("NAME:SimSetup", "DataBlockID:=", 15, "SimSetupID:=", 1,
"OptionName:=", _
"(Default Options)", "AdditionalOptions:=", "", "AlterBlockName:=", "", "FilterText:=", _
```

```
"", "AnalysisEnabled:=", 1, Array("NAME:OutputQuantities"), Array("NAME:NoiseOutputQuantities"),
"Name:=", _
"DCAnalysis")
```

Python Syntax	AddDCAnalysis(<setupData[]>)
Python Example	`oModule.AddDCAnalysis(` `  [` `  "NAME:SimSetup",` `  "DataBlockID:=" , 15,` `  "SimSetupID:=" , 3,` `  "OptionName:=" , "(Default Options)",` `  "AdditionalOptions:=" , "",` `  "AlterBlockName:=" , "",` `  "FilterText:=" , "",` `  "AnalysisEnabled:=" , 1,` `  [` `  "NAME:OutputQuantities"` `  ],` `  [`

```
            "NAME:NoiseOutputQuantities"
    ],
    "Name:="                , "DCAnalysis1"
  ])
```

Add Envelope Setup [Nexxim]

Use: Adds a new Envelope Analysis setup

Syntax: AddEnvelopeSetup(

```
ARRAY newsetup) // new setup data
```

VB Example:

```
oModule.AddEnvelopeSetup Array("NAME:SimSetup", "DataBlockID:=", 21, "SimSetupID:=", 28,
"OptionName:=", _
"(Default Options)", "AdditionalOptions:=", "", "AlterBlockName:=", "", "FilterText:=", _
"", "AnalysisEnabled:=", 1, Array("NAME:OutputQuantities", Array("NAME:Quantity", "NodeType:=",
_
"Harmonics", "CompID:=", "", "CompName:=", "", "QuantityName:=", "2F1", "Selected:=", _
true, "UnitType:=", "NoUnit", "NetlistFUT:=", "", "DataType:=", "Real", "TypeInfo:=", Array( _
"", 0, "", "IsUnconstrained:=", false), "CircuitInstanceID:=", ""), Array("NAME:Quantity",
"NodeType:=", _
"Harmonics", "CompID:=", "", "CompName:=", "", "QuantityName:=", "3F1", "Selected:=", _
true, "UnitType:=", "NoUnit", "NetlistFUT:=", "", "DataType:=", "Real", "TypeInfo:=", Array( _
"", 0, "", "IsUnconstrained:=", false), "CircuitInstanceID:=", ""), Array("NAME:Quantity",
"NodeType:=", _
"Harmonics", "CompID:=", "", "CompName:=", "", "QuantityName:=", "4F1", "Selected:=", _
```

```
true, "UnitType:=", "NoUnit", "NetlistFUT:=", "", "DataType:=", "Real", "TypeInfo:=", Array( _
"", 0, "", "IsUnconstrained:=", false), "CircuitInstanceID:=", ""), Array("NAME:Quantity",
"NodeType:=", _
"Harmonics", "CompID:=", "", "CompName:=", "", "QuantityName:=", "DC", "Selected:=", _
true, "UnitType:=", "NoUnit", "NetlistFUT:=", "", "DataType:=", "Real", "TypeInfo:=", Array( _
"", 0, "", "IsUnconstrained:=", false), "CircuitInstanceID:=", ""), Array("NAME:Quantity",
"NodeType:=", _
"Harmonics", "CompID:=", "", "CompName:=", "", "QuantityName:=", "F1", "Selected:=", _
true, "UnitType:=", "NoUnit", "NetlistFUT:=", "", "DataType:=", "Real", "TypeInfo:=", Array( _
"", 0, "", "IsUnconstrained:=", false), "CircuitInstanceID:=", "")), Array("NAME:NoiseOut-
putQuantities"), "Name:=", _
"Envelope1", "EnvelopeData:=", Array(1, "0", "1e-008", "", "", "trap", "HB", _
"No", 0), "EnvelopeMaxkValues:=", Array("F1", "7"), "EnvelopeTonesValues:=", Array( _
"1Ghz"), "HarmonicsText:=", "2F1 3F1 4F1 DC F1")
```

Python Syntax	AddEnvelopeSetup(<setupData[]>)
Python Example	oModule.AddEnvelopeSetup(["NAME:SimSetup", "DataBlockID:=" , 21,

```
"SimSetupID:="          , 26,
"OptionName:="          , "(Default Options)",
"AdditionalOptions:="   , "",
"AlterBlockName:="      , "",
"FilterText:="          , "",
"AnalysisEnabled:="     , 1,
[
        "NAME:OutputQuantities",
        [
                "NAME:Quantity",
                "NodeType:="            , "Harmonics",
                "CompID:="              , "",
                "CompName:="            , "",
                "QuantityName:="        , "2F1",
                "Selected:="            , True,
                "UnitType:="            , "NoUnit",
                "NetlistFUT:="          , "",
                "DataType:="            , "Real",
                "TypeInfo:="            , ["",0,"",
                "IsUnconstrained:="     , False],
                "CircuitInstanceID:="   , ""
```

```
        ],
        [
                "NAME:Quantity",
                "NodeType:="            , "Harmonics",
                "CompID:="              , "",
                "CompName:="            , "",
                "QuantityName:="        , "3F1",
                "Selected:="            , True,
                "UnitType:="            , "NoUnit",
                "NetlistFUT:="          , "",
                "DataType:="            , "Real",
                "TypeInfo:="            , ["",0,"",
                "IsUnconstrained:="     , False],
                "CircuitInstanceID:="   , ""
        ],
        [
                "NAME:Quantity",
                "NodeType:="            , "Harmonics",
                "CompID:="              , "",
                "CompName:="            , "",
```

```
                    "QuantityName:="        , "4F1",
                    "Selected:="            , True,
                    "UnitType:="            , "NoUnit",
                    "NetlistFUT:="          , "",
                    "DataType:="            , "Real",
                    "TypeInfo:="            , ["",0,"",
                    "IsUnconstrained:="     , False],
                    "CircuitInstanceID:="   , ""
                ],
                [
                    "NAME:Quantity",
                    "NodeType:="            , "Harmonics",
                    "CompID:="              , "",
                    "CompName:="            , "",
                    "QuantityName:="        , "DC",
                    "Selected:="            , True,
                    "UnitType:="            , "NoUnit",
                    "NetlistFUT:="          , "",
                    "DataType:="            , "Real",
                    "TypeInfo:="            , ["",0,"",
                    "IsUnconstrained:="     , False],
```

```
                "CircuitInstanceID:="   , ""
            ],
            [
                "NAME:Quantity",
                "NodeType:="            , "Harmonics",
                "CompID:="              , "",
                "CompName:="            , "",
                "QuantityName:="        , "F1",
                "Selected:="            , True,
                "UnitType:="            , "NoUnit",
                "NetlistFUT:="          , "",
                "DataType:="            , "Real",
                "TypeInfo:="            , ["",0,"",
                "IsUnconstrained:="     , False],
                "CircuitInstanceID:="   , ""
            ]
        ],
        [
            "NAME:NoiseOutputQuantities"
        ],
```

```
            "Name:="          , "Envelope",
            "EnvelopeData:="          , [1,"0","1e-8","","","trap","HB","No",0],
            "EnvelopeMaxkValues:="    , ["F1","7"],
            "EnvelopeTonesValues:="   , ["1Ghz"],
            "HarmonicsText:=" , "2F1 3F1 4F1 DC F1",
  ])
```

Add HB 1-Tone [Nexxim]

Use: Adds a new Harmonic Balance 1-Tone setup

Syntax: AddHB1Tone(

```
ARRAY newsetup) // new setup data
```

VB Example:

```
oModule.AddHB1Tone Array("NAME:SimSetup", "DataBlockID:=", 9, "SimSetupID:=", 3, "OptionName:=",
_
"(Default Options)", "AdditionalOptions:=", "", "AlterBlockName:=", "", "FilterText:=", _
"", "AnalysisEnabled:=", 1, Array("NAME:OutputQuantities", Array("NAME:Quantity", "NodeType:=",
_
"Harmonics", "CompID:=", "", "CompName:=", "", "QuantityName:=", "2F1", "Selected:=", _
true, "UnitType:=", "NoUnit", "NetlistFUT:=", "", "DataType:=", "Real", "TypeInfo:=", Array( _
"", 0, "", "IsUnconstrained:=", false), "CircuitInstanceID:=", ""), Array("NAME:Quantity",
"NodeType:=", _
"Harmonics", "CompID:=", "", "CompName:=", "", "QuantityName:=", "3F1", "Selected:=", _
true, "UnitType:=", "NoUnit", "NetlistFUT:=", "", "DataType:=", "Real", "TypeInfo:=", Array( _
```

```
"", 0, "", "IsUnconstrained:=", false), "CircuitInstanceID:=", ""), Array("NAME:Quantity",
"NodeType:=", _
"Harmonics", "CompID:=", "", "CompName:=", "", "QuantityName:=", "4F1", "Selected:=", _
true, "UnitType:=", "NoUnit", "NetlistFUT:=", "", "DataType:=", "Real", "TypeInfo:=", Array( _
"", 0, "", "IsUnconstrained:=", false), "CircuitInstanceID:=", ""), Array("NAME:Quantity",
"NodeType:=", _
"Harmonics", "CompID:=", "", "CompName:=", "", "QuantityName:=", "DC", "Selected:=", _
true, "UnitType:=", "NoUnit", "NetlistFUT:=", "", "DataType:=", "Real", "TypeInfo:=", Array( _
"", 0, "", "IsUnconstrained:=", false), "CircuitInstanceID:=", ""), Array("NAME:Quantity",
"NodeType:=", _
"Harmonics", "CompID:=", "", "CompName:=", "", "QuantityName:=", "F1", "Selected:=", _
true, "UnitType:=", "NoUnit", "NetlistFUT:=", "", "DataType:=", "Real", "TypeInfo:=", Array( _
"", 0, "", "IsUnconstrained:=", false), "CircuitInstanceID:=", "")), Array("NAME:NoiseOut-
putQuantities"), "Name:=", _
"HB1Tone1", "HB1ToneData:=", Array(7, "1GHz", "HB", "No", "0.0"), "HarmonicsText:=", _
"2F1 3F1 4F1 DC F1")
```

Python Syntax	AddHB1Tone(<setupData[]>)
Python Example	oModule.AddHB1Tone(["NAME:SimSetup",

```
"DataBlockID:="         , 9,
"SimSetupID:="          , 1,
"OptionName:="          , "(Default Options)",
"AdditionalOptions:="   , "",
"AlterBlockName:="      , "",
"FilterText:="          , "",
"AnalysisEnabled:="     , 1,
[
  "NAME:OutputQuantities",
  [
    "NAME:Quantity",
    "NodeType:="          , "Harmonics",
    "CompID:="            , "",
    "CompName:="          , "",
    "QuantityName:="      , "2F1",
    "Selected:="          , True,
    "UnitType:="          , "NoUnit",
    "NetlistFUT:="                , "",
    "DataType:="          , "Real",
    "TypeInfo:="          , ["",0,"",
    "IsUnconstrained:="   , False],
```

```
      "CircuitInstanceID:=" , ""
    ],
    [
      "NAME:Quantity",
      "NodeType:="          , "Harmonics",
      "CompID:="            , "",
      "CompName:="          , "",
      "QuantityName:="      , "3F1",
      "Selected:="          , True,
      "UnitType:="          , "NoUnit",
      "NetlistFUT:="              , "",
      "DataType:="          , "Real",
      "TypeInfo:="          , ["",0,"",
      "IsUnconstrained:="   , False],
      "CircuitInstanceID:=" , ""
    ],
    [
      "NAME:Quantity",
      "NodeType:="          , "Harmonics",
      "CompID:="            , "",
```

```
      "CompName:="          , "",
      "QuantityName:="      , "4F1",
      "Selected:="          , True,
      "UnitType:="          , "NoUnit",
      "NetlistFUT:="        , "",
      "DataType:="          , "Real",
      "TypeInfo:="          , ["",0,"",
      "IsUnconstrained:="   , False],
      "CircuitInstanceID:=" , ""
    ],
    [
     "NAME:Quantity",
      "NodeType:="          , "Harmonics",
      "CompID:="            , "",
      "CompName:="          , "",
      "QuantityName:="      , "DC",
      "Selected:="          , True,
      "UnitType:="          , "NoUnit",
      "NetlistFUT:="        , "",
      "DataType:="          , "Real",
      "TypeInfo:="          , ["",0,"",
```

```
      "IsUnconstrained:="   , False],
      "CircuitInstanceID:=" , ""
    ],
    [
     "NAME:Quantity",
      "NodeType:="          , "Harmonics",
      "CompID:="            , "",
      "CompName:="          , "",
      "QuantityName:="      , "F1",
      "Selected:="          , True,
      "UnitType:="          , "NoUnit",
      "NetlistFUT:="        , "",
      "DataType:="          , "Real",
      "TypeInfo:="          , ["",0,"",
      "IsUnconstrained:="   , False],
      "CircuitInstanceID:=" , ""
    ],
  ],
  [
    "NAME:NoiseOutputQuantities"
```

```
  ],
  "Name:="                , "HB1Tone",
  "HB1ToneData:="         , [7,"1GHz","HB","No","0.0"],
  "HarmonicsText:="       , "2F1 3F1 4F1 DC F1"
])
```

Add HB N-Tone [Nexxim]

Use: Adds a new Harmonic Balance N-Tone setup

Syntax: AddHBnTone(

```
ARRAY newsetup) // new setup data
```

VB Example:

```
oModule.AddHBnTone Array("NAME:SimSetup", "DataBlockID:=", 17, "SimSetupID:=", 10,
"OptionName:=", _
"(Default Options)", "AdditionalOptions:=", "", "AlterBlockName:=", "", "FilterText:=", _
"", "AnalysisEnabled:=", 1, Array("NAME:OutputQuantities", Array("NAME:Quantity", "NodeType:=",
_
"Harmonics", "CompID:=", "", "CompName:=", "", "QuantityName:=", "2F1", "Selected:=", _
true, "UnitType:=", "NoUnit", "NetlistFUT:=", "", "DataType:=", "Real", "TypeInfo:=", Array( _
"", 0, "", "IsUnconstrained:=", false), "CircuitInstanceID:=", ""), Array("NAME:Quantity",
"NodeType:=", _
"Harmonics", "CompID:=", "", "CompName:=", "", "QuantityName:=", "3F1", "Selected:=", _
true, "UnitType:=", "NoUnit", "NetlistFUT:=", "", "DataType:=", "Real", "TypeInfo:=", Array( _
```

```
"", 0, "", "IsUnconstrained:=", false), "CircuitInstanceID:=", ""), Array("NAME:Quantity",
"NodeType:=", _
"Harmonics", "CompID:=", "", "CompName:=", "", "QuantityName:=", "4F1", "Selected:=", _
true, "UnitType:=", "NoUnit", "NetlistFUT:=", "", "DataType:=", "Real", "TypeInfo:=", Array( _
"", 0, "", "IsUnconstrained:=", false), "CircuitInstanceID:=", ""), Array("NAME:Quantity",
"NodeType:=", _
"Harmonics", "CompID:=", "", "CompName:=", "", "QuantityName:=", "DC", "Selected:=", _
true, "UnitType:=", "NoUnit", "NetlistFUT:=", "", "DataType:=", "Real", "TypeInfo:=", Array( _
"", 0, "", "IsUnconstrained:=", false), "CircuitInstanceID:=", ""), Array("NAME:Quantity",
"NodeType:=", _
"Harmonics", "CompID:=", "", "CompName:=", "", "QuantityName:=", "F1", "Selected:=", _
true, "UnitType:=", "NoUnit", "NetlistFUT:=", "", "DataType:=", "Real", "TypeInfo:=", Array( _
"", 0, "", "IsUnconstrained:=", false), "CircuitInstanceID:=", "")), Array("NAME:NoiseOut-
putQuantities"), "Name:=", _
"HBnTone1", "HBnTone:=", Array(1, "HB", "No", "0.0", "0.0", "F1", "7"), HBnValues:=", Array( _
"1GHz"), "HarmonicsText:=", "2F1 3F1 4F1 DC F1")
```

Python Syntax	AddHBnTone(<setupData[]>)
Python Example	oModule.AddHBnTone(["NAME:SimSetup",

```
"DataBlockID:="         , 17,
"SimSetupID:="          , 8,
"OptionName:="          , "(Default Options)",
"AdditionalOptions:="   , "",
"AlterBlockName:="      , "",
"FilterText:="          , "",
"AnalysisEnabled:="     , 1,
[
  "NAME:OutputQuantities",
  [
    "NAME:Quantity",
    "NodeType:="          , "Harmonics",
    "CompID:="            , "",
    "CompName:="          , "",
    "QuantityName:="      , "2F1",
    "Selected:="          , True,
    "UnitType:="          , "NoUnit",
    "NetlistFUT:="                , "",
    "DataType:="          , "Real",
    "TypeInfo:="          , ["",0,"",
    "IsUnconstrained:="   , False],
```

```
      "CircuitInstanceID:=" , ""
    ],
    [
      "NAME:Quantity",
      "NodeType:="          , "Harmonics",
      "CompID:="            , "",
      "CompName:="          , "",
      "QuantityName:="      , "3F1",
      "Selected:="          , True,
      "UnitType:="          , "NoUnit",
      "NetlistFUT:="              , "",
      "DataType:="          , "Real",
      "TypeInfo:="          , ["",0,"",
      "IsUnconstrained:="   , False],
      "CircuitInstanceID:=" , ""
    ],
    [
      "NAME:Quantity",
      "NodeType:="          , "Harmonics",
      "CompID:="            , "",
```

```
      "CompName:="          , "",
      "QuantityName:="      , "4F1",
      "Selected:="          , True,
      "UnitType:="          , "NoUnit",
      "NetlistFUT:="        , "",
      "DataType:="          , "Real",
      "TypeInfo:="          , ["",0,"",
      "IsUnconstrained:="   , False],
      "CircuitInstanceID:=" , ""
    ],
    [
     "NAME:Quantity",
      "NodeType:="          , "Harmonics",
      "CompID:="            , "",
      "CompName:="          , "",
      "QuantityName:="      , "DC",
      "Selected:="          , True,
      "UnitType:="          , "NoUnit",
      "NetlistFUT:="        , "",
      "DataType:="          , "Real",
      "TypeInfo:="          , ["",0,"",
```

```
      "IsUnconstrained:="   , False],
      "CircuitInstanceID:=" , ""
    ],
    [
     "NAME:Quantity",
      "NodeType:="          , "Harmonics",
      "CompID:="            , "",
      "CompName:="          , "",
      "QuantityName:="      , "F1",
      "Selected:="          , True,
      "UnitType:="          , "NoUnit",
      "NetlistFUT:="        , "",
      "DataType:="          , "Real",
      "TypeInfo:="          , ["",0,"",
      "IsUnconstrained:="   , False],
      "CircuitInstanceID:=" , ""
    ],
  ],
  [
    "NAME:NoiseOutputQuantities"
```

	], "Name:=" , "HBnTone", "HB1ToneData:=" , [1,"HB","No","0.0", "0.0","F1","7"], "HBnValues:=" , ["1GHz"] "HarmonicsText:=" , "2F1 3F1 4F1 DC F1"])

Add HSPICE Transient [Nexxim]

Use: Adds a new HSPICE Transient Analysis setup

Syntax: AddHSPICETransient(

```
ARRAY newsetup) // new setup data
```

VB Example:

```
oModule.AddHSPICETransient Array("NAME:SimSetup", "DataBlockID:=", 30, "SimSetupID:=", _
11, "OptionName:=", "(Default Options)", "AdditionalOptions:=", "", "AlterBlockName:=", _
"", "FilterText:=", "", "AnalysisEnabled:=", 1, Array("NAME:OutputQuantities"), Array
("NAME:NoiseOutputQuantities"), "Name:=", _
"HSPICETransient1", ""HSPICETransientData:=", Array("0.1ns", "10ns"), "HSPICETran-
sientOtherData:=", Array( _
3))
```

Python Syntax	AddHSPICETransient(<setupData[]>)
Python Example	oModule.AddHSPICETransient(

```
[
  "NAME:SimSetup",
  "DataBlockID:=" , 30,
  "SimSetupID:=" , 9,
  "OptionName:=" , "(Default Options)",
  "AdditionalOptions:=" , "",
  "AlterBlockName:=" , "",
  "FilterText:=" , "",
  "AnalysisEnabled:=" , 1,
  [
    "NAME:OutputQuantities"
  ],
  [
    "NAME:NoiseOutputQuantities"
  ],
  "Name:=" , "HSPICETransient",
  "HSPICETransientData:=" , ["0.1ns","10ns"],
  "HSPICETransientOtherData:=", [3]
])
```

Add Linear Network Analysis [Nexxim]

Use: Adds a new Linear Network analysis setup

Syntax: AddLinearNetworkAnalysis(

```
ARRAY newsetup) // new setup data
```

VB Example:

```
oModule.AddLinearNetworkAnalysis Array("NAME:SimSetup", "DataBlockID:=", 16, "SimSetupID:=", _
17, "OptionName:=", "(Default Options)", "AdditionalOptions:=", "", "AlterBlockName:=", _
"", "FilterText:=", "", "AnalysisEnabled:=", 1, Array("NAME:OutputQuantities"), Array
("NAME:NoiseOutputQuantities"), "Name:=", _
"LinearFrequency1", "LinearFrequencyData:=", Array(false, 0.1, false, "", _
false), Array("NAME:SweepDefinition", "Variable:=", "F", "Data:=", _
"LINC 1GHz 100GHz 10", "OffsetF1:=", false, "Synchronize:=", 0))
```

Python Syntax	AddLinearNetworkAnalysis(<setupData[]>)
Python Example	`oModule.AddLinearNetworkAnalysis(` `  [` `  "NAME:SimSetup",` `  "DataBlockID:=" , 16,` `  "SimSetupID:=" , 15,` `  "OptionName:=" , "(Default Options)",`

```
    "AdditionalOptions:="   , "",
    "AlterBlockName:="      , "",
    "FilterText:="          , "",
    "AnalysisEnabled:="     , 1,
    [
            "NAME:OutputQuantities"
    ],
    [
            "NAME:NoiseOutputQuantities"
    ],
    "Name:="                , "LinearFrequency",
    "LinearFrequencyData:=" , [False,"0.1",False,"",False],
    [
            "NAME:SweepDefinition",
            "Variable:="    , "F",
            "Data:="        , "LINC 1GHz 100GHz 10",
            "OffsetF1:="            , False,
            "Synchronize:="         , 0
    ]
  ])
```

Add Oscillator Analysis [Nexxim]

Use: Adds a new Oscillator 1-Tone analysis setup

Syntax: AddOscillatorAnalysis(

```
ARRAY newsetup) // new setup data
```

VB Example:

```
oModule.AddOscillatorAnalysis Array("NAME:SimSetup", "DataBlockID:=", 21, "SimSetupID:=", 28,
"OptionName:=", _
"(Default Options)", "AdditionalOptions:=", "", "AlterBlockName:=", "", "FilterText:=", _
"", "AnalysisEnabled:=", 1, Array("NAME:OutputQuantities", Array("NAME:Quantity", "NodeType:=",
_
"Harmonics", "CompID:=", "", "CompName:=", "", "QuantityName:=", "2F1", "Selected:=", _
true, "UnitType:=", "NoUnit", "NetlistFUT:=", "", "DataType:=", "Real", "TypeInfo:=", Array( _
"", 0, "", "IsUnconstrained:=", false), "CircuitInstanceID:=", ""), Array("NAME:Quantity",
"NodeType:=", _
"Harmonics", "CompID:=", "", "CompName:=", "", "QuantityName:=", "3F1", "Selected:=", _
true, "UnitType:=", "NoUnit", "NetlistFUT:=", "", "DataType:=", "Real", "TypeInfo:=", Array( _
"", 0, "", "IsUnconstrained:=", false), "CircuitInstanceID:=", ""), Array("NAME:Quantity",
"NodeType:=", _
"Harmonics", "CompID:=", "", "CompName:=", "", "QuantityName:=", "4F1", "Selected:=", _
true, "UnitType:=", "NoUnit", "NetlistFUT:=", "", "DataType:=", "Real", "TypeInfo:=", Array( _
"", 0, "", "IsUnconstrained:=", false), "CircuitInstanceID:=", ""), Array("NAME:Quantity",
"NodeType:=", _
"Harmonics", "CompID:=", "", "CompName:=", "", "QuantityName:=", "DC", "Selected:=", _
```

```
true, "UnitType:=", "NoUnit", "NetlistFUT:=", "", "DataType:=", "Real", "TypeInfo:=", Array( _
"", 0, "", "IsUnconstrained:=", false), "CircuitInstanceID:=", ""), Array("NAME:Quantity",
"NodeType:=", _
"Harmonics", "CompID:=", "", "CompName:=", "", "QuantityName:=", "F1", "Selected:=", _
true, "UnitType:=", "NoUnit", "NetlistFUT:=", "", "DataType:=", "Real", "TypeInfo:=", Array( _
"", 0, "", "IsUnconstrained:=", false), "CircuitInstanceID:=", "")), Array("NAME:NoiseOut-
putQuantities"), "Name:=", _
"Oscillator1", "OscillatorData:=", Array(7, "1Ghz", false, 1),
"HarmonicsText:=", "2F1 3F1 4F1 DC F1")
```

Python Syntax	AddOscillatorAnalysis(<setupData[]>)
Python Example	oModule.AddOscillatorAnalysis(["NAME:SimSetup", "DataBlockID:=" , 21, "SimSetupID:=" , 26, "OptionName:=" , "(Default Options)", "AdditionalOptions:=" , "", "AlterBlockName:=" , "",

```
    "FilterText:="          , "",
    "AnalysisEnabled:="     , 1,
    [
            "NAME:OutputQuantities",
            [
                    "NAME:Quantity",
                    "NodeType:="            , "Harmonics",
                    "CompID:="              , "",
                    "CompName:="            , "",
                    "QuantityName:="        , "2F1",
                    "Selected:="            , True,
                    "UnitType:="            , "NoUnit",
                    "NetlistFUT:="          , "",
                    "DataType:="            , "Real",
                    "TypeInfo:="            , ["",0,"",
                    "IsUnconstrained:="     , False],
                    "CircuitInstanceID:="   , ""
            ],
            [
                    "NAME:Quantity",
                    "NodeType:="            , "Harmonics",
```

```
                    "CompID:="              , "",
                    "CompName:="            , "",
                    "QuantityName:="        , "3F1",
                    "Selected:="            , True,
                    "UnitType:="            , "NoUnit",
                    "NetlistFUT:="          , "",
                    "DataType:="            , "Real",
                    "TypeInfo:="            , ["",0,"",
                    "IsUnconstrained:="     , False],
                    "CircuitInstanceID:="   , ""
                ],
                [
                    "NAME:Quantity",
                    "NodeType:="            , "Harmonics",
                    "CompID:="              , "",
                    "CompName:="            , "",
                    "QuantityName:="        , "4F1",
                    "Selected:="            , True,
                    "UnitType:="            , "NoUnit",
                    "NetlistFUT:="          , "",
```

```
                "DataType:="            , "Real",
                "TypeInfo:="            , ["",0,"",
                "IsUnconstrained:="     , False],
                "CircuitInstanceID:="   , ""
        ],
        [
                "NAME:Quantity",
                "NodeType:="            , "Harmonics",
                "CompID:="              , "",
                "CompName:="            , "",
                "QuantityName:="        , "DC",
                "Selected:="            , True,
                "UnitType:="            , "NoUnit",
                "NetlistFUT:="          , "",
                "DataType:="            , "Real",
                "TypeInfo:="            , ["",0,"",
                "IsUnconstrained:="     , False],
                "CircuitInstanceID:="   , ""
        ],
        [
                "NAME:Quantity",
```

```
                    "NodeType:="            , "Harmonics",
                    "CompID:="              , "",
                    "CompName:="            , "",
                    "QuantityName:="        , "F1",
                    "Selected:="            , True,
                    "UnitType:="            , "NoUnit",
                    "NetlistFUT:="          , "",
                    "DataType:="            , "Real",
                    "TypeInfo:="            , ["",0,"",
                    "IsUnconstrained:="     , False],
                    "CircuitInstanceID:="   , ""
          ]
        ],
        [
          "NAME:NoiseOutputQuantities"
        ],
        "Name:="          , "Oscillator",
        "OscillatorData:="        , [7,"1Ghz",False,1],
        "HarmonicsText:=" , "2F1 3F1 4F1 DC F1",
])
```

Add Oscillator N-Tone Analysis [Nexxim]

Use: Adds a new Oscillator N-Tone analysis setup

Syntax: AddOscillatorNtoneAnalysis(

```
ARRAY newsetup) // new setup data
```

VB Example:

```
oModule.AddOscillatorNtoneAnalysis Array("NAME:SimSetup", "DataBlockID:=", 23, "SimSetupID:=", _
5, "OptionName:=", "(Default Options)", "AdditionalOptions:=", "", "AlterBlockName:=", _
"", "FilterText:=", "", "AnalysisEnabled:=", 1, Array("NAME:OutputQuantities", Array("NAME:Quant-
ity", "NodeType:=", _
"Harmonics", "CompID:=", "", "CompName:=", "", "QuantityName:=", "2F2", "Selected:=", _
true, "UnitType:=", "NoUnit", "NetlistFUT:=", "", "DataType:=", "Real", "TypeInfo:=", Array( _
"", 0, "", "IsUnconstrained:=", false), "CircuitInstanceID:=", ""), Array("NAME:Quantity",
"NodeType:=", _
"Harmonics", "CompID:=", "", "CompName:=", "", "QuantityName:=", "3F2", "Selected:=", _
true, "UnitType:=", "NoUnit", "NetlistFUT:=", "", "DataType:=", "Real", "TypeInfo:=", Array( _
"", 0, "", "IsUnconstrained:=", false), "CircuitInstanceID:=", ""), Array("NAME:Quantity",
"NodeType:=", _
"Harmonics", "CompID:=", "", "CompName:=", "", "QuantityName:=", "4F2", "Selected:=", _
true, "UnitType:=", "NoUnit", "NetlistFUT:=", "", "DataType:=", "Real", "TypeInfo:=", Array( _
"", 0, "", "IsUnconstrained:=", false), "CircuitInstanceID:=", ""), Array("NAME:Quantity",
"NodeType:=", _
"Harmonics", "CompID:=", "", "CompName:=", "", "QuantityName:=", "DC", "Selected:=", _
true, "UnitType:=", "NoUnit", "NetlistFUT:=", "", "DataType:=", "Real", "TypeInfo:=", Array( _
```

```
"", 0, "", "IsUnconstrained:=", false), "CircuitInstanceID:=", ""), Array("NAME:Quantity",
"NodeType:=", _
"Harmonics", "CompID:=", "", "CompName:=", "", "QuantityName:=", "F2", "Selected:=", _
true, "UnitType:=", "NoUnit", "NetlistFUT:=", "", "DataType:=", "Real", "TypeInfo:=", Array( _
"", 0, "", "IsUnconstrained:=", false), "CircuitInstanceID:=", "")), Array("NAME:NoiseOut-
putQuantities"), "Name:=", _
"OscillatorNtone2", "OscillatorNtoneData:=", Array(1, 1, false, 1), "OscMaxkValues:=", Array( _
"F1", "7"), "OscTonesValues:=", Array("1Ghz"), "DrivenMaxkValues:=", Array("RF1", _
"7"), "DrivenTonesValues:=", Array("1Ghz"), "HarmonicsText:=", _
"2F2 3F2 4F2 DC F2", Array("NAME:SweepDefinition", "Variable:=", "FNoi", "Data:=", _
"LINC 30GHz 300GHz 10", "OffsetF1:=", false, "Synchronize:=", 0))
```

Python Syntax	AddOscillatorNtoneAnalysis(<setupData[]>)
Python Example	oModule.AddOscillatorNtoneAnalysis(["NAME:SimSetup", "DataBlockID:=" , 23, "SimSetupID:=" , 1, "OptionName:=" , "(Default Options)", "AdditionalOptions:=" , "",

```
"AlterBlockName:=" , "",
"FilterText:=" , "",
"AnalysisEnabled:=" , 1,
[
"NAME:OutputQuantities",
[
"NAME:Quantity",
"NodeType:=" , "Harmonics",
"CompID:=" , "",
"CompName:=" , "",
"QuantityName:=" , "2F2",
"Selected:=" , True,
"UnitType:=" , "NoUnit",
"NetlistFUT:=" , "",
"DataType:=" , "Real",
"TypeInfo:=" , ["",0,"", "IsUnconstrained:=" , False],
"CircuitInstanceID:=" , ""
],
[
"NAME:Quantity",
"NodeType:=" , "Harmonics",
```

```
"CompID:=" , "",
"CompName:=" , "",
"QuantityName:=" , "3F2",
"Selected:=" , True,
"UnitType:=" , "NoUnit",
"NetlistFUT:=" , "",
"DataType:=" , "Real",
"TypeInfo:=" , ["",0,"", "IsUnconstrained:=" , False],
"CircuitInstanceID:=" , ""
],
[
"NAME:Quantity",
"NodeType:=" , "Harmonics",
"CompID:=" , "",
"CompName:=" , "",
"QuantityName:=" , "4F2",
"Selected:=" , True,
"UnitType:=" , "NoUnit",
"NetlistFUT:=" , "",
"DataType:=" , "Real",
```

```
"TypeInfo:=" , ["",0,"", "IsUnconstrained:=" , False],
"CircuitInstanceID:=" , ""
],
[
"NAME:Quantity",
"NodeType:=" , "Harmonics",
"CompID:=" , "",
"CompName:=" , "",
"QuantityName:=" , "DC",
"Selected:=" , True,
"UnitType:=" , "NoUnit",
"NetlistFUT:=" , "",
"DataType:=" , "Real",
"TypeInfo:=" , ["",0,"", "IsUnconstrained:=" , False],
"CircuitInstanceID:=" , ""
],
[
"NAME:Quantity",
"NodeType:=" , "Harmonics",
"CompID:=" , "",
"CompName:=" , "",
```

```
    "QuantityName:=" , "F2",
    "Selected:=" , True,
    "UnitType:=" , "NoUnit",
    "NetlistFUT:=" , "",
    "DataType:=" , "Real",
    "TypeInfo:=" , ["",0,"", "IsUnconstrained:=" , False],
    "CircuitInstanceID:=" , ""
    ],
  ],
  [
    "NAME:NoiseOutputQuantities"
  ],
  "Name:=" , "OscillatorNtone",
  "OscillatorNtoneData:=" , [1,1,False,1],
  "OscMaxkValues:=" , ["F1","7"],
  "OscTonesValues:=" , ["1Ghz"],
  "DrivenMaxkValues:=" , ["RF1","7"],
  "DrivenTonesValues:=" , ["1Ghz"],
  "HarmonicsText:=" , "2F2 3F2 4F2 DC F2",
    [
```

```
    "NAME:SweepDefinition",
    "Variable:=" , "FNoi",
    "Data:=" , "LINC 1GHz 200GHz 10",
    "OffsetF1:=" , False,
    "Synchronize:=" , 0
    ]
  ])
```

Add Oscillator Resonant Frequency Search [Nexxim]

Use: Adds a new Oscillator Resonant Frequency Search setup

Syntax: AddOscResonantFrequencySearch(

```
ARRAY newsetup) // new setup data
```

VB Example:

```
oModule.AddOscResonantFrequencySearch Array("NAME:SimSetup", "DataBlockID:=", 21,
"SimSetupID:=", 28, "OptionName:=", _
"(Default Options)", "AdditionalOptions:=", "", "AlterBlockName:=", "", "FilterText:=", _
"", "AnalysisEnabled:=", 1, Array("NAME:OutputQuantities", Array("NAME:Quantity", "NodeType:=",
_
"Harmonics", "CompID:=", "", "CompName:=", "", "QuantityName:=", "2F1", "Selected:=", _
true, "UnitType:=", "NoUnit", "NetlistFUT:=", "", "DataType:=", "Real", "TypeInfo:=", Array( _
"", 0, "", "IsUnconstrained:=", false), "CircuitInstanceID:=", ""), Array("NAME:Quantity",
"NodeType:=", _
"Harmonics", "CompID:=", "", "CompName:=", "", "QuantityName:=", "3F1", "Selected:=", _
```

```
true, "UnitType:=", "NoUnit", "NetlistFUT:=", "", "DataType:=", "Real", "TypeInfo:=", Array( _
"", 0, "", "IsUnconstrained:=", false), "CircuitInstanceID:=", ""), Array("NAME:Quantity",
"NodeType:=", _
"Harmonics", "CompID:=", "", "CompName:=", "", "QuantityName:=", "4F1", "Selected:=", _
true, "UnitType:=", "NoUnit", "NetlistFUT:=", "", "DataType:=", "Real", "TypeInfo:=", Array( _
"", 0, "", "IsUnconstrained:=", false), "CircuitInstanceID:=", ""), Array("NAME:Quantity",
"NodeType:=", _
"Harmonics", "CompID:=", "", "CompName:=", "", "QuantityName:=", "DC", "Selected:=", _
true, "UnitType:=", "NoUnit", "NetlistFUT:=", "", "DataType:=", "Real", "TypeInfo:=", Array( _
"", 0, "", "IsUnconstrained:=", false), "CircuitInstanceID:=", ""), Array("NAME:Quantity",
"NodeType:=", _
"Harmonics", "CompID:=", "", "CompName:=", "", "QuantityName:=", "F1", "Selected:=", _
true, "UnitType:=", "NoUnit", "NetlistFUT:=", "", "DataType:=", "Real", "TypeInfo:=", Array( _
"", 0, "", "IsUnconstrained:=", false), "CircuitInstanceID:=", "")), Array("NAME:NoiseOut-
putQuantities"), "Name:=", _
"OscResonantFrequency1", "OscResonantFrequencyData:=", Array(1000, "1000000000", _
"5000000000", false, 1), "OscMaxkValues:=", Array("F1", 7),
"OscToneValues:=", Array("1Ghz")
```

Python Syntax	AddOscResonantFreqencySearch(<setupData[]>)
Python Example	

```
oModule.AddOscResonantFrequencySearch(
  [
    "NAME:SimSetup",
    "DataBlockID:="         , 21,
    "SimSetupID:="          , 26,
    "OptionName:="          , "(Default Options)",
    "AdditionalOptions:="   , "",
    "AlterBlockName:="      , "",
    "FilterText:="          , "",
    "AnalysisEnabled:="     , 1,
    [
            "NAME:OutputQuantities",
            [
                    "NAME:Quantity",
                    "NodeType:="            , "Harmonics",
                    "CompID:="              , "",
                    "CompName:="            , "",
                    "QuantityName:="        , "2F1",
                    "Selected:="            , True,
                    "UnitType:="            , "NoUnit",
                    "NetlistFUT:="          , "",
```

```
                "DataType:="            , "Real",
                "TypeInfo:="            , ["",0,"",
                "IsUnconstrained:="     , False],
                "CircuitInstanceID:="   , ""
        ],
        [
                "NAME:Quantity",
                "NodeType:="            , "Harmonics",
                "CompID:="              , "",
                "CompName:="            , "",
                "QuantityName:="        , "3F1",
                "Selected:="            , True,
                "UnitType:="            , "NoUnit",
                "NetlistFUT:="          , "",
                "DataType:="            , "Real",
                "TypeInfo:="            , ["",0,"",
                "IsUnconstrained:="     , False],
                "CircuitInstanceID:="   , ""
        ],
        [
```

```
                    "NAME:Quantity",
                    "NodeType:="            , "Harmonics",
                    "CompID:="              , "",
                    "CompName:="            , "",
                    "QuantityName:="        , "4F1",
                    "Selected:="            , True,
                    "UnitType:="            , "NoUnit",
                    "NetlistFUT:="          , "",
                    "DataType:="            , "Real",
                    "TypeInfo:="            , ["",0,"",
                    "IsUnconstrained:="     , False],
                    "CircuitInstanceID:="   , ""
            ],
            [
                    "NAME:Quantity",
                    "NodeType:="            , "Harmonics",
                    "CompID:="              , "",
                    "CompName:="            , "",
                    "QuantityName:="        , "DC",
                    "Selected:="            , True,
                    "UnitType:="            , "NoUnit",
```

```
                "NetlistFUT:="          , "",
                "DataType:="            , "Real",
                "TypeInfo:="            , ["",0,"",
                "IsUnconstrained:="     , False],
                "CircuitInstanceID:="   , ""
        ],
        [
                "NAME:Quantity",
                "NodeType:="            , "Harmonics",
                "CompID:="              , "",
                "CompName:="            , "",
                "QuantityName:="        , "F1",
                "Selected:="            , True,
                "UnitType:="            , "NoUnit",
                "NetlistFUT:="          , "",
                "DataType:="            , "Real",
                "TypeInfo:="            , ["",0,"",
                "IsUnconstrained:="     , False],
                "CircuitInstanceID:="   , ""
        ]
```

```
          ],
          [
            "NAME:NoiseOutputQuantities"
          ],
          "Name:="          , "OscResonantFrequency",
          "OscResonantFrequencyData:="      ,
              [1000,"1000000000:,"5000000000","1Ghz",False,1],
          "OscMaxkValues:=" , ["F1","7"
          "OscToneValues:=" , ["1Ghz"]
  ])
```

Add PXF Setup [Nexxim]

Use: Adds a new Periodic Transfer Function (PXF) setup

Syntax: AddPXFSetup(

```
ARRAY newsetup) // new setup data
```

VB Example:

```
oModule.AddPXFSetup Array("NAME:SimSetup", "DataBlockID:=", 21, "SimSetupID:=", 28,
"OptionName:=", _
"(Default Options)", "AdditionalOptions:=", "", "AlterBlockName:=", "", "FilterText:=", _
"", "AnalysisEnabled:=", 1, Array("NAME:OutputQuantities", Array("NAME:Quantity", "NodeType:=",
_
"Harmonics", "CompID:=", "", "CompName:=", "", "QuantityName:=", "2F1", "Selected:=", _
true, "UnitType:=", "NoUnit", "NetlistFUT:=", "", "DataType:=", "Real", "TypeInfo:=", Array( _
```

```
"", 0, "", "IsUnconstrained:=", false), "CircuitInstanceID:=", ""), Array("NAME:Quantity",
"NodeType:=", _
"Harmonics", "CompID:=", "", "CompName:=", "", "QuantityName:=", "3F1", "Selected:=", _
true, "UnitType:=", "NoUnit", "NetlistFUT:=", "", "DataType:=", "Real", "TypeInfo:=", Array( _
"", 0, "", "IsUnconstrained:=", false), "CircuitInstanceID:=", ""), Array("NAME:Quantity",
"NodeType:=", _
"Harmonics", "CompID:=", "", "CompName:=", "", "QuantityName:=", "4F1", "Selected:=", _
true, "UnitType:=", "NoUnit", "NetlistFUT:=", "", "DataType:=", "Real", "TypeInfo:=", Array( _
"", 0, "", "IsUnconstrained:=", false), "CircuitInstanceID:=", ""), Array("NAME:Quantity",
"NodeType:=", _
"Harmonics", "CompID:=", "", "CompName:=", "", "QuantityName:=", "DC", "Selected:=", _
true, "UnitType:=", "NoUnit", "NetlistFUT:=", "", "DataType:=", "Real", "TypeInfo:=", Array( _
"", 0, "", "IsUnconstrained:=", false), "CircuitInstanceID:=", ""), Array("NAME:Quantity",
"NodeType:=", _
"Harmonics", "CompID:=", "", "CompName:=", "", "QuantityName:=", "F1", "Selected:=", _
true, "UnitType:=", "NoUnit", "NetlistFUT:=", "", "DataType:=", "Real", "TypeInfo:=", Array( _
"", 0, "", "IsUnconstrained:=", false), "CircuitInstanceID:=", "")), Array("NAME:NoiseOut-
putQuantities"), "Name:=", _
"PXF1", "PXFData:=", Array(1, "HB", false, "HB", "No", 0), "PXFMaxkValues:=", Array( _
"F1", "7"), "PXFTonesValues:=", Array("1Ghz"), "PXFRelHarmNums:=", Array(), "PXFMaxSideband:=",
Array( _
```

```
"0"), "PXFSideband:=", Array("0"), "HarmonicsText:=", "2F1 3F1 4F1 DC F1", Array("NAME:Sweep-
Definition", "Variable:=", _
"F", "Data:=", "LINC 20GHz 200GHz 10", "OffsetF1:=", false, "Synchronize:=", 0), Array
("NAME:FSweep", "Variable:=", _
"F", "Data:=", "LINC 20GHz 200GHz 10", "OffsetF1:=", false, "Synchronize:=", 0))
```

Python Syntax	AddPXFSetup(<setupData[]>)
Python Example	oModule.AddPXFSetup(["NAME:SimSetup", "DataBlockID:=" , 21, "SimSetupID:=" , 26, "OptionName:=" , "(Default Options)", "AdditionalOptions:=" , "", "AlterBlockName:=" , "", "FilterText:=" , "", "AnalysisEnabled:=" , 1, ["NAME:OutputQuantities", ["NAME:Quantity",

```
                    "NodeType:="            , "Harmonics",
                    "CompID:="              , "",
                    "CompName:="            , "",
                    "QuantityName:="        , "2F1",
                    "Selected:="            , True,
                    "UnitType:="            , "NoUnit",
                    "NetlistFUT:="          , "",
                    "DataType:="            , "Real",
                    "TypeInfo:="            , ["",0,"",
                    "IsUnconstrained:="     , False],
                    "CircuitInstanceID:="   , ""
                ],
                [
                    "NAME:Quantity",
                    "NodeType:="            , "Harmonics",
                    "CompID:="              , "",
                    "CompName:="            , "",
                    "QuantityName:="        , "3F1",
                    "Selected:="            , True,
                    "UnitType:="            , "NoUnit",
```

```
                "NetlistFUT:="          , "",
                "DataType:="            , "Real",
                "TypeInfo:="            , ["",0,"",
                "IsUnconstrained:="     , False],
                "CircuitInstanceID:="   , ""
            ],
            [
                "NAME:Quantity",
                "NodeType:="            , "Harmonics",
                "CompID:="              , "",
                "CompName:="            , "",
                "QuantityName:="        , "4F1",
                "Selected:="            , True,
                "UnitType:="            , "NoUnit",
                "NetlistFUT:="          , "",
                "DataType:="            , "Real",
                "TypeInfo:="            , ["",0,"",
                "IsUnconstrained:="     , False],
                "CircuitInstanceID:="   , ""
            ],
            [
```

```
                    "NAME:Quantity",
                    "NodeType:="            , "Harmonics",
                    "CompID:="              , "",
                    "CompName:="            , "",
                    "QuantityName:="        , "DC",
                    "Selected:="            , True,
                    "UnitType:="            , "NoUnit",
                    "NetlistFUT:="          , "",
                    "DataType:="            , "Real",
                    "TypeInfo:="            , ["",0,"",
                    "IsUnconstrained:="     , False],
                    "CircuitInstanceID:="   , ""
            ],
            [
                    "NAME:Quantity",
                    "NodeType:="            , "Harmonics",
                    "CompID:="              , "",
                    "CompName:="            , "",
                    "QuantityName:="        , "F1",
                    "Selected:="            , True,
```

```
                "UnitType:="            , "NoUnit",
                "NetlistFUT:="          , "",
                "DataType:="            , "Real",
                "TypeInfo:="            , ["",0,"",
                "IsUnconstrained:="     , False],
                "CircuitInstanceID:="   , ""
            ]
        ],
        [
          "NAME:NoiseOutputQuantities"
        ],
        "Name:="          , "PXF",
        "PXFData:="               , [1,"HB",False,"HB","No",0],
        "PXFMaxkValues:=" , ["F1","7"],
        "PXFTonesValues:="        , ["1Ghz"],
        "PXFRelHarmNums:="        , [],
        "PXFMaxSideband:="        , ["0"],
        "PXFSideband:="           , ["0"],
        "HarmonicsText:=" , "2F1 3F1 4F1 DC F1",
        [
           "NAME:SweepDefinition",
```

```
                "Variable:="          , "F",
                "Data:="              , "LINC 10GHz 100GHz 10",
                "OffsetF1:="          , False,
                "Synchronize:="               , 0
            ],
            [
                "NAME:SweepDefinition",
                "Variable:="          , "F",
                "Data:="              , "LINC 10GHz 100GHz 10",
                "OffsetF1:="          , False,
                "Synchronize:="               , 0
            ],
    ])
```

AddQuickEyeAnalysis

Adds a QuickEye analysis to the design.

Command: Context menu of Analysis icon in project tree -> AddSolution Setup... -> QuickEye Analysis

Syntax: AddQuickEyeAnalysis (Array("NAME:SimSetupName",

"DataBlockID:=", <int>,

"SimSetupID:=", <int>,

"OptionName:=", <string>,

```
"AdditionalOptions:=", <string>,
"AlterBlockName:=", <string>,
"FilterText:=", <string>,
"AnalysisEnabled:=", <int>, // 1 if enabled, 0 if disabled
Array("NAME:OutputQuantities", <QuantityArray>, <QuantityArray>...)
Array("NAME:NoiseOutputQuantities", <QuantityArray>, <QuantityArray>...) ),
"Name:=", <string>, // name for new analysis
"QuickEyeAnalysis:=", Array(<string>, // Input rise time
<string>, // Input low voltage
<string>, // Input high voltage
<string>, // bits per second
<string>, // number of FFE taps
<string>, // Random Jitter Standard Deviation
<string>, // Delay
<string>, // Duty cycle distortion
<bool>, // true if unit interval, false if bits per second
<string>, // number of DFE taps
<bool>, // true to calculate FFE, false if weights are specified
<bool>, // true to calculate DFE, false if weights are specified
<string>, // DFE decision threshold
<string>, // DFE decision high
```

```
<string>, // DFE decision low
<bool>), // true if using specified equalization, false if disabled
Array("NAME:SweepDefinition",
"Variable:=", <string>,
"Data:=", <string>, // sweep
"OffsetF1:=", <bool>,
"Synchronize:=", <int>), // 1 to Synchronize, 0 otherwise
"FFEWts:=", Array(".5", "-2"), // optional, specified weights
"DFEWts:=", Array("2"))) // optional, specified weights
```

Return Value: <string> – // Name of the analysis that was added

// If the name requested conflicts with the name of an existing

// analysis, the requested name is altered to be unique.

// The name returned reflects any change made to be unique.

Parameters: Parameters: <QuantityArray>:

Array("NAME:Quantity",

"NodeType:=", <string>, // CompInst, Variable, Net, Harmonics, or Custom

"CompID:=", <string>,

"CompName:=", <string>,

"QuantityName:=", <string>,

"Selected:=", <bool>,

"UnitType:=", <string>,

"DataType:=", <string>, // Real, Complex, Integer, Enum, Char, Free, Array, Record

"CircuitInstanceID:=", <string)

VB Example:

```
dim name

name = oModule.AddQuickEyeAnalysis (Array("NAME:SimSetup", "DataBlockID:=", 28, "SimSetupID:=",
_
3, "OptionName:=", "(Default Options)", "AdditionalOptions:=", "", "AlterBlockName:=", _

"", "FilterText:=", "", "AnalysisEnabled:=", 1, Array("NAME:OutputQuantities", Array("NAME:Quant-
ity", "NodeType:=", "CompInst", "CompID:=", _

"5", "CompName:=", "Level01_NPN_Model_5", "QuantityName:=", "I", "Selected:=", _

true, "UnitType:=", "NoUnit", "DataType:=", "Real", "CircuitInstanceID:=", ""), Array
("NAME:Quantity", "NodeType:=", _

"CompInst", "CompID:=", "10", "CompName:=", "RES__10", "QuantityName:=", "I", "Selected:=", _

true, "UnitType:=", "NoUnit", "DataType:=", "Real", "CircuitInstanceID:=", ""), Array
("NAME:Quantity", "NodeType:=", _

"Net", "CompID:=", "", "CompName:=", "net_47", "QuantityName:=", "V", "Selected:=", _

true, "UnitType:=", "NoUnit", "DataType:=", "Real", "CircuitInstanceID:=", "")), Array
("NAME:NoiseOutputQuantities"), "Name:=", _

"MyQuickEyeAnalysis", "QuickEyeAnalysis:=", Array("5e-10", "0", "1", "1e-9", "2", _

"0", "1e-9", "0", true, "1", false, false, "0", "1", "-1", true), Array("NAME:SweepDefinition",
"Variable:=", _

"Temp", "Data:=", "LINC 1cel 10cel 10", "OffsetF1:=", false, "Synchronize:=", _

0), "FFEWts:=", Array(".5", "-2"), "DFEWts:=", Array("2")))
```

Python Syntax	AddQuickEyeAnalysis(<data_block>)
Python Example	name = oModule.AddQuickEyeAnalysis (["NAME:SimSetup", "DataBlockID:=", 28, "SimSetupID:=", 3, "OptionName:=", "(Default Options)", "AdditionalOptions:=","", "AlterBlockName:=", "", "FilterText:=", "", "AnalysisEnabled:=",1, ["NAME:OutputQuantities", ["NAME:Quantity", "NodeType:=","CompInst", "CompID:=", "5",

```
            "CompName:=", "Level01_NPN_Model_5",
            "QuantityName:=", "I",
            "Selected:=", True,
            "UnitType:=", "NoUnit",
            "DataType:=","Real",
            "CircuitInstanceID:=", ""
        ],
        [
            "NAME:Quantity",
            "NodeType:=", "CompInst",
            "CompID:=", "10",
            "CompName:=", "RES__10",
            "QuantityName:=", "I",
            "Selected:=", True,
            "UnitType:=", "NoUnit",
            "DataType:=", "Real",
            "CircuitInstanceID:=", ""
        ],
        [
            "NAME:Quantity",
            "NodeType:=", Net",
```

```
            "CompID:=", "",
            "CompName:=", "net_47",
            "QuantityName:=", "V",
            "Selected:=", True,
            "UnitType:=", "NoUnit",
            "DataType:=", "Real",
            "CircuitInstanceID:=", ""
         ]
      ],
      [
         "NAME:NoiseOutputQuantities"
      ],
     "Name:=", "MyQuickEyeAnalysis",
     "QuickEyeAnalysis:=",
      [
         "5e-10", "0", "1", "1e-9", "2", "0", "1e-9", "0",
         True, "1", False, False,"0", "1", "-1", True
      ],
      [
         "NAME:SweepDefinition",
```

```
          "Variable:=", "Temp",
          "Data:=", "LINC 1cel 10cel 10",
          "OffsetF1:=", False,
          "Synchronize:=", 0),
          "FFEWts:=",
          [
             ".5", "-2"
          ]
          DFEWts:=",
          [
             "2"
          ]
       ]
    ]
)
```

Add System FD Analysis [Nexxim]

Use: Adds a new System Frequency Domain (FD) Analysis setup

Syntax: AddSystemFDAnalysis(

`ARRAY newsetup) // new setup data`

VB Example:

```
oModule.AddSystemFDAnalysis Array("NAME:SimSetup", "DataBlockID:=", 29, "SimSetupID:=", _
3, "OptionName:=", "(Default Options)", "AdditionalOptions:=", "", "AlterBlockName:=", _
"", "FilterText:=", "", "AnalysisEnabled:=", 1, Array("NAME:OutputQuantities"), Array
("NAME:NoiseOutputQuantities"), "Name:=", _
"SystemFDAnalysis1", "SystemFDAnalysis:=", Array(false))
```

Python Syntax	AddSystemFDAnalysis(<setupData[]>)
Python Example	`oModule.AddSystemFDAnalysis(` `[` `  "NAME:SimSetup",` `  "DataBlockID:=" , 32,` `  "SimSetupID:=" , 16,` `  "OptionName:=" , "(Default Options)",` `  "AdditionalOptions:=" , "",` `  "AlterBlockName:=" , "",` `  "FilterText:=" , "",` `  "AnalysisEnabled:=" , 1,` `  [` `  "NAME:OutputQuantities"` `  ],`

	["NAME:NoiseOutputQuantities"], "Name:=" , "SystemFDAnalysis", "SystemFDAnalysis:=" , [False]])

Add Transient Analysis [Nexxim]

Use: Adds a new Transient analysis setup

Syntax: AddTransient(

ARRAY newsetup) // new setup data

VB Example:

```
oModule.AddTransient Array("NAME:SimSetup", "DataBlockID:=", 10, "SimSetupID:=", 9,
"OptionName:=", _
"(Default Options)", "AdditionalOptions:=", "", "AlterBlockName:=", "", "FilterText:=", _
"", "AnalysisEnabled:=", 1, Array("NAME:OutputQuantities"), Array("NAME:NoiseOutputQuantities"),
"Name:=", _
"NexximTransient", "TransientData:=", Array("0.1ns", "10ns"), "TransientNoiseData:=", Array( _
false, "", "", 0, 1, 0, false, 1), "TransientOtherData:=", Array("default"))
```

Python Syntax	AddTransient(<setupData[]>)

Python Example

```
oModule.AddTransient(
   [
      "NAME:SimSetup",
      "DataBlockID:="         , 10,
      "SimSetupID:="          , 11,
      "OptionName:="          , "(Default Options)",
      "AdditionalOptions:="   , "",
      "AlterBlockName:="      , "",
      "FilterText:="          , "",
      "AnalysisEnabled:="     , 1,
      [
              "NAME:OutputQuantities"
      ],
      [
              "NAME:NoiseOutputQuantities"
      ],
      "Name:="                , "NexximTransient1",
      "TransientData:="       , ["0.1ns","10ns"],
      "TransientNoiseData:="  , [False,"","",0,1,0,False,1],
```

	"TransientOtherData:=" , ["default"]])

Add TV Noise [Nexxim]

Use: Adds a new TV Noise analysis setup

Syntax: AddTVNoise(

ARRAY newsetup) // new setup data

VB Example:

```
oModule.AddTVNoise Array("NAME:SimSetup", "DataBlockID:=", 20, "SimSetupID:=", 23,
"OptionName:=", _
"(Default Options)", "AdditionalOptions:=", "", "AlterBlockName:=", "", "FilterText:=", _
"", "AnalysisEnabled:=", 1, Array("NAME:OutputQuantities", Array("NAME:Quantity", "NodeType:=",
_
"Harmonics", "CompID:=", "", "CompName:=", "", "QuantityName:=", "2F1", "Selected:=", _
true, "UnitType:=", "NoUnit", "NetlistFUT:=", "", "DataType:=", "Real", "TypeInfo:=", Array( _
"", 0, "", "IsUnconstrained:=", false), "CircuitInstanceID:=", ""), Array("NAME:Quantity",
"NodeType:=", _
"Harmonics", "CompID:=", "", "CompName:=", "", "QuantityName:=", "3F1", "Selected:=", _
true, "UnitType:=", "NoUnit", "NetlistFUT:=", "", "DataType:=", "Real", "TypeInfo:=", Array( _
"", 0, "", "IsUnconstrained:=", false), "CircuitInstanceID:=", ""), Array("NAME:Quantity",
"NodeType:=", _
"Harmonics", "CompID:=", "", "CompName:=", "", "QuantityName:=", "4F1", "Selected:=", _
true, "UnitType:=", "NoUnit", "NetlistFUT:=", "", "DataType:=", "Real", "TypeInfo:=", Array( _
```

```
"", 0, "", "IsUnconstrained:=", false), "CircuitInstanceID:=", ""), Array("NAME:Quantity",
"NodeType:=", _
"Harmonics", "CompID:=", "", "CompName:=", "", "QuantityName:=", "DC", "Selected:=", _
true, "UnitType:=", "NoUnit", "NetlistFUT:=", "", "DataType:=", "Real", "TypeInfo:=", Array( _
"", 0, "", "IsUnconstrained:=", false), "CircuitInstanceID:=", ""), Array("NAME:Quantity",
"NodeType:=", _
"Harmonics", "CompID:=", "", "CompName:=", "", "QuantityName:=", "F1", "Selected:=", _
true, "UnitType:=", "NoUnit", "NetlistFUT:=", "", "DataType:=", "Real", "TypeInfo:=", Array( _
"", 0, "", "IsUnconstrained:=", false), "CircuitInstanceID:=", "")), Array("NAME:NoiseOut-
putQuantities"), "Name:=", _
"TVNoise1", "TVNoiseData:=", Array(1, "", "", "", "", "HB", false, "HB", "No", _
0), "TVNoiseMaxkValues:=", Array("F1", "7"), "TVNoiseTonesValues:=", Array( _
"1Ghz"), "TVNoiseRelHarmNums:=", Array(), "TVNoiseRefSideband:=", Array("1"), "HarmonicsText:=",
_
"2F1 3F1 4F1 DC F1", Array("NAME:SweepDefinition", "Variable:=", "F", "Data:=", _
"LINC 10GHz 100GHz 10", "OffsetF1:=", false, "Synchronize:=", 0), Array("NAME:FSweep", "Vari-
able:=", _
"F", "Data:=", "LINC 10GHz 100GHz 10", "OffsetF1:=", false, "Synchronize:=", 0))
```

Python Syntax	AddTVNoise(<setupData[]>)
Python Example	

```
oModule.AddTVNoise(
  [
    "NAME:SimSetup",
    "DataBlockID:="         , 20,
    "SimSetupID:="          , 21,
    "OptionName:="          , "(Default Options)",
    "AdditionalOptions:="   , "",
    "AlterBlockName:="      , "",
    "FilterText:="          , "",
    "AnalysisEnabled:="     , 1,
    [
            "NAME:OutputQuantities",
            [
                    "NAME:Quantity",
                    "NodeType:="            , "Harmonics",
                    "CompID:="              , "",
                    "CompName:="            , "",
                    "QuantityName:="        , "2F1",
                    "Selected:="            , True,
                    "UnitType:="            , "NoUnit",
                    "NetlistFUT:="          , "",
```

```
                    "DataType:="            , "Real",
                    "TypeInfo:="            , ["",0,"",
                    "IsUnconstrained:="     , False],
                    "CircuitInstanceID:="   , ""
                ],
                [
                    "NAME:Quantity",
                    "NodeType:="            , "Harmonics",
                    "CompID:="              , "",
                    "CompName:="            , "",
                    "QuantityName:="        , "3F1",
                    "Selected:="            , True,
                    "UnitType:="            , "NoUnit",
                    "NetlistFUT:="          , "",
                    "DataType:="            , "Real",
                    "TypeInfo:="            , ["",0,"",
                    "IsUnconstrained:="     , False],
                    "CircuitInstanceID:="   , ""
                ],
                [
```

```
                    "NAME:Quantity",
                    "NodeType:="            , "Harmonics",
                    "CompID:="              , "",
                    "CompName:="            , "",
                    "QuantityName:="        , "4F1",
                    "Selected:="            , True,
                    "UnitType:="            , "NoUnit",
                    "NetlistFUT:="          , "",
                    "DataType:="            , "Real",
                    "TypeInfo:="            , ["",0,"",
                    "IsUnconstrained:="     , False],
                    "CircuitInstanceID:="   , ""
            ],
            [
                    "NAME:Quantity",
                    "NodeType:="            , "Harmonics",
                    "CompID:="              , "",
                    "CompName:="            , "",
                    "QuantityName:="        , "DC",
                    "Selected:="            , True,
                    "UnitType:="            , "NoUnit",
```

```
                "NetlistFUT:="          , "",
                "DataType:="            , "Real",
                "TypeInfo:="            , ["",0,"",
                "IsUnconstrained:="     , False],
                "CircuitInstanceID:="   , ""
        ],
        [
                "NAME:Quantity",
                "NodeType:="            , "Harmonics",
                "CompID:="              , "",
                "CompName:="            , "",
                "QuantityName:="        , "F1",
                "Selected:="            , True,
                "UnitType:="            , "NoUnit",
                "NetlistFUT:="          , "",
                "DataType:="            , "Real",
                "TypeInfo:="            , ["",0,"",
                "IsUnconstrained:="     , False],
                "CircuitInstanceID:="   , ""
        ]
```

```
        ],
        [
          "NAME:NoiseOutputQuantities"
        ],
        "Name:="          , "TVNoise",
        "TVNoiseData:="           , [1,"","","","","HB",False,"HB","No",0],
        "TVNoiseMaxkValues:="     , ["F1","7"],
        "TVNoiseTonesValues:="    , ["1Ghz"],
        "TVNoiseRelHarmNums:="    , [],
        "TVNoiseRefSideband:="    , ["1"],
        "HarmonicsText:=" , "2F1 3F1 4F1 DC F1",
        [
          "NAME:SweepDefinition",
          "Variable:="            , "F",
          "Data:="                , "LINC 10GHz 100GHz 10",
          "OffsetF1:="            , False,
          "Synchronize:="         , 0
        ],
        [
          "NAME:FSweep",
          "Variable:="            , "F",
```

```
            "Data:="                , "LINC 10GHz 100GHz 10",
            "OffsetF1:="            , False,
            "Synchronize:="         , 0
          ]
  ])
```

Add VerifEye Analysis

Adds a VerifEye analysis to the design.

Command: Context menu of Analysis icon in project tree -> AddSolution Setup... -> VerifEye (Statistical Eye) Analysis

Syntax: AddVerifEyeAnalysis (Array("NAME:SimSetupName",

"DataBlockID:=", <int>,

"SimSetupID:=", <int>,

"OptionName:=", <string>,

"AdditionalOptions:=", <string>,

"AlterBlockName:=", <string>,

"FilterText:=", <string>,

"AnalysisEnabled:=", <int>, // 1 if enabled, 0 if disabled

Array("NAME:OutputQuantities", <QuantityArray>, <QuantityArray>...)

Array("NAME:NoiseOutputQuantities", <QuantityArray>, <QuantityArray>...)),

"Name:=", <string>, // name for new analysis

"VerifEyeAnalysis:=", Array(<string>, // Input rise time

<string>, // Input low voltage

<string>, // Input high voltage

<string>, // bits per second

<string>, // number of FFE taps

<string>, // Random Jitter Standard Deviation

<string>, // Delay

<string>, // Duty cycle distortion

<bool>, // true if unit interval, false if bits per second

<string>, // number of DFE taps

<bool>, // true to calculate FFE, false if weights are specified

<bool>, // true to calculate DFE, false if weights are specified

<string>, // DFE decision threshold

<string>, // DFE decision high

<string>, // DFE decision low

<bool>), // true if using specified equalization, false if disabled

Array("NAME:SweepDefinition",

"Variable:=", <string>,

"Data:=", <string>, // sweep

"OffsetF1:=", <bool>,

"Synchronize:=", <int>), // 1 to Synchronize, 0 otherwise

"FFEWts:=", Array(".5", "-2"), // optional, specified weights

"DFEWts:=", Array("2"))) // optional, specified weights

Return Value: <string> – // Name of the analysis that was added

// If the name requested conflicts with the name of an existing

// analysis, the requested name is altered to be unique.

// The name returned reflects any change made to be unique.

Parameters: <QuantityArray>:

Array("NAME:Quantity",

"NodeType:=", <string>, // CompInst, Variable, Net, Harmonics, or Custom

"CompID:=", <string>,

"CompName:=", <string>,

"QuantityName:=", <string>,

"Selected:=", <bool>,

"UnitType:=", <string>,

"DataType:=", <string>, // Real, Complex, Integer, Enum, Char, Free, Array, Record

"CircuitInstanceID:=", <string)

VB Example:

```
dim name
name = oModule.AddVerifEyeAnalysis (Array("NAME:SimSetup", "DataBlockID:=", 27, "SimSetupID:=", _
1, "OptionName:=", "Options", "AdditionalOptions:=", "", "AlterBlockName:=", "", "FilterText:=",
_
```

```
"", "AnalysisEnabled:=", 1, Array("NAME:OutputQuantities", Array("NAME:Quantity", "NodeType:=",
"CompInst", "CompID:=", _
"5", "CompName:=", "Level01_NPN_Model_5", "QuantityName:=", "I", "Selected:=", _
true, "UnitType:=", "NoUnit", "DataType:=", "Real", "CircuitInstanceID:=", ""), Array
("NAME:Quantity", "NodeType:=", _
"CompInst", "CompID:=", "10", "CompName:=", "RES__10", "QuantityName:=", "I", "Selected:=", _
true, "UnitType:=", "NoUnit", "DataType:=", "Real", "CircuitInstanceID:=", ""), Array
("NAME:Quantity", "NodeType:=", _
"Net", "CompID:=", "", "CompName:=", "net_47", "QuantityName:=", "V", "Selected:=", _
true, "UnitType:=", "NoUnit", "DataType:=", "Real", "CircuitInstanceID:=", "")), Array
("NAME:NoiseOutputQuantities"), "Name:=", _
"MyVerifEyeAnalysis", "VerifEyeAnalysis:=", Array("5e-10", "0", "1", "1e-9", "2", _
"0", "1e-9", "0", true, "1", false, false, "0", "1", "-1", true), Array("NAME:SweepDefinition",
"Variable:=", _
"Temp", "Data:=", "LINC 1cel 10cel 10", "OffsetF1:=", false, "Synchronize:=", _
0), "FFEWts:=", Array(".5", "-2"), "DFEWts:=", Array("2")))
```

Python Syntax	AddVerifEyeAnalysis(<data_block>)
Python Example	name = oModule.AddVerifEyeAnalysis (["NAME:SimSetup", "DataBlockID:=", 28,

```
"SimSetupID:=", 3,
"OptionName:=", "(Default Options)",
"AdditionalOptions:=","",
"AlterBlockName:=", "",
"FilterText:=", "",
"AnalysisEnabled:=",1,
   [
      "NAME:OutputQuantities",
      [
         "NAME:Quantity",
         "NodeType:=","CompInst",
         "CompID:=", "5",
         "CompName:=", "Level01_NPN_Model_5",
         "QuantityName:=", "I",
         "Selected:=", True,
         "UnitType:=", "NoUnit",
         "DataType:=","Real",
         "CircuitInstanceID:=", ""
      ],
      [
```

```
            "NAME:Quantity",
            "NodeType:=", "CompInst",
            "CompID:=", "10",
            "CompName:=", "RES__10",
            "QuantityName:=", "I",
            "Selected:=", True,
            "UnitType:=", "NoUnit",
            "DataType:=", "Real",
            "CircuitInstanceID:=", ""
        ],
        [
            "NAME:Quantity",
            "NodeType:=", Net",
            "CompID:=", "",
            "CompName:=", "net_47",
            "QuantityName:=", "V",
            "Selected:=", True,
            "UnitType:=", "NoUnit",
            "DataType:=", "Real",
            "CircuitInstanceID:=", ""
        ]
```

```
        ],
        [
           "NAME:NoiseOutputQuantities"
        ],
      "Name:=", "MyVerifEyeAnalysis",
      "VerifEyeAnalysis:=",
        [
           "5e-10", "0", "1", "1e-9", "2", "0", "1e-9", "0",
           True, "1", False, False,"0", "1", "-1", True
        ],
        [
           "NAME:SweepDefinition",
           "Variable:=", "Temp",
           "Data:=", "LINC 1cel 10cel 10",
           "OffsetF1:=", False,
           "Synchronize:=", 0),
           "FFEWts:=",
           [
              ".5", "-2"
           ]
```

```
          DFEWts:=",
          [
             "2"
          ]
       ]
    ]
)
```

AddSweep [Nexxim]

Use: Adds a sweep to a setup

Syntax: AddSweep(

```
STRING setup) // setup name
STRING newsweep) // new sweep name
```

VB Example:

```
oModule. AddSweep <setup> <newsweep>
```

Analyze All Nominal [Nexxim]

Simulates all defined setups

Syntax: AnalyzeAllNominal( )

VB Example:

```
oModule. AnalyzeAllNominal
```

Python Syntax	AnalyzeAllNominal()
Python Example	`oDesign.AnalyzeAllNominal()`

Analyze [Nexxim]

Simulates the given setup

Syntax: Analyze(STRING setup1) // setup name

VB Example:

```
oModule. Analyze <setup1>
```

Python Syntax	Analyze(<setupName>)
Python Example	`oDesign.Analyze("QuickEyeAnalysis")`

AnalyzeSweep [Nexxim]

Use: Simulates the given sweep

Syntax: AnalyzeSweep(

```
STRING setup1 // setup name
STRING sweep1) // sweep name
```

VB Example: `oModule. AnalyzeSweep <setup1> <sweep1>`

Delete [Nexxim]

Use: Removes a simulation setup

Syntax: Delete(

```
STRING name) // The specified setup to delete
```

VB Example: oModule. Delete <name>

DisplayBiasPointInfo

Use: Display voltage/current bias

Command: None

Syntax: DisplayBiasPointInfo <display_preference>

Return Value: None

Parameters: <display_preference>

Type: string

VB Example: oDesign.DisplayBiasPointInfo "1"

```
Where display_preference is one of the following:
0 : hide the info
1 : display voltage
2 : display current
3 : display both
```

Python Syntax	DisplayBiasPointInfo(<display_preference>)
Python Example	`oDesign.DisplayBiasPointInfo(3)`

DeleteSweep [Nexxim]

Use: Removes a sweep from a setup

Syntax: DeleteSweep(

```
STRING setup1) // The specified setup

STRING sweep1) // The specified sweep
```

VB Example: `oModule. DeleteSweep <setup1> <sweep1>`

DynamicMeshOverlays

Use: Turns on dynamic mesh overlays

Syntax: DynamicMeshOverlays(

```
STRING setup) // setup name
```

VB Example: `oModule. DynamicMeshOverlay <setup>`

Edit [Nexxim]

Use: Edit a setup

Syntax: Edit(

```
STRING setup) // setup name

ARRAY solvesetup_data) // setup data
```

VB Example: `oModule. Edit <setup> <solvesetup_data>`

Edit Alter Block [Nexxim]

Use: Edits an Alter Block

Syntax: EditAlterBlock(

```
STRING alterblockname,
ARRAY alterdata) // Alter block data
```

VB Example:

```
oModule.EditAlterBlock "AlterData1", Array("NAME:AlterData", "DataBlockID:=", 22, "Name:=",
"AlterData1", "Alter:=", Array("Alter1", "V:=", _
".ALTER second_one" & Chr(13) & Chr(10) & _
".param myval=3000" & Chr(13) & Chr(10) & _
".LNA POI 3 1000 3000" & Chr(13) & Chr(10) & _
".PRINT LNA V(2)"))
```

Python Syntax	EditAlterBlock(<name>,<data>[])
Python Example	`oModule.EditAlterBlock("AlterData",` `  [` `  "NAME:AlterData",` `  "DataBlockID:=" , 22,` `  "Name:=" , "AlterData",` `  "Alter:=" ,` `  ["Alter1",`

```
            "V:="           , '''.ALTER first_one
            .param myval=2000
            .HB TONES=2000 MAXk=5
            .PRINT HB V(2)'''
    ]
  ])
```

Edit AMI Analysis [Nexxim]

Use: Edits an AMI Analysis setup

Syntax: EditAMIAnalysis(

```
STRING name,
ARRAY newsetup) // new setup data
```

VB Example:

```
oModule.AddAMIAnalysis "AMIAnalysis1", Array("NAME:SimSetup", "DataBlockID:=", 29,
"SimSetupID:=", _
3, "OptionName:=", "(Default Options)", "AdditionalOptions:=", "", "AlterBlockName:=", _
"", "FilterText:=", "", "AnalysisEnabled:=", 1, Array("NAME:OutputQuantities"), Array
("NAME:NoiseOutputQuantities"), "Name:=", _
"AMIAnalysis1", "InputType:=", 1, "AMIAnalysis:=", Array(32, false))), Array("NAME:Sweep-
Definition", "Variable:=", _
"Temp", "Data:=", "LINC 30cel 300cel 10 LINC 10cel 100cel 10", "OffsetF1:=", _
false, "Synchronize:=", 0))
```

Python Syntax	AddAMIAnalysis(<name>,<setupData[]>)
Python Example	oModule.AddAMIAnalysis("AMIAnalysis1", ["NAME:SimSetup", "DataBlockID:=" , 29, "SimSetupID:=" , 1, "OptionName:=" , "(Default Options)", "AdditionalOptions:=" , "", "AlterBlockName:=" , "", "FilterText:=" , "", "AnalysisEnabled:=" , 1, ["NAME:OutputQuantities"], ["NAME:NoiseOutputQuantities"], "Name:=" , "AMIAnalysis", "InputType:=" , 1, "AMIAnalysis:=" , [32,False]

	["NAME:SweepDefinition", "Variable:=" , "Temp", "Data:=" , "LINC 30cel 300cel 10", "OffsetF1:=" , False, "Synchronize:=" , 0]])

Edit DC Analysis [Nexxim]

Use: Edits a DC analysis setup

Syntax: EditDCAnalysis(

```
ARRAY setup) // modified setup data
```

VB Example:

```
oModule.EditDCAnalysis "DCAnalysis", Array("NAME:SimSetup", "DataBlockID:=", 15, "SimSetupID:=",
_
1, "OptionName:=", "(Default Options)", "AdditionalOptions:=", "", "AlterBlockName:=", _
"", "FilterText:=", "", "AnalysisEnabled:=", 1, Array("NAME:OutputQuantities"), Array
("NAME:NoiseOutputQuantities"), "Name:=", _
"DCAnalysis", Array("NAME:SweepDefinition", "Variable:=", "Temp", "Data:=", _
"LINC 10cel 200cel 10", "OffsetF1:=", false, "Synchronize:=", 0))
```

Python Syntax	EditDCAnalysis(<modifiedsetupData[]>)
Python Example	oModule.AddDCAnalysis(["NAME:SimSetup", "DataBlockID:=" , 15, "SimSetupID:=" , 3, "OptionName:=" , "(Default Options)", "AdditionalOptions:=" , "", "AlterBlockName:=" , "", "FilterText:=" , "", "AnalysisEnabled:=" , 1, ["NAME:OutputQuantities"], ["NAME:NoiseOutputQuantities"], "Name:=" , "DCAnalysis1" [

```
            "NAME:SweepDefinition",
            "Variable:="            , "Temp",
            "Data:="                , "LINC 10cel 200cel 10 LINC 20cel 400cel 10",
            "OffsetF1:="            , False,
            "Synchronize:="         , 0
    ]
  ])
```

Edit Envelope Setup[Nexxim]

Use: Edits an Envelope Analysis setup

Syntax: EditEnvelopeSetup(

```
STRING name,
ARRAY newsetup) // new setup data
```

VB Example:

```
oModule.EditEnvelopeSetup "Envelope1", Array("NAME:SimSetup", "DataBlockID:=", 21,
"SimSetupID:=", 28, "OptionName:=", _
"(Default Options)", "AdditionalOptions:=", "", "AlterBlockName:=", "", "FilterText:=", _
"", "AnalysisEnabled:=", 1, Array("NAME:OutputQuantities", Array("NAME:Quantity", "NodeType:=", _
"Harmonics", "CompID:=", "", "CompName:=", "", "QuantityName:=", "2F1", "Selected:=", _
true, "UnitType:=", "NoUnit", "NetlistFUT:=", "", "DataType:=", "Real", "TypeInfo:=", Array( _
```

```
"", 0, "", "IsUnconstrained:=", false), "CircuitInstanceID:=", ""), Array("NAME:Quantity",
"NodeType:=", _
"Harmonics", "CompID:=", "", "CompName:=", "", "QuantityName:=", "3F1", "Selected:=", _
true, "UnitType:=", "NoUnit", "NetlistFUT:=", "", "DataType:=", "Real", "TypeInfo:=", Array( _
"", 0, "", "IsUnconstrained:=", false), "CircuitInstanceID:=", ""), Array("NAME:Quantity",
"NodeType:=", _
"Harmonics", "CompID:=", "", "CompName:=", "", "QuantityName:=", "4F1", "Selected:=", _
true, "UnitType:=", "NoUnit", "NetlistFUT:=", "", "DataType:=", "Real", "TypeInfo:=", Array( _
"", 0, "", "IsUnconstrained:=", false), "CircuitInstanceID:=", ""), Array("NAME:Quantity",
"NodeType:=", _
"Harmonics", "CompID:=", "", "CompName:=", "", "QuantityName:=", "DC", "Selected:=", _
true, "UnitType:=", "NoUnit", "NetlistFUT:=", "", "DataType:=", "Real", "TypeInfo:=", Array( _
"", 0, "", "IsUnconstrained:=", false), "CircuitInstanceID:=", ""), Array("NAME:Quantity",
"NodeType:=", _
"Harmonics", "CompID:=", "", "CompName:=", "", "QuantityName:=", "F1", "Selected:=", _
true, "UnitType:=", "NoUnit", "NetlistFUT:=", "", "DataType:=", "Real", "TypeInfo:=", Array( _
"", 0, "", "IsUnconstrained:=", false), "CircuitInstanceID:=", "")), Array("NAME:NoiseOut-
putQuantities"), "Name:=", _
"Envelope1", "EnvelopeData:=", Array(1, "0", "1e-008", "", "", "trap", "HB", _
"No", 0), "EnvelopeMaxkValues:=", Array("F1", "7"), "EnvelopeTonesValues:=", Array( _
"1Ghz"), "HarmonicsText:=", "2F1 3F1 4F1 DC F1")
```

Python Syntax	EditEnvelopeSetup(<name>,<setupData[]>)
Python Example	oModule.EditEnvelopeSetup("Envelope", ["NAME:SimSetup", "DataBlockID:=" , 21, "SimSetupID:=" , 26, "OptionName:=" , "(Default Options)", "AdditionalOptions:=" , "", "AlterBlockName:=" , "", "FilterText:=" , "", "AnalysisEnabled:=" , 1, ["NAME:OutputQuantities", ["NAME:Quantity", "NodeType:=" , "Harmonics", "CompID:=" , "", "CompName:=" , "", "QuantityName:=" , "2F1",

```
                    "Selected:="            , True,
                    "UnitType:="            , "NoUnit",
                    "NetlistFUT:="          , "",
                    "DataType:="            , "Real",
                    "TypeInfo:="            , ["",0,"",
                    "IsUnconstrained:="     , False],
                    "CircuitInstanceID:="   , ""
                ],
                [
                    "NAME:Quantity",
                    "NodeType:="            , "Harmonics",
                    "CompID:="              , "",
                    "CompName:="            , "",
                    "QuantityName:="        , "3F1",
                    "Selected:="            , True,
                    "UnitType:="            , "NoUnit",
                    "NetlistFUT:="          , "",
                    "DataType:="            , "Real",
                    "TypeInfo:="            , ["",0,"",
                    "IsUnconstrained:="     , False],
                    "CircuitInstanceID:="   , ""
```

```
        ],
        [
                "NAME:Quantity",
                "NodeType:="            , "Harmonics",
                "CompID:="              , "",
                "CompName:="            , "",
                "QuantityName:="        , "4F1",
                "Selected:="            , True,
                "UnitType:="            , "NoUnit",
                "NetlistFUT:="          , "",
                "DataType:="            , "Real",
                "TypeInfo:="            , ["",0,"",
                "IsUnconstrained:="     , False],
                "CircuitInstanceID:="   , ""
        ],
        [
                "NAME:Quantity",
                "NodeType:="            , "Harmonics",
                "CompID:="              , "",
                "CompName:="            , "",
```

```
                    "QuantityName:="        , "DC",
                    "Selected:="            , True,
                    "UnitType:="            , "NoUnit",
                    "NetlistFUT:="          , "",
                    "DataType:="            , "Real",
                    "TypeInfo:="            , ["",0,"",
                    "IsUnconstrained:="     , False],
                    "CircuitInstanceID:="   , ""
                ],
                [
                    "NAME:Quantity",
                    "NodeType:="            , "Harmonics",
                    "CompID:="              , "",
                    "CompName:="            , "",
                    "QuantityName:="        , "F1",
                    "Selected:="            , True,
                    "UnitType:="            , "NoUnit",
                    "NetlistFUT:="          , "",
                    "DataType:="            , "Real",
                    "TypeInfo:="            , ["",0,"",
                    "IsUnconstrained:="     , False],
```

```
                "CircuitInstanceID:="   , ""
            ]
        ],
        [
            "NAME:NoiseOutputQuantities"
        ],
        "Name:="          , "PXF",
        "PXFData:="       , [1,"HB",False,"HB","No",0],
        "PXFMaxkValues:=" , ["F1","7"],
        "PXFTonesValues:="        , ["1Ghz"],
        "PXFRelHarmNums:="        , [],
        "PXFMaxSideband:="        , ["0"],
        "PXFSideband:="           , ["0"],
        "HarmonicsText:=" , "2F1 3F1 4F1 DC F1",
        [
            "NAME:SweepDefinition",
            "Variable:="          , "F",
            "Data:="              , "LINC 10GHz 100GHz 10",
            "OffsetF1:="          , False,
            "Synchronize:="               , 0
```

```
            ],
            "Name:="          , "Envelope",
            "EnvelopeData:="          , [1,"0","1e-8","","","trap","HB","No",0],
            "EnvelopeMaxkValues:="    , ["F1","7"],
            "EnvelopeTonesValues:="   , ["1Ghz"],
            "HarmonicsText:=" , "2F1 3F1 4F1 DC F1",
  ])
```

Edit HB 1-Tone [Nexxim]

Use: Edits a Harmonic Balance 1-Tone setup

Syntax: EditHB1Tone(

```
STRING setupname

ARRAY setup) // setup data
```

VB Example:

```
oModule.EditHB1Tone "HB1Tone", Array("NAME:SimSetup", "DataBlockID:=", 9, "SimSetupID:=", 3,
"OptionName:=", _

"(Default Options)", "AdditionalOptions:=", "", "AlterBlockName:=", "", "FilterText:=", _

"", "AnalysisEnabled:=", 1, Array("NAME:OutputQuantities", Array("NAME:Quantity", "NodeType:=",

_
"Harmonics", "CompID:=", "", "CompName:=", "", "QuantityName:=", "2F1", "Selected:=", _

true, "UnitType:=", "NoUnit", "NetlistFUT:=", "", "DataType:=", "Real", "TypeInfo:=", Array( _

"", 0, "", "IsUnconstrained:=", false), "CircuitInstanceID:=", ""), Array("NAME:Quantity",
"NodeType:=", _
```

```
"Harmonics", "CompID:=", "", "CompName:=", "", "QuantityName:=", "3F1", "Selected:=", _
true, "UnitType:=", "NoUnit", "NetlistFUT:=", "", "DataType:=", "Real", "TypeInfo:=", Array( _
"", 0, "", "IsUnconstrained:=", false), "CircuitInstanceID:=", ""), Array("NAME:Quantity",
"NodeType:=", _
"Harmonics", "CompID:=", "", "CompName:=", "", "QuantityName:=", "4F1", "Selected:=", _
true, "UnitType:=", "NoUnit", "NetlistFUT:=", "", "DataType:=", "Real", "TypeInfo:=", Array( _
"", 0, "", "IsUnconstrained:=", false), "CircuitInstanceID:=", ""), Array("NAME:Quantity",
"NodeType:=", _
"Harmonics", "CompID:=", "", "CompName:=", "", "QuantityName:=", "DC", "Selected:=", _
true, "UnitType:=", "NoUnit", "NetlistFUT:=", "", "DataType:=", "Real", "TypeInfo:=", Array( _
"", 0, "", "IsUnconstrained:=", false), "CircuitInstanceID:=", ""), Array("NAME:Quantity",
"NodeType:=", _
"Harmonics", "CompID:=", "", "CompName:=", "", "QuantityName:=", "F1", "Selected:=", _
true, "UnitType:=", "NoUnit", "NetlistFUT:=", "", "DataType:=", "Real", "TypeInfo:=", Array( _
"", 0, "", "IsUnconstrained:=", false), "CircuitInstanceID:=", "")), Array("NAME:NoiseOut-
putQuantities"), "Name:=", _
"HB1Tone1", "HB1ToneData:=", Array(7, "1GHz", "HB", "No", "0.0"), "HarmonicsText:=", _
"2F1 3F1 4F1 DC F1", Array("NAME:SweepDefinition", "Variable:=", "Temp", "Data:=", _
"LINC 10cel 200cel 10", "OffsetF1:=", false, "Synchronize:=", 0)
)
```

Python Syntax	EditHB1Tone(<setupName>,<setupData[]>)
Python Example	oModule.EditHB1Tone("HB1Tone", ["NAME:SimSetup", "DataBlockID:=" , 9, "SimSetupID:=" , 1, "OptionName:=" , "(Default Options)", "AdditionalOptions:=" , "", "AlterBlockName:=" , "", "FilterText:=" , "", "AnalysisEnabled:=" , 1, ["NAME:OutputQuantities", ["NAME:Quantity", "NodeType:=" , "Harmonics", "CompID:=" , "", "CompName:=" , "", "QuantityName:=" , "2F1", "Selected:=" , True, "UnitType:=" , "NoUnit",

```
      "NetlistFUT:="                , "",
      "DataType:="          , "Real",
      "TypeInfo:="          , ["",0,"",
      "IsUnconstrained:="   , False],
      "CircuitInstanceID:=" , ""
    ],
    [
      "NAME:Quantity",
      "NodeType:="          , "Harmonics",
      "CompID:="            , "",
      "CompName:="          , "",
      "QuantityName:="      , "3F1",
      "Selected:="          , True,
      "UnitType:="          , "NoUnit",
      "NetlistFUT:="                , "",
      "DataType:="          , "Real",
      "TypeInfo:="          , ["",0,"",
      "IsUnconstrained:="   , False],
      "CircuitInstanceID:=" , ""
    ],
```

```
    [
      "NAME:Quantity",
      "NodeType:="          , "Harmonics",
      "CompID:="            , "",
      "CompName:="          , "",
      "QuantityName:="      , "4F1",
      "Selected:="          , True,
      "UnitType:="          , "NoUnit",
      "NetlistFUT:="        , "",
      "DataType:="          , "Real",
      "TypeInfo:="          , ["",0,"",
      "IsUnconstrained:="   , False],
      "CircuitInstanceID:=" , ""
    ],
    [
     "NAME:Quantity",
      "NodeType:="          , "Harmonics",
      "CompID:="            , "",
      "CompName:="          , "",
      "QuantityName:="      , "DC",
      "Selected:="          , True,
```

```
      "UnitType:="          , "NoUnit",
      "NetlistFUT:="        , "",
      "DataType:="          , "Real",
      "TypeInfo:="          , ["",0,"",
      "IsUnconstrained:="   , False],
      "CircuitInstanceID:=" , ""
    ],
    [
     "NAME:Quantity",
      "NodeType:="          , "Harmonics",
      "CompID:="            , "",
      "CompName:="          , "",
      "QuantityName:="      , "F1",
      "Selected:="          , True,
      "UnitType:="          , "NoUnit",
      "NetlistFUT:="        , "",
      "DataType:="          , "Real",
      "TypeInfo:="          , ["",0,"",
      "IsUnconstrained:="   , False],
      "CircuitInstanceID:=" , ""
```

```
    ],
  ],
  [
    "NAME:NoiseOutputQuantities"
  ],
  "Name:="                , "HB1Tone",
  "HB1ToneData:="         , [7,"1GHz","HB","No","0.0"],
  "HarmonicsText:="       , "2F1 3F1 4F1 DC F1"
  [
    "NAME:SweepDefinition",
    "Variable:="            , "Temp",
    "Data:="                , "LINC 10cel 200cel 10",
    "OffsetF1:="            , False,
    "Synchronize:="         , 0
  ]
])
```

Edit HB N-Tone [Nexxim]

Use: Edits a Harmonic Balance N-Tone setup

Syntax: EditHBnTone(

```
STRING setupname
ARRAY setup) // setup data
```

VB Example:

```
oModule.EditHBnTone "HBnTone", Array("NAME:SimSetup", "DataBlockID:=", 17, "SimSetupID:=", 8,
"OptionName:=", _
"(Default Options)", "AdditionalOptions:=", "", "AlterBlockName:=", "", "FilterText:=", _
"", "AnalysisEnabled:=", 1, Array("NAME:OutputQuantities", Array("NAME:Quantity", "NodeType:=", _
"Harmonics", "CompID:=", "", "CompName:=", "", "QuantityName:=", "2F1", "Selected:=", _
true, "UnitType:=", "NoUnit", "NetlistFUT:=", "", "DataType:=", "Real", "TypeInfo:=", Array( _
"", 0, "", "IsUnconstrained:=", false), "CircuitInstanceID:=", ""), Array("NAME:Quantity",
"NodeType:=", _
"Harmonics", "CompID:=", "", "CompName:=", "", "QuantityName:=", "3F1", "Selected:=", _
true, "UnitType:=", "NoUnit", "NetlistFUT:=", "", "DataType:=", "Real", "TypeInfo:=", Array( _
"", 0, "", "IsUnconstrained:=", false), "CircuitInstanceID:=", ""), Array("NAME:Quantity",
"NodeType:=", _
"Harmonics", "CompID:=", "", "CompName:=", "", "QuantityName:=", "4F1", "Selected:=", _
true, "UnitType:=", "NoUnit", "NetlistFUT:=", "", "DataType:=", "Real", "TypeInfo:=", Array( _
"", 0, "", "IsUnconstrained:=", false), "CircuitInstanceID:=", ""), Array("NAME:Quantity",
"NodeType:=", _
"Harmonics", "CompID:=", "", "CompName:=", "", "QuantityName:=", "DC", "Selected:=", _
true, "UnitType:=", "NoUnit", "NetlistFUT:=", "", "DataType:=", "Real", "TypeInfo:=", Array( _
"", 0, "", "IsUnconstrained:=", false), "CircuitInstanceID:=", ""), Array("NAME:Quantity",
"NodeType:=", _
"Harmonics", "CompID:=", "", "CompName:=", "", "QuantityName:=", "F1", "Selected:=", _
```

```
true, "UnitType:=", "NoUnit", "NetlistFUT:=", "", "DataType:=", "Real", "TypeInfo:=", Array( _
"", 0, "", "IsUnconstrained:=", false), "CircuitInstanceID:=", "")), Array("NAME:NoiseOut-
putQuantities"), "Name:=", _
"HBnTone1", "HBnTone:=", Array(1, "HB", "No", "0.0","0.0","F1","7"), HBnValues:=" Array( _
"1GHz"), "HarmonicsText:=", "2F1 3F1 4F1 DC F1", Array("NAME:SweepDefinition", "Variable:=", _
"Temp", "Data:=", "LINC 10cel 200cel 10", "OffsetF1:=", false, "Synchronize:=", 0)
)
```

Python Syntax	EditHBnTone(<setupName>,<setupData[]>)
Python Example	oModule.EditHBnTone("HBnTone", ["NAME:SimSetup", "DataBlockID:=" , 17, "SimSetupID:=" , 8, "OptionName:=" , "(Default Options)", "AdditionalOptions:=" , "", "AlterBlockName:=" , "", "FilterText:=" , "", "AnalysisEnabled:=" , 1, ["NAME:OutputQuantities",

```
    [
      "NAME:Quantity",
      "NodeType:="          , "Harmonics",
      "CompID:="            , "",
      "CompName:="          , "",
      "QuantityName:="      , "2F1",
      "Selected:="          , True,
      "UnitType:="          , "NoUnit",
      "NetlistFUT:="                , "",
      "DataType:="          , "Real",
      "TypeInfo:="          , ["",0,"",
      "IsUnconstrained:="   , False],
      "CircuitInstanceID:=" , ""
    ],
    [
      "NAME:Quantity",
      "NodeType:="          , "Harmonics",
      "CompID:="            , "",
      "CompName:="          , "",
      "QuantityName:="      , "3F1",
```

```
      "Selected:="          , True,
      "UnitType:="          , "NoUnit",
      "NetlistFUT:="                , "",
      "DataType:="          , "Real",
      "TypeInfo:="          , ["",0,"",
      "IsUnconstrained:="   , False],
      "CircuitInstanceID:=" , ""
    ],
    [
      "NAME:Quantity",
      "NodeType:="          , "Harmonics",
      "CompID:="            , "",
      "CompName:="          , "",
      "QuantityName:="      , "4F1",
      "Selected:="          , True,
      "UnitType:="          , "NoUnit",
      "NetlistFUT:="        , "",
      "DataType:="          , "Real",
      "TypeInfo:="          , ["",0,"",
      "IsUnconstrained:="   , False],
      "CircuitInstanceID:=" , ""
```

```
    ],
    [
     "NAME:Quantity",
      "NodeType:="          , "Harmonics",
      "CompID:="            , "",
      "CompName:="          , "",
      "QuantityName:="      , "DC",
      "Selected:="          , True,
      "UnitType:="          , "NoUnit",
      "NetlistFUT:="        , "",
      "DataType:="          , "Real",
      "TypeInfo:="          , ["",0,"",
      "IsUnconstrained:="   , False],
      "CircuitInstanceID:=" , ""
    ],
    [
     "NAME:Quantity",
      "NodeType:="          , "Harmonics",
      "CompID:="            , "",
      "CompName:="          , "",
```

```
      "QuantityName:="      , "F1",
      "Selected:="          , True,
      "UnitType:="          , "NoUnit",
      "NetlistFUT:="        , "",
      "DataType:="          , "Real",
      "TypeInfo:="          , ["",0,"",
      "IsUnconstrained:="   , False],
      "CircuitInstanceID:=" , ""
    ],
  ],
  [
    "NAME:NoiseOutputQuantities"
  ],
  "Name:="               , "HBnTone",
  "HBnTone:="            , [7,"HB","No","0.0","0.0","F1","7"],
  "HBnValues:="          , [1GHz],
  "HarmonicsText:="      , "2F1 3F1 4F1 DC F1"
  [
    "NAME:SweepDefinition",
    "Variable:="           , "Temp",
    "Data:="               , "LINC 10cel 200cel 10",
```

	"OffsetF1:=" , False, "Synchronize:=" , 0]])

Edit HSPICE Transient [Nexxim]

Use: Edits an HSPICE Transient Analysis setup

Syntax: EditHSPICETransient(

```
STRING setupname,

ARRAY newsetup) // new setup data
```

VB Example:

```
oModule.AddHSPICETransient "HSPICETransient1", Array("NAME:SimSetup", "DataBlockID:=", 30,
"SimSetupID:=", _

11, "OptionName:=", "(Default Options)", "AdditionalOptions:=", "", "AlterBlockName:=", _

"", "FilterText:=", "", "AnalysisEnabled:=", 1, Array("NAME:OutputQuantities"), Array
("NAME:NoiseOutputQuantities"), "Name:=", _

"HSPICETransient1", ""HSPICETransientData:=", Array("0.1ns", "10ns"), "HSPICETran-
sientOtherData:=", Array( _

3))
```

Python Syntax	EditHSPICETransient(<name>,<setupData[]>)
Python Example	oModule.AddHSPICETransient("HSPICETransient",

```
[
  "NAME:SimSetup",
  "DataBlockID:=" , 30,
  "SimSetupID:=" , 9,
  "OptionName:=" , "(Default Options)",
  "AdditionalOptions:=" , "",
  "AlterBlockName:=" , "",
  "FilterText:=" , "",
  "AnalysisEnabled:=" , 1,
  [
    "NAME:OutputQuantities"
  ],
  [
    "NAME:NoiseOutputQuantities"
  ],
  "Name:=" , "HSPICETransient",
  "HSPICETransientData:=" , ["0.1ns","10ns"],
  "HSPICETransientOtherData:=", [3]
])
```

Edit Linear Network Analysis [Nexxim]

Use: Edits a Linear Network analysis setup

Syntax: EditLinearNetworkAnalysis(

```
STRING setupname

ARRAY newsetup) // modified setup data
```

VB Example:

```
oModule.EditLinearNetworkAnalysis "LinearFrequency", Array("NAME:SimSetup", "DataBlockID:=", _

16, "SimSetupID:=", 15, "OptionName:=", "(Default Options)", "AdditionalOptions:=", _

"", "AlterBlockName:=", "", "FilterText:=", "", "AnalysisEnabled:=", 1, Array("NAME:Out-
putQuantities"), Array("NAME:NoiseOutputQuantities"), "Name:=", _

"LinearFrequency", "LinearFrequencyData:=", Array(true, 0.1, false, "", true), Array("NAME:Sweep-
Definition", "Variable:=", _

"F", "Data:=", "LINC 1GHz 100GHz 10", "OffsetF1:=", false, "Synchronize:=", 0))
```

Python Syntax	EditLinearNetworkAnalysis(<setupName>,<setupData[]>)
Python Example	oModule.EditLinearNetworkAnalysis("LinearFrequency1", ["NAME:SimSetup", "DataBlockID:=" , 16,

```
    "SimSetupID:="          , 17,
    "OptionName:="          , "(Default Options)",
    "AdditionalOptions:="   , "",
    "AlterBlockName:="      , "",
    "FilterText:="          , "",
    "AnalysisEnabled:="     , 1,
    [
            "NAME:OutputQuantities"
    ],
    [
            "NAME:NoiseOutputQuantities"
    ],
    "Name:="                , "LinearFrequency",
    "LinearFrequencyData:=" , [True,"0.1",False,"",True],
    [
            "NAME:SweepDefinition",
            "Variable:="    , "F",
            "Data:="        , "LINC 1GHz 100GHz 10",
            "OffsetF1:="            , False,
            "Synchronize:="         , 0
    ]
```

	])

Edit Oscillator Analysis [Nexxim]

Use: Edits an Oscillator analysis setup

Syntax: EditOscillatorAnalysis(

```
STRING setup

ARRAY setupdata) // modified setup data
```

VB Example:

```
oModule.EditOscillatorAnalysis "Oscillator1", Array("NAME:SimSetup", "DataBlockID:=", 20,
"SimSetupID:=", 23, "OptionName:=", _

"(Default Options)", "AdditionalOptions:=", "", "AlterBlockName:=", "", "FilterText:=", _

"", "AnalysisEnabled:=", 1, Array("NAME:OutputQuantities", Array("NAME:Quantity", "NodeType:=", _

"Harmonics", "CompID:=", "", "CompName:=", "", "QuantityName:=", "2F1", "Selected:=", _

true, "UnitType:=", "NoUnit", "NetlistFUT:=", "", "DataType:=", "Real", "TypeInfo:=", Array( _

"", 0, "", "IsUnconstrained:=", false), "CircuitInstanceID:=", ""), Array("NAME:Quantity",
"NodeType:=", _

"Harmonics", "CompID:=", "", "CompName:=", "", "QuantityName:=", "3F1", "Selected:=", _

true, "UnitType:=", "NoUnit", "NetlistFUT:=", "", "DataType:=", "Real", "TypeInfo:=", Array( _

"", 0, "", "IsUnconstrained:=", false), "CircuitInstanceID:=", ""), Array("NAME:Quantity",
"NodeType:=", _

"Harmonics", "CompID:=", "", "CompName:=", "", "QuantityName:=", "4F1", "Selected:=", _

true, "UnitType:=", "NoUnit", "NetlistFUT:=", "", "DataType:=", "Real", "TypeInfo:=", Array( _
```

```
"", 0, "", "IsUnconstrained:=", false), "CircuitInstanceID:=", ""), Array("NAME:Quantity",
"NodeType:=", _
"Harmonics", "CompID:=", "", "CompName:=", "", "QuantityName:=", "DC", "Selected:=", _
true, "UnitType:=", "NoUnit", "NetlistFUT:=", "", "DataType:=", "Real", "TypeInfo:=", Array( _
"", 0, "", "IsUnconstrained:=", false), "CircuitInstanceID:=", ""), Array("NAME:Quantity",
"NodeType:=", _
"Harmonics", "CompID:=", "", "CompName:=", "", "QuantityName:=", "F1", "Selected:=", _
true, "UnitType:=", "NoUnit", "NetlistFUT:=", "", "DataType:=", "Real", "TypeInfo:=", Array( _
"", 0, "", "IsUnconstrained:=", false), "CircuitInstanceID:=", "")), Array("NAME:NoiseOut-
putQuantities"), "Name:=", _
"TVNoise1", "TVNoiseData:=", Array(1, "", "", "", "", "HB", false, "HB", "No", _
0), "TVNoiseMaxkValues:=", Array("F1", "7"), "TVNoiseTonesValues:=", Array( _
"1Ghz"), "TVNoiseRelHarmNums:=", Array(), "TVNoiseRefSideband:=", Array("1"), "HarmonicsText:=",
_
"2F1 3F1 4F1 DC F1", Array("NAME:SweepDefinition", "Variable:=", "F", "Data:=", _
"LINC 10GHz 100GHz 10", "OffsetF1:=", false, "Synchronize:=", 0), Array("NAME:FSweep", "Vari-
able:=", _
"F", "Data:=", "LINC 10GHz 100GHz 10", "OffsetF1:=", false, "Synchronize:=", 0))
```

Python Syntax	EditOscillatorAnalysis(<setupName>,<setupData[]>)
Python Example	oModule.EditOscillatorAnalysis("Oscillator", [

```
    "NAME:SimSetup",
    "DataBlockID:="         , 21,
    "SimSetupID:="          , 26,
    "OptionName:="          , "(Default Options)",
    "AdditionalOptions:="   , "",
    "AlterBlockName:="      , "",
    "FilterText:="          , "",
    "AnalysisEnabled:="     , 1,
    [
            "NAME:OutputQuantities",
            [
                    "NAME:Quantity",
                    "NodeType:="            , "Harmonics",
                    "CompID:="              , "",
                    "CompName:="            , "",
                    "QuantityName:="        , "2F1",
                    "Selected:="            , True,
                    "UnitType:="            , "NoUnit",
                    "NetlistFUT:="          , "",
                    "DataType:="            , "Real",
```

```
                "TypeInfo:="            , ["",0,"",
                "IsUnconstrained:="     , False],
                "CircuitInstanceID:="   , ""
        ],
        [
                "NAME:Quantity",
                "NodeType:="            , "Harmonics",
                "CompID:="              , "",
                "CompName:="            , "",
                "QuantityName:="        , "3F1",
                "Selected:="            , True,
                "UnitType:="            , "NoUnit",
                "NetlistFUT:="          , "",
                "DataType:="            , "Real",
                "TypeInfo:="            , ["",0,"",
                "IsUnconstrained:="     , False],
                "CircuitInstanceID:="   , ""
        ],
        [
                "NAME:Quantity",
                "NodeType:="            , "Harmonics",
```

```
                    "CompID:="              , "",
                    "CompName:="            , "",
                    "QuantityName:="        , "4F1",
                    "Selected:="            , True,
                    "UnitType:="            , "NoUnit",
                    "NetlistFUT:="          , "",
                    "DataType:="            , "Real",
                    "TypeInfo:="            , ["",0,"",
                    "IsUnconstrained:="     , False],
                    "CircuitInstanceID:="   , ""
                ],
                [
                    "NAME:Quantity",
                    "NodeType:="            , "Harmonics",
                    "CompID:="              , "",
                    "CompName:="            , "",
                    "QuantityName:="        , "DC",
                    "Selected:="            , True,
                    "UnitType:="            , "NoUnit",
                    "NetlistFUT:="          , "",
```

```
                    "DataType:="            , "Real",
                    "TypeInfo:="            , ["",0,"",
                    "IsUnconstrained:="     , False],
                    "CircuitInstanceID:="   , ""
                ],
                [
                    "NAME:Quantity",
                    "NodeType:="            , "Harmonics",
                    "CompID:="              , "",
                    "CompName:="            , "",
                    "QuantityName:="        , "F1",
                    "Selected:="            , True,
                    "UnitType:="            , "NoUnit",
                    "NetlistFUT:="          , "",
                    "DataType:="            , "Real",
                    "TypeInfo:="            , ["",0,"",
                    "IsUnconstrained:="     , False],
                    "CircuitInstanceID:="   , ""
                ]
            ],
            [
```

```
            "NAME:NoiseOutputQuantities"
          ],
          "Name:="          , "Oscillator",
          "OscillatorData:="        , [7,"1Ghz",False,1],
          "HarmonicsText:=" , "2F1 3F1 4F1 DC F1",
          [
            "NAME:SweepDefinition",
            "Variable:="            , "Fnoi",
            "Data:="                , "LINC 10GHz 100GHz 10",
            "OffsetF1:="            , False,
            "Synchronize:="         , 0
          ],
  ])
```

Edit Oscillator N-Tone Analysis [Nexxim]

Use: Edits an Oscillator N-Tone analysis setup

Syntax: EditOscillatorNtoneAnalysis(

```
STRING setupname,
ARRAY newsetup) // new setup data
```

VB Example:

```
oModule.EditOscillatorNtoneAnalysis "OscillatorNtone2", Array("NAME:SimSetup", "DataBlockID:=",
23, "SimSetupID:=", _
5, "OptionName:=", "(Default Options)", "AdditionalOptions:=", "", "AlterBlockName:=", _
"", "FilterText:=", "", "AnalysisEnabled:=", 1, Array("NAME:OutputQuantities", Array("NAME:Quant-
ity", "NodeType:=", _
"Harmonics", "CompID:=", "", "CompName:=", "", "QuantityName:=", "2F2", "Selected:=", _
true, "UnitType:=", "NoUnit", "NetlistFUT:=", "", "DataType:=", "Real", "TypeInfo:=", Array( _
"", 0, "", "IsUnconstrained:=", false), "CircuitInstanceID:=", ""), Array("NAME:Quantity",
"NodeType:=", _
"Harmonics", "CompID:=", "", "CompName:=", "", "QuantityName:=", "3F2", "Selected:=", _
true, "UnitType:=", "NoUnit", "NetlistFUT:=", "", "DataType:=", "Real", "TypeInfo:=", Array( _
"", 0, "", "IsUnconstrained:=", false), "CircuitInstanceID:=", ""), Array("NAME:Quantity",
"NodeType:=", _
"Harmonics", "CompID:=", "", "CompName:=", "", "QuantityName:=", "4F2", "Selected:=", _
true, "UnitType:=", "NoUnit", "NetlistFUT:=", "", "DataType:=", "Real", "TypeInfo:=", Array( _
"", 0, "", "IsUnconstrained:=", false), "CircuitInstanceID:=", ""), Array("NAME:Quantity",
"NodeType:=", _
"Harmonics", "CompID:=", "", "CompName:=", "", "QuantityName:=", "DC", "Selected:=", _
true, "UnitType:=", "NoUnit", "NetlistFUT:=", "", "DataType:=", "Real", "TypeInfo:=", Array( _
"", 0, "", "IsUnconstrained:=", false), "CircuitInstanceID:=", ""), Array("NAME:Quantity",
"NodeType:=", _
"Harmonics", "CompID:=", "", "CompName:=", "", "QuantityName:=", "F2", "Selected:=", _
true, "UnitType:=", "NoUnit", "NetlistFUT:=", "", "DataType:=", "Real", "TypeInfo:=", Array( _
```

```
"", 0, "", "IsUnconstrained:=", false), "CircuitInstanceID:=", "")), Array("NAME:NoiseOut-
putQuantities"), "Name:=", _
"OscillatorNtone2", "OscillatorNtoneData:=", Array(1, 1, true, 1), "OscMaxkValues:=", Array( _
"F1", "7"), "OscTonesValues:=", Array("1Ghz"), "DrivenMaxkValues:=", Array("RF1", _
"7"), "DrivenTonesValues:=", Array("1Ghz"), "HarmonicsText:=", _
"2F2 3F2 4F2 DC F2", Array("NAME:SweepDefinition", "Variable:=", "FNoi", "Data:=", _
"LINC 30GHz 300GHz 10", "OffsetF1:=", false, "Synchronize:=", 0))
```

Python Syntax	EditOscillatorNtoneAnalysis(<name>,<setupData[]>)
Python Example	oModule.EditOscillatorNtoneAnalysis("OscillatorNtone", ["NAME:SimSetup", "DataBlockID:=" , 23, "SimSetupID:=" , 1, "OptionName:=" , "(Default Options)", "AdditionalOptions:=" , "", "AlterBlockName:=" , "", "FilterText:=" , "", "AnalysisEnabled:=" , 1, [

```
"NAME:OutputQuantities",
[
"NAME:Quantity",
"NodeType:=" , "Harmonics",
"CompID:=" , "",
"CompName:=" , "",
"QuantityName:=" , "2F2",
"Selected:=" , True,
"UnitType:=" , "NoUnit",
"NetlistFUT:=" , "",
"DataType:=" , "Real",
"TypeInfo:=" , ["",0,"", "IsUnconstrained:=" , False],
"CircuitInstanceID:=" , ""
],
[
"NAME:Quantity",
"NodeType:=" , "Harmonics",
"CompID:=" , "",
"CompName:=" , "",
"QuantityName:=" , "3F2",
"Selected:=" , True,
```

```
    "UnitType:=" , "NoUnit",
    "NetlistFUT:=" , "",
    "DataType:=" , "Real",
    "TypeInfo:=" , ["",0,"", "IsUnconstrained:=" , False],
    "CircuitInstanceID:=" , ""
    ],
    [
    "NAME:Quantity",
    "NodeType:=" , "Harmonics",
    "CompID:=" , "",
    "CompName:=" , "",
    "QuantityName:=" , "4F2",
    "Selected:=" , True,
    "UnitType:=" , "NoUnit",
    "NetlistFUT:=" , "",
    "DataType:=" , "Real",
    "TypeInfo:=" , ["",0,"", "IsUnconstrained:=" , False],
    "CircuitInstanceID:=" , ""
    ],
    [
```

```
"NAME:Quantity",
"NodeType:=" , "Harmonics",
"CompID:=" , "",
"CompName:=" , "",
"QuantityName:=" , "DC",
"Selected:=" , True,
"UnitType:=" , "NoUnit",
"NetlistFUT:=" , "",
"DataType:=" , "Real",
"TypeInfo:=" , ["",0,"", "IsUnconstrained:=" , False],
"CircuitInstanceID:=" , ""
],
[
"NAME:Quantity",
"NodeType:=" , "Harmonics",
"CompID:=" , "",
"CompName:=" , "",
"QuantityName:=" , "F2",
"Selected:=" , True,
"UnitType:=" , "NoUnit",
"NetlistFUT:=" , "",
```

```
    "DataType:=" , "Real",
    "TypeInfo:=" , ["",0,"", "IsUnconstrained:=" , False],
    "CircuitInstanceID:=" , ""
    ],
  ],
  [
    "NAME:NoiseOutputQuantities"
  ],
  "Name:=" , "OscillatorNtone",
  "OscillatorNtoneData:=" , [1,1,True,1],
  "OscMaxkValues:=" , ["F1","7"],
  "OscTonesValues:=" , ["1Ghz"],
  "DrivenMaxkValues:=" , ["RF1","7"],
  "DrivenTonesValues:=" , ["1Ghz"],
  "HarmonicsText:=" , "2F2 3F2 4F2 DC F2",
    [
    "NAME:SweepDefinition",
    "Variable:=" , "FNoi",
    "Data:=" , "LINC 1GHz 200GHz 10",
    "OffsetF1:=" , False,
```

	"Synchronize:=" , 0]])

Edit Oscillator Resonant Frequency Search [Nexxim]

Use: Edits an Oscillator Resonant Frequency Search setup

Syntax: EditOscResonantFrequencySearch(

```
STRING name,

ARRAY newsetup) // new setup data
```

VB Example:

```
oModule.EditOscResonantFrequencySearch "OscResonantFrequency1", Array("NAME:SimSetup", "DataB-
lockID:=", 21, "SimSetupID:=", 28, "OptionName:=", _

"(Default Options)", "AdditionalOptions:=", "", "AlterBlockName:=", "", "FilterText:=", _

"", "AnalysisEnabled:=", 1, Array("NAME:OutputQuantities", Array("NAME:Quantity", "NodeType:=",

_
"Harmonics", "CompID:=", "", "CompName:=", "", "QuantityName:=", "2F1", "Selected:=", _

true, "UnitType:=", "NoUnit", "NetlistFUT:=", "", "DataType:=", "Real", "TypeInfo:=", Array( _

"", 0, "", "IsUnconstrained:=", false), "CircuitInstanceID:=", ""), Array("NAME:Quantity",
"NodeType:=", _

"Harmonics", "CompID:=", "", "CompName:=", "", "QuantityName:=", "3F1", "Selected:=", _

true, "UnitType:=", "NoUnit", "NetlistFUT:=", "", "DataType:=", "Real", "TypeInfo:=", Array( _

"", 0, "", "IsUnconstrained:=", false), "CircuitInstanceID:=", ""), Array("NAME:Quantity",
"NodeType:=", _

"Harmonics", "CompID:=", "", "CompName:=", "", "QuantityName:=", "4F1", "Selected:=", _
```

```
true, "UnitType:=", "NoUnit", "NetlistFUT:=", "", "DataType:=", "Real", "TypeInfo:=", Array( _
"", 0, "", "IsUnconstrained:=", false), "CircuitInstanceID:=", ""), Array("NAME:Quantity",
"NodeType:=", _
"Harmonics", "CompID:=", "", "CompName:=", "", "QuantityName:=", "DC", "Selected:=", _
true, "UnitType:=", "NoUnit", "NetlistFUT:=", "", "DataType:=", "Real", "TypeInfo:=", Array( _
"", 0, "", "IsUnconstrained:=", false), "CircuitInstanceID:=", ""), Array("NAME:Quantity",
"NodeType:=", _
"Harmonics", "CompID:=", "", "CompName:=", "", "QuantityName:=", "F1", "Selected:=", _
true, "UnitType:=", "NoUnit", "NetlistFUT:=", "", "DataType:=", "Real", "TypeInfo:=", Array( _
"", 0, "", "IsUnconstrained:=", false), "CircuitInstanceID:=", "")), Array("NAME:NoiseOut-
putQuantities"), "Name:=", _
"OscResonantFrequency1", "OscResonantFrequencyData:=", Array(1000, "1000000000", _
"5000000000", true, 1), "OscMaxkValues:=", Array("F1", 7),
"OscToneValues:=", Array("1Ghz")
```

Python Syntax	EditOscResonantFreqencySearch(<name>,<setupData[]>)
Python Example	oModule.EditOscResonantFrequencySearch("OscResonantFrequency", ["NAME:SimSetup", "DataBlockID:=" , 21,

```
"SimSetupID:="          , 26,
"OptionName:="          , "(Default Options)",
"AdditionalOptions:="   , "",
"AlterBlockName:="      , "",
"FilterText:="          , "",
"AnalysisEnabled:="     , 1,
[
        "NAME:OutputQuantities",
        [
                "NAME:Quantity",
                "NodeType:="            , "Harmonics",
                "CompID:="              , "",
                "CompName:="            , "",
                "QuantityName:="        , "2F1",
                "Selected:="            , True,
                "UnitType:="            , "NoUnit",
                "NetlistFUT:="          , "",
                "DataType:="            , "Real",
                "TypeInfo:="            , ["",0,"",
                "IsUnconstrained:="     , False],
                "CircuitInstanceID:="   , ""
```

```
        ],
        [
                "NAME:Quantity",
                "NodeType:="            , "Harmonics",
                "CompID:="              , "",
                "CompName:="            , "",
                "QuantityName:="        , "3F1",
                "Selected:="            , True,
                "UnitType:="            , "NoUnit",
                "NetlistFUT:="          , "",
                "DataType:="            , "Real",
                "TypeInfo:="            , ["",0,"",
                "IsUnconstrained:="     , False],
                "CircuitInstanceID:="   , ""
        ],
        [
                "NAME:Quantity",
                "NodeType:="            , "Harmonics",
                "CompID:="              , "",
                "CompName:="            , "",
```

```
                    "QuantityName:="        , "4F1",
                    "Selected:="            , True,
                    "UnitType:="            , "NoUnit",
                    "NetlistFUT:="          , "",
                    "DataType:="            , "Real",
                    "TypeInfo:="            , ["",0,"",
                    "IsUnconstrained:="     , False],
                    "CircuitInstanceID:="   , ""
                ],
                [
                    "NAME:Quantity",
                    "NodeType:="            , "Harmonics",
                    "CompID:="              , "",
                    "CompName:="            , "",
                    "QuantityName:="        , "DC",
                    "Selected:="            , True,
                    "UnitType:="            , "NoUnit",
                    "NetlistFUT:="          , "",
                    "DataType:="            , "Real",
                    "TypeInfo:="            , ["",0,"",
                    "IsUnconstrained:="     , False],
```

```
                "CircuitInstanceID:="   , ""
            ],
            [
                "NAME:Quantity",
                "NodeType:="            , "Harmonics",
                "CompID:="              , "",
                "CompName:="            , "",
                "QuantityName:="        , "F1",
                "Selected:="            , True,
                "UnitType:="            , "NoUnit",
                "NetlistFUT:="          , "",
                "DataType:="            , "Real",
                "TypeInfo:="            , ["",0,"",
                "IsUnconstrained:="     , False],
                "CircuitInstanceID:="   , ""
            ]
        ],
        [
            "NAME:NoiseOutputQuantities"
        ],
```

```
            "Name:="          , "OscResonantFrequency",
            "OscResonantFrequencyData:="      ,
                [1000,"1000000000","5000000000","1Ghz",True,1],
            "OscMaxkValues:=" , ["F1","7"
            "OscToneValues:=" , ["1Ghz"]
  ])
```

Edit PXF Setup[Nexxim]

Use: Edits a Periodic Transfer Function (PXF) setup

Syntax: EditPXFSetup(

```
STRING name,
ARRAY newsetup) // new setup data
```

VB Example:

```
oModule.EditPXFSetup "PXF1", Array("NAME:SimSetup", "DataBlockID:=", 21, "SimSetupID:=", 28,
"OptionName:=", _
"(Default Options)", "AdditionalOptions:=", "", "AlterBlockName:=", "", "FilterText:=", _
"", "AnalysisEnabled:=", 1, Array("NAME:OutputQuantities", Array("NAME:Quantity", "NodeType:=",
_
"Harmonics", "CompID:=", "", "CompName:=", "", "QuantityName:=", "2F1", "Selected:=", _
true, "UnitType:=", "NoUnit", "NetlistFUT:=", "", "DataType:=", "Real", "TypeInfo:=", Array( _
"", 0, "", "IsUnconstrained:=", false), "CircuitInstanceID:=", ""), Array("NAME:Quantity",
"NodeType:=", _
"Harmonics", "CompID:=", "", "CompName:=", "", "QuantityName:=", "3F1", "Selected:=", _
```

```
true, "UnitType:=", "NoUnit", "NetlistFUT:=", "", "DataType:=", "Real", "TypeInfo:=", Array( _
"", 0, "", "IsUnconstrained:=", false), "CircuitInstanceID:=", ""), Array("NAME:Quantity",
"NodeType:=", _
"Harmonics", "CompID:=", "", "CompName:=", "", "QuantityName:=", "4F1", "Selected:=", _
true, "UnitType:=", "NoUnit", "NetlistFUT:=", "", "DataType:=", "Real", "TypeInfo:=", Array( _
"", 0, "", "IsUnconstrained:=", false), "CircuitInstanceID:=", ""), Array("NAME:Quantity",
"NodeType:=", _
"Harmonics", "CompID:=", "", "CompName:=", "", "QuantityName:=", "DC", "Selected:=", _
true, "UnitType:=", "NoUnit", "NetlistFUT:=", "", "DataType:=", "Real", "TypeInfo:=", Array( _
"", 0, "", "IsUnconstrained:=", false), "CircuitInstanceID:=", ""), Array("NAME:Quantity",
"NodeType:=", _
"Harmonics", "CompID:=", "", "CompName:=", "", "QuantityName:=", "F1", "Selected:=", _
true, "UnitType:=", "NoUnit", "NetlistFUT:=", "", "DataType:=", "Real", "TypeInfo:=", Array( _
"", 0, "", "IsUnconstrained:=", false), "CircuitInstanceID:=", "")), Array("NAME:NoiseOut-
putQuantities"), "Name:=", _
"PXF1", "PXFData:=", Array(1, "HB", false, "HB", "No", 0), "PXFMaxkValues:=", Array( _
"F1", "7"), "PXFTonesValues:=", Array("1Ghz"), "PXFRelHarmNums:=", Array(), "PXFMaxSideband:=",
Array( _
"0"), "PXFSideband:=", Array("0"), "HarmonicsText:=", "2F1 3F1 4F1 DC F1", Array("NAME:Sweep-
Definition", "Variable:=", _
"F", "Data:=", "LINC 20GHz 200GHz 10", "OffsetF1:=", false, "Synchronize:=", 0), Array
("NAME:FSweep", "Variable:=", _
"F", "Data:=", "LINC 20GHz 200GHz 10", "OffsetF1:=", false, "Synchronize:=", 0))
```

Python Syntax	EditPXFSetup(<name>,<setupData[]>)
Python Example	oModule.EditPXFSetup("PXF", ["NAME:SimSetup", "DataBlockID:=" , 21, "SimSetupID:=" , 26, "OptionName:=" , "(Default Options)", "AdditionalOptions:=" , "", "AlterBlockName:=" , "", "FilterText:=" , "", "AnalysisEnabled:=" , 1, ["NAME:OutputQuantities", ["NAME:Quantity", "NodeType:=" , "Harmonics", "CompID:=" , "", "CompName:=" , "", "QuantityName:=" , "2F1",

```
                    "Selected:="            , True,
                    "UnitType:="            , "NoUnit",
                    "NetlistFUT:="          , "",
                    "DataType:="            , "Real",
                    "TypeInfo:="            , ["",0,"",
                    "IsUnconstrained:="     , False],
                    "CircuitInstanceID:="   , ""
                ],
                [
                    "NAME:Quantity",
                    "NodeType:="            , "Harmonics",
                    "CompID:="              , "",
                    "CompName:="            , "",
                    "QuantityName:="        , "3F1",
                    "Selected:="            , True,
                    "UnitType:="            , "NoUnit",
                    "NetlistFUT:="          , "",
                    "DataType:="            , "Real",
                    "TypeInfo:="            , ["",0,"",
                    "IsUnconstrained:="     , False],
```

```
                "CircuitInstanceID:="   , ""
        ],
        [
                "NAME:Quantity",
                "NodeType:="            , "Harmonics",
                "CompID:="              , "",
                "CompName:="            , "",
                "QuantityName:="        , "4F1",
                "Selected:="            , True,
                "UnitType:="            , "NoUnit",
                "NetlistFUT:="          , "",
                "DataType:="            , "Real",
                "TypeInfo:="            , ["",0,"",
                "IsUnconstrained:="     , False],
                "CircuitInstanceID:="   , ""
        ],
        [
                "NAME:Quantity",
                "NodeType:="            , "Harmonics",
                "CompID:="              , "",
                "CompName:="            , "",
```

```
                    "QuantityName:="        , "DC",
                    "Selected:="            , True,
                    "UnitType:="            , "NoUnit",
                    "NetlistFUT:="          , "",
                    "DataType:="            , "Real",
                    "TypeInfo:="            , ["",0,"",
                    "IsUnconstrained:="     , False],
                    "CircuitInstanceID:="   , ""
                ],
                [
                    "NAME:Quantity",
                    "NodeType:="            , "Harmonics",
                    "CompID:="              , "",
                    "CompName:="            , "",
                    "QuantityName:="        , "F1",
                    "Selected:="            , True,
                    "UnitType:="            , "NoUnit",
                    "NetlistFUT:="          , "",
                    "DataType:="            , "Real",
                    "TypeInfo:="            , ["",0,"",
```

```
                "IsUnconstrained:="     , False],
                "CircuitInstanceID:="   , ""
              ]
            ],
            [
              "NAME:NoiseOutputQuantities"
            ],
            "Name:="          , "PXF",
            "PXFData:="       , [1,"HB",False,"HB","No",0],
            "PXFMaxkValues:=" , ["F1","7"],
            "PXFTonesValues:="        , ["1Ghz"],
            "PXFRelHarmNums:="        , [],
            "PXFMaxSideband:="        , ["0"],
            "PXFSideband:="           , ["0"],
            "HarmonicsText:=" , "2F1 3F1 4F1 DC F1",
            [
               "NAME:SweepDefinition",
               "Variable:="           , "F",
               "Data:="               , "LINC 10GHz 100GHz 10",
               "OffsetF1:="           , False,
               "Synchronize:="                , 0
```

```
            ],
            [
               "NAME:SweepDefinition",
               "Variable:="          , "F",
               "Data:="              , "LINC 10GHz 100GHz 10",
               "OffsetF1:="          , False,
               "Synchronize:="               , 0
            ],
  ])
```

Edit QuickEye Analysis

Modifies an existing QuickEye analysis.

Command: Double-click on the analysis in the project tree.

Syntax: Edit QuickEye Analysis (<string>, // name of analysis to edit

Array("NAME:SimSetupName",

"DataBlockID:=", <int>,

"SimSetupID:=", <int>,

"OptionName:=", <string>,

"AdditionalOptions:=", <string>,

"AlterBlockName:=", <string>,

"FilterText:=", <string>,

"AnalysisEnabled:=", <int>, // 1 if enabled, 0 if disabled

Array("NAME:OutputQuantities", <QuantityArray>, <QuantityArray>...)

Array("NAME:NoiseOutputQuantities", <QuantityArray>, <QuantityArray>...)),

"Name:=", <string>, // name for new analysis

"QuickEyeAnalysis:=", Array(<string>, // Input rise time

<string>, // Input low voltage

<string>, // Input high voltage

<string>, // bits per second

<string>, // number of FFE taps

<string>, // Random Jitter Standard Deviation

<string>, // Delay

<string>, // Duty cycle distortion

<bool>, // true if unit interval, false if bits per second

<string>, // number of DFE taps

<bool>, // true to calculate FFE, false if weights are specified

<bool>, // true to calculate DFE, false if weights are specified

<string>, // DFE decision threshold

<string>, // DFE decision high

<string>, // DFE decision low

<bool>), // true if using specified equalization, false if disabled

Array("NAME:SweepDefinition",

"Variable:=", <string>,

"Data:=", <string>, // sweep

"OffsetF1:=", <bool>,

"Synchronize:=", <int>), // 1 to Synchronize, 0 otherwise

"FFEWts:=", Array(".5", "-2"), // optional, specified weights

"DFEWts:=", Array("2"))) // optional, specified weights

Return Value: <string> – // Name of the analysis after being modified

// If the name requested conflicts with the name of an existing

// analysis, the requested name is altered to be unique.

// The name returned reflects any change made to be unique.

Parameters: <QuantityArray>:

Array("NAME:Quantity",

"NodeType:=", <string>, // CompInst, Variable, Net, Harmonics, or Custom

"CompID:=", <string>,

"CompName:=", <string>,

"QuantityName:=", <string>,

"Selected:=", <bool>,

"UnitType:=", <string>,

"DataType:=", <string>, // Real, Complex, Integer, Enum, Char, Free, Array, Record

"CircuitInstanceID:=", <string)

VB Example:

```
dim name

name = oModule.EditQuickEyeAnalysis ("MyQuickEyeAnalysis", Array("NAME:SimSetup", "DataB-
lockID:=", 28, "SimSetupID:=", _

3, "OptionName:=", "(Default Options)", "AdditionalOptions:=", "", "AlterBlockName:=", _

"", "FilterText:=", "", "AnalysisEnabled:=", 1, Array("NAME:OutputQuantities", Array("NAME:Quant-
ity", "NodeType:=", "CompInst", "CompID:=", _

"5", "CompName:=", "Level01_NPN_Model_5", "QuantityName:=", "I", "Selected:=", _

true, "UnitType:=", "NoUnit", "DataType:=", "Real", "CircuitInstanceID:=", ""), Array
("NAME:Quantity", "NodeType:=", _

"CompInst", "CompID:=", "10", "CompName:=", "RES__10", "QuantityName:=", "I", "Selected:=", _

true, "UnitType:=", "NoUnit", "DataType:=", "Real", "CircuitInstanceID:=", ""), Array
("NAME:Quantity", "NodeType:=", _

"Net", "CompID:=", "", "CompName:=", "net_47", "QuantityName:=", "V", "Selected:=", _

true, "UnitType:=", "NoUnit", "DataType:=", "Real", "CircuitInstanceID:=", "")), Array
("NAME:NoiseOutputQuantities"), "Name:=", _

"MyQuickEyeAnalysis", "QuickEyeAnalysis:=", Array("5e-10", "0", "1", "1e-9", "2", _

"0", "1e-9", "0", true, "1", false, false, "0", "1", "-1", true), Array("NAME:SweepDefinition",
"Variable:=", _

"Temp", "Data:=", "LINC 1cel 10cel 10", "OffsetF1:=", false, "Synchronize:=", _

0), "FFEWts:=", Array(".5", "-2"), "DFEWts:=", Array("2")))
```

Python Syntax	EditQuickEyeAnalysis(<data_block>)

Python Example

```
newname = oModule.EditQuickEyeAnalysis (
[
   "NAME:SimSetup",
   "DataBlockID:=",
   28,
   "SimSetupID:=", 3,
   "OptionName:=", "(Default Options)",
   "AdditionalOptions:=","",
   "AlterBlockName:=", "",
   "FilterText:=", "",
   "AnalysisEnabled:=",1,
      [
         "NAME:OutputQuantities",
         [
            "NAME:Quantity",
            "NodeType:=","CompInst",
            "CompID:=", "5",
            "CompName:=", "Level01_NPN_Model_5",
            "QuantityName:=", "I",
            "Selected:=", True,
```

```
            "UnitType:=", "NoUnit",
            "DataType:=","Real",
            "CircuitInstanceID:=", ""
         ],
         [
            "NAME:Quantity",
            "NodeType:=", "CompInst",
            "CompID:=", "10",
            "CompName:=", "RES__10",
            "QuantityName:=", "I",
            "Selected:=", True,
            "UnitType:=", "NoUnit",
            "DataType:=", "Real",
            "CircuitInstanceID:=", ""
         ],
         [
            "NAME:Quantity",
            "NodeType:=", Net",
            "CompID:=", "",
            "CompName:=", "net_47",
            "QuantityName:=", "V",
```

```
            "Selected:=", True,
            "UnitType:=", "NoUnit",
            "DataType:=", "Real",
            "CircuitInstanceID:=", ""
         ]
      ],
      [
         "NAME:NoiseOutputQuantities"
      ],
     "Name:=", "MyQuickEyeAnalysis",
     "QuickEyeAnalysis:=",
      [
         "5e-10", "0", "1", "1e-9", "2", "0", "1e-9", "0",
         True, "1", False, False,"0", "1", "-1", True
      ],
      [
         "NAME:SweepDefinition",
         "Variable:=", "Temp",
         "Data:=", "LINC 1cel 10cel 10",
         "OffsetF1:=", False,
```

```
        "Synchronize:=", 0),
        "FFEWts:=",
        [
           ".5", "-2"
        ]
        DFEWts:=",
        [
           "2"
        ]
     ]
  ]
```

Edit System FD Analysis [Nexxim]

Use: Edits a System Frequency Domain (FD) Analysis setup

Syntax: EditSystemFDAnalysis(

```
STRING name,
ARRAY newsetup) // new setup data
```

VB Example:

```
oModule.EditSystemFDAnalysis "SystemFDAnalysis1", Array("NAME:SimSetup", "DataBlockID:=", 29,
"SimSetupID:=", _
3, "OptionName:=", "(Default Options)", "AdditionalOptions:=", "", "AlterBlockName:=", _
```

```
"", "FilterText:=", "", "AnalysisEnabled:=", 1, Array("NAME:OutputQuantities"), Array
("NAME:NoiseOutputQuantities"), "Name:=", _
"SystemFDAnalysis1", "SystemFDAnalysis:=", Array(false))
```

Python Syntax	EditSystemFDAnalysis(<name>,<setupData[]>)
Python Example	oModule.AddSystemFDAnalysis("SystemFDAnalysis", ["NAME:SimSetup", "DataBlockID:=" , 32, "SimSetupID:=" , 16, "OptionName:=" , "(Default Options)", "AdditionalOptions:=" , "", "AlterBlockName:=" , "", "FilterText:=" , "", "AnalysisEnabled:=" , 1, ["NAME:OutputQuantities"], ["NAME:NoiseOutputQuantities"

	], "Name:=" , "SystemFDAnalysis", "SystemFDAnalysis:=" , [False]])

Edit Transient [Nexxim]

Use: Edits a Transient analysis setup

Syntax: EditTransient(

ARRAY setup) // modified setup data

VB Example:

```
oModule.EditTransient "NexximTransient", Array("NAME:SimSetup", "DataBlockID:=", 10,
"SimSetupID:=", _
9, "OptionName:=", "(Default Options)", "AdditionalOptions:=", "", "AlterBlockName:=", _
"", "FilterText:=", "", "AnalysisEnabled:=", 1, Array("NAME:OutputQuantities"), Array
("NAME:NoiseOutputQuantities"), "Name:=", _
"NexximTransient", "TransientData:=", Array("0.1ns", "10ns"), "TransientNoiseData:=", Array( _
false, "", "", 0, 1, 0, false, 1), "TransientOtherData:=", Array("default"), Array("NAME:Sweep-
Definition", "Variable:=", _
"Temp", "Data:=", "LINC 10cel 200cel 10", "OffsetF1:=", false, "Synchronize:=", _
0))
```

Python Syntax	EditTransient(<modifiedsetupData[]>)
Python Example	oModule.EditTransient("NexximTransient1", ["NAME:SimSetup", "DataBlockID:=" , 10, "SimSetupID:=" , 11, "OptionName:=" , "(Default Options)", "AdditionalOptions:=" , "", "AlterBlockName:=" , "", "FilterText:=" , "", "AnalysisEnabled:=" , 1, ["NAME:OutputQuantities"], ["NAME:NoiseOutputQuantities"], "Name:=" , "NexximTransient1", "TransientData:=" , ["0.1ns","10ns"],

```
    "TransientNoiseData:="  , [False,"","",0,1,0,False,1],
    "TransientOtherData:="  , ["default"],
    [
            "NAME:SweepDefinition",
            "Variable:="            , "Temp",
            "Data:="                , "LINC 10cel 200cel 10",
            "OffsetF1:="            , False,
            "Synchronize:="         , 0
    ]
  ])
```

Edit TV Noise [Nexxim]

Use: Edits a TV Noise analysis setup

Syntax: EditTVNoise(

```
STRING setup

ARRAY setupdata) // modified setup data
```

VB Example:

```
oModule.EditTVNoise "TVNoise1", Array("NAME:SimSetup", "DataBlockID:=", 20, "SimSetupID:=", 23,
"OptionName:=", _

"(Default Options)", "AdditionalOptions:=", "", "AlterBlockName:=", "", "FilterText:=", _

"", "AnalysisEnabled:=", 1, Array("NAME:OutputQuantities", Array("NAME:Quantity", "NodeType:=",

_
"Harmonics", "CompID:=", "", "CompName:=", "", "QuantityName:=", "2F1", "Selected:=", _
```

```
true, "UnitType:=", "NoUnit", "NetlistFUT:=", "", "DataType:=", "Real", "TypeInfo:=", Array( _
"", 0, "", "IsUnconstrained:=", false), "CircuitInstanceID:=", ""), Array("NAME:Quantity",
"NodeType:=", _
"Harmonics", "CompID:=", "", "CompName:=", "", "QuantityName:=", "3F1", "Selected:=", _
true, "UnitType:=", "NoUnit", "NetlistFUT:=", "", "DataType:=", "Real", "TypeInfo:=", Array( _
"", 0, "", "IsUnconstrained:=", false), "CircuitInstanceID:=", ""), Array("NAME:Quantity",
"NodeType:=", _
"Harmonics", "CompID:=", "", "CompName:=", "", "QuantityName:=", "4F1", "Selected:=", _
true, "UnitType:=", "NoUnit", "NetlistFUT:=", "", "DataType:=", "Real", "TypeInfo:=", Array( _
"", 0, "", "IsUnconstrained:=", false), "CircuitInstanceID:=", ""), Array("NAME:Quantity",
"NodeType:=", _
"Harmonics", "CompID:=", "", "CompName:=", "", "QuantityName:=", "DC", "Selected:=", _
true, "UnitType:=", "NoUnit", "NetlistFUT:=", "", "DataType:=", "Real", "TypeInfo:=", Array( _
"", 0, "", "IsUnconstrained:=", false), "CircuitInstanceID:=", ""), Array("NAME:Quantity",
"NodeType:=", _
"Harmonics", "CompID:=", "", "CompName:=", "", "QuantityName:=", "F1", "Selected:=", _
true, "UnitType:=", "NoUnit", "NetlistFUT:=", "", "DataType:=", "Real", "TypeInfo:=", Array( _
"", 0, "", "IsUnconstrained:=", false), "CircuitInstanceID:=", "")), Array("NAME:NoiseOut-
putQuantities"), "Name:=", _
"TVNoise1", "TVNoiseData:=", Array(1, "", "", "", "", "HB", false, "HB", "No", _
0), "TVNoiseMaxkValues:=", Array("F1", "7"), "TVNoiseTonesValues:=", Array( _
```

```
"1Ghz"), "TVNoiseRelHarmNums:=", Array(), "TVNoiseRefSideband:=", Array("1"), "HarmonicsText:=",
_
"2F1 3F1 4F1 DC F1", Array("NAME:SweepDefinition", "Variable:=", "F", "Data:=", _
"LINC 10GHz 100GHz 10", "OffsetF1:=", false, "Synchronize:=", 0), Array("NAME:FSweep", "Vari-
able:=", _
"F", "Data:=", "LINC 10GHz 100GHz 10", "OffsetF1:=", false, "Synchronize:=", 0))
```

Python Syntax	EditTVNoise(<setupName>,<setupData[]>)
Python Example	oModule.EditTVNoise(["NAME:SimSetup", "DataBlockID:=" , 20, "SimSetupID:=" , 21, "OptionName:=" , "(Default Options)", "AdditionalOptions:=" , "", "AlterBlockName:=" , "", "FilterText:=" , "", "AnalysisEnabled:=" , 1, ["NAME:OutputQuantities", [

```
                    "NAME:Quantity",
                    "NodeType:="            , "Harmonics",
                    "CompID:="              , "",
                    "CompName:="            , "",
                    "QuantityName:="        , "2F1",
                    "Selected:="            , True,
                    "UnitType:="            , "NoUnit",
                    "NetlistFUT:="          , "",
                    "DataType:="            , "Real",
                    "TypeInfo:="            , ["",0,"",
                    "IsUnconstrained:="     , False],
                    "CircuitInstanceID:="   , ""
            ],
            [
                    "NAME:Quantity",
                    "NodeType:="            , "Harmonics",
                    "CompID:="              , "",
                    "CompName:="            , "",
                    "QuantityName:="        , "3F1",
                    "Selected:="            , True,
```

```
                "UnitType:="            , "NoUnit",
                "NetlistFUT:="          , "",
                "DataType:="            , "Real",
                "TypeInfo:="            , ["",0,"",
                "IsUnconstrained:="     , False],
                "CircuitInstanceID:="   , ""
        ],
        [
                "NAME:Quantity",
                "NodeType:="            , "Harmonics",
                "CompID:="              , "",
                "CompName:="            , "",
                "QuantityName:="        , "4F1",
                "Selected:="            , True,
                "UnitType:="            , "NoUnit",
                "NetlistFUT:="          , "",
                "DataType:="            , "Real",
                "TypeInfo:="            , ["",0,"",
                "IsUnconstrained:="     , False],
                "CircuitInstanceID:="   , ""
        ],
```

```
        [
                "NAME:Quantity",
                "NodeType:="            , "Harmonics",
                "CompID:="              , "",
                "CompName:="            , "",
                "QuantityName:="        , "DC",
                "Selected:="            , True,
                "UnitType:="            , "NoUnit",
                "NetlistFUT:="          , "",
                "DataType:="            , "Real",
                "TypeInfo:="            , ["",0,"",
                "IsUnconstrained:="     , False],
                "CircuitInstanceID:="   , ""
        ],
        [
                "NAME:Quantity",
                "NodeType:="            , "Harmonics",
                "CompID:="              , "",
                "CompName:="            , "",
                "QuantityName:="        , "F1",
```

```
                "Selected:="            , True,
                "UnitType:="            , "NoUnit",
                "NetlistFUT:="          , "",
                "DataType:="            , "Real",
                "TypeInfo:="            , ["",0,"",
                "IsUnconstrained:="     , False],
                "CircuitInstanceID:="   , ""
            ]
        ],
        [
            "NAME:NoiseOutputQuantities"
        ],
        "Name:="          , "TVNoise",
        "TVNoiseData:="           , [2,"","","","","HB",False,"HB","No",0],
        "TVNoiseMaxkValues:="     , ["F1","7","F2","7"],
        "TVNoiseTonesValues:="    , ["1Ghz","2Ghz"],
        "TVNoiseRelHarmNums:="    , [],
        "TVNoiseRefSideband:="    , ["1","1"],
        "HarmonicsText:=" , "2F1 3F1 4F1 DC F1",
        [
            "NAME:SweepDefinition",
```

```
            "Variable:="            , "F",
            "Data:="                , "LINC 10GHz 100GHz 10",
            "OffsetF1:="            , False,
            "Synchronize:="         , 0
          ],
          [
            "NAME:FSweep",
            "Variable:="            , "F",
            "Data:="                , "LINC 10GHz 100GHz 10",
            "OffsetF1:="            , False,
            "Synchronize:="         , 0
          ]
  ])
```

Edit VerifEye Analysis

Edits an existing VerifEye analysis.

Command: Double-click on the analysis in the project tree.

Syntax: Edit VerifEye Analysis (<string>, // name of analysis to edit

Array("NAME:SimSetupName",

"DataBlockID:=", <int>,

"SimSetupID:=", <int>,

"OptionName:=", <string>,

"AdditionalOptions:=", <string>,

"AlterBlockName:=", <string>,

"FilterText:=", <string>,

"AnalysisEnabled:=", <int>, // 1 if enabled, 0 if disabled

Array("NAME:OutputQuantities", <QuantityArray>, <QuantityArray>...)

Array("NAME:NoiseOutputQuantities", <QuantityArray>, <QuantityArray>...)),

"Name:=", <string>, // name for modified analysis

"VerifEyeAnalysis:=", Array(<string>, // Input rise time

<string>, // Input low voltage

<string>, // Input high voltage

<string>, // bits per second

<string>, // number of FFE taps

<string>, // Random Jitter Standard Deviation

<string>, // Delay

<string>, // Duty cycle distortion

<bool>, // true if unit interval, false if bits per second

<string>, // number of DFE taps

<bool>, // true to calculate FFE, false if weights are specified

<bool>, // true to calculate DFE, false if weights are specified

<string>, // DFE decision threshold

<string>, // DFE decision high

<string>, // DFE decision low

<bool>), // true if using specified equalization, false if disabled

Array("NAME:SweepDefinition",

"Variable:=", <string>,

"Data:=", <string>, // sweep

"OffsetF1:=", <bool>,

"Synchronize:=", <int>), // 1 to Synchronize, 0 otherwise

"FFEWts:=", Array(".5", "-2"), // optional, specified weights

"DFEWts:=", Array("2"))) // optional, specified weights

Return Value: <string> – // Name of the analysis after being modified

// If the name requested conflicts with the name of an existing

// analysis, the requested name is altered to be unique.

// The name returned reflects any change made to be unique.

Parameters: <QuantityArray>:

Array("NAME:Quantity",

"NodeType:=", <string>, // CompInst, Variable, Net, Harmonics, or Custom

"CompID:=", <string>,

"CompName:=", <string>,

"QuantityName:=", <string>,

"Selected:=", <bool>,

"UnitType:=", <string>,

"DataType:=", <string>, // Real, Complex, Integer, Enum, Char, Free, Array, Record

"CircuitInstanceID:=", <string)

VB Example:

```
dim name

name = oModule.EditVerifEyeAnalysis ("MyVerifEyeAnalysis", Array("NAME:SimSetup", "DataB-
lockID:=", 27, "SimSetupID:=", _

1, "OptionName:=", "Options", "AdditionalOptions:=", "", "AlterBlockName:=", "", "FilterText:=",

_

"", "AnalysisEnabled:=", 1, Array("NAME:OutputQuantities", Array("NAME:Quantity", "NodeType:=",
"CompInst", "CompID:=", _

"5", "CompName:=", "Level01_NPN_Model_5", "QuantityName:=", "I", "Selected:=", _

true, "UnitType:=", "NoUnit", "DataType:=", "Real", "CircuitInstanceID:=", ""), Array
("NAME:Quantity", "NodeType:=", _

"CompInst", "CompID:=", "10", "CompName:=", "RES__10", "QuantityName:=", "I", "Selected:=", _

true, "UnitType:=", "NoUnit", "DataType:=", "Real", "CircuitInstanceID:=", ""), Array
("NAME:Quantity", "NodeType:=", _

"Net", "CompID:=", "", "CompName:=", "net_47", "QuantityName:=", "V", "Selected:=", _

true, "UnitType:=", "NoUnit", "DataType:=", "Real", "CircuitInstanceID:=", "")), Array
("NAME:NoiseOutputQuantities"), "Name:=", _

"MyVerifEyeAnalysis", "VerifEyeAnalysis:=", Array("5e-10", "0", "1", "1e-9", "2", _

"0", "1e-9", "0", true, "1", false, false, "0", "1", "-1", true), Array("NAME:SweepDefinition",
"Variable:=", _
```

```
"Temp", "Data:=", "LINC 1cel 10cel 10", "OffsetF1:=", false, "Synchronize:=", _
0), "FFEWts:=", Array(".5", "-2"), "DFEWts:=", Array("2")))
```

Python Syntax	EditVerifEyeAnalysis(<data_block>)
Python Example	newname = oModule.EditVerifEyeAnalysis (["NAME:SimSetup", "DataBlockID:=", 28, "SimSetupID:=", 3, "OptionName:=", "(Default Options)", "AdditionalOptions:=","", "AlterBlockName:=", "", "FilterText:=", "", "AnalysisEnabled:=",1, ["NAME:OutputQuantities", ["NAME:Quantity",

```
            "NodeType:=","CompInst",
            "CompID:=", "5",
            "CompName:=", "Level01_NPN_Model_5",
            "QuantityName:=", "I",
            "Selected:=", True,
            "UnitType:=", "NoUnit",
            "DataType:=","Real",
            "CircuitInstanceID:=", ""
        ],
        [
            "NAME:Quantity",
            "NodeType:=", "CompInst",
            "CompID:=", "10",
            "CompName:=", "RES__10",
            "QuantityName:=", "I",
            "Selected:=", True,
            "UnitType:=", "NoUnit",
            "DataType:=", "Real",
            "CircuitInstanceID:=", ""
        ],
        [
```

```
            "NAME:Quantity",
            "NodeType:=", Net",
            "CompID:=", "",
            "CompName:=", "net_47",
            "QuantityName:=", "V",
            "Selected:=", True,
            "UnitType:=", "NoUnit",
            "DataType:=", "Real",
            "CircuitInstanceID:=", ""
         ]
      ],
      [
         "NAME:NoiseOutputQuantities"
      ],
   "Name:=", "MyVerifEyeAnalysis",
   "VerifEyeAnalysis:=",
      [
         "5e-10", "0", "1", "1e-9", "2", "0", "1e-9", "0",
         True, "1", False, False,"0", "1", "-1", True
      ],
```

```
      [
         "NAME:SweepDefinition",
         "Variable:=", "Temp",
         "Data:=", "LINC 1cel 10cel 10",
         "OffsetF1:=", False,
         "Synchronize:=", 0),
         "FFEWts:=",
         [
            ".5", "-2"
         ]
         DFEWts:=",
         [
            "2"
         ]
      ]
   ]
```

EditSweep [Nexxim]

Use: Edit a sweep

Syntax: EditSweep(

```
STRING setup) // setup name
ARRAY solvesetup_data) // setup data
```

VB Example: oModule. EditSweep <setup> <solvesetup_data>

GetAllSolutionSetups [Nexxim]

Use: Returns an array of the names of the simulation setups in the design

Syntax: GetAllSolutionSetups( )

VB Example:

```
Set oModule = oDesign.GetModule("SimSetup")
Dim names
names = oModule.GetAllSolutionSetups()
MsgBox("GetAllSolutionSetups:" & Chr(13) & Join(names,", "))
```

Python Syntax	GetAllSolutionSetups( )
Python Example	`names = oModule.GetAllSolutionSetups()`

ListVariations [Nexxim]

Use: Get a list of solution variations for a given solution

Syntax: ListVariations(

```
STRING SolutionName) // solution name
```

VB Example:

```
oModule.ListVariations("NWA1")
```

Python Syntax	ListVariations(SolutionName)
Python Example	`variationArr = oModule.ListVariations("NWA1")`

RefreshMeshOverlays

Use: Refreshes mesh display for a setup

Syntax: RefreshMeshOverlays(

```
STRING setup) // setup name
```

VB Example: `oModule.RefreshMeshOverlays <setup>`

RenameSweep [Nexxim]

Use: Renames a NexximSimulation setup

Syntax: RenameSweep(

```
STRING oldname) // old setup name
STRING newname) // new setup name
```

VB Example: `oModule. RenameSweep <oldname> <newname>`

Nexxim Linear Network Analysis

The following linear network analysis commands are available.

AddAnalysisOptions

AddSimulationSetup

EditAnalysisOptions

EditSimulationSetup

ExportForSpice

ExportForHSpice

RemoveAnalysisOptions

RemoveSimulationSetup

RenameAnalysisOptions

RenameSimulationSetup

Some of the following commands are available only in the "SolveSetups" module:

For example,

```
oDesign.GetModule("SolveSetups")
oModule.CommandName <args>
```

Add Analysis Options [Nexxim]

Use: Add an analysis option

Command: None

Syntax: AddAnalysisOptions

```
ARRAY <Option_Data>
```

Return Value: None

Parameters: `<Option_Data>`

Type:

array

VB Example:

```
oModule.AddAnalysisOptions Array("NAME:DataBlock", "DataBlockID:=", 8, "Name:=", _
"Nexxim Options1", Array("NAME:ModifiedOptions", "gen_timedomsmodel:=", true, "eye_sym_step_
resp:=", _
false, "tmi_flag:=", "2.01"))
```

Python Syntax	AddAnalysisOptions (option_data[])
Python Example	`oModule.AddAnalysisOptions(` `[` `  "NAME:DataBlock",` `  "DataBlockID:="  , 8,` `  "Name:="  , "Nexxim Options",` `  [` `  "NAME:ModifiedOptions",` `  "eye_sym_step_resp:="  , False,` `  "tmi_flag:="  , "2.01"` `  ]` `])`

AddSimulationSetup [Nexxim]

Use: Adds a new simulation setup

Command: None

Syntax: AddSimulationSetup <Simulationsetup_data>

Return Value: None

Parameters: <Simulationsetup_Data>

VB Example: oModule.AddSimulationSetup <SimulationSetup_Data>

Edit Analysis Options [Nexxim]

Use: Edit an analysis option

Command: None

Syntax: EditAnalysisOptions

```
STRING name,
ARRAY <Option_Data>
```

Return Value: None

VB Example:

```
oModule.EditAnalysisOptions "Nexxim Options", Array("NAME:DataBlock", "DataBlockID:=", _
8, "Name:=", "Nexxim Options", Array("NAME:ModifiedOptions", "eye_sym_step_resp:=", _
false, "tmi_flag:=", "2.01", "gen_timedomsmodel:=", true, "eye_num_initial_ui:=", _
200))
```

Python Syntax	EditAnalysisOptions(<name>,<Option_Data>[])
Python Example	oModule.EditAnalysisOptions("Nexxim Options", ["NAME:DataBlock", "DataBlockID:=" , 8, "Name:=" , "Nexxim Options", ["NAME:ModifiedOptions", "eye_sym_step_resp:=" , False, "tmi_flag:=" , "2.01", "gen_timedomsmodel:=" , True]])

EditSimulationSetup [Nexxim]

Use: Edit a simulation setup

Command: None

Syntax: EditSimulationSetup <Simulationsetup_data>

Return Value: None

Parameters: <Simulationsetup_Data>

VB Example: oModule.EditSimulationSetup <SimulationSetup_Data>

ExportForSpice [Nexxim]

Export matrix solution data to a file in the given spice format.

Command: None

Syntax: ExportForSpice <theVariationKey> <sourceNames> <theType> <bandwidth> <fullwaveSpiceFilename> <lumpedElementFilename> <Unused> <Unused> <partialFractionFilename> <fittingError> <maximumOrder> <useCommonGround> <doPassivityCheck>

Return Value: None

Parameters: <theVariationKey>

Type: string

<sourceNames>

Type: array

<theType: 0=PSpice, 2=MaxwellSpice, 3=Spectre>

Type: long

<bandwidth: 0=Low Bandwidth, 1=Broad Bandwidth>

Type: long

<fullwaveSpiceFilename>

Type: string

<lumpedElementFilename>

Type: string

<Unused>

Type: string

<Unused>

Type: string

<partialFractionFilename>

Type: string

<fittingError [Default:0.5]>

Type: double

<maximumOrder [Default:200]>

Type: int

<useCommonGround [Default:FALSE]>

Type: boolean

<doPassivityCheck [Default:FALSE]>

Type: boolean

VB Example:

```
oDesign.ExportForSpice "", Array("LNA:LNA"), 0, 0, _
"C:/Documents and Settings/Marcia.BMT/My Documents/Code Modifications/Test" & _
" Designs/s-params_fws_fws.lib", _
"C:/Documents and Settings/Marcia.BMT/My Documents/Code Modifications/Test" & _
" Designs/s-params_fws_lfws.lib", "", "", "", 0.52, 210, 1, 1
```

Python Syntax	ExportForSpice(<theVariationKey> <sourceNames> <theType> <bandwidth> <fullwaveSpiceFilename> <lumpedElementFilename> <Unused> <Unused> <partialFractionFilename> <fittingError> <maximumOrder> <useCommonGround> <doPassivityCheck>)

Python Example	oDesign.ExportForSpice("", ["LNA:LNA"] , 0, 0, "C:/Documents and Settings/Marcia.BMT/MyDocuments/Code Modifications/Test" & " Designs/s-params_fws_fws.lib", "C:/Documents and Settings/Marcia.BMT/MyDocuments/Code Modifications/Test" & " Designs/s-params_fws_lfws.lib", "", "", "", 0.52, 210, 1, 1)

ExportForHSpice [Nexxim]

Export matrix solution data to an HSpice subcircuit.

Command: None

Syntax: ExportForHSpice <theVariationKey> <sourceNames> <theType> <bandwidth> <fullwaveSpiceFilename> <lumpedElementFilename> <Unused> <Unused> <partialFractionFilename> <fitting Error> <maximumOrder> <useCommonGround> <doPassivityCheck>

Return Value: None

Parameters: <theVariationKey>

Type: string

<sourceNames>

Type: array

<theType: 1=HSPICE>

Type:long

<bandwidth: 0=Low Bandwidth, 1=Broad Bandwidth>

Type: long

<fullwaveSpiceFilename>

Type: string

<lumpedElementFilename>

Type: string

<Unused>

Type: string

<Unused>

Type: string

<partialFractionFilename>

Type: string

<fittingError [Default:0.5]>

Type: double

<maximumOrder [Default: 200]>

Type: int

<useCommonGround [Default: FALSE]>

Type: boolean

<doPassivityCheck [Default: FALSE]>

Type: boolean

VB Example:

```
oDesign.ExportForHSpice "", Array("Setup 1:Sweep 1"), 1, 0, "C:/Projects/MyTRL_fws.sp", "", "",
"", "", 0.5, 0, 200, 0, 0
```

Python Syntax	ExportForHSpice(<theVariationKey> <sourceNames> <theType> <bandwidth> <fullwaveSpiceFilename> <lumpedElementFilename> <Unused> <Unused> <partialFractionFilename> <fitting Error> <maximumOrder> <useCommonGround> <doPassivityCheck>)
Python Example	`oDesign.ExportForHSpice( "", ["Setup1:Sweep 1"], 1, 0, "C:/Projects/MyTRL_fws.sp", "", "", "", "", 0.5, 0, 200, 0, 0)`

RemoveAnalysisOptions [Nexxim]

Use: Remove an analysis option

Command: None

Syntax: RemoveAnalysisOptions <Option_Data>

Return Value: None

Parameters: `<Option_Data>`

VB Example: `oModule.RemoveAnalysisOptions <Option_Data>`

RemoveSimulationSetup [Nexxim]

Use: Remove a simulation setup

Command: None

Syntax: RemoveSimSetup <Simulationsetup_data>

Return Value: None

Parameters: `<Simulationsetup_Data>`

VB Example: `oModule.RemoveSimSetup <SimulationSetup_Data>`

Python Syntax	RemoveSimSetup(<setupName1>,<setupName2>...)
Python Example	`oDesign.RemoveSimSetup("QuickEyeAnalysis", "VerifyEyeAnalysis")`

RenameAnalysisOptions[Nexxim]

Use: Rename an analysis option

Command: None

Syntax: RenameAnalysisOptions <oldname> <newname>

Return Value: None

Parameters: <oldname>

Type: string

<newname>

Type: string

VB Example: oModule.RenameAnalysisOptions <oldname> <newname>

RenameSimulationSetup [Nexxim]

Use: Rename a simulation setup

Command: None

Syntax: RenameSimulationSetup <oldname> <newname>

Return Value: None

Parameters: <oldname>

Type: string

<newname>

Type: string

VB Example: oModule.RenameSimulationSetup "oldname", "newname"

Interface Ports And Sources Commands

The following Ports and Sources commands for Nexxim are available.

ChangePortProperty

ChangeSourceProperty

DeletePort

DeleteSource

GetAllPorts

GetAllSources

RenameSource

ChangePortProperty [Nexxim]

Use: Change a port property

Command: None

Syntax: ChangePortProperty <Port Name> <Port Info> <Port Properties>

Return Value: None

Parameters: `<Port Name> - Type: string`

<Port Info> - Type: array

<Port Properties> - Type:array

VB Example:

```
oDesign.ChangePortProperty "Port2", Array("NAME:Port2", "IIPortName:=", "Port2", "SymbolType:=",
_
0), Array(Array("NAME:Properties", Array("NAME:NewProps", Array("NAME:term", "PropType:=", _
"TextProp", "OverridingDef:=", true, "Value:=", "Model1")), Array("NAME:ChangedProps", Array
("NAME:TerminationData", "Value:=", _
"Zo"), Array("NAME:pnum", "Value:=", "2"), Array("NAME:noisetemp", "Value:=", "16.85")))
```

Python Syntax	ChangePortProperty(<portName> [<portInfo>] [<portProperties>])

Python Example	oDesign.ChangePortProperty("Port2", ["NAME:Port2","IIPortName:=", "Port2", "SymbolType:=",0, [["NAME:Properties", ["NAME:NewProps",["NAME:term", "PropType:=","TextProp", "OverridingDef:=", true, "Value:=","Model1"]], ["NAME:ChangedProps", ["NAME:TerminationData","Value:=", "Zo"], ["NAME:pnum", "Value:=", "2"], ["NAME:noisetemp", "Value:=", "16.85"]]])

ChangeSourceProperty [Nexxim]

Use: Change a source property

Command: None

Syntax: ChangeSourceProperty <Source_Name> <Source_Data>

Return Value: None

Parameters: <Source_Name>

Type: string

<Source_Data>

Type: array

VB Example:

```
oDesign.ChangeSourceProperty "Sinusoidal1", Array(Array("NAME:Properties",

Array("NAME:NewProps", _

Array("NAME:noise", "PropType:=", "TextProp", "OverridingDef:=", true, "UserDef:=", true,
"Value:=", "SourceNoise1"), _

Array("NAME:mod", "PropType:=", "TextProp", "OverridingDef:=", true, "UserDef:=", true,
"Value:=", "SourceModulation1"))), _
```

```
Array("NAME:DataBlocks", Array("NAME:NewDataBlk", Array("NAME:DataBlock", "Name:=",
"SourceNoise1", _
"noisedata:=", Array("FDEV=1Hz ", "RFUP=1 ", "RFLO=2 ", "NCOR=2 2 ")), _
Array("NAME:DataBlock", "Name:=", "SourceModulation1", "ModulationSourceDatas:=", Array( _
"IS95", "Butterworth", "Br=1.2288MHz", "Dly=0.5", "Iasc=1", "Qasc=1")))))
```

Python Syntax	ChangeSourceProperty(<sourceName>,[<Properties])
Python Example	`oDesign.ChangeSourceProperty( "Sinusoidal1",` `  [` `  [` `  "NAME:Properties",` `  [` `  "NAME:NewProps",` `  [` `  "NAME:noise",` `  "PropType:=", "TextProp",` `  "OverridingDef:=", true,` `  "UserDef:=", true,` `  "Value:=", "SourceNoise1"` `  ],`

```
            [
              "NAME:mod",
              "PropType:=", "TextProp",
              "OverridingDef:=", true,
              "UserDef:=", true,
              "Value:=", "SourceModulation1"
            ]
         ]
      ],
      [
         "NAME:DataBlocks",
         [
            "NAME:NewDataBlk",
            [
               "NAME:DataBlock",
               "Name:=", "SourceNoise1",
               "noisedata:=",
               [
                   "FDEV=1Hz ",
                   "RFUP=1",
```

```
                    "RFLO=2 ",
                    "NCOR=2 2 "
                ]
            ],
            [
              "NAME:DataBlock",
              "Name:=", "SourceModulation1"
              "ModulationSourceDatas:=",
               [
                    "IS95",
                    "Butterworth",
                    "Br=1.2288MHz",
                    "Dly=0.5",
                    "Iasc=1",
                    "Qasc=1"
               ]
           ]
        ]
     ]
])
```

DeletePort [Nexxim]

Use: Delete a port

Command: None

Syntax: DeletePort <Port_Name>

Return Value: None

Parameters: <Port_Name>

Type: string

VB Example: oDesign.DeletePort "Port1"

Python Syntax	DeletePort(<portName>)
Python Example	`oDesign.DeletePort("Port1")`

DeleteSource [Nexxim]

Use: Delete the source

Command: None

Syntax: DeleteSource <srcname>

Return Value: None

Parameters: <srcname>

Type: string

VB Example: `oDesign.DeleteSource "srcname"`

Python Syntax	DeleteSource(<sourceName>)
Python Example	`oDesign.DeleteSource("VoltageSource1")`

GetAllPorts [Nexxim]

Use: Returns a list of all port names in the Nexxim Design

Syntax: GetAllPorts

Return Value: Array of Strings

Parameters: None

VB Example:

```
Dim portsArr
    portsArr= oDesign.GetAllPorts
```

Python Syntax	GetAllPorts()
Python Example	`portsArr = oDesign.GetAllPorts()`

GetAllSources [Nexxim]

Use: Returns a list of all sources names in the Nexxim Design

Syntax: GetAllSources

Return Value: Array of Strings

Parameters: None

VB Example: Dim sourceArr

```
sourceArr = oDesign.GetAllSources
```

Python Syntax	GetAllSources()
Python Example	`sourceArr = oDesign.GetAllSources()`

RenameSource [Nexxim]

Use: Rename the source

Command: None

Syntax: RenameSource <oldname>, <newname>

Return Value: None

Parameters: <oldname>

Type: string

<newname>

Type: string

VB Example: oDesign.RenameSource "oldname", "newname"

Python Syntax	RenameSource(<oldName>,<newName>)
Python Example	

	`oDesign.RenameSource("VoltageSource1", "MySource1")`

Nexxim Component Manager Commands

The following Component Manager commands for Nexxim are available.

ImportSandWComponent

ImportXparamComponent

ImportSandWComponent

Use: Import S-element or W-element

Command: None

Syntax: ImportSandWComponent <Files><Options>

Return Value: Name of the component that is created

Parameters: `<Files>, <Options>`

Type: array

```
oComponentManager.ImportSandWComponent
// Array of files selected
Array("NAME:Files", "Files:=",Array(_
"C:\Projects\100ohm.s2p",_
"C:\Projects\50ohm.s2p",_
"C:\Projects\25ohm.s2p",_
// S-element options
Array("NAME:Options",
"IsWElement:=", false, // true=W-element, false=S-element
"NumPortsorLines:’", 2 // ports for S, lines for W
"CustomNetlist:=", "INTDATTYP=MA HIGHPASS=10 LOWPASS=10_
convolution=1 enforce_passivity=1",
// Netlist representing selected options for S-element
"CreateArray:="’ true // true=file array false=no array
"PinConfig:="’, 0, // 0=Odd/Even, 1=I/I+N
```

```
"AddCommonRef:="', 1 // 0=implied to gnd, 1=show ref pin
)
```

ImportXparamComponent

Use: Import X-Parameter component

Command: None

Syntax: ImportXparamComponent <File>

Return Value: Name of the component that is created

Parameters: <File>

Type: string

VB Example: oComponentManager.ImportXparamComponent <filename>

Index

C

E

F

G

H

I

S

V

W

Z