SIwave 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 2023 R1
January 2023

Table of Contents

1 - Introduction to Scripting

SIwave has the ability to run saved scripts in VBScript or IronPython. It also contains a native IronPython command shell.

The following topics contain more information about scripting in SIwave:

Running a Script

To run a script file in SIwave:

1. Click **Tools**. In the **Scripting** area, click **Run Script**.

   The **Open** window appears.
2. Use the file browser to locate and select a *.vbs or *.py script file.
3. Click **Open**.

   The script is executed.

To run a script file from a command line:

1. Type the following:

   ```
   <path to siwave.exe> <path to *.siw file> -RunScript <path to *.py file>
   ```

   SIwave opens and the script is executed.

   Alternatively, type the following to run the script and close SIwave:

   ```
   <path to siwave.exe> <path to *.siw file> -RunScriptAndExit <path to *.py file>
   ```

To run a script from the IronPython command shell:

1. Click **Tools**. In the **Scripting** area, click **IronPython Comand Shell**.

   The command window appears.
2. Type the desired script commands and press **Enter**.

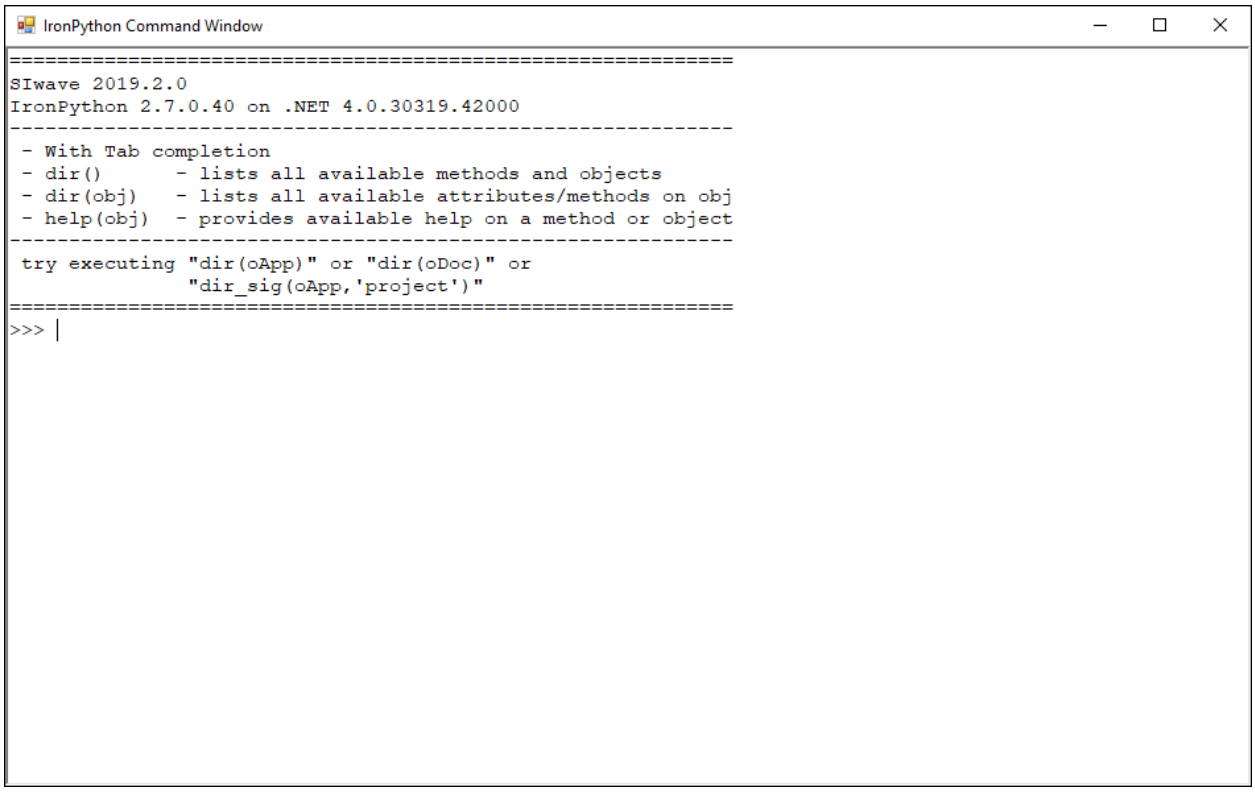
Using the IronPython Command Shell

This section describes IronPython and using the IronPython Command Shell in SIwave. It is written for those already familiar with scripting in VBScript or JavaScript.

To launch the IronPython Command Shell:

1. Click **Tools**. In the **Scripting** area, click **IronPython Command Shell**.

   The **IronPython Command Window** appears.



Introduction to IronPython

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

Scope

This section of the help provides a basic introduction to the use of IronPython in SIwave. If you require a full tutorial on Python or IronPython, there are many online resources.

Python Compatibility

The version of IronPython in use is **2.7** and built on .NET framework version 4.0. 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) cannot be expected to work in 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 provides are significant:

- Python has a large ecosystem with plenty of supporting libraries, Visual IDEs and debuggers. It is actively developed and enhanced.
- IronPython has access to the entire .NET ecosystem. 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.
- The Python syntax of dictionaries is somewhat easier to read and write when supplying arguments to the scripting methods.

Scripting Using IronPython

Interacting with script objects in IronPython is 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.

If you have existing VBScript/Javascript scripts, use them by either embedding into the IronPython script or invoking them via run methods.

If you must write an IronPython script from scratch, consult this help, along with internet resources. See IronPython Samples for a collection of pure IronPython snippets. These should serve as a guide and reference.

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`

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.

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.

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 sample scripts.

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
- A function (described next)

Named Functions

Named functions are arrays which start with `Array(` without 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, in the form `keyName:=` followed by a primitive value
- Another function (nested function)

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"),
"offset:=",_

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()
```

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. The main thing to notice here is that Boolean values in IronPython have their first letter capitalized.

(`True` instead of `true` and `False` instead of `false`)

The recommended approach is to simply replace a VBScript array with a Python array. The mapping is quite simple:

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

Named Array Arguments

Formatting (which helps readability immensely) is not needed. 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]
]
```

Figure 1-1 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]
}
```

Figure 1-2 Sample Script: Named array with nested named array as mixed dictionary + array

```
{ "NAME" : "name",
"key1" : 1,
"key2" : 2,
"name2" : {"R" : 255}
}
```

Figure 1-3 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 [ ]

2 - Script Commands

The following scripts are available in SIwave:

- ScrDeleteResonantModeSolution
- ScrDeleteSpiceSubcktSolution
- ScrDeleteSyzParameterSolution
- ScrDrawCapacitor
- ScrDrawCircle
- ScrDrawInductor
- ScrDrawPolygon
- ScrDrawPort
- ScrDrawRectangle
- ScrDrawResistor
- ScrDrawTrace
- ScrDrawVia
- ScrDrawVoltageProbe
- ScrDrawVoltageSource
- ScrEditCktElemName
- ScrEditLayerName
- ScrEditMaterial
- ScrEditNetName
- ScrEditPadStackName
- ScrEnableCavityFieldCoupling
- ScrEnableCoPlaneCoupling
- ScrEnableErcSimSetup
- ScrEnableFwsRelativeErrorTol
- ScrEnableIntraPlaneCoupling
- ScrEnableSplitPlaneCoupling
- ScrEnableTraceCoupling
- ScrExport3DModel
- ScrExportAnf
- ScrExportComponentFile
- ScrExportCpaSimReport
- ScrExportDcPowerDataToIcepak
- ScrExportDcPowerTree
- ScrExportDcSimReport
- ScrExportDcSimReportColorBarProperties
- ScrExportDcSimReportOptions
- ScrExportDcSimReportScaling
- ScrExportDcSimReportUnits

- ScrMergeConnectedNets
- ScrNetGetLength
- ScrNetIsDisjoint
- ScrNetIsSelected
- ScrNetSeparate
- ScrNetSetDummy
- ScrNetSetSelected
- ScrPlaceCircuitElement
- ScrPlaceCircuitElementsToNearestRefPin
- ScrPlaceFreqDependentSrc
- ScrPlacePortsAcrossRLCs
- ScrPlacePortsAtPinsOnSelectedNets
- ScrPlacePortsAtPinsOnSelectedNetsExcludePart
- ScrPlacePortsAtPinsOnSelectedNetsPinNamesOut
- ScrPlotResModeVoltageDiff
- ScrPreserveNetsGivenInFile
- ScrReadDCLoopResInfo
- ScrRestoreResonantModeMinFreq
- ScrRunDcSimulation
- ScrRunFarFieldSimulation
- ScrRunFrequencySweepSimulation
- ScrRunIcepakSimulation
- ScrRunInducedVoltageSimulation
- ScrRunNearFieldSimulation
- ScrRunResonantModeSimulation
- ScrRunSimulation
- ScrRunSpiceSubcktSimulation
- ScrRunSyzParameterSimulation
- ScrRunValidationCheck
- ScrRunValidationCheckWithOptions
- ScrSanitizeLayout
- ScrSanitizeNets
- ScrSaveProjectAs
- ScrSaveSimulationMessages
- ScrSaveToPngFile
- ScrSelectDcConnectedNets
- ScrSelectNet

- ScrSetPadstackMaterial
- ScrSetPadstackViaPlatingAbsolute
- ScrSetPadstackViaPlatingRatio
- ScrSetPlotAfterDcSimulation
- ScrSetPlotLayers
- ScrSetPlotSyzMag
- ScrSetPlotSyzPhase
- ScrSetPortNamingConvention
- ScrSetPowerGroundNets
- ScrSetPowerGroundNetsFromFile
- ScrSetProjectModified
- ScrSetPsiOptionsFromFile
- ScrSetPsiPortType
- ScrSetPsiSyzInterpOptions
- ScrSetRefineBondwiresInDcSimulation
- ScrSetRefineDcSimulation
- ScrSetRefineViasInDcSimulation
- ScrSetRemoveCutoutsByArea
- ScrSetResonantModeMaxFreq
- ScrSetResonantModeMinFreq
- ScrSetRLCValues
- ScrSetSignalNets
- ScrSetSignalNetsFromFile
- ScrSetSimulationName
- ScrSetSketchedBwProfile
- ScrSetSketchedBwProfileFromArray
- ScrSetSnapLengthThreshold
- ScrSetSolderballMaterial
- ScrSetSolderballParameters
- ScrSetSourceMagnitude
- ScrSetSparamModelSetup
- ScrSetSpiceModelSetup
- ScrSetSpiceSubcktFormat
- ScrSetStackupLayerThickness
- ScrSetStackupLayerThicknessUnit
- ScrSetSweepFreqRange
- ScrSetSweepMaxFreq

- ScrSetSweepMinFreq
- ScrSetSweepNumFreqPoints
- ScrSetSyzInterpSweep
- ScrSetSyzInterpSweepParams
- ScrSetTDCrosstalkScanParameters
- ScrSetThermalPadOnLayer
- ScrSetTouchstoneExportFormatToDb
- ScrSetTouchstonePortOrder
- ScrSetTouchstonePortRemapping
- ScrSetTraceCouplingDistance
- ScrSetUniformTemperature
- ScrSetZ0ScanParameters
- ScrSetZ0ScanReportImageHeight
- ScrShowSelectedNetsOnly
- ScrSIwaveEnable_3D_DDM
- ScrSIwaveEnableHFSSRegions
- ScrSIwaveEnableReturnCurrentDistribution
- ScrSIwaveIncludeSourceParasitics
- ScrSIwaveSyzComputeExactDcPoint
- ScrSIwaveSyzEnforceCausality
- ScrSIwaveSyzEnforcePassivity
- ScrUnselectAll
- ScrUpdateComponentTree
- ScrUseIcepakTemperatureDataInDc
- ScrUseTouchstonePortRemapping
- Solve
- StopSimLink
- SupportSParamLink

Note:

- The above list contains all current SIwave scripting functions. Running `dir(oApp)` or `dir(oDoc)` from the Command Window may show additional scripts. These are scripts that are either obsolete or that are part of the Electronics Desktop scripting environment and do not serve any function in SIwave.
- TPA is no longer supported. Ansys recommends using CPA.

CloseProject

Closes the specified project and opens a new "Untitled" project.	
UI Command:	Click **File** > **New**.
Syntax:	`obj.CloseProject(<projectName>)`
Parameters:	BSTR projectName
Return Value:	None.
VB Example:	`obj.CloseProject "project.siw"`
IPY Example:	`oApp.CloseProject('project.siw')`

Note:

The behavior of CloseProject() and CloseProjectNoForce() is identical when the SIwave UI is invoked in graphical/interactive mode.

However, when invoked in extractor/non-interactive mode:

- CloseProject() will result in termination of the siwave.exe process.
- CloseProjectNoForce() will close any open projects, but will keep the siwave.exe process active (so that subsequent script commands can be issued).

CloseProjectNoForce

Closes the specified project and opens a new "Untitled" project.	
UI Command:	Click **File** > **New**.
Syntax:	`obj.CloseProjectNoForce(<projectName>)`
Parameters:	BSTR projectName
Return Value:	None.
VB Example:	`obj.CloseProjectNoForce "project.siw"`
IPY Example:	`oApp.CloseProjectNoForce('project.siw')`

Note:

The behavior of CloseProjectNoForce() and CloseProject() is identical when the SIwave UI is invoked in graphical/interactive mode.

However, when invoked in extractor/non-interactive mode:

- CloseProject() will result in termination of the siwave.exe process.
- CloseProjectNoForce() will close any open projects, but will keep the siwave.exe process active (so that subsequent script commands can be issued).

Equals

Comparison operator. Returns whether two objects are equal.	
UI Command:	N/A
Syntax:	`obj.Equals(<comparisonObject>)`
Parameters:	VAR comparisonObject
Return Value:	TRUE or FALSE
VB Example:	`obj.Equals(object)`
IPY Example:	`oDoc.Equals(oDoc)`

GetActiveProject

Returns the active project in SIwave.	
UI Command:	None.
Syntax:	`obj.GetActiveProject()`
Parameters:	None.
Return Value:	Object of active project.
VB Example:	`Set obj = app.GetActiveProject`
IPY Example:	`oDoc = oApp.GetActiveProject()`

GetFileDir

Returns the directory path of the open project.	
UI Command:	None.
Syntax:	`obj.GetFileDir()`
Parameters:	None.
Return Value:	BSTR directory path
VB Example:	`fileDirPath = obj.GetFileDir`
IPY Example:	`oDoc.GetFileDir()`

GetFilePath

Returns the directory path of the open project.	
UI Command:	None.
Syntax:	`obj.GetFilePath()`
Parameters:	None.
Return Value:	BSTR directory path
VB Example:	`fileDirPath = obj.GetFilePath`
IPY Example:	`oDoc.GetFilePath()`

GetName

Returns the name of the open SIwave project.	
UI Command:	None.
Syntax:	`obj.GetName()`
Parameters:	None.
Return Value:	BSTR project name
VB Example:	`fileName = obj.GetName`
IPY Example:	`oDoc.GetName()`

GetNetworkDataSolution

Returns network data for a previously run SYZ simulation.	
UI Command:	None.
Syntax:	`obj.GetNetworkDataSolution(<solnName>)`
Parameters:	BSTR solnName (name of a previously run SYZ simulation)
Return Value:	BSTR network data
VB Example:	`obj.GetNetworkDataSolution("SYZ Sim 1")`
IPY Example:	`oDoc.GetNetworkDataSolution('SYZ Sim 1')`

GetNetworkDataSolutionDefinition

Returns port names for a previously run SYZ simulation.	
UI Command:	None.
Syntax:	`obj.GetNetworkDataSolutionDefinition(<solnName>)`
Parameters:	BSTR solnName (name of a previously run SYZ simulation)
Return Value:	ARRAY of BSTR port names
VB Example:	`obj.GetNetworkDataSolutionDefinition("SYZ Sim 1")`
IPY Example:	`oDoc.GetNetworkDataSolutionDefinition('SYZ Sim 1')`

GetProjectDirectory

Returns the directory path of the open project.	
UI Command:	None.
Syntax:	`obj.GetProjectDirectory()`
Parameters:	None.
Return Value:	BSTR directory path
VB Example:	`obj.GetProjectDirectory()`
IPY Example:	`oApp.GetProjectDirectory()`

GetProjectList

Returns a list of all projects that are open in SIwave.	
UI Command:	None.
Syntax:	`obj.GetProjectList()`
Parameters:	None.
Return Value:	ARRAY of strings containing names of all open projects.
VB Example:	`obj.GetProjectList()`
IPY Example:	`oDoc.GetProjectList()`

GetTopDesignList

Returns a list of all designs that are open in SIwave.	
UI Command:	None.
Syntax:	`obj.GetTopDesignList()`
Parameters:	None.
Return Value:	ARRAY of strings containing names of all open designs.
VB Example:	`obj.GetTopDesignList()`
IPY Example:	`oDoc.GetTopDesignList()`

GetVersion

Returns SIwave version information.	
UI Command:	None.
Syntax:	`obj.GetVersion()`
Parameters:	None.
Return Value:	BSTR version information
VB Example:	`obj.GetVersion()`
IPY Example:	`oApp.GetVersion()`

ImportAnfFile

Imports an ANF file into a new project.	
UI Command:	**Import** > **ANF**.
Syntax:	`obj.ImportAnfFile(<filePath>)`
Parameters:	BSTR filePath
Return Value:	None.
VB Example:	`Set obj = app.ImportAnfFile ( "C:\KWH\GSG.anf" )`
IPY Example:	`oApp.ImportAnfFile('C:\KWH\GSG.anf')`

ImportOdb

Imports an ODB++ file into a new project.	
UI Command:	**Import** > **ODB++**.
Syntax:	`obj.ImportOdb (<ODB++Filename>, <controlFilename>)`
Parameters:	BSTR ODB++Filename BSTR controlFilename (this string can be empty if no XML control file is used)
Return Value:	None.
VB Example:	`Set oTool = oDesktop.GetTool("ImportExport")` `oTool.ImportOdb "test.tgz", "test.xml"`
IPY Example:	`oApp.ImportOdb('test.tgz','test.xml')`

IsSolutionDataAvailable

Checks for the existence of an SYZ simulation.	
UI Command:	None.
Syntax:	`obj.IsSolutionDataAvailable(<solnName>)`
Parameters:	BSTR solnName (name of a previously run SYZ simulation)
Return Value:	BOOL • **0** – Failure • **1** – Success
VB Example:	`obj.IsSolutionDataAvailable("SYZ Sim 1")`
IPY Example:	`oDoc.IsSolutionDataAvailable('SYZ Sim 1')`

OpenProject

Opens a previously created project, or imports an EDB folder, IPC2581 file, or ODB++ file.	
UI Command:	**File** > **Open**.
Syntax:	`obj.OpenProject(<filePath>)`
Parameters:	BSTR filePath
Return Value:	None.
VB Example:	`Set obj = app.OpenProject ( "C:\KWH\GSG_model.siw" )`
IPY Example:	`oApp.OpenProject('C:\KWH\GSG_model.siw')`

Quit

Quits SIwave and closes the IronPython Command Window.	
UI Command:	**File** > **Exit**.
Syntax:	`obj.Quit`
Parameters:	None.
Return Value:	None.
VB Example:	`app.Quit`
IPY Example:	`oApp.Quit()`

ReferenceEquals

Compares two objects and returns whether they are the same.	
UI Command:	None.
Syntax:	`obj.ReferenceEquals(<obj1>, <obj2>)`
Parameters:	obj1, obj2 (any type)
Return Value:	BOOL
VB Example:	`obj.ReferenceEquals(obj1, obj2)`
IPY Example:	`oDoc.ReferenceEquals(obj1, obj2)`

RestoreWindow

Restores a minimized SIwave window.	
UI Command:	None.
Syntax:	`obj.RestoreWindow`
Parameters:	None.
Return Value:	None.
VB Example:	`obj.RestoreWindow()`
IPY Example:	`oApp.RestoreWindow()`

Save

Saves all changes made until the point in the scripting file where this function is called.	
UI Command:	**File** > **Save**.
Syntax:	`obj.Save`
Parameters:	None.
Return Value:	None.
VB Example:	`obj.Save()`
IPY Example:	`oDoc.Save()`

ScrActivateCktElem

Activates or deactivates specified circuit elements.	
UI Command:	None.
Syntax:	`obj.ScrActivateCktElem(<elementName>, <elementType>, <activate?>)`
Parameters:	BSTR elementName BSTR elementType (select from: cap, ind, res, port, vprobe, csource, vsource) BOOL activate? (True to activate or False to deactivate)
Return Value:	INT result: • **0** – Function succeeded • **1** – Invalid input parameters (i.e., empty strings) • **2** – Invalid second parameter (does not match cap, ind, res, port, vprobe, csource, vsource) • **3** – Specified circuit element could not be located in layout
VB Example:	`result = obj.ScrActivateCktElem ("Port1","port",false)` `result = obj.ScrActivateCktElem ("Port2","port",true)`
IPY Example:	`oDoc.ScrActivateCktElem ('Port1','port',False)` `oDoc.ScrActivateCktElem ('Port2','port',True)`

ScrAddEquipotentialRegion

Creates an equipotential region located at a specified pin.	
UI Command:	None.
Syntax:	`obj.ScrAddEquipotentialRegion (<partName>, <refDes>, <pinName> , <regionOnTop>)`
Parameters:	BSTR partName BSTR refDes BSTR pinName BOOL regionOnTop (True to create the equipotential region above the top metal layer, False to create it below the bottom metal layer)
Return Value:	BOOL • **0** – Failure • **1** – Success
VB Example:	`outcome = doc.ScrAddEquipotentialRegion "T1_A", "U1", "10", True`
IPY Example:	`oDoc.ScrAddEquipotentialRegion('T1_A','U1','10',True)`

ScrAddError

Logs an error to the Information / Errors / Warnings window.	
UI Command:	None.
Syntax:	`obj.ScrAddError(<error>)`
Parameters:	BSTR error
Return Value:	None.
VB Example:	`obj.ScrAddError("This is my error.")`
IPY Example:	`oDoc.ScrAddError('This is my error.')`

ScrAddInfo

Logs information to the Information / Errors / Warnings window.	
UI Command:	None.
Syntax:	`obj.ScrAddInfo(<information>)`
Parameters:	BSTR information
Return Value:	None.
VB Example:	`obj.ScrAddInfo("This is my information.")`
IPY Example:	`oDoc.ScrAddInfo('This is my information.')`

ScrAddLayer

Adds a new layer above/below a given reference layer.	
UI Command:	Click **Home** > **Layer stackup Editor**. Then click either **Add Above Selected Layer** or **Add Below Selected Layer**.
Syntax:	`obj. ScrAddLayer (<newLayerName>, <referenceLayerName>, <aboveBelow>, <layerTypeIndex>, <layerThickness>, <materialName>)`
Parameters:	BSTR newLayerName BSTR referenceLayerName BOOL aboveBelow (True to place new layer above; False to place it below) INT layerTypeIndex (0 for dielectric, 1 for metal, 2 for wirebond) INT layerThickness BSTR materialName
Return Value:	None.
VB Example:	`obj.ScrAddLayer("new layer name", "reference layer name", true, 1, 0.1, "copper")`
IPY Example:	`oDoc.ScrAddLayer('new layer name','reference layer name',True,1,0.1,'copper')`

ScrAddMaterial

Adds a new material to the database.	
UI Command:	**Home** > **Edit Materials** > **[Conductors/Dielectrics]** > **Add**.
Syntax:	`obj.ScrAddMaterial (<materialType>, <materialName>, <materialProperty1>, <materialProperty2>)`
Parameters:	BSTR materialType (conductor or dielectric) BSTR materialName DOUBLE materialProperty1 (conductivity for conductor or permittivity for dielectric) DOUBLE materialProperty2 (permeability for conductor or loss tangent for dielectric)
Return Value:	INT result: • **0** – Function succeeded • **1** – Material type is not "conductor" or "dielectric" • **2** – Material name already exists in library • **3** – Material parameters are not within an acceptable range
VB Example:	`obj.ScrAddMaterial("conductor", "Name", 0.1, 0.1)`
IPY Example:	`oDoc.ScrAddMaterial('dielectric','Name',5,0.02)`

ScrAddOneLayerPadstack

Adds a new padstack.	
UI Command:	**Home** > **Edit Padstacks** > **Add**.
Syntax:	`obj.ScrAddOneLayerPadstack (<padstackName>, <layerName>, <shapeName>, <width>, <height>)`
Parameters:	BSTR padstackName BSTR layerName BSTR shapeName (None, Circle, Oblong, or Rectangle) BSTR width BSTR height
Return Value:	BOOL • **0** – Failure • **1** – Success
VB Example:	`obj.ScrAddOneLayerPadstack( "NEW_PADSTACK", "METAL-1", "Circle", "0.5mm", "0.5mm")`
IPY Example:	`oDoc.ScrAddOneLayerPadstack('NEW_PADSTACK','METAL-1','Circle','0.5mm','0.5mm')`

ScrAddWarning

Logs a warning to the Information / Errors / Warnings window.	
UI Command:	None.
Syntax:	`obj.ScrAddWarning(<warning>)`
Parameters:	BSTR warning
Return Value:	None.
VB Example:	`obj.ScrAddWarning("This is my warning.")`
IPY Example:	`oDoc.ScrAddWarning('This is my warning.')`

ScrAppendSteppedSweep

Defines a frequency sweep with the specified step size for the specified simulation type.	
UI Command:	From any simulation sweep setup tab, add a frequency sweep with a **Linear Step** distribution.
Syntax:	`obj.ScrAppendSteppedSweep(<simType>, <minimumFrequency>, <maximumFrequency>, <stepSize>)`
Parameters:	BSTR simType ("ac", "ff", "nf", "syz", "psi_syz", "psi_ac", or "hfss_syz") DOUBLE minimumFrequency DOUBLE maximumFrequency DOUBLE stepSize (in Hz)
Return Value:	BOOL • **0** – Failure • **1** – Success
VB Example:	`outcome = obj.ScrAppendSteppedSweep("syz", 5000000.0, 5005000000.0, 100000000.0)`
IPY Example:	`oDoc.ScrAppendSteppedSweep('syz', 5000000.0, 5005000000.0, 100000000.0)`

ScrAppendSweep

Defines a frequency sweep with the specified number of frequencies for the specified simulation type.	
UI Command:	From any simulation sweep setup tab, add a frequency sweep with a **Linear** or **By Decade** distribution.
Syntax:	`obj.ScrAppendSweep(<simType>, <minimumFrequency>, <maximumFrequency>, <numPts>, <isLog>)`
Parameters:	BSTR simType ("ac", "ff", "nf", "syz", "psi_syz", "psi_ac", or "hfss_syz") DOUBLE minimumFrequency DOUBLE maximumFrequency INT numPts (number of frequencies in the frequency distribution) BOOL isLog (True for **By Decade** distribution, False for **Linear** distribution)
Return Value:	BOOL • **0** – Failure • **1** – Success
VB Example:	`outcome = obj.ScrAppendSweep("syz", 5000000.0, 5000000000.0, 4, TRUE)` This creates a **By Decade** frequency sweep from 5MHz to 5GHz.
IPY Example:	`oDoc.ScrAppendSweep('syz',5000000.0,5000000000.0,4,True)` This creates a **By Decade** frequency sweep from 5MHz to 5GHz

ScrAssign4PtBondwireProfile

Creates a 4 point bondwire profile for a layer.	
UI Command:	**Home** > **Bondwire Model Editor**. Select **JEDEC 4-Point** from the **Model** drop-down menu.
Syntax:	`obj.ScrAssign4PtBondwireProfile (<layerOrProfileName>, <h1>, <h2>, <radius>, <supportLayerName>, <terminationLayerName>)`
Parameters:	BSTR layerOrProfileName (either the name of a layer containing bondwires or a specific bondwire model name) DOUBLE h1 DOUBLE h2 DOUBLE radius BSTR supportLayerName BSTR terminationLayerName
Return Value:	BOOL • **0** – Failure • **1** – Success
VB Example:	`obj.ScrAssign4PtBondwireProfile "die2_die3", 0.3, 0.3, 0.01, "Signal", "Power"`
IPY Example:	`oDoc.ScrAssign4PtBondwireProfile('die2_die3',0.3,0.3,0.01,'Signal','Power')`

ScrAssign5PtBondwireProfile

Creates a 5 point bondwire profile for a layer.	
UI Command:	**Home** > **Bondwire Model Editor**. Select **JEDEC 5-Point** from the **Model** drop-down menu.
Syntax:	`obj.ScrAssign5PtBondwireProfile(<layerOrProfileName>, <h1>, <h2>, <radius>, <alpha>, <beta>, <supportLayerName>, <terminationLayerName>)`
Parameters:	BSTR layerOrProfileName (either the name of a layer containing bondwires or a specific bondwire model name) DOUBLE h1 DOUBLE h2 DOUBLE radius DOUBLE alpha (angle in degrees) DOUBLE beta (angle in degrees) BSTR supportLayerName BSTR terminationLayerName
Return Value:	BOOL • **0** – Failure • **1** – Success
VB Example:	`obj.ScrAssign5PtBondwireProfile "die1_die3", 0.31, 0.32, 0.011, 85.1, 5.1, "Top", "Ground"`
IPY Example:	`oDoc.ScrAssign5PtBondwireProfile('die1_die3',0.31,0.32,0.011,85.1,5.1,'Top','Ground')`

ScrAssignBondwireTerminalType

Sets bondwire terminals to either sink or source terminal type.	
UI Command:	**Home** > **Bondwire Model Editor**.
Syntax:	`obj.ScrAssignBondwireTerminalType(<netNameRegExp>, <refDesRegExp>, <pinNameRegExp>, <isSink>)`
Parameters:	BSTR netNameRegExp BSTR refDesRegExp BSTR pinNameRegExp BOOL isSink (True for Sink, False for Source)
Return Value:	BOOL • **0** – Failure • **1** – Success
VB Example:	`obj.ScrAssignBondwireTerminalType "*Net_2*", "RT*", "100-*", true`
IPY Example:	`oDoc.ScrAssignBondwireTerminalType('*Net_2*','RT*','100-*',True)`

ScrAssignComplexSolderballProfile

Creates a complex solderball profile for a layer.	
UI Command:	**Home** > **Solderball Properties**. Select **Complex** from the **Type** drop-down menu.
Syntax:	`obj.ScrAssignComplexSolderballProfile(<padstackName>, <height>, <radius>, <midRadius>, <frustumHeight>, <placement>, <terminalType>)`
Parameters:	BSTR padstackName DOUBLE height DOUBLE radius DOUBLE midRadius DOUBLE frustrumHeight INT placement (0 for Above, 1 for Below) INT terminalType (0 for Sink, 1 for Source)
Return Value:	BOOL • **0** – Failure • **1** – Success
VB Example:	`obj.ScrAssignComplexSolderballProfile "BGA", 0.5, 0.18, 0.225, 0.1666, 0, 0`
IPY Example:	`oDoc.ScrAssignComplexSolderballProfile ('BGA',0.5,0.18,0.225,0.1666,0,0)`

ScrAssignLowBondwireProfile

Creates a low bondwire profile for a layer.	
UI Command:	**Home** > **Bondwire Model Editor**. Select **Low** from the **Model** drop-down menu.
Syntax:	`obj.ScrAssignLowBondwireProfile(<layerName>, <h1>, <h2>, <radius>, <alpha>, <beta>, <supportLayerName>, <terminationLayerName>)`
Parameters:	BSTR layerName (name of a layer containing bondwires) DOUBLE h1 DOUBLE h2 DOUBLE radius DOUBLE alpha DOUBLE beta BSTR supportLayerName BSTR terminationLayerName
Return Value:	BOOL • **0** – Failure • **1** – Success
VB Example:	`obj.ScrAssignLowBondwireProfile "die2_die3", 0.3, 0.3, 0.01, 0.1, 0.1, "Signal", "Power"`
IPY Example:	`oDoc.ScrAssignLowBondwireProfile('die2_die3',0.3,0.3,0.01,0.1,0.1,'Signal','Power')`

ScrAssignSimpleSolderballProfile

Creates a simple solderball profile for a layer.	
UI Command:	**Home** > **Solderball Properties**. Select **Simple** from the **Type** drop-down menu.
Syntax:	`obj.ScrAssignSimpleSolderballProfile(<padstackName>, <height>, <radius>, <placement>, <terminalType>)`
Parameters:	BSTR padstackName DOUBLE height DOUBLE radius INT placement (0 for Above, 1 for Below) INT terminalType (0 for Sink, 1 for Source)
Return Value:	BOOL • **0** – Failure • **1** – Success
VB Example:	`obj.ScrAssignSimpleSolderballProfile "BGA", 0.5, 0.225, 0, 0`
IPY Example:	`oDoc.ScrAssignSimpleSolderballProfile('BGA', 0.5, 0.225, 0, 0)`

ScrAssignSketchedBondwireProfile

Creates a sketched bondwire profile for a layer.	
UI Command:	**Home** > **Bondwire Model Editor**. Select **Sketched** from the **Model** drop-down menu.
Syntax:	`obj.ScrAssignSketchedBondwireProfile (<layerOrProfileName>, <filePath>, <radius>, <supportLayerName>, <terminationLayerName>)`
Parameters:	BSTR layerOrProfileName (can be either the name of a layer containing bondwires or a specific bondwire model name) BSTR filePath INT radius BSTR supportLayerName BSTR terminationLayerName
Return Value:	BOOL • **0** – Failure • **1** – Success
VB Example:	`obj.ScrAssignSketchedBondwireProfile "die1_die2", "F:\TestScriptsForDocumentation\bw_pts.bwp", 0.012, "Top", "Ground"`
IPY Example:	`oDoc.ScrAssignSketchedBondwireProfile('die1_die2', 'F:\TestScriptsForDocumentation\bw_pts.bwp', 0.012, 'Top', 'Ground')`

ScrAssignSketchedBondwireProfileFromArray

Creates a sketched bondwire profile for a layer, from an array.	
UI Command:	**Home** > **Bondwire Model Editor**. Select **Sketched** from the **Model** drop-down menu.
Syntax:	`obj.ScrAssignSketchedBondwireProfile(<layerName>, <units>, <bwPoints>, <radius>, <supportLayerName>, <terminationLayerName>)`
Parameters:	BSTR layerName BSTR units ARRAY bwPoints (array of xy coordinates) INT radius BSTR supportLayerName BSTR terminationLayerName
Return Value:	BOOL • **0** – Failure • **1** – Success
VB Example:	`obj.ScrAssignSketchedBondwireProfileFromArray "die1_die2", "mm", xyPoints, 0.012, "Top", "Ground"`
IPY Example:	`oDoc.ScrAssignSketchedBondwireProfileFromArray('die1_die2', 'mm', xyPoints, 0.012, 'Top', 'Ground')`

ScrAssignSolderballTerminalType

Sets specified Solderball Terminals to either Sink or Source type.	
UI Command:	**Home** > **Solderball Properties**.
Syntax:	`obj.ScrAssignSolderballTerminalType(<netNameRegExp>, <referenceDesigRegExp>, <pinNameRegExp>, <isSink>)`
Parameters:	BSTR netNameRegExp (narrows solderball terminals by net name) BSTR referenceDesigRegExp (narrows solderball terminals by reference designator) BSTR pinNameRegExp (narrows solderball terminals by pin name) BOOL isSink (TRUE sets sinks; FALSE sets sources)
Return Value:	BOOL • **0** – Failure • **1** – Success
VB Example:	`obj.ScrAssignSolderballTerminalType "*Net_2*", "RT*", "100-*", true`
IPY Example:	`oDoc.ScrAssignSolderballTerminalType('*Net_2*', 'RT*', '100-*', True)`

ScrBooleanUnite

Performs a boolean unite on geometry from the specified nets.	
UI Command:	**Tools** > **Unite**.
Syntax:	`obj.ScrBooleanUnite(<netNameList>)`
Parameters:	ARRAY netNameList
Return Value:	BOOL • **0** – Failure • **1** – Success
VB Example:	`outcome = doc.ScrBooleanUnite ("netName1","netName2","netName3")`
IPY Example:	`oDoc.ScrBooleanUnite(['netName1','netName2','netName3'])`

ScrChangePartType

Changes the part type for a specified part name.	
UI Command:	None.
Syntax:	`obj.ScrChangePartType(<partName>, <newType>)`
Parameters:	BSTR partName BSTR newType • For existing parts with two pins, valid types: capacitor, inductor, resistor, port, voltage probe, current source, voltage source • For other existing parts, newType cannot be any of these
Return Value:	BOOL • **0** – Failure • **1** – Success
VB Example:	`obj.ScrChangePartType ("288DIMMDDR4_EDGE_CONN-BASE", "Discrete Device")`
IPY Example:	`oDoc.ScrChangePartType('288DIMMDDR4_EDGE_CONN-BASE','Discrete Device')`

ScrCleanUpOverlappingtraces

Converts overlapping traces to planes and then merges them.	
UI Command:	None.
Syntax:	`obj.ScrCleanUpOverlappingtraces(<layerNames>)`
Parameters:	ARRAY layerNames
Return Value:	BOOL: • **0** – Failure • **1** – Success
VB Example:	`obj.ScrCleanUpOverlappingtraces ("layer1","layer2","layer3")`
IPY Example:	`oDoc.ScrCleanUpOverlappingtraces (['layer1','layer2','layer3'])`

ScrClearAllSweeps

Removes all frequency sweeps assigned to a particular simulation type.	
UI Command:	**Simulation** > **SIwave** > **[Simulation Type]**. Select the values in the **Frequency Range Setup** box and click **Delete Selection**.
Syntax:	`obj.ScrClearAllSweeps (<sweepType>)`
Parameters:	BSTR sweepType (must be one of the following: "ac", "ff", "nf", "syz" or "hfss_syz")
Return Value:	BOOL • **0** – Failure (sweepType is not a valid type) • **1** – Success
VB Example:	`obj.ScrClearAllSweeps ("ff")`
IPY Example:	`oDoc.ScrClearAllSweeps('ff')`

ScrClipDesign

Clips designated nets using a specified polygon.	
UI Command:	None.
Syntax:	`obj.ScrClipDesign (<netNames>, <points>)`
Parameters:	ARRAY netNames (nets to be clipped) ARRAY points (vertices of clipping polygon)
Return Value:	BOOL • **0** – Failure • **1** – Success

VB Example:	`dim netNames (5)`
	`netNames(0)="VCC"`
	`netNames(1)="GND"`
	`netNames(2)="Heg"`
	`netNames(3)="NET-1"`
	`netNames(4)="NET-2"`
	`netNames(5)="PWR"`
	`' points for polygon: (0.0, 0.0), (10.0, 20.0),`
	`' (40.0, 40.0), (40.0, 0.0), (16.0, 4.0)`

Clips designated nets using a specified polygon.	

	`dim points(9)`
	`points(0)=0.0`
	`points(1)=0.0`
	`points(2)=10.0`
	`points(3)=20.0`
	`points(4)=40.0`
	`points(5)=40.0`
	`points(6)=40.0`
	`points(7)=0.0`
	`points(8)=16.0`
	`points(9)=4.0`
	`' outcome is set to TRUE on success`
	`outcome = obj.ScrClipDesign ( netNames, points )`

IPY Example:	`netNames = ['VCC', 'GND', 'Heg', 'NET-1', 'NET-2', 'PWR']`
	`points = [0.0, 0.0, 10.0, 20.0, 40.0, 40.0, 40.0, 0.0, 16.0, 4.0]`
	`outcome = obj.ScrClipDesign (netNames, points)`

ScrClipDesignAroundNets

Clips the design around nets.	
UI Command:	None.
Syntax:	`obj.ScrClipDesignAroundNets(<netNames>, <clipExtentDistance>, <simplifiedExtent>, <traceCuttingOption>, <ignoreLayerVisibility>, <reverseCutting>)`
Parameters:	ARRAY netNames (array of strings holding the net names of the selected nets) BSTR clipExtentDistance (distance to push from the selected nets; must include units and will not support wavelength based distance) BOOL simplifiedExtent (TRUE to simplify edges or FALSE) INT traceCuttingOption • **0** – Cut traces that cross the boundary. • **1** – Include all traces that overlap the extent. • **2** – Include only traces that are completely inside the extent. BOOL ignoreLayerVisibility (TRUE to ignore or FALSE) BOOL reverseCutting (TRUE to clip outside the polygon or FALSE to keep objects inside the clipping polygon)
Return Value:	BOOL • **0** – Failure • **1** – Success
VB Example:	`obj.ScrClipDesignAroundNets (netNames, "1mm", TRUE, 0, TRUE, FALSE)`
IPY Example:	`oDoc.ScrClipDesignAroundNets (netNames, '1mm', True, 0, True, False)`

ScrCloseProject

Closes the current active project and opens a new "Untitled" project.	
UI Command:	Click **File** > **New**.
Syntax:	`obj.ScrCloseProject()`
Parameters:	None.
Return Value:	None.
VB Example:	`obj.ScrCloseProject()`
IPY Example:	`oDoc.ScrCloseProject()`

ScrCloseProjectNoSave

Closes the current active project and opens a new "Untitled" project.	
UI Command:	Click **File** > **New**.
Syntax:	`obj.ScrCloseProjectNoSave()`
Parameters:	None.
Return Value:	None.
VB Example:	`obj.ScrCloseProjectNoSave()`
IPY Example:	`oDoc.ScrCloseProjectNoSave()`

ScrComputeFwsSubckt

Computes a full-wave Spice subcircuit representing the specified S-parameter solution. The ScrSetFwsSubcktFormat() function specifies the Spice format to use.	
UI Command:	Click **Results** > **SYZ** > **[simulation name]** > **Compute FWS sub-circuit**.
Syntax:	`obj.ScrComputeFwsSubckt(<syzSimName>, <path>)`
Parameters:	BSTR syzSimName BSTR path (save location)
Return Value:	INT • **0** – Success • **Non-zero value** – Failure
VB Example:	`outcome=obj.ScrComputeFwsSubckt("SYZ Sweep 1", "C:\sweep1" )`
IPY Example:	`oDoc.ScrComputeFwsSubckt('SYZ Sweep 1', 'C:\sweep1')`

ScrComputeFwsSubcktForNamedSim

Computes a full-wave Spice subcircuit for a specified simulation. **The ScrSetFwsSubcktFormat() function specifies the Spice format to use.**	
UI Command:	Click **Results** > **[simulation type]** > **[simulation name]** > **Compute FWS sub-circuit**.
Syntax:	`obj.ScrComputeFwsSubcktForNamedSim(<simType>, <simName>, <path>)`
Parameters:	BSTR simType BSTR simName BSTR path
Return Value:	INT • **0** – Success • **Non-zero value** – Failure
VB Example:	`outcome=obj.ScrComputeFwsSubcktForNamedSim('syz', "SYZ Sweep 1", "C:\sweep1" )`
IPY Example:	`oDoc.ScrComputeFwsSubcktForNamedSim('syz', 'SYZ Sweep 1', 'C:\sweep1')`

ScrConvertPlanesToTraces

Performs plane to trace conversion on specified nets.	
UI Command:	Click **Tools** > **Convert Planes to Traces**.
Syntax:	`obj.ScrConvertPlanesToTraces (<netNameList>)`
Parameters:	ARRAY netNameList
Return Value:	BOOL • **0** – Failure • **1** – Success

VB Example:	`dim netNames`
	`netNames = Array("GND", "P28VA")`
	`outcome = obj.ScrConvertPlanesToTraces(netNames)`

IPY Example:	`oDoc.ScrConvertPlanesToTraces(['GND','P28VA'])`

ScrConvertTracesToPlanes

Performs trace to plane conversion on a specified trace.	
UI Command:	Click **Tools** > **Convert Traces to Planes**.
Syntax:	`obj.ScrConvertTracesToPlanes (<layerName>, <netName>, <mergeAll>, <minVoidArea>, <unitName>)`
Parameters:	BSTR layerName BSTR netName (If net name is "all", traces of all nets on the layer will be converted) BOOL mergeAll (TRUE merges all planes; FALSE does not) DOUBLE minVoidArea (all voids smaller than this value will be deleted) BSTR unitName ("m", "meters", "mm", "cm", "um", "microns", "inches", "mils", "nanometers", "nm", or "feet")
Return Value:	BOOL • **0** – Failure • **1** – Success
VB Example:	`obj.ScrConvertTracesToPlanes ("BOTTOM","GND",true,0.001,"mm")`
IPY Example:	`oDoc.ScrConvertTracesToPlanes ('BOTTOM','GND',True,0.001,'mm')`

ScrConvertTracesToPlanesByNet

Performs trace to plane conversion by a specified net.	
UI Command:	Click **Tools** > **Convert Traces to Planes**.
Syntax:	`obj.ScrConvertTracesToPlanesByNet (<netNameList)`
Parameters:	ARRAY netNameList
Return Value:	BOOL • **0** – Failure • **1** – Success

VB Example:	`dim netNames`
	`netNames = Array("GND", "P28VA")`
	`outcome = obj.ScrConvertTracesToPlanes(netNames)`

IPY Example:	`oDoc.ScrConvertTracesToPlanes(['GND','P28VA'])`

ScrCopyImageToClipBoard

Copies the Modeling workspace to the clipboard.	
UI Command:	Right-click in the Modeling workspace and select **Copy Image**.
Syntax:	`obj.ScrCopyImageToClipBoard()`
Parameters:	None.
Return Value:	None.
VB Example:	`obj.ScrCopyImageToClipBoard()`
IPY Example:	`oDoc.ScrCopyImageToClipBoard()`

ScrCreatePinGroups

Creates a pin group containing specified pins.	
UI Command:	**Tools** > **Create/Manage Pin Groups**.
Syntax:	`obj.ScrCreatePinGroups(<partName>, <refDes>, <pinNumbers>, <groupName>, <applyToAllComponents>)`
Parameters:	BSTR partName BSTR refDes ARRAY pinNumbers BSTR groupName BOOL applyToAllComponents (True = pin group should be created for all parts with the given part name. When set to True, refDes is ignored.)
Return Value:	BOOL • **0** – Failure • **1** – Success

VB Example:	`dim pinNumbers`
	`pinNumbers = Array("1","2","3","4")`
	`outcome = obj.ScrCreatePinGroups("CSP_BGA", "BGA", pinNumbers, "new_group_1", False)`

IPY Example:	`oDoc.ScrCreatePinGroups('CSP_BGA', 'BGA', ['1','2','3','4'], 'new_group_1', False)`

ScrCreatePinGroupByDist

Creates a pin group for pins a given distance from a specified pin.	
UI Command:	**Tools** > **Create/Manage Pin Groups**.
Syntax:	`obj.ScrCreatePinGroupByDist (<partName>, <refDes>, <pinNumber>, <groupName>, <maxDistance>, <selectFromAllNets>)`
Parameters:	BSTR partName BSTR refDes BSTR pinNumber BSTR groupName BSTR maxDistance BOOL selectFromAllNets (True = pins will be selected regardless of their nets. False = only pins from the same net as the specified pin are included.)
Return Value:	BOOL • **0** – Failure • **1** – Success
VB Example:	`obj.ScrCreatePinGroupByDist("T1_A", "U1", "14", "TestPinGroupA", "450um", true)`
IPY Example:	`oDoc.ScrCreatePinGroupByDist('T1_A', 'U1', '14', 'TestPinGroupA', '450um', True)`

ScrCreatePinGroupsByGrid

Creates a pin group by dividing a component into a grid.	
UI Command:	**Tools** > **Create/Manage Pin Groups**.
Syntax:	`obj.ScrCreatePinGroupsByGrid (<partName>, <refDes>, <numRows>, <numCols>, <applyPerNet>, <applyToAllComponents>)`
Parameters:	BSTR partName BSTR refDes LONG numRows LONG numCols BOOL applyPerNet BOOL applyToAllComponents
Return Value:	BOOL • **0** – Failure • **1** – Success
VB Example:	`obj.ScrCreatePinGroupsByGrid ("DDR4_X4_FBGA78-10X13,,", "U1", 3, 2, FALSE, TRUE)`
IPY Example:	`oDoc.ScrCreatePinGroupsByGrid ('DDR4_X4_FBGA78-10X13,,', 'U1', 3, 2, False, True)`

ScrCreatePinGroupByNet

Creates a pin group consisting of pins from a specified part within a specified net.	
UI Command:	**Tools** > **Create/Manage Pin Groups**.
Syntax:	`obj.ScrCreatePinGroupByNet (<partName>, <refDes>, <netName>, <groupName>, <applyToAllComponents>)`
Parameters:	BSTR partName BSTR refDes BSTR netName BSTR groupName BOOL applyToAllComponents (True= pin groups will be created on all parts of the specified partName regardless of the refDes; the resulting group names will have an index number appended for uniqueness)
Return Value:	BOOL • **0** – Failure • **1** – Success
VB Example:	`obj.ScrCreatePinGroupByNet "T1_A", "U1", "GND", "U1_GND", false`
IPY Example:	`oDoc.ScrCreatePinGroupByNet ('T1_A', 'U1', 'GND', 'U1_GND', False)`

ScrCreatePortsOnPart

Creates ports between pins on the specified part. **Note:** The positive terminals are either the specified pins, the pins on the specified net, or the intersection of the two sets if both parameters are not empty. The negative terminal is chosen as the closest pin on the reference net.	
UI Command:	**Tools** > **Create/Manage Pin Groups**.
Syntax:	`obj.ScrCreatePortsOnPart (<partName>, <refDes>, <posNet>, <posPinList>, <refNet>, <impedance>)`
Parameters:	BSTR partName BSTR refDes BSTR posNet ARRAY posPinList BSTR refNet BSTR impedance
Return Value:	BOOL • **0** – Failure • **1** – Success

VB Example:	`dim posPinList`
	`posPinList = Array("pin1","pin2","pin3")`
	`doc.ScrCreatePortsOnPart("CP90-P4969-90", "V1P1", "VREG_S9A_0P8", posPinList, "AGND", "35ohm")`

IPY Example:	`oDoc.ScrCreatePortsOnPart('CP90-P4969-90', 'V1P1', 'VREG_S9A_0P8', ['pin1','pin2','pin3'], 'AGND', '35ohm')`

ScrDeleteAllNets

Deletes all nets in the design.	
UI Command:	Right-click nets, **Edit** > **Delete**.
Syntax:	`obj.ScrDeleteAllNets()`
Parameters:	None.
Return Value:	None.
VB Example:	`obj.ScrDeleteAllNets()`
IPY Example:	`oDoc.ScrDeleteAllNets()`

ScrDeleteCktElem

Deletes a specified circuit element.	
UI Command:	Right-click element, **Edit** > **Delete**.
Syntax:	`obj.ScrDeleteCktElem(<refDes>)`
Parameters:	BSTR refDes
Return Value:	BOOL • **0** – Failure • **1** – Success
VB Example:	`obj.ScrDeleteCktElem("C_1")`
IPY Example:	`oDoc.ScrDeleteCktElem('C_1')`

ScrDeleteDcSolution

Deletes a DC solution.	
UI Command:	**Results** > **DC IR Drop** > **[Solution Name]** > **Delete Solution**.
Syntax:	`obj.ScrDeleteDcSolution()`
Parameters:	None.
Return Value:	None.
VB Example:	`obj.ScrDeleteDcSolution()`
IPY Example:	`oDoc.ScrDeleteDcSolution()`

ScrDeleteFrequencySweepSolution

Deletes a DC solution.	
UI Command:	**Results** > **Frequency Sweep** > **[Solution Name]** > **Delete Solution**.
Syntax:	`obj.ScrDeleteFrequencySweepSolution()`
Parameters:	None.
Return Value:	None.
VB Example:	`obj.ScrDeleteFrequencySweepSolution()`
IPY Example:	`oDoc.ScrDeleteFrequencySweepSolution()`

ScrDeleteLayer

Deletes a specified layer.	
UI Command:	**Home** > **Layer Stackup Editor**. Select layer and click **Delete Selected Layers**.
Syntax:	`obj.ScrDeleteLayer(<layerName>)`
Parameters:	None.
Return Value:	BOOL • **0** – Failure • **1** – Success
VB Example:	`obj.ScrDeleteLayer("BOTTOM")`
IPY Example:	`oDoc.ScrDeleteLayer('BOTTOM')`

ScrDeleteNearFieldSolutions

Deletes any Near Field solutions.	
UI Command:	**Results** > **Near Field** > **[Solution Name]** > **Delete Solution**.
Syntax:	`obj.ScrDeleteNearFieldSolutions()`
Parameters:	None.
Return Value:	None.
VB Example:	`obj.ScrDeleteNearFieldSolutions()`
IPY Example:	`oDoc.ScrDeleteNearFieldSolutions()`

ScrDeleteNet

Deletes a specified net.	
UI Command:	None.
Syntax:	`obj.ScrDeleteNet(<netName>)`
Parameters:	BSTR netName
Return Value:	BOOL • **0** – Failure • **1** – Success
VB Example:	`obj.ScrDeleteNet("net69")`
IPY Example:	`oDoc.ScrDeleteNet('net69')`

ScrDeleteNets

Deletes all nets in a given array.	
UI Command:	None.
Syntax:	`obj.ScrDeleteNets(<netNames>)`
Parameters:	ARRAY netNames
Return Value:	None.

VB Example:	`dim netNames`
	`netNames = Array("net5","net6","net7","net8")`
	`outcome = obj.ScrDeleteNets(netNames)`

IPY Example:	`oDoc.ScrDeleteNets(['net5','net6','net7','net8'])`

ScrDeleteNetsGivenInFile

Deletes all nets named in a given file.	
UI Command:	None.
Syntax:	`obj.ScrDeleteNetsGivenInFile(<fileName>)`
Parameters:	BSTR fileName (file path; file must have net names enclosed in double quotes)
Return Value:	None.
VB Example:	`obj.ScrDeleteNetsGivenInFile("C:\NetFiles\nets_to_delete.txt")`
IPY Example:	`oDoc.ScrDeleteNetsGivenInFile('C:\NetFiles\nets_to_delete.txt')`

ScrDeletePadstack

Deletes a specified padstack.	
UI Command:	None.
Syntax:	`obj.ScrDeletePadstack(<padstackName>)`
Parameters:	BSTR padstackName
Return Value:	BOOL • **0** – Failure • **1** – Success
VB Example:	`obj.ScrDeletePadstack("Stack")`
IPY Example:	`oDoc.ScrDeletePadstack('Stack')`

ScrDeletePinGroup

Deletes a specified pin group.	
UI Command:	None.
Syntax:	`obj.ScrDeletePinGroup(<pin_group_name>,<deleteRefCktElems>)`
Parameters:	BSTR pin_group_name BOOL deleteRefCktElems (True = deletes references circuit element(s); False = referenced circuit element(s) remove the reference to the pin group and change the reference to the first pin of the pin group.)
Return Value:	BOOL • **0** – Failure • **1** – Success
VB Example:	`obj.ScrDeletePinGroup("U1_GND_Group", true)`
IPY Example:	`oDoc.ScrDeletePinGroup('U1_GND_Group', True)`

ScrDeleteResonantModeSolution

Deletes any Resonant Mode solution.	
UI Command:	**Results** > **Resonant Mode** > **[Solution Name]** > **Delete Solution**.
Syntax:	`obj.ScrDeleteResonantModeSolution()`
Parameters:	None.
Return Value:	None.
VB Example:	`obj.ScrDeleteResonantModeSolution()`
IPY Example:	`oDoc.ScrDeleteResonantModeSolution()`

ScrDeleteSpiceSubcktSolution

Deletes any Spice Subcircuit solution.	
UI Command:	None.
Syntax:	`obj.ScrDeleteSpiceSubcktSolution()`
Parameters:	None.
Return Value:	None.
VB Example:	`obj.ScrDeleteSpiceSubcktSolution()`
IPY Example:	`oDoc.ScrDeleteSpiceSubcktSolution()`

ScrDeleteSyzParameterSolution

Deletes any SYZ solution.	
UI Command:	**Results** > **SYZ** > **[Solution Name]** > **Delete Solution**.
Syntax:	`obj.ScrDeleteSyzParameterSolution()`
Parameters:	None.
Return Value:	None.
VB Example:	`obj.ScrDeleteSyzParameterSolution()`
IPY Example:	`oDoc.ScrDeleteSyzParameterSolution()`

ScrDrawCapacitor

Draws a capacitor in the specified location with specified parameters.	
UI Command:	**Home** > **Circuit Elements** > **Add Capacitor**.
Syntax:	`obj.ScrDrawCapacitor(<capName>, <partName>, <px>, <py>, <nx>, <ny>, <posLayerName>, <negLayerName>, <capVal>, <seriesIndVal>, <seriesResVal>)`
Parameters:	BSTR capName BSTR partName DOUBLE px (positive terminal x location) DOUBLE py (positive terminal y location) DOUBLE nx (negative terminal x location) DOUBLE ny (negative terminal y location) BSTR posLayerName BSTR negLayerName DOUBLE capVal (capacitance, in Farads) DOUBLE seriesIndVal (parasitic inductance, in Henries) DOUBLE seriesResVal (parasitic resistance, in Ohms)
Return Value:	BOOL: • **0** – Failure • **1** – Success
VB Example:	`obj.ScrDrawCapacitor ("Cappy","CL03A103KP3NNN",8500,4500,6000,-500,"SURFACE","BASE",1E-07,1E-11,0)`
IPY Example:	`oDoc.ScrDrawCapacitor ('Cappy','CL03A103KP3NNN',8500,4500,6000,-500,'SURFACE','BASE',1E-07,1E-11,0)`

ScrDrawCircle

Draws a circular plane.	
UI Command:	**Home** > **Draw Circle**.
Syntax:	`obj.ScrDrawCircle(<ctrX>, <ctrY>, <radius>, <layerName>, <netName>, <unit>)`
Parameters:	DOUBLE ctrX (x coordinate of the center point) DOUBLE ctrY (y coordinate of the center point) DOUBLE radius BSTR layerName BSTR netName BSTR unit
Return Value:	BOOL: • **0** – Failure • **1** – Success
VB Example:	`obj.ScrDrawCircle(100, 100, 20, "Top Metal", "NET-1", "mm")`
IPY Example:	`oDoc.ScrDrawCircle(100, 100, 20, 'Top Metal', 'NET-1', 'mm')`

ScrDrawInductor

Draws an inductor in the specified location with specified parameters.	
UI Command:	**Home** > **Circuit Elements** > **Add Inductor**.
Syntax:	`obj.ScrDrawInductor(<indName>, <partName>, <px>, <py>, <nx>, <ny>, <posLayerName>, <negLayerName>, <indVal>)`
Parameters:	BSTR indName BSTR partName DOUBLE px (positive terminal x location) DOUBLE py (positive terminal y location) DOUBLE nx (negative terminal x location) DOUBLE ny (negative terminal y location) BSTR posLayerName BSTR negLayerName DOUBLE indVal (inductance, in Henries)
Return Value:	BOOL: • **0** – Failure • **1** – Success
VB Example:	`obj.ScrDrawInductor("Indy","RLC_XYZ_I",8500,4500,6000,1000,"SURFACE","BASE",1E-09)`
IPY Example:	`oDoc.ScrDrawInductor('Indy','RLC_XYZ_I',8500,4500,6000,1000,'SURFACE','BASE',1E-09)`

ScrDrawPolygon

Draws a polygonal plane.	
UI Command:	**Home** > **Draw Polygon**.
Syntax:	`obj.ScrDrawPolygon(<points>, <layerName>, <netName>, <unit>)`
Parameters:	ARRAY points (x, y, x, y, x, y, ...) BSTR layerName BSTR netName BSTR unit
Return Value:	BOOL: • **0** – Failure • **1** – Success

VB Example:	`dim points`
	`points = Array(-10,10,5,-5,-25,-20,20,0)`
	`outcome = obj.ScrDrawPolygon(points,"Top Metal","NET-1","mm")`

IPY Example:	`oDoc.ScrDrawPolygon([-10,10,5,-5,-25,-20,20,0], 'Top Metal', 'NET-1', 'mm')`

ScrDrawPort

Draws a port in the specified location with specified parameters.	
UI Command:	**Home** > **Circuit Elements** > **Add Port**.
Syntax:	`obj.ScrDrawPort(<portName>, <px>, <py>, <nx>, <ny>, <posLayerName>, <negLayerName>, <refZe>)`
Parameters:	BSTR portName DOUBLE px (positive terminal x location) DOUBLE py (positive terminal y location) DOUBLE nx (negative terminal x location) DOUBLE ny (negative terminal y location) BSTR posLayerName BSTR negLayerName DOUBLE refZe (reference impedance, in Ohms)
Return Value:	BOOL: • **0** – Failure • **1** – Success
VB Example:	`obj.ScrDrawPort ("Porty",8500,4500,6000,1000,"SURFACE","BASE",0.1)`
IPY Example:	`oDoc.ScrDrawPort ('Porty',8500,4500,6000,1000,'SURFACE','BASE',0.1)`

ScrDrawRectangle

Draws a rectangular plane.	
UI Command:	**Home** > **Draw Rectangle**.
Syntax:	`obj.ScrDrawRectangle(<x1>, <y1>, <x2>, <y2>, <layerName>, <netName>, <unit>)`
Parameters:	DOUBLE x1 DOUBLE y1 DOUBLE x2 DOUBLE y2 BSTR layerName BSTR netName BSTR unit
Return Value:	BOOL: • **0** – Failure • **1** – Success
VB Example:	`obj.ScrDrawRectangle(100, 100, 200, 200, "Top Metal", "NET-1", "mm")`
IPY Example:	`oDoc.ScrDrawRectangle (100, 100, 200, 200, 'Top Metal', 'NET-1', 'mm')`

ScrDrawResistor

Draws a resistor in the specified location with specified parameters.	
UI Command:	**Home** > **Circuit Elements** > **Add Resistor**.
Syntax:	`obj.ScrDrawResistor(<resName>, <partName>, <px>, <py>, <nx>, <ny>, <posLayerName>, <negLayerName>, <resVal>)`
Parameters:	BSTR resName BSTR partName DOUBLE px (positive terminal x location) DOUBLE py (positive terminal y location) DOUBLE nx (negative terminal x location) DOUBLE ny (negative terminal y location) BSTR posLayerName BSTR negLayerName DOUBLE resVal (resistance, in Ohms)
Return Value:	BOOL: • **0** – Failure • **1** – Success
VB Example:	`obj.ScrDrawResistor("Resist","BBQ_4L1FE",8500,4500,6000,1000,"SURFACE","BASE",50)`
IPY Example:	`oDoc.ScrDrawResistor('Resist','BBQ_4L1FE',8500,4500,6000,1000,'SURFACE','BASE',50)`

ScrDrawTrace

Draws a trace.	
UI Command:	**Home** > **Draw Trace**.
Syntax:	`obj.ScrDrawTrace(<points>, <width>, <layerName>, <netName>, <unit>)`
Parameters:	ARRAY points (x, y, x, y, x, y, ...) DOUBLE width BSTR layerName BSTR netName BSTR unit
Return Value:	BOOL: • **0** – Failure • **1** – Success

VB Example:	`dim points`
	`points = Array(10,-10,-5,5,25,20,-20,0)`
	`outcome = obj.ScrDrawTrace(points,0.5,"Top Metal","NET-1","mm")`

IPY Example:	`oDoc.ScrDrawTrace([-10,10,5,-5,-25,-20,20,0], '0.5', 'Top Metal', 'NET-1', 'mm')`

ScrDrawVia

Draws a via.	
UI Command:	**Home** > **Drop Via**.
Syntax:	`obj.ScrDrawVia(<ctrX>, <ctrY>, <topLayer>, <botLayer>, <padstack>, <netName>, <offsetX>, <offsetY>, <rotAngle>, <unit>)`
Parameters:	DOUBLE ctrX DOUBLE ctrY BSTR topLayer BSTR botLayer BSTR padstack BSTR netName DOUBLE offsetX DOUBLE offsetY DOUBLE rotAngle BSTR unit
Return Value:	BOOL: • **0** – Failure • **1** – Success
VB Example:	`obj.ScrDrawVia (100, 100, "Top Metal", "Bottom Metal", "Thru Via", "NET-1", 0.0, 0.0, 0.0, "mm")`
IPY Example:	`oDoc.ScrDrawVia(100, 100, 'Top Metal', 'Bottom Metal', 'Thru Via', 'NET-1', 0.0, 0.0, 0.0, 'mm')`

ScrDrawVoltageProbe

Draws a voltage probe in the specified location with specified parameters.	
UI Command:	**Home** > **Circuit Elements** > **Add Voltage Probe**.
Syntax:	`obj.ScrDrawVoltageProbe(<probeName>, <px>, <py>, <nx>, <ny>, <posLayerName>, <negLayerName>)`
Parameters:	BSTR probeName DOUBLE px (positive terminal x location) DOUBLE py (positive terminal y location) DOUBLE nx (negative terminal x location) DOUBLE ny (negative terminal y location) BSTR posLayerName BSTR negLayerName
Return Value:	BOOL: • **0** – Failure • **1** – Success
VB Example:	`obj.ScrDrawVoltageProbe ("ProbeOno",6500,4000,5000,1500,"SURFACE","BASE")`
IPY Example:	`oDoc.ScrDrawVoltageProbe ('ProbeOno',6500,4000,5000,1500,'SURFACE','BASE')`

ScrDrawVoltageSource

Draws a frequency independent voltage source in the specified location with specified parameters.	
UI Command:	**Home** > **Circuit Elements** > **Add Voltage Source**.
Syntax:	`obj.ScrDrawVoltageSource(<sourceName>, <partName>, <px>, <py>, <nx>, <ny>, <posLayerName>, <negLayerName>, <mag>, <phase>, <seriesRes>)`
Parameters:	BSTR sourceName BSTR partName DOUBLE px (positive terminal x location) DOUBLE py (positive terminal y location) DOUBLE nx (negative terminal x location) DOUBLE ny (negative terminal y location) BSTR posLayerName BSTR negLayerName DOUBLE mag (magnitude, in Volts) DOUBLE phase (phase, in degrees) DOUBLE seriesRes (parasitic resistance, in Ohms)
Return Value:	BOOL: • **0** – Failure • **1** – Success
VB Example:	`obj.ScrDrawVoltageSource("Sourcey","PRT1_00543",5000,4500,4500,3000,"SURFACE","BASE",1,0,1E-06)`
IPY Example:	`oDoc.ScrDrawVoltageSource('Sourcey','PRT1_00543',5000,4500,4500,3000,'SURFACE','BASE',1,0,1E-06)`

ScrEditCktElemName

Edits an existing circuit element's name.	
UI Command:	**Components** window. Right-click **[circuit element]** > **Edit Circuit Element**. Change name.
Syntax:	`obj.ScrEditCktElemName(<name>, <type> <newName>)`
Parameters:	BSTR name BSTR type ("cap", "ind", "res", "port", "vprobe", "csource", or "vsource") BSTR newName
Return Value:	INT: • **0** – Success • **1** – One of the input parameters is an empty string. • **2** – type value is unacceptable. • **3** – Cannot find component to rename. • **4** – newName is already taken.
VB Example:	`obj.ScrEditCktElemName("port_old", "port", "port_new")`
IPY Example:	`oDoc.ScrEditCktElemName('port_old', 'port', 'port_new')`

ScrEditLayerName

Sets a new name for a specified layer.	
UI Command:	Click **Home** > **Layer Stackup Editor**. Enter a value in the **Name** field.
Syntax:	`obj.ScrEditLayerName(<layerName>, <newLayerName>)`
Parameters:	BSTR layerName BSTR newLayerName
Return Value:	None.
VB Example:	`obj.ScrEditLayerName("MY LAYER","MY NEW LAYER")`
IPY Example:	`oDoc.ScrEditLayerName('MY LAYER','MY NEW LAYER')`

ScrEditMaterial

Edits an existing material.	
UI Command:	**Home** > **Edit Materials**.
Syntax:	`obj.ScrEditMaterial(<matType>, <matName>, <epsOrSigma>, <ltOrPerm>)`
Parameters:	BSTR matType (conductor or dielectric) BSTR matName DOUBLE epsOrSigma (conductivity for conductor or permittivity for dielectric) DOUBLE ltOrPerm (permeability for conductor or loss tangent for dielectric)
Return Value:	INT result: • **0** – Success • **1** – Material type is not "conductor" or "dielectric" • **2** – Material name does not exist in library • **3** – Material parameters are not within an acceptable range
VB Example:	`obj.ScrEditMaterial("conductor", "unobtanium", 5.7E+07, 0.5)`
IPY Example:	`oDoc.ScrEditMaterial('conductor', 'unobtanium', 5.7E+07, 0.5)`

ScrEditNetName

Edits an existing net's name.	
UI Command:	Nets window. Right-click net and select **Edit Net Name**.
Syntax:	`obj.ScrEditNetName(<netName>, <newNetName>)`
Parameters:	BSTR netName BSTR newNetName
Return Value:	None.
VB Example:	`obj.ScrEditNetName("MY NET", "MY NEW NET")`
IPY Example:	`oDoc.ScrEditNetName('MY NET', 'MY NEW NET')`

ScrEditPadStackName

Edits an existing padstack's name.	
UI Command:	**Home** > **Edit Padstacks**. Enter a new **Name**.
Syntax:	`obj.ScrEditPadStackName(<oldPadstackName>, <newPadstackName>)`
Parameters:	BSTR oldPadstackName BSTR newPadstackName
Return Value:	BOOL: • **0** – Failure • **1** – Success
VB Example:	`obj.ScrEditPadStackName("MY PADSTACK", "MY NEW PADSTACK")`
IPY Example:	`oDoc.ScrEditPadStackName('MY PADSTACK', 'MY NEW PADSTACK')`

ScrEnableCavityFieldCoupling

Enables or disables cavity field coupling detection and solution during simulation. **Note:** Does not apply to DC IR simulations.	
UI Command:	**Simulation** > **Options** to open the **SIwave Options** window. On **SI/PI Advanced** tab, select **Cavity field** check box.
Syntax:	`obj.ScrEnableCavityFieldCoupling <flag>`
Parameters:	BOOL flag (TRUE/1 = Enable; FALSE/0 = Disable)
Return Value:	None.
VB Example:	`obj.ScrEnableCavityFieldCoupling True`
IPY Example:	`oDoc.ScrEnableCavityFieldCoupling(1)`

ScrEnableCoPlaneCoupling

Enables or disables coplanar waveguide coupling detection and solution during simulation. **Note:** Does not apply to DC IR simulations.	
UI Command:	**Simulation** > **Options** to open the **SIwave Options** window. On **SI/PI Advanced** tab, select **Coplane** check box.
Syntax:	`obj.ScrEnableCoPlaneCoupling <flag>`
Parameters:	BOOL flag (TRUE/1 = Enable; FALSE/0 = Disable)
Return Value:	None.
VB Example:	`obj.ScrEnableCoPlaneCoupling True`
IPY Example:	`oDoc.ScrEnableCoPlaneCoupling(1)`

ScrEnableErcSimSetup

For PSI simulations, enables or disables option to perform ERC during simulation setup	
UI Command:	**Simulation** > **PSI Options**. Select **Perform ERC during simulation setup**.
Syntax:	`obj.ScrEnableErcSimSetup(<flag>)`
Parameters:	BOOL flag (TRUE/1 = Enable; FALSE/0 = Disable)
Return Value:	None.
VB Example:	`obj.ScrEnableErcSimSetup True`
IPY Example:	`oDoc.ScrEnableErcSimSetup(1)`

ScrEnableFwsRelativeErrorTol

Enables or disables relative error tolerance.	
UI Command:	**Results** > **SYZ** > **[Simulation Name]** > **Compute FWS Sub-circuit**. Select **Advanced** > **Enable Relative Error Tolerance**.
Syntax:	`obj.ScrEnableFwsRelativeErrorTol <flag>`
Parameters:	BOOL flag (True enables; False disables)
Return Value:	None.
VB Example:	`obj.ScrEnableFwsRelativeErrorTol True`
IPY Example:	`oDoc.ScrEnableFwsRelativeErrorTol(True)`

ScrEnableIntraPlaneCoupling

Enables or disables low-frequency intra-plane coupling detection and solution during simulation. **Note:** Does not apply to DC IR simulations.	
UI Command:	**Simulation** > **Options** to open the **SIwave Options** window. On **SI/PI Advanced** tab, select **Intra-plane** check box.
Syntax:	`obj.ScrEnableIntraPlaneCoupling <flag>`
Parameters:	BOOL flag (TRUE/1 = Enable; FALSE/0 = Disable)
Return Value:	None.
VB Example:	`obj.ScrEnableIntraPlaneCoupling True`
IPY Example:	`oDoc.ScrEnableIntraPlaneCoupling(1)`

ScrEnableSplitPlaneCoupling

Enables or disables split plane coupling detection and solution during simulation. **Note:** Does not apply to DC IR simulations.	
UI Command:	**Simulation** > **Options** to open the **SIwave Options** window. On **SI/PI Advanced** tab, select **Split-plane** check box.
Syntax:	`obj.ScrEnableSplitPlaneCoupling <flag>`
Parameters:	BOOL flag (TRUE/1 = Enable; FALSE/0 = Disable)
Return Value:	None.
VB Example:	`obj.ScrEnableSplitPlaneCoupling True`
IPY Example:	`oDoc.ScrEnableSplitPlaneCoupling(1)`

ScrEnableTraceCoupling

Enables or disables trace coupling detection and solution during simulation. **Note:** Does not apply to DC IR simulations.	
UI Command:	**Simulation** > **Options** to open the **SIwave Options** window. On **SI/PI Advanced** tab, select **Trace** check box.
Syntax:	`obj.ScrEnableTraceCoupling <flag>`
Parameters:	BOOL flag (TRUE/1 = Enable; FALSE/0 = Disable)
Return Value:	None.
VB Example:	`obj.ScrEnableTraceCoupling True`
IPY Example:	`oDoc.ScrEnableTraceCoupling(1)`

ScrExport3DModel

Exports a 3D Model file.	
UI Command:	**Export** > **Export to [HFSS 3D Layout, Q3D Extractor]**.
Syntax:	`obj.ScrExport3DModel <exportTypeName> <outFilePath>`
Parameters:	BSTR exportTypeName (HFSS or Q3D) BSTR outFilePath (including extension)
Return Value:	BOOL • **0** – Failure • **1** – Success
VB Example:	`obj.ScrExport3DModel("Q3D", "C:\SampleFiles\test.aedt")`
IPY Example:	`oDoc.ScrExport3DModel ('Q3D','C:\SampleFiles\test.aedt')`

ScrExportAnf

Exports an Ansys Neutral File (*.anf).	
UI Command:	**Export** > **ANF**.
Syntax:	`obj.ScrExportAnf <filePath>`
Parameters:	BSTR filePath (including extension)
Return Value:	BOOL • **0** – Failure • **1** – Success
VB Example:	`obj.ScrExportAnf("D:\Tests\testExport.anf")`
IPY Example:	`oDoc.ScrExportAnf('D:\Tests\testExport.anf')`

ScrExportComponentFile

Exports a component file (*.cmp) to be paired with an exported ANF file.	
UI Command:	**Export** > **Component File**.
Syntax:	`obj.ScrExportComponentFile <filePath>`
Parameters:	BSTR filePath (including extension)
Return Value:	BOOL • **0** – Failure • **1** – Success
VB Example:	`obj.ScrExportComponentFile ("D:\Tests\testExport.cmp")`
IPY Example:	`oDoc.ScrExportComponentFile ('D:\Tests\testExport.cmp')`

ScrExportCpaSimReport (IronPython)

Exports a CPA simulation report in HTML format.	
UI Command:	**Results** > **RLGC** > **[Simulation Name]** > **Export Simulation Report**.
Syntax:	`obj.ScrExportCpaSimReport(<simName>, <filePath>)`
Parameters:	BSTR simName BSTR filePath
Return Value:	INT: • **0** – Success • **1** – Cannot find CPA simulation by simName provided. • **2** – Export failed.
VB Example:	`obj.ScrExportCpaSimReport("CPA Sim 1", "C:\Directory\")`
IPY Example:	`oDoc.ScrExportCpaSimReport('CPA Sim 1','C:\Directory')`

ScrExportDcPowerDataToIcepak

Enables or disables generation of DC power data for use in thermal simulations.	
UI Command:	None.
Syntax:	`obj.ScrExportDcPowerDataToIcepak(<exportPowerData>)`
Parameters:	BOOL exportPowerData (True = enable; False = disable)
Return Value:	None.
VB Example:	`obj.ScrExportDcPowerDataToIcepak True`
IPY Example:	`oDoc.ScrExportDcPowerDataToIcepak(True)`

ScrExportDcPowerTree

Exports a DC power tree of a previously run DC simulation.	
UI Command:	**Results** > **DC IR** > **[Simulation Name]** > **Export Power Tree**.
Syntax:	`obj.ScrExportDcPowerTree(<simName>, <thresholds_csv_file>, <output_image_file>)`
Parameters:	BSTR simName (DC IR simulation must already be completed) BSTR thresholds_csv_file (path to CSV file containing voltage and current threshold values for every node; include file extension) BSTR output_image_file (path to the file the tree will be exported to; include file extension)
Return Value:	INT: • **0** – Success • **1** – Cannot find DC simulation by simName provided. • **2** – Export failed.
VB Example:	`obj.ScrExportDcPowerTree ("DC Drop 1", "D:\thresholds.csv", "D:\pwrtree.png")`
IPY Example:	`oDoc.ScrExportDcPowerTree('DC Drop 1', 'D:\thresholds.csv', 'D:\pwrtree.png')`

ScrExportDcSimReport

Exports a simulation report for a previously run DC simulation. **To specify additional options, see: ScrExportDcSimReportColorBarProperties, ScrExportDcSimReportOptions, ScrExportDcSimReportScaling, and ScrExportDcSimReportUnits.**	
UI Command:	**Results** > **DC IR** > **[Simulation Name]** > **Export Report**.
Syntax:	`obj.ScrExportDcSimReport(<simName>, <backgroundColor>, <htmReportFilenameWithPath>)`
Parameters:	BSTR simName (DC IR simulation must already be completed) BSTR backgroundColor ('black', 'white', or '', where empty string uses the current SIwave background colors) BSTR htmReportFilenameWithPath
Return Value:	INT: • **0** – Success • **1** – Cannot find DC simulation by simName provided. • **2** – Export failed.
VB Example:	`obj.ScrExportDcSimReport("DC IR Sim 1", "white", "C:\Project1\report.htm")`
IPY Example:	`oDoc.ScrExportDcSimReport('DC IR Sim 1', 'white', 'C:\Project1\report.htm')`

ScrExportDcSimReportColorBarProperties

Used before ScrExportDcSimReport to set color bar properties for DC report generation.	
UI Command:	**Advanced** > **Color Scale**. Select **Color Bar Properties**.
Syntax:	`obj.ScrExportDcSimReportColorBarProperties (<numDiv>, <numDigit>, <bFlipColorScale>, <bWhiteBeyondMinMax>)`
Parameters:	INT numDiv INT numDigit BOOL bFlipColorScale BOOL bWhiteBeyondMinMax
Return Value:	INT: • **0** – Success

Used before ScrExportDcSimReport to set color bar properties for DC report generation.	
	• **Else** – Failure
VB Example:	`obj.ScrExportDcSimReportColorBarProperties (14,3,False,True)`
IPY Example:	`oDoc.ScrExportDcSimReportColorBarProperties (14,3,False,True)`

ScrExportDcSimReportOptions

Used before ScrExportDcSimReport to set options for DC report generation.	
UI Command:	**Results** > **DC IR** > **[Simulation Name]** > **Export Report**. Specify options.
Syntax:	`obj.ScrExportDcSimReportOptions(<showDevices>, <filtersXmlFilenameWithPath>)`
Parameters:	BOOL showDevices (True to show devices and device names for devices with pins on signal nets that are involved in the current simulation) BSTR filtersXmlFilenameWithPath (full path to XML file containing filters for current, voltage or power plots on selected layers and nets)
Return Value:	INT: • **0** – Success • **Else** – Failure
VB Example:	`obj.ScrExportDcSimReportOptions(TRUE, "C:\Projects\filter.xml")`
IPY Example:	`oDoc.ScrExportDcSimReportOptions(True, 'C:\Projects\filter.xml')`

ScrExportDcSimReportScaling

Used before ScrExportDcSimReport to set scaling options for DC report generation.	
UI Command:	None.
Syntax:	`obj.ScrExportDcSimReportScaling(<layerName>, <plotType>, <minVal>, <maxVal>, <bLogScale>)`
Parameters:	BSTR layerName (specify a layer name, "All", or "All Bondwires/Vias") BSTR plotType ("Current Density", "Voltage", "Power Density", "Via Current", or "All") DOUBLE minVal DOUBLE maxVal

Used before ScrExportDcSimReport to set scaling options for DC report generation.	
	BOOL bLogScale (True = Log; False = Linear) **Use -1 for both minVal & maxVal to set the plot range back to original and also to set bLogScale independently without affecting the range.
Return Value:	INT: • **0** – Success • **1** – layerName value is unacceptable. • **2** – plotType value is unacceptable.
VB Example:	`obj.ScrExportDcSimReportScaling ("All","Voltage",0.00,50.0,TRUE)`
IPY Example:	`oDoc.ScrExportDcSimReportScaling ('All','Voltage',0.00,50.0,True)`

ScrExportDcSimReportUnits

Used before ScrExportDcSimReport to set units for DC report generation.	
UI Command:	**Results** > **DC IR** > **[Simulation Name]** > **Export Report**.
Syntax:	`obj.ScrExportDcSimReportUnits(<curDenUnits>, <vltUnits>, <pwrDenUnits>)`
Parameters:	BSTR curDenUnits ("A/um^2", "A/mil^2", "A/mm^2", "A/cm^2", "A/in^2", or "A/m^2") BSTR vltUnits ("uV", "mV", or "V") BSTR pwrDenUnits ("W/um^3", "W/mil^3", "W/mm^3", "W/cm^3", "W/in^3", or "W/m^3")
Return Value:	INT: • **0** – Success • **1** – curDenUnits value is unacceptable. • **2** – vltUnits value is unacceptable. • **3** – curDenUnits value is unacceptable.
VB Example:	`obj.ScrExportDcSimReportUnits ("A/um^2","V","W/um^3")`
IPY Example:	`oDoc.ScrExportDcSimReportUnits ('A/um^2','V','W/um^3')`

ScrExportElementData

Exports the element data from a DC simulation report.	
UI Command:	**Results** > **DC IR** > **[Simulation Name]** > **Export Element Data**.
Syntax:	`obj.ScrExportElementData(<simName>, <fileName>, <tabTitle>)`
Parameters:	BSTR simName BSTR fileName BSTR tabTitle
Return Value:	INT: • **0** – Success • **3** – Cannot find DC simulation by simName provided. • **Else** – Export failed.
VB Example:	`obj.ScrExportElementData("DC Script Sim", "D:\Tests\viaReportOut.txt", "Vias")`
IPY Example:	`oDoc.ScrExportElementData('DC Script Sim', 'D:\Tests\viaReportOut.txt', 'Vias')`

ScrExportEmiScanReport

Generates an HTML EMI scanner report, with images of violations.	
UI Command:	**Results** > **EMI Scanner** > **[SimulationName]** > **Export Report with Images**.
Syntax:	`obj.ScrExportEmiScanReport(<simName>, <reportPath>)`
Parameters:	BSTR simName BSTR reportPath (include file extension)
Return Value:	INT: • **0** – Success • **1** – Cannot find simulation by simName provided. • **2** – Export failed.
VB Example:	`obj.ScrExportEmiScanReport("EMI Scan Sim 1", "D:\AutomationTest\EMI.htm")`
IPY Example:	`oDoc.ScrExportEmiScanReport("EMI Scan Sim 1", "D:\AutomationTest\EMI.htm")`

ScrExportIcepakProject

Exports an Icepak project for standalone use. A DC simulation may be specified to provide power data to the Icepak project.	
UI Command:	**Export** > **Icepak**.
Syntax:	`obj.ScrExportIcepakProject(<projectPath>, <dcSimName>)`
Parameters:	BSTR projectPath BSTR dcSimName (leave string empty to omit)
Return Value:	INT • **0** – Success • **1** – Project directory could not be created. • **2** – An EDB could not be exported for the project. • **3** – Error generating the project import script. • **4** – Specified DC simulation could not be found OR failure generating Icepak launch script.
VB Example:	`obj.ScrExportIcepakProject("D:\icepakProj","")`
IPY Example:	`oDoc.ScrExportIcepakProject('D:\icepakProj', 'DC Sim 1')`

ScrExportIcepakSimReport

Generates an HTML report file containing Icepak thermal plots taken at various elevations. **To specify additional options, see: ScrExportIcepakSimReportColorBarProperties, ScrExportIcepakSimReportScaling, and ScrExportIcepakSimReportUnits.**	
UI Command:	**Results** > **Temperature** > **Export Report**.
Syntax:	`obj.ScrExportIcepakSimReport(<simName>, <fileNameWithPath>)`
Parameters:	BSTR simName BSTR fileNameWithPath (include file extension)
Return Value:	INT: • **0** – Success • **1** – Cannot find Icepak simulation by simName provided. • **2** – Export failed.
VB Example:	`obj.ScrExportIcepakSimReport("Icepak Script Sim", "D:\AutomationTest\ScriptReportTest.htm")`
IPY Example:	`oDoc.ScrExportIcepakSimReport("Icepak Script Sim", "D:\AutomationTest\ScriptReportTest.htm")`

ScrExportIcepakSimReportColorBarProperties

Used before ScrExportIcepakSimReport to set color bar properties for Icepak report generation.	
UI Command:	**Advanced** > **Color Scale**. Select **Color Bar Properties**.
Syntax:	`obj.ScrExportIcepakSimReportColorBarProperties (<numDiv>, <numDigit>, <bFlipColorScale>, <bWhiteBeyondMinMax>)`
Parameters:	INT numDiv INT numDigit BOOL bFlipColorScale BOOL bWhiteBeyondMinMax
Return Value:	INT: • **0** – Success • **Else** – Failure
VB Example:	`obj.ScrExportIcepakSimReportColorBarProperties (14,3,False,True)`

Used before ScrExportIcepakSimReport to set color bar properties for Icepak report generation.	
IPY Example:	`oDoc.ScrExportIcepakSimReportColorBarProperties (14,3,False,True)`

ScrExportIcepakSimReportScaling

Used before ScrExportIcepakSimReport to set scaling options for Icepak report generation.	
UI Command:	None.
Syntax:	`obj.ScrExportIcepakSimReportScaling(<minVal>, <maxVal>, <bLogScale>)`
Parameters:	DOUBLE minVal DOUBLE maxVal BOOL bLogScale (True = Log; False = Linear) **Use -1 for both minVal & maxVal to set the plot range back to original and also to set bLogScale independently without affecting the range.
Return Value:	INT: • **0** – Success • **Else** – Failure
VB Example:	`obj.ScrExportIcepakSimReportScaling (0.00,50.0,FALSE)`
IPY Example:	`oDoc.ScrExportIcepakSimReportScaling (0.00,50.0,False)`

ScrExportIcepakSimReportUnits

Used before ScrExportIcepakSimReport to set units for Icepak report generation.	
UI Command:	None.
Syntax:	`obj.ScrExportIcepakSimReportUnits(<tempUnits>)`
Parameters:	BSTR tempUnits ("C", "F", or "K")
Return Value:	INT: • **0** – Success • **1** – tempUnits value is unacceptable.
VB Example:	`obj.ScrExportIcepakSimReportUnits("F")`
IPY Example:	`oDoc.ScrExportIcepakSimReportUnits('F')`

ScrExportLayerStackup

Exports the current layer stackup to a specified file (*.stk OR *.xml).	
UI Command:	**Export** > **Layer Stackup**, OR **Export** > **Layer Stackup XML**.
Syntax:	`obj.ScrExportLayerStackup(<outputFileName>)`
Parameters:	BSTR outputFileName (include extension)
Return Value:	None.
VB Example:	`obj.ScrExportLayerStackup("C:\Documents\stack.xml")`
IPY Example:	`oDoc.ScrExportLayerStackup ('C:\Documents\stack.stk')`

ScrExportNamedSimToTouchstone

Exports SYZ simulation results to a specified touchstone file.	
UI Command:	**Results** > **SYZ** > **[Simulation Name]** > **Export Touchstone File**.
Syntax:	`obj.ScrExportNamedSimToTouchstone(<simType>, <simName>, <file>)`
Parameters:	BSTR simType (syz, psi_syz, or hfss_syz) BSTR simName (previously run simulation) BSTR file (full path for the touchstone file to be generated; do not include file extension)
Return Value:	INT: • **0** – Success • **1** – simType value is unacceptable. • **2** – simName does not exist OR path cannot be created.
VB Example:	`obj.ScrExportNamedSimToTouchstone("hfss_syz", "Sim 1", "D:\results")`
IPY Example:	`oDoc.ScrExportNamedSimToTouchstone('hfss_syz', 'Sim 1', 'D:\results')`

ScrExportNetDelayReport

Exports a net delay report in HTML format.	
UI Command:	**Simulation** > **Signal Net Analyzer**. Click **Export Net Delays** and select appropriate options.
Syntax:	`obj.ScrExportNetDelayReport(<reportPath>, <netNameRegExp>, <lengthUnits>, <delayUnits>, <onlyDieToBall>)`
Parameters:	BSTR reportPath (full file path for report to be exported) BSTR netNameRegExp (regular expression; use '.*' to include all nets) BSTR lengthUnits (pass empty string to use default 'microns') BSTR delayUnits (pass empty string to use default 'ps') BOOL onlyDieToBall (True to enforce **Only include paths from Die pin to BGA solderball** option; else False)
Return Value:	INT • **0** – Success • **1** – Licensing Failure
VB Example:	`obj.ScrExportNetDelayReport ("C:/MyFiles/netdelay.htm", ".*", "", "", True)`
IPY Example:	`oDoc.ScrExportNetDelayReport ('C:/MyFiles/netdelay.htm','.*','','',1)`

ScrExportSettingsFile

Exports an SIwave Settings file (*.sef). **Use ScrExportSettingsFileSetOptions to specify additional options.**	
UI Command:	**Export** > **Settings File**.
Syntax:	`obj.ScrExportSettingsFile(<filePath>)`
Parameters:	BSTR filePath (full path)
Return Value:	BOOL: • **0** – Failure • **1** – Success
VB Example:	`obj.ScrExportSettingsFile "C:\Path\my_settings.sef"`
IPY Example:	`oDoc.ScrExportSettingsFile('C:/Path/my_settings.sef')`

ScrExportSettingsFileSetOptions

Selects Settings File (*.sef) options used with ScrExportSettingsFile.	
UI Command:	**Export** > **Settings File**. Use the check boxes to select export options.
Syntax:	`obj.ScrExportSettingsFileSetOptions(<optionArray>)`
Parameters:	ARRAY optionArray of BOOLs (1 = enable export; 0 = disable export) INDEX: • 0 – Plane Extents • 1 – Solder Balls and Bumps • 2 – Bondwires • 3 – Net Selections • 4 – Differential Nets and Extended Differential Nets • 5 – Extended Nets • 6 – Padstacks (includes Padshapes)
Return Value:	BOOL: • **0** – Failure • **1** – Success

VB Example:	`Dim expOptions (7)`
	`expOptions(0) = 1`
	`expOptions(1) = 0`
	`expOptions(2) = 1`
	`expOptions(3) = 0`
	`expOptions(4) = 0`
	`expOptions(5) = 1`
	`expOptions(6) = 0`
	`obj.ScrExportSettingsFileSetOptions expOptions`

IPY Example:	`oDoc.ScrExportSettingsFileSetOptions([1,0,1,0,0,1,0])`

ScrExportSNAReport

Exports a Signal Net Analyzer output report in either HTML or CSV format.	
UI Command:	**Simulation** > **Signal Net Analyzer**. From the **Signal Net Analyzer** window, click **Export Table**.
Syntax:	`obj.ScrExportSNAReport(<reportPath>, <netNameRegExp>,`

Exports a Signal Net Analyzer output report in either HTML or CSV format.	
	`<lengthUnits>, <delayUnits>, <onlyDieToBall>,<exportToCSV>)`
Parameters:	BSTR reportPath (full file path for report to be exported) BSTR netNameRegExp (regular expression; use '.*' to include all nets) BSTR lengthUnits (pass empty string to use default 'mm') BSTR delayUnits (pass empty string to use default 'ps') BOOL onlyDieToBall (True to enforce **Only include paths from Die pin to BGA solderball** option; else False) BOOL exportToCSV (True to export to CSV file; False to export to HTML file)
Return Value:	INT • **0** – Success • **1** – Licensing Failure

VB Example:	**HTML File:**
	`obj.ScrExportSNAReport "C:/MyFiles/snaRpt.htm", ".*", "", "", False, False`
	CSV File:
	`obj.ScrExportSNAReport "C:/MyFiles/snaRpt.csv", ".*", "microns", "ns", False, True`

IPY Example:	**HTML File:**
	`oDoc.ScrExportSNAReport ('C:/MyFiles/snaRpt.htm','.*','','',0,0)`
	CSV File:
	`oDoc.ScrExportSNAReport ('C:/MyFiles/snaRpt.csv','.*','microns','ns',0,1)`

ScrExportSyzSimToTouchstone

Exports SYZ simulation results to a specified touchstone file.	
UI Command:	**Results** > **SYZ** > **[Simulation Name]** > **Export Touchstone File**.
Syntax:	`obj.ScrExportSyzSimToTouchstone(<syzSimName>, <touchstonePath>)`
Parameters:	BSTR syzSimName BSTR touchstonePath
Return Value:	INT: • **0** – Success • **1** – Simulation name does not exist OR path cannot be created
VB Example:	`obj.ScrExportSyzSimToTouchstone("SYZ Sweep 1", "C:\sweep1")`
IPY Example:	`oDoc.ScrExportSyzSimToTouchstone('SYZ Sweep 1', 'C:\sweep1')`

ScrExportToTouchstone

Exports the active dataset as SYZ data in touchstone format.	
UI Command:	**Results** > **SYZ** > **[Simulation Name]** > **Export Touchstone File**.
Syntax:	`obj.ScrExportToTouchstone(<filePath>)`
Parameters:	BSTR filePath
Return Value:	INT: • **0** – Success • **1** – SYZ Simulation does not exist OR path cannot be created.
VB Example:	`obj.ScrExportToTouchstone ("D:\Tests\testExport.s2p")`
IPY Example:	`oDoc.ScrExportToTouchstone ('D:\Tests\testExport.s2p')`

ScrExportVprobeData

Exports voltage probe data for a specified AC sweep to a text file.	
UI Command:	None.
Syntax:	`obj.ScrExportVprobeData(<acSimName>, <bstrVprobeFilePath>)`
Parameters:	BSTR acSimName BSTR bstrVprobeFilePath (include file extension *.vpb)
Return Value:	INT • **0** – Success • **1** – AC simulation name does not exist OR file path cannot be created.
VB Example:	`obj.ScrExportVprobeData("AC Sweep 1", "C:\Files\probe_data.vpb")`
IPY Example:	`oDoc.ScrExportVprobeData('AC Sweep 1', 'C:\Files\probe_data.vpb')`

ScrExportXfl

Exports an XFL file.	
UI Command:	**Export** > **XFL**.
Syntax:	`obj.ScrExportXfl(<filePath>)`
Parameters:	BSTR filePath (path for export)
Return Value:	BOOL • **0** – Failure • **1** – Success
VB Example:	`obj.ScrExportXfl("C:\Directory\filename.xfl")`
IPY Example:	`oDoc.ScrExportXfl('C:\Directory\filename.xfl')`

ScrExportZ0ScanReport

Generates an HTML report file containing impedance plots taken for every layer. **To specify additional report options, use ScrExportZ0ScanReportColorBarProperties and ScrExportZ0ScanReportScaling.**	
UI Command:	**Results** > **Impedance Scan** > **Export Report**.
Syntax:	`obj.ScrExportZ0ScanReport(<simName>, <fileNameWithPath>)`
Parameters:	BSTR simName BSTR fileNameWithPath (include file extension)
Return Value:	BOOL • **0** – Failure • **1** – Success
VB Example:	`obj.ScrExportZ0ScanReport("Z0 Scan 1", "D:\AutomationTest\ScriptReportTest.htm")`
IPY Example:	`oDoc.ScrExportZ0ScanReport('Z0 Scan 1', 'D:\AutomationTest\ScriptReportTest.htm')`

ScrExportZ0ScanReportColorBarProperties

Used before ScrExportZ0ScanReport to set color bar properties for impedance report generation.	
UI Command:	**Advanced** > **Color Scale**. Select **Color Bar Properties**.
Syntax:	`obj.ScrExportZ0ScanReportColorBarProperties (<numDiv>, <numDigit>, <bFlipColorScale>, <bWhiteBeyondMinMax>)`
Parameters:	INT numDiv INT numDigit BOOL bFlipColorScale BOOL bWhiteBeyondMinMax
Return Value:	INT: • **0** – Success • **Else** – Failure
VB Example:	`obj.ScrExportZ0ScanReportColorBarProperties (14,3,False,True)`
IPY Example:	`oDoc.ScrExportZ0ScanReportColorBarProperties (14,3,False,True)`

ScrExportZ0ScanReportScaling

Used before ScrExportZ0ScanReport to set scaling options for impedance report generation.	
UI Command:	None.
Syntax:	`obj.ScrExportZ0ScanReportScaling(<minVal>, <maxVal>, <bLogScale>)`
Parameters:	DOUBLE minVal DOUBLE maxVal BOOL bLogScale (True = Log; False = Linear) **Use -1 for both minVal & maxVal to set the plot range back to original and also to set bLogScale independently without affecting the range.
Return Value:	INT: • **0** – Success • **Else** – Failure
VB Example:	`obj.ScrExportZ0ScanReportScaling(0.00,50.0,TRUE)`
IPY Example:	`oDoc.ScrExportZ0ScanReportScaling(0.00,50.0,True)`

ScrFitAll

Fits the design to the modeling workspace.	
UI Command:	**View** > **Fit All**.
Syntax:	`obj.ScrFitAll`
Parameters:	None.
Return Value:	None.
VB Example:	`obj.ScrFitAll()`
IPY Example:	`oDoc.ScrFitAll()`

ScrFitSelection

Fits selected object(s) to the modeling workspace.	
UI Command:	**View** > **Fit Selection**.
Syntax:	`obj.ScrFitSelection`
Parameters:	None.
Return Value:	None.
VB Example:	`obj.ScrFitSelection()`
IPY Example:	`oDoc.ScrFitSelection()`

ScrFitToViewingWindow

Fits selected object(s) to the viewing window. See: GetCurrentViewingWindow.	
UI Command:	None.
Syntax:	`obj.ScrFitToViewingWindow(<unit>, <viewBox>)`
Parameters:	BSTR unit (unit of measure) ARRAY viewBox (structured array containing X,Y coordinates)
Return Value:	None.

VB Example:	`Dim obj`
	
	`Dim viewBox`
	
	`viewBox = obj.GetCurrentViewingWindow("mm")`
	
	`obj.ScrFitToViewingWindow "mm", viewBox`

IPY Example:	`oDoc.ScrFitToViewingWindow('um',['-54','-27','54','27'])`

ScrFwsEnforceCausality

Controls option to check whether s-parameter data is passive.	
UI Command:	**Results** > **SYZ** > **[Simulation Name]** > **Compute FWS Sub-circuit**. Select **Enforce Causality** check box.
Syntax:	`obj.ScrFwsEnforceCausality(<flag>)`
Parameters:	BOOL flag (True to enforce)
Return Value:	None.
VB Example:	`obj.ScrFwsEnforceCausality(true)`
IPY Example:	`oDoc.ScrFwsEnforceCausality(True)`

ScrGenerateConnectionReport

Exports a net connection report to a specified HTML file in the project directory.	
UI Command:	**Advanced** > **Export Connection Report**.
Syntax:	`obj.ScrGenerateConnectionReport(<fileName>)`
Parameters:	BSTR fileName (including file extension)
Return Value:	INT: • **0** – Success • **Else** – Failure
VB Example:	`obj.ScrGenerateConnectionReport("connRpt.html")`
IPY Example:	`oDoc.ScrGenerateConnectionReport('connRpt.html')`

ScrGenerateICDieNetwork

Generates an IC die network.	
UI Command:	**Tools** > **Create IC Die Network**. Set options in **IC Die Network Generation** window.
Syntax:	`obj.ScrGenerateICDieNetwork (<icPartName>, <refDes>, <net>, <networkName>, <resVal>, <useStarPattern>, <capVal>, <esr>, <toNet>, <useAutoRadius>, <resRadius>, <capRadius>)`
Parameters:	BSTR icPartName BSTR refDes BSTR net BSTR networkName BSTR resVal BOOL useStarPattern (True = star pattern; False = grid pattern) BSTR capVal BSTR esr BSTR toNet BOOL useAutoRadius (True = auto radius; False = specify radius) BSTR resRadius (leave string blank for auto radius) BSTR capRadius (leave string blank for auto radius)
Return Value:	BOOL • **0** – Failure • **1** – Success
VB Example:	`obj.doc.ScrGenerateICDieNetwork("CD90-P2913-1", "U13", "AGND", "U13_AGND", "3.0mOhm", TRUE, "", "", "", TRUE, "", "")`
IPY Example:	`oDoc.ScrGenerateICDieNetwork('CD90-P2913-1', 'U13', 'AGND', 'U13_AGND', '3.0mOhm', True, '', '', '', True, '', '')`

ScrGetActiveComponentList

Returns list of all active components of a specified type.	
UI Command:	None.
Syntax:	`obj.ScrGetActiveComponentList(<compType>)`
Parameters:	BSTR compType (can be "All", "RLC", or any category shown in the Components workspace, e.g. "Discrete Devices", "Pin Groups")
Return Value:	ARRAY nameList
VB Example:	`obj.ScrGetActiveComponentList("All")`
IPY Example:	`oDoc.ScrGetActiveComponentList('Current Sources')`

ScrGetBondwiresOfBwModel

Returns list of all bondwires associated with the given bondwire model.	
UI Command:	None.
Syntax:	`obj.ScrGetBondwiresOfBwModel(<modelName>)`
Parameters:	BSTR modelName
Return Value:	ARRAY bwList (list of bondwires)
VB Example:	`obj.ScrGetBondwiresOfBwModel("Z1-MT8530-LOOP1")`
IPY Example:	`oDoc.ScrGetBondwiresOfBwModel('Z1-MT8530-LOOP1')`

ScrGetBwModelNameList

Returns list of all bondwire models.	
UI Command:	None.
Syntax:	`obj.ScrGetBwModelNameList()`
Parameters:	None.
Return Value:	ARRAY bwModelNameList (list of bondwire models)
VB Example:	`Dim bwModelNameList = obj.ScrGetBwModelNameList()`
IPY Example:	`oDoc.ScrGetBwModelNameList()`

ScrGetCktElemTerminalNetNames

Returns names of the two nets to which a specified circuit element is connected.	
UI Command:	None.
Syntax:	`obj.ScrGetCktElemTerminalNetNames(<name>, <type>, <pnet>, <nnet>)`
Parameters:	BSTR name (the circuit element Reference Designator) BSTR type ("cap", "ind", "res", "port", "vprobe", "csource" or "vsource") ARRAY pnet ARRAY nnet
Return Value:	INT • **0** – Success • **1** – Either name OR type is not specified. • **2** – Type is unrecognized. • **3** – Name cannot be found. • **4** – Circuit element's positive terminal is floating (has not been connected to a net). • **5** – Circuit element's negative terminal is floating (has not been connected to a net).
VB Example:	`obj.ScrGetCktElemTerminalNetNames ("C1","cap",pnet,nnet)`
IPY Example:	`oDoc.ScrGetCktElemTerminalNetNames ('C1','cap',pnet,nnet)`

ScrGetComponentList

Returns an array containing names of all parts of the specified type.	
UI Command:	None.
Syntax:	`obj.ScrGetComponentList(<compType>)`
Parameters:	BSTR compType ("all", "rlc", "capacitors", "inductors", "resistors", "ports", "voltage probes", "current sources", "voltage sources", "integrated circuits", "input/output", or "discrete devices"; separate using a comma to specify more than one type)
Return Value:	ARRAY componentList
VB Example:	`obj.ScrGetComponentList("rlc, ports, integrated circuits")`
IPY Example:	`oDoc.ScrGetComponentList('rlc, ports, integrated circuits')`

ScrGetCurrentViewingWindow

Returns the position of the viewing window.	
UI Command:	None.
Syntax:	`obj.ScrGetCurrentViewingWindow(<units>)`
Parameters:	BSTR units
Return Value:	ARRAY containing X,Y coordinates. Results can be used with ScrFitToViewingWindow.
VB Example:	`obj.ScrGetCurrentViewingWindow("um")`
IPY Example:	`oDoc.ScrGetCurrentViewingWindow('um')`

ScrGetDcConnectedNets

Returns an array containing all nets and RLCs that form a connection to the specified net.	
UI Command:	None.
Syntax:	`obj.ScrGetDcConnectedNets(<netNameList>, <nets>, <cktElems>)`
Parameters:	ARRAY netNameList ARRAY nets ARRAY cktElems
Return Value:	INT
VB Example:	`obj.ScrGetDcConnectedNets(netNameList, nets, cktElems)`
IPY Example:	`oDoc.ScrGetDcConnectedNets(netNameList, nets, cktElems)`

ScrGetDcThermalDataDir

Sets variable thermalDataDirBstr to the path of the DC Thermal directory.	
UI Command:	None.
Syntax:	`obj.ScrGetDcThermalDataDir(<simName>, <thermalDataDirBstr>)`
Parameters:	BSTR simName (DC simulation name) ARRAY thermalDataDirBstr (empty)
Return Value:	BOOL: • **0** – Failure • **1** – Success
VB Example:	`Dim thermalDataDirBstr()` `obj.ScrGetDcThermalDataDir("DC IR Sim 1",thermalDataDirBstr)`
IPY Example:	`thermalDataDirBstr = []` `oDoc.ScrGetDcThermalDataDir('DC IR Sim 1',thermalDataDirBstr)`

ScrGetDesignBoundingBox

Assigns the design's bounding box, in specified units, to variable designBBox.	
UI Command:	None.
Syntax:	`obj.ScrGetDesignBoundingBox(<units>, <designBBox>)`
Parameters:	BSTR units ('um', 'in', 'm', 'microns', 'mils', 'mm', 'cm', 'inches', or 'meters') ARRAY designBBox (empty)
Return Value:	ARRAY designBBox populates with 3D bounding box coordinates, represented by corner points (x1,y1,z1) and (x2, y2, z2). BOOL: • **1** – Success • **0** – Failure
VB Example:	`Dim designBBox()` `obj.ScrGetDesignBoundingBox("mm",designBBox)`
IPY Example:	`designBBox= []` `oDoc.ScrGetDesignBoundingBox('mm',designBBox)`

ScrGetDieLayerName

Returns the layer on which a specified die exists.	
UI Command:	None.
Syntax:	`obj.ScrGetDieLayerName(<dieName>)`
Parameters:	BSTR dieName
Return Value:	BSTR layer name
VB Example:	`obj.ScrGetDieLayerName "DIE"`
IPY Example:	`oDoc.ScrGetDieLayerName('DIE')`

ScrGetDieNameList

Returns an array containing all die names.	
UI Command:	None.
Syntax:	`obj.ScrGetDieNameList()`
Parameters:	None.
Return Value:	ARRAY dieNameList
VB Example:	`obj.ScrGetDieNameList()`
IPY Example:	`oDoc.ScrGetDieNameList()`

ScrGetLayerMaterial

Returns the material of a specified layer.	
UI Command:	None.
Syntax:	`obj.ScrGetLayerMaterial(<layerName>)`
Parameters:	BSTR layerName
Return Value:	BSTR materialName
VB Example:	`obj.ScrGetLayerMaterial("BOTTOM")`
IPY Example:	`oDoc.ScrGetLayerMaterial('BOTTOM')`

ScrGetLayerNameList

Returns an array containing names of all layers in the project.	
UI Command:	None.
Syntax:	`obj.ScrGetLayerNameList()`
Parameters:	None.
Return Value:	ARRAY layerNameList
VB Example:	`obj.ScrGetLayerNameList()`
IPY Example:	`oDoc.ScrGetLayerNameList()`

ScrGetLayerThickness

Returns the thickness of a specified layer.	
UI Command:	None.
Syntax:	`obj.ScrGetLayerThickness(<layerName>)`
Parameters:	BSTR layerName
Return Value:	DOUBLE layer thickness
VB Example:	`obj.ScrGetLayerThickness "TOP_LAYER"`
IPY Example:	`oDoc.ScrGetLayerThickness('TOP_LAYER')`

ScrGetLayerType

Returns a specified layer's type.	
UI Command:	None.
Syntax:	`obj.ScrGetLayerType(<layerName>)`
Parameters:	BSTR layerName
Return Value:	INT • **0** – Dielectric • **1** – Metal • **2** – Wirebond
VB Example:	`obj.ScrGetLayerType("SURFACE_LAYER")`
IPY Example:	`oDoc.ScrGetLayerType('SURFACE_LAYER')`

ScrGetLayoutLengthUnit

Returns the current unit of measure.	
UI Command:	None.
Syntax:	`obj.ScrGetLayoutLengthUnit()`
Parameters:	None.
Return Value:	BSTR unit
VB Example:	`obj.ScrGetLayoutLengthUnit()`
IPY Example:	`oDoc.ScrGetLayoutLengthUnit()`

ScrGetMetalLayerFillerMaterial

Returns a specified metal layer's filler material.	
UI Command:	None.
Syntax:	`obj.ScrGetMetalLayerFillerMaterial(<layerName>)`
Parameters:	BSTR layerName
Return Value:	BSTR materialName
VB Example:	`obj.ScrGetMetalLayerFillerMaterial("LAYER3")`
IPY Example:	`oDoc.ScrGetMetalLayerFillerMaterial('LAYER3')`

ScrGetNetlistOfBondwireProfile

Returns a list of nets associated with a given bondwire profile	
UI Command:	None.
Syntax:	`obj.ScrGetNetlistOfBondwireProfile(<profileName>)`
Parameters:	BSTR profileName
Return Value:	ARRAY netNameList
VB Example:	`obj.ScrGetNetlistOfBondwireProfile("Bw Profile")`
IPY Example:	`oDoc.ScrGetNetlistOfBondwireProfile('Bw Profile')`

ScrGetNetNameList

Returns a list of nets in the project.	
UI Command:	None.
Syntax:	`obj.ScrGetNetNameList()`
Parameters:	None.
Return Value:	ARRAY netNameList
VB Example:	`obj.ScrGetNetNameList()`
IPY Example:	`oDoc.ScrGetNetNameList()`

ScrGetNetsAndCktElemsBetweenComponents

Returns a list of all power/ground nets and RLCs that form a connection between specified components.	
UI Command:	None.
Syntax:	`obj.ScrGetNetsAndCktElemsBetweenComponents (<partName1>, <refDes1>, <partName2>, <refDes2>, <nets>, <cktElems>)`
Parameters:	BSTR partName1 BSTR refDes1 BSTR partName2 BSTR refDes2 ARRAY nets (empty) ARRAY cktElems (empty)
Return Value:	ARRAY nets (filled) ARRAY cktElems (filled) INT outcome: • **0** – Success • **Else** – Failure
VB Example:	`obj.ScrGetNetsAndCktElemsBetweenComponents ("288DIMMDDR4_EDGE_CONN-BASE", "J1", "DDR4_X4_FBGA78-10X13", "U1", nets2, elems2)`
IPY Example:	`oDoc.ScrGetNetsAndCktElemsBetweenComponents ('288DIMMDDR4_EDGE_CONN-BASE', 'J1', 'DDR4_X4_FBGA78-10X13','U1', nets2, elems2)`

ScrGetNetsAndCktElemsBetweenNets

Returns a list of all nets and RLCs that form a connection between specified components.	
UI Command:	None.
Syntax:	`obj.ScrGetNetsAndCktElemsBetweenNets(<net1>, <net2>, <refDes2>, <nets>, <cktElems>)`
Parameters:	BSTR net1 BSTR net2 ARRAY nets (empty) ARRAY cktElems (empty)
Return Value:	ARRAY nets (filled) ARRAY cktElems (filled) INT outcome: • **0** – Success • **Else** – Failure
VB Example:	`obj.ScrGetNetsAndCktElemsBetweenNets("VDD", "GND", nets, elems)`
IPY Example:	`oDoc.ScrGetNetsAndCktElemsBetweenNets('VDD', 'GND', nets, elems)`

ScrGetPadstackNameList

Returns a list of all padstack names in the project.	
UI Command:	None.
Syntax:	`obj.ScrGetPadstackNameList()`
Parameters:	None.
Return Value:	ARRAY padstackNameList
VB Example:	`obj.ScrGetPadstackNameList()`
IPY Example:	`oDoc.ScrGetPadstackNameList()`

ScrGetPinGroupNameList

Returns a list of all pin group names on specified part(s) or in the entire project.	
UI Command:	None.
Syntax:	`obj.ScrGetPinGroupNameList(<partName>,<refDes>)`
Parameters:	BSTR partName (leave empty to search entire project) BSTR refDes (leave empty to search entire project)
Return Value:	ARRAY pinGroupNameList
VB Example:	`obj.ScrGetPinGroupNameList("288DIMMDDR4_EDGE_CONN-BASE", "J1")`
IPY Example:	`oDoc.ScrGetPinGroupNameList()`

ScrGetPinPadstackName

Returns the padstack name associated with a specified pin.	
UI Command:	None.
Syntax:	`obj.ScrGetPinPadstackName(<partName>, <refDes>, <pinName>)`
Parameters:	BSTR partName BSTR refDes BSTR pinName
Return Value:	BSTR padstackName
VB Example:	`obj.ScrGetPinPadstackName("DIE", "U1", "394")`
IPY Example:	`oDoc.ScrGetPinPadstackName('DIE','U1','394')`

ScrGetPinsOnNet

Returns a list of identifying information for pins on a specified net.	
UI Command:	None.
Syntax:	`obj.ScrGetPinsOnNet(<netName>, <partName>, <refDes>, <pinNames>, <partNames>, <refDesOut>)`
Parameters:	BSTR netName BSTR partName (use empty string or "any" to disable this filter) BSTR refDes (use empty string or "any" to disable this filter) ARRAY pinNames (empty) ARRAY partNames (empty) ARRAY refDesOut (empty)
Return Value:	ARRAY pinNames (filled) ARRAY partNames (filled) ARRAY refDesOut (filled) BOOL outcome: • **0** – Failure • **1** – Success
VB Example:	`obj.ScrGetPinsOnNet("GND", "ANY", "", pins, parts, refDesList)`
IPY Example:	`oDoc.ScrGetPinsOnNet ('GND', 'ANY', '', pins, parts, refDesList)`

ScrGetPinsOnPart

Returns a list of identifying information for pins on a specified part.	
UI Command:	None.
Syntax:	`obj.ScrGetPinsOnPart(<partName>, <refDes>, <pinNames>, <netNames>)`
Parameters:	BSTR partName BSTR refDes ARRAY pinNames (empty) ARRAY netNames (empty)
Return Value:	ARRAY pinNames (filled) ARRAY netNames (filled) BOOL outcome: • **0** – Failure • **1** – Success
VB Example:	`obj.ScrGetPinsOnPart("T1_A", "U1", pins, nets)`
IPY Example:	`oDoc.ScrGetPinsOnPart('T1_A', 'U1', pins, nets)`

ScrGetPwrGndNetNameList

Returns a list of Power/Ground nets in the project.	
UI Command:	None.
Syntax:	`obj.ScrGetPwrGndNetNameList()`
Parameters:	None.
Return Value:	ARRAY pwrGndNetNameList
VB Example:	`obj.ScrGetPwrGndNetNameList()`
IPY Example:	`oDoc.ScrGetPwrGndNetNameList()`

ScrGetRLCsBetweenNets

Finds RLCs that directly connect any pair of nets from the specified list.	
UI Command:	None.
Syntax:	`obj.ScrGetRLCsBetweenNets(<netNameList>, <includeR>, <includeL>, <includeC>, <ctkElems>)`
Parameters:	ARRAY netNameList BOOL includeR BOOL includeL BOOL includeC ARRAY cktElems (empty)
Return Value:	ARRAY cktElems (filled) INT outcome: • **0** – Success • **Else** – Failure
VB Example:	`obj.ScrGetRLCsBetweenNets(netsIn, FALSE, FALSE, TRUE, elems1)`
IPY Example:	`oDoc.ScrGetRLCsBetweenNets(netsIn1, False, False, True, elems1)`

ScrGetStackupLayerThickness

Returns the thickness of a specified layer.	
UI Command:	None.
Syntax:	`obj.ScrGetStackupLayerThickness(<layerName>)`
Parameters:	BSTR layerName
Return Value:	BSTR layer thickness
VB Example:	`obj.ScrGetStackupLayerThickness("Layer 1")`
IPY Example:	`oDoc.ScrGetStackupLayerThickness('Layer 1')`

ScrGetUniqueSimulationName

Generates a unique simulation name based on a given simulation type.	
UI Command:	None.
Syntax:	`obj.ScrGetUniqueSimulatioName(<simType>)`
Parameters:	BSTR simType ("ac", "dc", "eigen", "ff", "nf", "syz", or "hfss_syz")
Return Value:	BSTR uniqueSimName
VB Example:	`obj.ScrGetUniqueSimulationName("syz")`
IPY Example:	`oDoc.ScrGetUniqueSimulationName('nf')`

ScrImportAnf

Imports a specified Ansys Neutral File (*.anf)	
UI Command:	**Import** > **ANF**.
Syntax:	`obj.ScrImportAnf(<anfBstr>)`
Parameters:	BSTR anfBstr (file path)
Return Value:	BOOL • **0** – Failure • **1** – Success
VB Example:	`obj.ScrImportAnf("C:\anfFiles\design1.anf")`
IPY Example:	`oDoc.ScrImportAnf('C:\anfFiles\design1.anf')`

ScrImportCapacitorDeratingTable

Assigns capacitor derating attributes and opens modified SIwave file.	
UI Command:	**Simulation** > **Options**. Under **DC Bias**, click **Open**.
Syntax:	`obj.ScrImportCapacitorDeratingTable (<bDeratingTablePath>, <errors>)`
Parameters:	BSTR bDeratingTablePath ARRAY errors
Return Value:	INT number of capacitors in the derating file that could not be found in the design. • **-1** – Failure
VB Example:	`obj.ScrImportCapacitorDeratingTable (objShell.CurrentDirectory + "\derating_table.csv",errors)`
IPY Example:	`oDoc.ScrImportCapacitorDeratingTable ('C:\\csvfiles\\derating_table.csv','errors')`

ScrImportComponentFile

Imports a specified component file (*.cmp)	
UI Command:	**Import** > **Component File**.
Syntax:	`obj.ScrImportComponentFile(<cmpBstr>)`
Parameters:	BSTR cmpBstr (file path)
Return Value:	BOOL • **0** – Failure • **1** – Success
VB Example:	`obj.ScrImportComponentFile ("C:\ComponentFiles\design1.cmp")`
IPY Example:	`oDoc.ScrImportComponentFile ('C:\ComponentFiles\design1.cmp')`

ScrImportComponentMapFile

Imports a specified component mapping file (*.cmp)	
UI Command:	**Import** > **Component File**.
Syntax:	`obj.ScrImportComponentMapFile(<fileName>)`
Parameters:	BSTR fileName (full path)
Return Value:	BOOL • **0** – Failure • **1** – Success
VB Example:	`obj.ScrImportComponentMapFile ("C:\ComponentFiles\design1.cmp")`
IPY Example:	`oDoc.ScrImportComponentMapFile ('C:\ComponentFiles\design1.cmp')`

ScrImportCpaSimulationOptions

Loads an SIwave Simulation Settings file (*.sws).	
UI Command:	Click **Simulation**. In the **CPA** area, select **Options**. In the options window, click **Import Settings**.
Syntax:	`obj.ScrImportCpaSimulationOptions(<filePath>)`
Parameters:	BSTR filePath (full path)
Return Value:	BOOL • **0** – Failure • **1** – Success

VB Example:	`Dim app, doc, outcome, objShell`
	`Set objShell = WScript.CreateObject ("WScript.Shell")`
	`Set app = CreateObject("SIwave.Application.2023.1")`
	`Set doc = app.OpenProject(objShell.CurrentDirectory + "\test_project.siw")`
	`doc.ScrImportCpaSimulationOptions (objShell.CurrentDirectory + "\simulation_settings.sws")`
	`doc.ScrSaveProjectAs(objShell.CurrentDirectory + "\test_project_w_new_sim_settings.siw")`

IPY Example:	`oDoc.ScrImportCpaSimulationOptions('C:\Path\simulation_settings.sws')`

ScrImportCpmOrPloc

Imports a CPM or PLOC file.	
UI Command:	**Import** > **Ansys CPM/PLOC File**.
Syntax:	`obj.ScrImportCpmOrPloc(<plocFileName>, <partName>, <refName>, <controlFileName>)`
Parameters:	BSTR plocFileName (can also be CPM file name) BSTR partName BSTR refName (Reference Designator) BSTR controlFileName (containing keywords listed below; for auto connection, use empty string)
Return Value:	BOOL • **0** – Failure • **1** – Success
VB Example:	`obj.ScrImportCpmOrPloc ("C:\SAMPLEFILES\cpmfile.cpm","CSP_BGA","BGA","")`
IPY Example:	`oDoc.ScrImportCpmOrPloc ('C:\SAMPLEFILES\cpmfile.cpm','CSP_BGA','BGA','')`

Option Keywords and Example Values

DieCenterX -149.997058

DieCenterY 484.129583

FlipDie TRUE

RotationAngle 0

ScalingFactor 0.9

Tolerance 0.0

CreatePorts TRUE

CreateSources FALSE

ScrImportEDB

Imports an EDB folder into a new project.	
UI Command:	**Import** > **Ansys EDB**.
Syntax:	`obj.ScrImportEDB (<folderPath>)`
Parameters:	BSTR filePath
Return Value:	INT: • **0** – Success • **Else** – Failure
VB Example:	`obj.ScrImportEDB "C:\Files\Edb"`
IPY Example:	`oApp.ScrImportEDB('C:\Files\Edb')`

ScrImportGDSII

Imports a GDSII file (*.strm, *.gds) into a new project.	
UI Command:	**Import** > **GDSII**.
Syntax:	`obj.ScrImportGDSII(<filePath>, <controlFilePath>)`
Parameters:	BSTR filePath BSTR controlFilePath (pass an empty string to use the default control file)
Return Value:	INT: • **0** – Success • **1** – Failure generating control file • **2** – Error translating the file
VB Example:	`obj.ScrImportGDSII "C:\Files\MyProject.gds" ""`
IPY Example:	`oApp.ScrImportGDSII('C:\Files\MyProject.gds','')`

ScrImportIPC2581

Imports an IPC2581 file, as well as optional RLC part value and XML control files, into a new project	
UI Command:	**Import** > **IPC2581**.
Syntax:	`obj.ScrImportIPC2581 (<designFile>, <controlFile>, <partFile>)`
Parameters:	BSTR designFile (full path) BSTR controlFile (full path, use empty string for none) BSTR partFile (full path, use empty string for none)
Return Value:	INT: • **0** – Success • **1** – Error translating the file. • **2** – Error reading part mapping file.
VB Example:	`obj.ScrImportIPC2581 "C:\Files\mydesign.cvg" "C:\Files\controlfile.xml" "C:\Files\partfile.dat"`
IPY Example:	`oApp.ScrImportIPC2581('C:\Files\mydesign.cvg', 'C:\Files\controlfile.xml', 'C:\Files\partfile.dat')`

ScrImportLayerStackup

Imports a layer stackup file.	
UI Command:	**Import** > **Layer Stackup**.
Syntax:	`obj.ScrImportLayerStackup(<fileName>)`
Parameters:	BSTR fileName (full path)
Return Value:	BOOL: • **0** – Failure • **1** – Success
VB Example:	`obj.ScrImportLayerStackup ("C:\StackupFiles\stack1.stk")`
IPY Example:	`oDoc.ScrImportLayerStackup('C:\StackupFiles\stack1.stk')`

ScrImportLayerStackupFile

Imports a layer stackup file.	
UI Command:	**Import** > **Layer Stackup**.
Syntax:	`obj.ScrImportLayerStackupFile(<fileName>)`
Parameters:	BSTR fileName (full path)
Return Value:	BOOL: • **0** – Failure • **1** – Success
VB Example:	`obj.ScrImportLayerStackupFile ("C:\StackupFiles\stack1.stk")`
IPY Example:	`oDoc.ScrImportLayerStackupFile ('C:\StackupFiles\stack1.stk')`

ScrImportLayerStackupXML

Imports a layer stackup XML file.	
UI Command:	**Import** > **Layer Stackup XML**.
Syntax:	`obj.ScrImportLayerStackupXML(<filePath>)`
Parameters:	BSTR filePath (full path)
Return Value:	BOOL • **0** – Failure • **1** – Success

VB Example:	`Dim app, doc, outcome, objShell`
	`Set objShell = WScript.CreateObject ("WScript.Shell")`
	`Set app = CreateObject("SIwave.Application.2023.1")`
	`Set doc = app.OpenProject(objShell.CurrentDirectory + "\test_project.siw")`
	`outcome = doc.ScrImportLayerStackupXML (objShell.CurrentDirectory + "\layer_stackup.xml")`

IPY Example:	`oDoc.ScrImportLayerStackupXML ('C:\StackupFiles\stack1.xml')`

ScrImportPmap

Imports a *.pmap file to map locally defined capacitor and inductor part names to s-parameter models from the component library	
UI Command:	None.
Syntax:	`obj.ScrImportPmap(<fileName>)`
Parameters:	BSTR fileName (full path)
Return Value:	BOOL • **0** – Failure • **1** – Success
VB Example:	`obj.ScrImportPmap ("E:\Files\foo.pmap")`
IPY Example:	`oDoc.ScrImportPmap('C:\Files\foo.pmap')`

ScrImportSettingsFile

Loads an SIwave Settings file (*.sef).	
UI Command:	**Import** > **Settings File**.
Syntax:	`obj.ScrImportSettingsFile(<filePath>)`
Parameters:	BSTR filePath (full path)
Return Value:	BOOL: • **0** – Failure • **1** – Success
VB Example:	`obj.ScrImportSettingsFile "C:\Path\my_settings.sef"`
IPY Example:	`oDoc.ScrImportSettingsFile('C:/Path/my_settings.sef')`

ScrImportSIwaveSimulationOptions

Loads an SIwave Simulation Settings file (*.sws).	
UI Command:	Click **Simulation**. In the **SIwave** area, select **Options**. In the options window, click **Import Settings**.
Syntax:	`obj.ScrImportSIwaveSimulationOptions(<filePath>)`
Parameters:	BSTR filePath (full path)
Return Value:	BOOL • **0** – Failure • **1** – Success

VB Example:	`Dim app, doc, outcome, objShell`
	`Set objShell = WScript.CreateObject ("WScript.Shell")`
	`Set app = CreateObject("SIwave.Application.2023.1")`
	`Set doc = app.OpenProject(objShell.CurrentDirectory + "\test_project.siw")`
	`doc.ScrImportSIwaveSimulationOptions (objShell.CurrentDirectory + "\simulation_ settings.sws")`
	`doc.ScrSaveProjectAs(objShell.CurrentDirectory + "\test_project_w_new_sim_settings.siw")`

IPY Example:	`oDoc.ScrImportSIwaveSimulationOptions ('C:\Path\simulation_settings.sws')`

ScrImportXfl

Imports an XFL file.	
UI Command:	**Import** > **XFL File**.
Syntax:	`obj.ScrImportXfl(<filePath>)`
Parameters:	BSTR filePath (full path)
Return Value:	BOOL • **0** – Failure • **1** – Success

VB Example:	`Dim version, doc, outcome`
	`Set app = CreateObject("SIwave.Application.2023.1")`
	`outcome = doc.ScrImportXfl ("C:\Directory\filename.xfl")`
	`doc.ScrSaveProjectAs("C:\Directory\filename.siw")`
	`app.Quit`

IPY Example:	`oDoc.ScrImportXfl('C:\Directory\filename.xfl')`

ScrInterpolateSpectrum

Specifies interpolation option for Frequency Sweep, Far Field, or Near Field simulations. **Important:** Interpolation is active by default and Ansys recommends not changing this setting.	
UI Command:	From the **Simulation** menu, click either **Compute Frequency Sweep**, **Compute Far Field**, or **Compute Near Field**. Then, select or deselect the **Interpolate spectrum at missing frequency points** check box .
Syntax:	`obj.ScrInterpolateSpectrum(<interpolate>)`
Parameters:	BOOL interpolate
Return Value:	None.
VB Example:	`obj.ScrInterpolateSpectrum True`
IPY Example:	`oDoc.ScrInterpolateSpectrum(True)`

ScrLogMessage

Logs a specified message to the Messages window and to the project *.log file.	
UI Command:	None.
Syntax:	`obj.ScrLogMessage(<message>)`
Parameters:	BSTR message
Return Value:	None.
VB Example:	`obj.LogMessage("This is my message.")`
IPY Example:	`oDoc.LogMessage('This is my message.')`

ScrMergeConnectedNets

Merges connected nets in a specified list.	
UI Command:	None.
Syntax:	`obj.ScrMergeConnectedNets(<inNetNameList>)`
Parameters:	ARRAY inNetNameList
Return Value:	ARRAY outNetNameList (if no nets are connected, this will be the same as inNetNameList)

VB Example:	`obj.ScrMergeConnectedNets inNetList, outNetList`
	`inNetNameList=Array("NET_1","NET_2")`
	`outNetNameList=obj.ScrMergeConnectedNets (inNetNameList)`
	`inNetNameList: list of nets to be merged`
	`outNetNameList: list of nets which are resulted from the merging operation`

IPY Example:	`inNetNameList = ['NET_1','NET_2']` `outNetNameList = obj.ScrMergeConnectedNets (inNetNameList)`

ScrNetGetLength

Computes the shortest length between a specified source and sink.	
UI Command:	None.
Syntax:	`Dim length = obj.ScrNetGetLength(<netName>, <sourceName>, <sinkName>)`
Parameters:	BSTR netName BSTR sourceName BSTR sinkName
Return Value:	DOUBLE length (returns 0.0 if net, source, or sink does not exist)
VB Example:	`Dim len = obj.ScrNetGetLength("HOT_INS_DIS", "D1:HOT_INS_DIS:292", "P1:HOT_INS_DIS:E4")`
IPY Example:	`oDoc.ScrNetGetLength('HOT_INS_DIS', 'D1:HOT_INS_DIS:292', 'P1:HOT_INS_DIS:E4')`

ScrNetIsDisjoint

Checks whether a net is disjoint.	
UI Command:	None.
Syntax:	`obj.ScrNetIsDisjoint(<netName>)`
Parameters:	BSTR netName
Return Value:	BOOL • **0** – Not Disjoint • **1** – Disjoint
VB Example:	`obj.ScrNetIsDisjoint("MY-NET")`
IPY Example:	`oDoc.ScrNetIsDisjoint('MY-NET')`

ScrNetIsSelected

Checks whether a net is selected.	
UI Command:	None.
Syntax:	`obj.ScrNetIsSelected(<netName>)`
Parameters:	BSTR netName
Return Value:	BOOL • **0** – Not Selected • **1** – Selected
VB Example:	`obj.ScrNetIsSelected("MY-NET")`
IPY Example:	`oDoc.ScrNetIsSelected('MY-NET')`

ScrNetSeparate

If a net is disjoint, separates it and sets new net names.	
UI Command:	None.
Syntax:	`obj.ScrNetSeparate(<netName>)`
Parameters:	BSTR netName
Return Value:	None.
VB Example:	`obj.ScrNetSeparate("MY-NET")`
IPY Example:	`oDoc.ScrNetSeparate('MY-NET')`

ScrNetSetDummy

Sets a specified net to be a dummy net and appends DUMMY_ to the front of the net name.	
UI Command:	None.
Syntax:	`obj.ScrNetSetDummy(<netName>)`
Parameters:	BSTR netName
Return Value:	None.
VB Example:	`obj.ScrNetSetDummy("MY-NET")`
IPY Example:	`oDoc.ScrNetSetDummy('MY-NET')`

ScrNetSetSelected

Selects or deselects a specified net.	
UI Command:	None.
Syntax:	`obj.ScrNetSetSelected(<netName>, <select>)`
Parameters:	BSTR netName BOOL select (1 = select; 0 = deselect)
Return Value:	None.
VB Example:	`obj.ScrNetSetSelected("MY-NET", 1)`
IPY Example:	`oDoc.ScrNetSetSelected('MY-NET', 1)`

ScrPlaceCircuitElement

Attaches a circuit element of the specified type to the design. The element can be connected between any two points, pins or pin groups.	
UI Command:	**Home** > **Circuit Elements** > **Add [Capacitor / Inductor / Resistor / Port / Voltage Probe / Voltage Source / Current Source]**.
Syntax:	`obj.ScrPlaceCircuitElement(<givenElementName>, <givenPartName>, <circuitElementType>, <posTermConnectionType>, <posTermParam1>, <posTermParam2>, <posTermParam3>, <refTermConnectionType>, <refTermParam1>, <refTermParam2>, <refTermParam3>, <capVal>, <indVal>, <resVal>, <refZRe>, <mag>, <phase>)`
Parameters:	BSTR givenElementName (name of circuit element to be created) BSTR givenPartName (part name of circuit element to be created) INT circuitElementType (0 = capacitor; 1 = inductor; 2 = resistor; 3 = port; 4 = current source; 5 = voltage source; 6 = voltage probe) INTposTermConnectionType (0 = attachment to PIN; 1 = attachment to PIN GROUP; 2 = attachment at COORDINATES) BSTR posTermParam1 (partName or x) BSTR posTermParam2 (refDes or y) BSTR posTermParam3 (pinName, groupName, or layerName) INT refTermConnectionType (0 = attachment to PIN; 1 = attachment to PIN GROUP; 2 = attachment at COORDINATES) BSTR refTermParam1 (partName or x) BSTR refTermParam2 (refDes or y) BSTR refTermParam3 (pinName, groupName, or layerName) double capVal (capacitance) double indVal (inductance) double resVal (resistance) double refZRe (impedance) double mag (magnitude) double phase (phase)

Attaches a circuit element of the specified type to the design. The element can be connected between any two points, pins or pin groups.	
	Use an empty string for any parameters that are not applicable.
Return Value:	BOOL: • **0** – Failure • **1** – Success

VB Example:	`' outcome is set to TRUE on success`
	`' Create a Capacitor:`
	`outcome = obj.ScrPlaceCircuitElement "cap_1", "cap_part", 0, 1, "100354431", "Q3", "pinGrp_4", 1, "100349132", "U26", "pinGroup_10", 4.7e-9, 2e-11, 3e-3, 0.0, 0.0, 0.0`
	`' Create a Resistor:`
	`outcome = obj.ScrPlaceCircuitElement "res_1", "res_part", 2, 0, "100354431", "Q3", "2_1", 0, "100349132", "U26", "HS_1", 50.0, 0.0, 0.0, 0.0, 0.0, 0.0`
	`' Create a Voltage Source:`
	`outcome = obj.ScrPlaceCircuitElement "vs_1", "vs_part", 5, 0, "100354431", "Q3", "2_1", 0, "100349132", "U26", "HS_1", 0.0, 0.0, 5e-6, 0.0, 3.0, 180.0`

IPY Example:	`oDoc.ScrPlaceCircuitElement ('cap_1', 'cap_part', 0, 1, '100354431', 'Q3', 'pinGrp_4', 1, '100349132', 'U26', 'pinGroup_10', 4.7e-9, 2e-11, 3e-3, 0.0, 0.0, 0.0)`

ScrPlaceCircuitElementsToNearestRefPin

Creates ports or sources connecting to the nearest reference pins of the given reference terminal net.	
UI Command:	None.
Syntax:	`obj.ScrPlaceCircuitElementsToNearestRefPin (<circuitElementType <val>, <posPartName>, <posUnitName>, <posNetName>, <refPartName>, <refUnitName>, <refNetName>, <newElemList>)`
Parameters:	INT circuitElementType (3 = port; 4 = current source; 5 = voltage source) DOUBLE val BSTR posPartName BSTR posUnitName BSTR posNetName BSTR refPartName BSTR refUnitName BSTR refNetName ARRAY newElemList
Return Value:	INT: • **0** – Success • **Else** – Failure
VB Example:	`Dim result, newElemList result = obj.ScrPlaceCircuitElementsToNearestRefPin(3, 50.0, "SQFP28X28_208", "U1", "ARBLINK", "SQFP28X28_208", "U2", "GND", newElemList)`
IPY Example:	`oDoc.ScrPlaceCircuitElementsToNearestRefPin (3, 50.0, 'SQFP28X28_208', 'U1', 'ARBLINK', 'SQFP28X28_208', 'U2', 'GND', newElemList)`

ScrPlaceFreqDependentSrc

Creates a frequency dependent source.	
UI Command:	None.
Syntax:	`obj.ScrPlaceFreqDependentSrc(<givenElementName>, <circuitElementType>, <posTermConnectionType>, <posTermParam1>, <posTermParam2>, <posTermParam3>, <refTermConnectionType>, <refTermParam1>, <refTermParam2>, <refTermParam3>, <filename>)`
Parameters:	BSTR givenElementName INT circuitElementType (4 = current source; 5 = voltage source) INT posTermConnectionType (0 = pin; 1 = pin group; 2 = coordinate) BSTR posTermParam1 (part name for pins or pin groups; x coordinate for coordinate connections) BSTR posTermParam2 (reference designator for pins or pin groups; y coordinate for coordinate connections) BSTR posTermParam3 (pin name or pin group name; ignored for coordinate connections) INT refTermConnectionType (0 = pin; 1 = pin group; 2 = coordinate) BSTR refTermParam1 (part name for pins or pin groups; x coordinate for coordinate connections) BSTR refTermParam2 (reference designator for pins or pin groups; y coordinate for coordinate connections) BSTR refTermParam3 (pin name or pin group name; ignored for coordinate connections) BSTR filename
Return Value:	BOOL: • **0** – Failure • **1** – Success
VB Example:	`result = doc.ScrPlaceFreqDependentSrc("I1", 4, 2, "1.0", "1.0", "METAL-1", 2, "2.0", "1.0", "METAL-1", "e:\SrcFreqData.txt")`
IPY Example:	`oDoc.ScrPlaceFreqDependentSrc('I1', 4, 2, '1.0', '1.0', 'METAL-1', 2, '2.0', '1.0', 'METAL-1', 'e:\SrcFreqData.txt')`

ScrPlacePortsAcrossRLCs

Creates ports across specified RLCs.	
UI Command:	None.
Syntax:	`obj.ScrPlacePortsAcrossRLCs(<zref>, <rlcName>, <rlcType>, <portsCreated>)`
Parameters:	DOUBLE zref BSTR rlcName BSTR rlcType ('cap' for capacitor; 'ind' for inductor; 'res' for resistor) ARRAY portsCreated (empty)
Return Value:	ARRAY portsCreated (list of port names) INT outcome: • 0 – Success • 1 – Name or type is an empty string • 2 – rlcType is invalid • 3 – Specified RLC could not be found • 4 – Error filling the array portsCreated
VB Example:	`Dim portsCreated()` `obj.ScrPlacePortsAcrossRLCs (50,"C3A3","cap",portsCreated)`
IPY Example:	`portsCreated=[]` `oDoc.ScrPlacePortsAcrossRLCs (50,'C3A3','cap',portsCreated)`

ScrPlacePortsAtPinsOnSelectedNets

Creates ports with specified impedance for each of the pins found in a specified reference net.	
UI Command:	None.
Syntax:	`obj.ScrPlacePortsAtPinsOnSelectedNets(<zref>, <refNetName>, <connectToPinGroup>, <portsCreated>)`
Parameters:	DOUBLE zref (impedance) BSTR refNetName (reference net) BOOL connectToPinGroup ARRAY portsCreated (empty)
Return Value:	ARRAY portsCreated (filled) INT: • **0** – Success • **Else** – Failure
VB Example:	`Dim portsCreated()` `obj.ScrPlacePortsAtPinsOnSelectedNets(50.0, "MY_NET", True, portsCreated)`
IPY Example:	`portsCreated = []` `oDoc.ScrPlacePortsAtPinsOnSelectedNets(50.0, 'MY_NET', 1, portsCreated)`

ScrPlacePortsAtPinsOnSelectedNetsExcludePart

Creates ports between pins on the selected nets and the reference net unless the pins belong to the specified part.	
UI Command:	None.
Syntax:	`obj.ScrPlacePortsAtPinsOnSelectedNetsExcludePart (<zref>, <refNetName>, <partName>, <refDes>, <connectToPinGroup>, <portsCreated>)`
Parameters:	DOUBLE zref BSTR refNetName BSTR partName BSTR refDes BOOL connectToPinGroup ARRAY portsCreated (empty)
Return Value:	ARRAY portsCreated (filled) INT: • **0** – Success • **Else** – Failure
VB Example:	`obj.ScrPlacePortsAtPinsOnSelectedNetsExcludePart (50.0, "GND", "T1_A", "U1", false, ports)`
IPY Example:	`oDoc.ScrPlacePortsAtPinsOnSelectedNetsExcludePart (50.0, 'GND', 'T1_A', 'U1', false, ports)`

ScrPlacePortsAtPinsOnSelectedNetsPinNamesOut

Creates ports between pins on the selected nets and the reference net while outputting the port names and pin names.	
UI Command:	None.
Syntax:	`obj.ScrPlacePortsAtPinsOnSelectedNetsPinNamesOut (<zref>, <refNetName>, <connectToPinGroup>, <portsCreated>, <posPinNames>, <refPinNames>)`
Parameters:	DOUBLE zref BSTR refNetName BOOL connectToPinGroup (True = connect; False = do not connect) ARRAY portsCreated (empty) ARRAY posPinNames (empty) ARRAY refPinNames (empty)
Return Value:	ARRAY portsCreated (filled) ARRAY posPinNames (filled) ARRAY refPinNames (filled) INT: • **0** – Success • **Else** – Failure
VB Example:	`obj.ScrPlacePortsAtPinsOnSelectedNetsPinNamesOut (50.0, "GND", false, ports, posPins, refPins)`
IPY Example:	`oDoc.ScrPlacePortsAtPinsOnSelectedNetsPinNamesOut (50.0, 'GND', false, ports, posPins, refPins)`

ScrPlotResModeVoltageDiff

Generates voltage difference surface plots for the specified resonant mode simulation. **Note:** A layer pair can be specified and, if present, will restrict the voltage difference computation to just these two layers. If the layer pair is omitted, plots will be generated for all possible layer pair combinations.	
UI Command:	None.
Syntax:	`obj.ScrPlotResModeVoltageDiff(<resonantSimName>, <layerA>, <layerB>)`
Parameters:	BSTR resonantSimName BSTR layerA BSTR layerB
Return Value:	INT: • **0** – Success • **Else** – Failure
VB Example:	`obj.ScrPlotResModeVoltageDiff("Resonant Sim 1", "SURFACE", "INNER1" )`
IPY Example:	`oDoc.ScrPlotResModeVoltageDiff('Resonant Sim 1', 'SURFACE', 'INNER1')`

ScrPreserveNetsGivenInFile

Deletes all nets except those specified in a text file. **Note:** The file should contain net names in quotation marks.	
UI Command:	None.
Syntax:	`obj.ScrPreserveNetsGivenInFile(<fileName>)`
Parameters:	BSTR fileName (full path)
Return Value:	None.
VB Example:	`obj.ScrPreserveNetsGivenInFile("C:/My Files/nets.txt")`
IPY Example:	`oDoc.ScrPreserveNetsGivenInFile('C:/My Files/nets.txt')`

ScrReadDCLoopResInfo

Given the name of an SIwave DC simulation, exports a list of sources and a corresponding list of loop resistance values.	
UI Command:	**Results** > **DC IR Drop** > **Loop Resistance Info**.
Syntax:	`obj.ScrReadDCLoopResInfo(<simName>, <sourceNames>, <loopResData>)`
Parameters:	BSTR simName ARRAY sourceNames (empty) ARRAY loopResData (empty)
Return Value:	ARRAY sourceNames (filled) ARRAY loopResData (filled) INT: • **0** – Success • **Else** – Failure

VB Example:	`Dim sourceNames()`
	`Dim loopResData()`
	`obj.ScrReadDCLoopResInfo("DC IR Sim 1", sourceNames, loopResData)`

IPY Example:	`sourceNames = []`
	`loopResData = []`
	`oDoc.ScrReadDCLoopResInfo('DC IR Sim 1', sourceNames, loopResData)`

ScrRestoreResonantModeMinFreq

Restores the minimum suggested resonant mode frequency.	
UI Command:	**Simulation** > **Compute Resonant Modes**. Click **Restore Recommended Minimum Frequency**.
Syntax:	`obj.ScrRestoreResonantModeMinFreq()`
Parameters:	None.
Return Value:	None.
VB Example:	`obj.ScrRestoreResonantModeMinFreq()`
IPY Example:	`oDoc.ScrRestoreResonantModeMinFreq()`

ScrRunDcSimulation

Runs a DC IR simulation.	
UI Command:	**Simulation** > **Compute DC IR**.
Syntax:	`obj.ScrRunDcSimulation(<reprocessGeom>)`
Parameters:	INT reprocessGeom (1 = runs the simulation; 0 = sets up the simulation but does not run it)
Return Value:	INT: • **0** – Success • **Else** – Failure
VB Example:	`obj.ScrRunDcSimulation(1)`
IPY Example:	`oDoc.ScrRunDcSimulation(1)`

ScrRunFarFieldSimulation

Runs a Far Field simulation.	
UI Command:	**Simulation** > **Compute Far Field**.
Syntax:	`obj.ScrRunFarFieldSimulation()`
Parameters:	None.
Return Value:	INT: • **0** – Success • **Else** – Failure
VB Example:	`obj.ScrRunFarFieldSimulation()`
IPY Example:	`oDoc.ScrRunFarFieldSimulation()`

ScrRunFrequencySweepSimulation

Runs a Frequency Sweep simulation.	
UI Command:	**Simulation** > **Compute Frequency Sweeps**.
Syntax:	`obj.ScrRunFrequencySweepSimulation()`
Parameters:	None.
Return Value:	INT: • **0** – Success • **Else** – Failure
VB Example:	`obj.ScrRunFrequencySweepSimulation()`
IPY Example:	`oDoc.ScrRunFrequencySweepSimulation()`

ScrRunIcepakSimulation

Runs an Icepak simulation.	
UI Command:	**Simulation** > **Icepak**.
Syntax:	`obj.ScrRunIcepakSimulation(<icepakSimName>, <dcSimName>)`
Parameters:	BSTR icepakSimName BSTR dcSimName
Return Value:	INT: • **0** – Success • **Else** – Failure
VB Example:	`obj.ScrRunIcepakSimulation("Icepak 4", "DC IR Sim 1")`
IPY Example:	`oDoc.ScrRunIcepakSimulation('Icepak 4', 'DC IR Sim 1')`

ScrRunInducedVoltageSimulation

Runs a Plane Wave Induced Voltage simulation for one frequency with a single incident wave.	
UI Command:	**Simulation** > **Compute Induced Voltage**.
Syntax:	`obj.ScrRunInducedVoltageSimulation(<freq>, <phi>, <theta>, <e0_phi>, <e0_theta>, <magnitude>)`
Parameters:	DOUBLE freq DOUBLE phi (for incident vector in spherical system) DOUBLE theta (for incident vector in spherical system) DOUBLE e0_phi (for polarization vector in XY cartesian system on the orthogonal plane of the incident vector) DOUBLE e0_theta (for polarization vector in XY cartesian system on the orthogonal plane of the incident vector) DOUBLE magnitude
Return Value:	INT: • **0** – Success • **Else** – Failure
VB Example:	`obj.ScrRunInducedVoltageSimulation(1500000, 10, 45, 1, 3, 1)`
IPY Example:	`oDoc.ScrRunInducedVoltageSimulation(1500000, 10, 45, 1, 3, 1)`

ScrRunNearFieldSimulation

Runs a Near Field simulation.	
UI Command:	**Simulation** > **Compute Near Field**.
Syntax:	`obj.ScrRunNearFieldSimulation(<freq>, <computeH>)`
Parameters:	DOUBLE freq(uency) INT computeH (obsolete variable; use either 0 or 1)
Return Value:	INT: • **0** – Success • **Else** – Failure
VB Example:	`obj.ScrRunNearFieldSimulation(5.0, 1)`
IPY Example:	`oDoc.ScrRunNearFieldSimulation(5.0, 0)`

ScrRunResonantModeSimulation

Runs a Resonant Modes simulation.	
UI Command:	**Simulation** > **Compute Resonant Modes**.
Syntax:	`obj.ScrRunResonantModeSimulation()`
Parameters:	None.
Return Value:	INT: • **0** – Success • **Else** – Failure
VB Example:	`obj.ScrRunResonantModeSimulation()`
IPY Example:	`oDoc.ScrRunResonantModeSimulation()`

ScrRunSimulation

Runs the specified simulation.	
UI Command:	**Simulation** > **[Simulation Type]**.
Syntax:	`obj.ScrRunSimulation(<simType>, <simName>)`
Parameters:	BSTR simType (choose from: "ac", "dc", "eigen", "ff", "nf", "psi_ac", "psi_syz", "syz", "hfss_syz", "pdn", "cpa", "icepak", "iv", "emi_scan", "em_mttf", "z0_scan", "crosstalk_scan", "td_crosstalk_scan") BSTR simName
Return Value:	INT: • **0** – Success • **Else** – Failure
VB Example:	`obj.ScrRunSimulation("syz", "SYZ Sweep 1" )`
IPY Example:	`oDoc.ScrRunSimulation('syz', 'SYZ Sweep 1')`

ScrRunSpiceSubcktSimulation

Computes Spice Subcircuit based on a previously run SYZ sweep.	
UI Command:	**Results** > **SYZ** > **[Simulation Name]** > **Compute FWS Sub-circuit**.
Syntax:	`obj.ScrRunSpiceSubcktSimulation()`
Parameters:	None.
Return Value:	INT: • **0** – Success • **Else** – Failure
VB Example:	`obj.ScrRunSpiceSubcktSimulation()`
IPY Example:	`oDoc.ScrRunSpiceSubcktSimulation()`

ScrRunSyzParameterSimulation

Computes an SYZ sweep.	
UI Command:	**Simulation** > **Compute SYZ Parameters**.
Syntax:	`obj.ScrRunSyzParameterSimulation()`
Parameters:	None.
Return Value:	INT: • **0** – Success • **Else** – Failure

Computes an SYZ sweep.	
VB Example:	`obj.ScrRunSyzParameterSimulation()`
IPY Example:	`oDoc.ScrRunSyzParameterSimulation()`

ScrRunValidationCheck

Runs a Validation Check on the current project.	
UI Command:	**Tools** > **Validation Check**.
Syntax:	`obj.ScrRunValidationCheck`
Parameters:	None.
Return Value:	ARRAY results (number of errors, number of warnings)
VB Example:	`obj.ScrRunValidationCheck()`
IPY Example:	`oDoc.ScrRunValidationCheck()`

ScrRunValidationCheckWithOptions

Runs a Validation Check using specified parameters.	
UI Command:	**Tools** > **Validation Check**.
Syntax:	`obj.ScrRunValidationCheckWithOptions(<optionArray>, <simType>)`
Parameters:	ARRAY optionArray (0 - deselects an option, 1 - selects an option) • Self-Intersecting Polygons • Disjoint Nets (Floating Nodes) • DC-Short Errors • Identical/Overlapping Vias • Bondwire Collisions • Illegal Bondwire Connections • Misalignments • Less Than Two Terminals INT simType • **0** – No Associated Simulation • **1** – Resonant Modes • **2** – Frequency Sweep • **3** – SYZ Parameters • **4** – Far Field • **5** – Near Field • **6** – DC Current/Voltage • **7** – PSI AC or SYZ • **8** – 3D Export • **9** – CPA Solution • **10** – Impedance/Crosstalk Scan
Return Value:	None.

VB Example:	`dim optionArray`
	`optionArray=Array("1","1","1","1","1","0","1","0")`
	`obj.ScrRunValidationCheckWithOptions (optionArray,"3")`

IPY Example:	`oDoc.ScrRunValidationCheckWithOptions (['1','1','1','1','1','0','1','0'],'3')`

ScrSanitizeLayout

Runs the Sanitize Layout operation on all Power/Ground nets in the project.	
UI Command:	**Tools** > **Sanitize Layout**.
Syntax:	`obj.ScrSanitizeLayout()`
Parameters:	None.
Return Value:	None.
VB Example:	`obj.ScrSanitizeLayout()`
IPY Example:	`oDoc.ScrSanitizeLayout()`

ScrSanitizeNets

Runs the Sanitize Layout operation on specified nets.	
UI Command:	**Tools** > **Sanitize Layout**.
Syntax:	`obj.ScrSanitizeNets(<netNameList>)`
Parameters:	ARRAY netNameList (containing strings)
Return Value:	BOOL: • **0** – Failure • **1** – Success

VB Example:	`dim netNameList(2)`
	`netNameList(0) = "NET-1"`
	`netNameList(1) = "NET-2"`
	`netNameList(2) = "NET-3"`
	`outcome = obj.ScrSanitizeNets(netNameList)`

IPY Example:	`oDoc.ScrSanitizeNets(['NET-1','NET-2','NET-3'])`

ScrSaveProjectAs

Saves the current project under a different file name.	
UI Command:	**File** > **Save As**.
Syntax:	`obj.ScrSaveProjectAs(<projName>)`
Parameters:	BSTR projName (without file extension)
Return Value:	BOOL • **0** – Failure • **1** – Success
VB Example:	`obj.ScrSaveProjectAs ("C:\Users\Ansys\Documents\Ansys\pcb1")`
IPY Example:	`oDoc.ScrSaveProjectAs ('C:\Users\Ansys\Documents\Ansys\pcb1')`

ScrSaveSimulationMessages

Saves messages for a specified simulation in a specified folder.	
UI Command:	None.
Syntax:	`obj.ScrSaveSimulationMessages(<simName>,<outFilePath>)`
Parameters:	BSTR simName BSTR outFilePath
Return Value:	BOOL: • **0** – Failure • **1** – Success
VB Example:	`obj.ScrSaveSimulationMessages("Sweep1", "C:\FilePath\")`
IPY Example:	`oDoc.ScrSaveSimulationMessages('Sweep1','C:\FilePath\')`

ScrSaveToPngFile

Saves the current modeling workspace as an image file (*.png).	
UI Command:	None.
Syntax:	`obj.ScrSaveToPngFile(<fileName>)`
Parameters:	BSTR fileName (full path with extension)
Return Value:	None.
VB Example:	`obj.ScrSaveToPngFile("D:/capture.png")`
IPY Example:	`oDoc.ScrSaveToPngFile('D:/capture.png')`

ScrSelectDcConnectedNets

Selects all nets and RLCs which form a connection to the specified net.	
UI Command:	None.
Syntax:	`obj.ScrSelectDcConnectedNets(<netNameList>)`
Parameters:	ARRAY netNameList
Return Value:	BOOL • **0** – Failure • **1** – Success

VB Example:	`dim netNameList`
	`netNameList(0)="VCC"`
	`netNameList(1)="GND"`
	`netNameList(2)="Heg"`
	`netNameList(3)="NET-1"`
	`netNameList(4)="NET-2"`
	`netNameList(5)="PWR"`
	`outcome = obj.ScrSelectDcConnectedNets(netNameList)`

IPY Example:	`oDoc.ScrSelectDcConnectedNets (['VCC','GND','Heq','NET-1','NET-2','PWR'])`

ScrSelectNet

Selects or deselects a specified net.	
UI Command:	None.
Syntax:	`obj.ScrSelectNet(<netName>,<select>)`
Parameters:	BSTR netName INT select (1 selects the net; 0 deselects it)
Return Value:	BOOL • **0** – Failure • **1** – Success
VB Example:	`obj.ScrSelectNet("GND", 1)`
IPY Example:	`oDoc.ScrSelectNet('GND', 1)`

ScrSelectNetsBetweenComponents

Selects all Power/Ground nets and RLCs that form a connection between the specified components.	
UI Command:	None.
Syntax:	`obj.ScrSelectNetsBetweenComponents(<partname1>, <refDes1>, <partname2>, <refDes2>)`
Parameters:	BSTR partname1 BSTR refDes1 BSTR partname2 BSTR refDes2
Return Value:	BOOL • **0** – Failure • **1** – Success
VB Example:	`obj.ScrSelectNetsBetweenComponents("288DIMMDDR4_EDGE_CONN-BASE", "J1", "DDR4_X4_FBGA78-10X13,,", "U1")`
IPY Example:	`oDoc.ScrSelectNetsBetweenComponents('288DIMMDDR4_EDGE_CONN-BASE', 'J1', 'DDR4_X4_FBGA78-10X13,,', 'U1')`

ScrSelectNetsBetweenNets

Selects all nets and RLCs that form a connection between the specified nets.	
UI Command:	None.
Syntax:	`obj.ScrSelectNetsBetweenNets(<net1>, <net2>)`
Parameters:	BSTR net1 BSTR net2
Return Value:	BOOL • **0** – Failure • **1** – Success
VB Example:	`obj.ScrSelectNetsBetweenNets("VDD", "GND")`
IPY Example:	`oDoc.ScrSelectNetsBetweenNets('VDD', 'GND')`

ScrSeparateDisjointNets

Separates all electrically disjoint nets into independent nets.	
UI Command:	None.
Syntax:	`obj.ScrSeparateDisjointNets()`
Parameters:	None.
Return Value:	INT numSeparatedNets
VB Example:	`obj.ScrSeparateDisjointNets()`
IPY Example:	`oDoc.ScrSeparateDisjointNets()`

ScrSet4PtBwProfile

Assigns a 4 point bondwire profile to all bondwires of a given model.	
UI Command:	None.
Syntax:	`obj.ScrSet4PtBwProfile(<modelName>, <h1>, <h2>, <radius>)`
Parameters:	BSTR modelName DOUBLE h1 DOUBLE h2 DOUBLE radius
Return Value:	BOOL • **0** – Failure • **1** – Success
VB Example:	`obj.ScrSet4PtBwProfile("WB_PROFILE_1", 100, 200, 20)`
IPY Example:	`oDoc.ScrSet4PtBwProfile('WB_PROFILE_1', 100, 200, 20)`

ScrSet5PtBwProfile

Assigns a 5 point bondwire profile to all bondwires of a given model.	
UI Command:	None.
Syntax:	`obj.ScrSet5PtBwProfile(<modelName>, <h1>, <h2>, <radius>, <alpha>, <beta>)`
Parameters:	BSTR modelName DOUBLE h1 DOUBLE h2 DOUBLE radius DOUBLE alpha DOUBLE beta
Return Value:	BOOL • **0** – Failure • **1** – Success
VB Example:	`obj.ScrSet5PtBwProfile("WB_PROFILE_1", 100, 200, 20, 85, 5)`
IPY Example:	`oDoc.ScrSet5PtBwProfile('WB_PROFILE_1', 100, 200, 20, 85, 5)`

ScrSetAntiPadOnLayer

Adds or changes a given padstack's antipads. Can also be used to delete antipads. **Note:** If an antipad already exists, the script alters it. If one does not exist, the script creates one.	
UI Command:	None.
Syntax:	`obj.ScrSetAntiPadOnLayer(<padstackName>, <layerName>, <shapeName>, <widthString>, <heightString>)`
Parameters:	BSTR padstackName layerName shapeName ("None" to delete an antipad; otherwise, "Circle", "Oblong", or "Rectangle") widthString (including unit of measure) heightString (including unit of measure)
Return Value:	BOOL • **0** – Failure • **1** – Success
VB Example:	`obj.ScrSetAntiPadOnLayer("VIA_M1_M2", "METAL-1", "Rectangle", "0.1cm", "0.1cm")`
IPY Example:	`oDoc.ScrSetAntiPadOnLayer('VIA_M1_M2', 'METAL-1', 'Rectangle', '0.1cm', '0.1cm')`

ScrSetBwModel

Sets the given model to bondwires	
UI Command:	None.
Syntax:	`obj.ScrSetBwModel(<bwIndexArray>, <bwModelName>)`
Parameters:	ARRAY bwIndexArray BSTR bwModelName
Return Value:	BOOL: • **0** – Failure • **1** – Success
VB Example:	`obj.ScrSetBwModel bwList, "WB_profile_1"`
IPY Example:	`oDoc.ScrSetBwModel(bwList, 'WB_profile_1')`

ScrSetBwSuppLayer

Sets the support layer of given bondwires.	
UI Command:	None.
Syntax:	`obj.ScrSetBwSuppLayer(<bwIndexArray>, <suppLayerName>)`
Parameters:	ARRAY bwIndexArray BSTR suppLayerName
Return Value:	BOOL: • **0** – Failure • **1** – Success
VB Example:	`obj.ScrSetBwSuppLayer bwList, "CU-1"`
IPY Example:	`oDoc.ScrSetBwSuppLayer(bwList, 'CU-1')`

ScrSetBwTermLayer

Sets the termination layer of given bondwires.	
UI Command:	None.
Syntax:	`obj.ScrSetBwTermLayer(<bwIndexArray>, <termLayerName>)`
Parameters:	ARRAY bwIndexArray BSTR termLayerName
Return Value:	BOOL: • **0** – Failure • **1** – Success
VB Example:	`obj.ScrSetBwTermLayer bwList, "WB_loop2"`
IPY Example:	`oDoc.ScrSetBwTermLayer(bwList, 'WB_loop2')`

ScrSetCapacitorDcBiasDeratingSim

Opens Set Capacitor Temperature and DC Bias Voltage window and select simulation for DC bias derating.	
UI Command:	Click **Simulation** > **Options** > **Use bias voltage computed by DC IR simulation**. Select simulation.
Syntax:	`obj.ScrSetCapacitorDcBiasDeratingSim (<simulationName>)`
Parameters:	BSTR simulationName
Return Value:	INT • **1** – Failure • **0** – Success

VB Example:	`Set objShell = WScript.CreateObject ("WScript.Shell")`
	`Set app = CreateObject("SIwave.Application.2023.1")`
	`Set doc = app.OpenProject(objShell.CurrentDirectory + "\temp_dep_caps1.siw")`
	`' pick DC voltage across caps computed by DC IR sim named "DC 3"`
	`' outcome == 0 indicates success`
	`' outcome == 1 indicates that specified DC simulation could not be found in existing results set`
	`outcome = doc.ScrSetCapacitorDcBiasDeratingSim("DC 3")`
	`doc.ScrSaveProjectAs(objShell.CurrentDirectory + "\modified_dcsim.siw")`
	`doc.ScrSetCapacitorDcBiasDeratingSim("") ' revert to user-defined DC bias derating values`
	`doc.ScrSaveProjectAs(objShell.CurrentDirectory + "\modified_user_def_dc.siw")`
	`app.Quit`

IPY Example:	`oDoc.ScrSetCapacitorDcBiasDeratingSim('Simulation Name')`

ScrSetCapacitorTemperatureDeratingSim

Opens Set Capacitor Temperature and DC Bias Voltage window and select simulation for temperature derating.	
UI Command:	Click **Simulation** > **Options** > **Use temperature computed by Icepak simulation**. Select simulation.
Syntax:	`obj.ScrSetCapacitorTemperatureDeratingSim (<simulationName>)`
Parameters:	BSTR simulationName
Return Value:	INT • **1** – Failure • **0** – Success

VB Example:	`Set objShell = WScript.CreateObject ("WScript.Shell")`
	`Set app = CreateObject("SIwave.Application.2023.1")`
	`Set doc = app.OpenProject(objShell.CurrentDirectory + "\temp_dep_caps1.siw")`
	`' pick temperature at caps computed by Icepak sim "Icepak Sim 2"`
	`' outcome == 0 indicates success`
	`' outcome == 1 indicates that specified Icepak simulation could not be found in existing results set`
	`outcome = doc.ScrSetCapacitorTemperatureDeratingSim ("Icepak Sim 2")`
	`doc.ScrSaveProjectAs(objShell.CurrentDirectory + "\modified_temp_sim.siw")`
	`doc.ScrSetCapacitorTemperatureDeratingSim("") ' revert to user-defined temperature derating values`
	`doc.ScrSaveProjectAs(objShell.CurrentDirectory + "\modified_user_def_temps.siw")`
	`app.Quit`

IPY Example:	`oDoc.ScrSetCapacitorTemperatureDeratingSim ('Simulation Name')`

ScrSetConformalCoatLayers

Introduce or remove conformal coating (a set of dielectric layers on the top and bottom of every package and PCB in the design). **These layers have a default material of "SolderMask" and a default thickness of 15 microns.**	
UI Command:	None.
Syntax:	`obj.ScrSetConformalCoatLayers(<setConformalCoat>)`
Parameters:	INT setConformalCoat (1 = add conformal coat layers, 0 = remove all conformal coat layers)
Return Value:	INT number of layers changed, added, or removed.
VB Example:	`obj.ScrSetConformalCoatLayers(1)`
IPY Example:	`oDoc.ScrSetConformalCoatLayers(1)`

ScrSetCrosstalkScanParameters

Specifies the parameters for running a Crosstalk Scan via ScrRunSimulation.	
UI Command:	**Simulation** > **Crosstalk Scan** > **Frequency Domain**. Set values for the solver options.
Syntax:	`obj.ScrSetCrosstalkScanParameters (<FEXTWarningLevel>, <FEXTViolationThreshold>, <NEXTWarningLevel>, <NEXTViolationThreshold>, <freq>, <minTraceLengthInMM>)`
Parameters:	DOUBLE FEXTWarningLevel DOUBLE FEXTViolationThreshold DOUBLE NEXTWarningLevel DOUBLE NEXTViolationThreshold DOUBLE freq DOUBLE minTraceLengthInMM
Return Value:	BOOL • **0** – Failure • **1** – Success
VB Example:	`obj.ScrSetCrosstalkScanParameters(15, 30, 12, 24, 1000000000, 1.2)`
IPY Example:	`oDoc.ScrSetCrosstalkScanParameters(15, 30, 12, 24, 1000000000, 1.2)`

ScrSetCrossTalkThreshold

Specifies the cross-talk threshold to use when determining which structures are coupled, in dB.	
UI Command:	Click **Simulation** > **Options** to open the **SIwave Options** window. Click **SI/PI Advanced** tab and enter a value in the **Cross-talk threshold** box.
Syntax:	`obj.ScrSetCrossTalkThreshold(<xtalkInDb>)`
Parameters:	DOUBLE xtalkInDb (specified value MUST be negative)
Return Value:	BOOL • **0** – Failure • **1** – Success
VB Example:	`obj.ScrSetCrossTalkThreshold(-60.0)`
IPY Example:	`oDoc.ScrSetCrossTalkThreshold(-60.0)`

ScrSetDcMinPlaneAreaToMesh

For DC IR simulations, sets the minimum plane area to be meshed.	
UI Command:	**Simulation** > **Options** > **SI/PI Advanced**. Enter a value in the **Ignore planes smaller than** field.
Syntax:	`obj.ScrSetDcMinPlaneAreaToMesh (<minPlaneAreaToMesh>)`
Parameters:	BSTR minPlaneAreaToMesh (including unit of measure)
Return Value:	None.
VB Example:	`obj.ScrSetDcMinPlaneAreaToMesh "5669.2mil2"`
IPY Example:	`oDoc.ScrSetDcMinPlaneAreaToMesh('5669.2mil2')`

ScrSetDcMinVoidAreaToMesh

For DC IR simulations, sets the minimum void area to be meshed.	
UI Command:	**Simulation** > **Options** > **DC Advanced**. Enter a value in the **Ignore voids smaller than** field.
Syntax:	`obj.ScrSetDcMinVoidAreaToMesh (<dcMinVoidAreaToMesh>)`
Parameters:	BSTR dcMinVoidAreaToMesh (including unit of measure)
Return Value:	None.
VB Example:	`obj.ScrSetMinVoidAreaToMesh "3199.01mil2"`
IPY Example:	`oDoc.ScrSetDcMinVoidAreaToMesh('3199.01mil2')`

ScrSetDcPowerDataThresholds

For DC IR simulations, sets the minimum thermal cell size and minimum power per cell. Note: These settings are no longer visible in SIwave, and are only applicable for exporting data for Icepak in an old format.	
UI Command:	None.
Syntax:	`obj.ScrSetDcPowerDataThresholds (<minThermCellSizeInUm>,<minPwrLossPerCellInMilliwatts)`
Parameters:	DOUBLE minThermCellSizeInUm DOUBLE minPwrLossPerCellInMilliwatts
Return Value:	None.
VB Example:	`obj.ScrSetDcPowerDataThresholds 12.0, 1.75`
IPY Example:	`oDoc.ScrSetDcPowerDataThresholds(12.0, 1.75)`

ScrSetDieElevation

Sets a specified die's elevation.	
UI Command:	**Advanced** > **Die Stackup**.
Syntax:	`obj.ScrSetDieElevation(<dieName>,<elevation>)`
Parameters:	BSTR dieName DOUBLE elevation
Return Value:	BOOL • **0** – Failure • **1** – Success
VB Example:	`obj.ScrSetDieElevation("DIE_1", 100.0)`
IPY Example:	`oDoc.ScrSetDieElevation('DIE_1', 100.0)`

ScrSetDieThickness

Sets a specified die's thickness.	
UI Command:	**Advanced** > **Die Stackup**.
Syntax:	`obj.ScrSetDieThickness(<dieName>,<thickness>)`
Parameters:	BSTR dieName DOUBLE thickness
Return Value:	BOOL • **0** – Failure • **1** – Success
VB Example:	`obj.ScrSetDieThickness("DIE_1", 100.0)`
IPY Example:	`oDoc.ScrSetDieThickness('DIE_1', 100.0)`

ScrSetEmiScannerParameters

Sets the parameters for running an EMI scan via ScrRunSimulation. **If this function is not called, the scan runs with the EMI parameters already set in the file.** **If none are set, the scan runs using the default "EM Rules" with auto-tagging.**	
UI Command:	**Simulation** > **EMI Scanner**.
Syntax:	`obj.ScrSetEmiScannerParameters (<rulesXmlFilenameWithPath>, <rulesProfileIndex>, <tagsXmlFilenameWithPath>)`
Parameters:	BSTR rulesXmlFilenameWithPath (If rulesXmlFilenameWithPath is left an empty string, rulesProfileIndex is used) INT rulesProfileIndex, where: • **0** – Rules file specified by `<rulesXmlFilenameWithPath>` • **1** – EM Rules – [install_dir]/config/EMIScanner/Defaults/EM_emsat.xml • **2** – SI Rules – [install_dir]/config/EMIScanner/Defaults/SI_emsat.xml • **3** – EM+SI Rules – [install_dir]/config/EMIScanner/Defaults/EM_SI_emsat.xml • **4** – Rules (0-100MHz) – [install_dir]/config/EMIScanner/Defaults/EMSAT-STGprofile-000-100Mb.cfg • **5** – Rules (100MHz-500MHz) – [install_dir]/config/EMIScanner/Defaults/EMSAT-STGprofile-100-500Mb.cfg • **6** – Rules (500MHz-1000MHz) – [install_dir]/config/EMIScanner/Defaults/EMSAT-STGprofile-500-1000Mb.cfg • **7** – Rules (1GHz+) – [install_dir]/config/EMIScanner/Defaults/EMSAT-STGprofile-1Gb-and-higher.cfg BSTR tagsXmlFilenameWithPath
Return Value:	BOOL • **0** – Failure • **1** – Success
VB Example:	`obj.ScrSetEmiScannerParameters "C:/Projects/rules.xml", 0, "C:/Projects/tags.tgs"`
IPY Example:	`oDoc.ScrSetEmiScannerParameters ('C:/Projects/rules.xml', 0, 'C:/Projects/tags.tgs')`

ScrSetEnergyErrorPercentInDcSimulation

For DC IR simulations, sets the minimum void area to be meshed.	
UI Command:	**Simulation** > **Options** > **DC Advanced**. Enter a value in the **Energy Error** field.
Syntax:	`obj.ScrSetEnergyErrorPercentInDcSimulation (<energyErrorPercent>)`
Parameters:	DOUBLE energyErrorPercent
Return Value:	None.
VB Example:	`obj.ScrSetEnergyErrorPercentInDcSimulation "5.1"`
IPY Example:	`oDoc.ScrSetEnergyErrorPercentInDcSimulation('5.1')`

ScrSetExternalExcitations

Directs SIwave to use external (non-linear) excitation files containing current or voltage source frequency response data.	
UI Command:	From the **Simulation** menu, click either **Compute Frequency Sweep**, **Compute Far Field**, or **Compute Near Field**. Select the **Use sources defined in an external file** check box and specify the source of the excitations.
Syntax:	`obj.ScrSetExternalExcitations(<filePath>)`
Parameters:	BSTR filePath
Return Value:	None.

VB Example:	`' set current/voltage sources to reference I(f)/V(f) data in "C:\sources.txt"`
	`obj.ScrSetExternalExcitations "C:\sources.txt"`
	`' set current/voltage sources back to linear magnitude/phase values`
	`obj.ScrSetExternalExcitations "`

IPY Example:	`oDoc.ScrSetExternalExcitations('C:\sources.txt')`

ScrSetFarFieldSimOptions

Sets options for Far Fields simulations.	
UI Command:	**Simulation** > **Compute Far Field**. Set **Phi** and **Theta** options.
Syntax:	`obj.ScrSetFarFieldSimOptions(<phiStart>, <phiStop>, <phiStepSize>, <thetaStart>, <thetaStop>, <thetaStepSize>)`
Parameters:	DOUBLE phiStart DOUBLE phiStop INT phiStepSize DOUBLE thetaStart DOUBLE thetaStop INT thetaStepSize
Return Value:	BOOL: • **0** – Failure • **1** – Success
VB Example:	`obj.ScrSetFarFieldSimOptions(0, 360, 10, 0, 180, 10)`
IPY Example:	`oDoc.ScrSetFarFieldSimOptions(0, 360, 10, 0, 180, 10)`

ScrSetFwsColFitOptions

Controls FWS Sub-circuit column fitting options.	
UI Command:	Click **Results** > **SYZ** > **[Simulation Name]** > **Compute FWS Sub-circuit** to open the **Compute Full Wave SPICE Subcircuit** window. Specify **Column Fitting** options.
Syntax:	`obj.ScrSetFwsColFitOptions(<option>)`
Parameters:	INT option, where: • **0** – One matrix entry at a time • **1** – One matrix column at a time
Return Value:	None.
VB Example:	`obj.ScrSetFwsColFitOptions 0`
IPY Example:	`oDoc.ScrSetFwsColFitOptions(0)`

ScrSetFwsLaunchDesignerNexxim

After computing a Full Wave SPICE Subcircuit, opens the results in Electronics Desktop.	
UI Command:	None.
Syntax:	`obj.ScrSetFwsLaunchDesignerNexxim(<launch>)`
Parameters:	INT launch (1 = launch; 0 = do not launch)
Return Value:	None.
VB Example:	`obj.ScrSetFwsLaunchDesignerNexxim 1`
IPY Example:	`oDoc.ScrSetFwsLaunchDesignerNexxim(1)`

ScrSetFwsPassivityAlg

Controls FWS Sub-circuit Full Wave SPICE export passivity options.	
UI Command:	Click **Results** > **SYZ** > **[Simulation Name]** > **Compute FWS Sub-circuit** to open the **Compute Full Wave SPICE Subcircuit** window. Select passivity enforcement options.
Syntax:	`obj.ScrSetFwsPassivityAlg(<option>)`
Parameters:	INT option, where: • **0** – Do not enforce passivity • **1** – Passivity enforcement through convex optimization • **2** – Passivity enforcement by perturbation • **3** – Passivity enforcement by IFPV (iterative fitting of passivity violations)
Return Value:	None.
VB Example:	`obj.ScrSetFwsPassivityAlg 0`
IPY Example:	`oDoc.ScrSetFwsPassivityAlg(0)`

ScrSetFwsPortRefZ

For Full Wave SPICE, sets whether port reference impedances are to be renormalized and, if so, specifies reference impedance.	
UI Command:	Click **Results** > **SYZ** > **[Simulation Name]** > **Compute FWS Sub-circuit** to open the **Compute Full Wave SPICE Subcircuit** window. Enter a value in the **Renormalize all S-parameters to __ ohms before generating model** option.
Syntax:	`obj.ScrSetFwsPortRefZ(renormalize, refZ)`
Parameters:	BOOL renormalize, where: • **TRUE** – Renormalize • **FALSE** – Do not renormalize DOUBLE refZ (reference impedence, value ignored if renormalize = FALSE)
Return Value:	None.
VB Example:	`obj.ScrSetFwsPortRefZ(True, 75.0)`
IPY Example:	`oDoc.ScrSetFwsPortRefZ(True, 75.0)`

ScrSetFwsPzOptions

For Full Wave SPICE, sets pole/zero fitting options.	
UI Command:	Click **Results** > **SYZ** > **[Simulation Name]** > **Compute FWS Sub-circuit** to open the **Compute Full Wave SPICE Subcircuit** window. Select the options in the **HSPICE/Spectre Pole/Zero Fitting Options** box for the fitting error tolerance, and maximum pole/zero order.
Syntax:	`obj.ScrSetFwsPzOptions(<fitError>, <maxOrder>)`
Parameters:	DOUBLE fitError (fitting error tolerance) INT maxOrder (maximum pole/zero order)
Return Value:	None.
VB Example:	`obj.ScrSetFwsPzOptions(0.001, 200)`
IPY Example:	`oDoc.ScrSetFwsPzOptions(0.001, 200)`

ScrSetFwsSsfAlg

For Full Wave SPICE, specifies State-Space Fitting Algorithm options.	
UI Command:	Click **Results** > **SYZ** > **[Simulation Name]** > **Compute FWS Sub-circuit** to open the **Compute Full Wave SPICE Subcircuit** window. Specify **State-Space Fitting Algorithm** options.
Syntax:	`obj.ScrSetFwsSsfAlg(<algorithm>)`
Parameters:	INT algorithm, where: • **0** – TWS • **1** – Iterative Rational Function
Return Value:	None.
VB Example:	`obj.ScrSetFwsSsfAlg 0`
IPY Example:	`oDoc.ScrSetFwsSsfAlg(0)`

ScrSetFwsSubcktFormat

Specifies the full-wave Spice subcircuit format for export.	
UI Command:	Click **Results** > **SYZ** > **[Simulation Name]** > **Compute FWS Sub-circuit** to open the **Compute Full Wave SPICE Subcircuit** window. Select the Spice subcircuit format from the **Full Wave Spice Subcircuit Format** box.
Syntax:	`obj.ScrSetFwsSubcktFormat(<format>)`
Parameters:	INT format, where: • **0** – HSPICE-compatible format • **1** – Maxwell Spice compatible format • **2** – PSPICE compatible format • **3** – Cadence Spectre compatible format • **4** – Nexxim S-element format • **5** – State space model • **6** – Simplorer model
Return Value:	None.
VB Example:	`obj.ScrSetFwsSubcktFormat 0`
IPY Example:	`oDoc.ScrSetFwsSubcktFormat(0)`

ScrSetFwsUseCommonGround

For full-wave Spice subcircuit generation, specifies whether or not to use a common ground for all ports.	
UI Command:	Click **Results** > **[Simulation Name]** > **Compute FWS Sub-circuit** to open the **Compute Full Wave SPICE Subcircuit** window. Select the **Use common ground for Spice output** check box.
Syntax:	`obj.ScrSetFwsUseCommonGround(<useCommonGround>)`
Parameters:	BOOL, where: • **FALSE** – Do not use common ground. • **TRUE** – Use common ground.
Return Value:	None.
VB Example:	`obj.ScrSetFwsUseCommonGround True`
IPY Example:	`oDoc.ScrSetFwsUseCommonGround(True)`

ScrSetHFSS3DLayoutSimOptions

Imports simulation settings for an HFSS 3D Layout Simulation.	
UI Command:	Click **Simulation** > **HFSS 3D Layout**. Click **Import Settings** and select a file.
Syntax:	`obj.ScrSetHFSS3DLayoutSimOptions(<filename>)`
Parameters:	BSTR filename (full file path)
Return Value:	BOOL • **0** – Failure • **1** – Success
VB Example:	`' outcome is TRUE on success, FALSE if specified "filename" does not exist or is not the right format (this file is most easily generated by exporting from the HFSS 3D Layout Simulation Setup window).` `outcome = obj.ScrSetHFSS3DLayoutSimOptions ( "c:\simsettings.dss" )`
IPY Example:	`oDoc.ScrSetHFSS3DLayoutSimOptions ('c:\simsettings.dss')`

ScrSetHpcLicenseType

Sets the HPC license type. **Important: Use ScrSetHpcLicenseVendor to choose license vendor.**	
UI Command:	**Simulation** > **Options**. On the **Multiprocessing** tab, select **Use HPC licensing** and either **Pool** or **Pack**.
Syntax:	`obj.ScrSetHpcLicenseType(<licenseType>)`
Parameters:	BSTR licenseType ('pool' or 'pack'; case insensitive; use 'mp' to deselect **Use HPC licensing**)
Return Value:	BOOL • **0** – Failure • **1** – Success
VB Example:	`obj.ScrSetHpcLicenseType "pool"`
IPY Example:	`oDoc.ScrSetHpcLicenseType('pack')`

ScrSetHpcLicenseVendor

Sets the HPC license vendor. **Important: Use ScrSetHpcLicenseType to choose license type.**	
UI Command:	**Simulation** > **Options**. On the **Multiprocessing** tab, select **Use HPC licensing** and either **Use Ansys HPC licenses** or **Use legacy Electronics HPC licenses**.
Syntax:	`obj.ScrSetHpcLicenseVendor(<vendor>)`
Parameters:	BSTR vendor ('ansys' or 'electronics', case insensitive)
Return Value:	BOOL • **0** – Failure • **1** – Success
VB Example:	`obj.ScrSetHpcLicenseVendor "Ansys"`
IPY Example:	`oDoc.ScrSetHpcLicenseVendor('Electronics')`

ScrSetIcepakBoardOutlineFidelity

Specifies the minimum edge length when modifying the board outline for export to Icepak. This minimum edge length is used when discretizing arcs into a series of straight lines and when simplifying the outline to remove very small edges.	
UI Command:	None.
Syntax:	`obj.ScrSetIcepakBoardOutlineFidelity(<distInMM>)`
Parameters:	DOUBLE distInMM
Return Value:	INT • **0** – Success • **1** – Specified distance is too small
VB Example:	`obj.ScrSetIcepakBoardOutlineFidelity(1.5)`
IPY Example:	`oDoc.ScrSetIcepakBoardOutlineFidelity(1.5)`

ScrSetIcepakCabinetDimensions

Configures the cabinet settings for Icepak convection simulations.	
UI Command:	None.
Syntax:	`obj.ScrSetIcepakCabinetDimensions (<horizPadPercent>, <vertAbovePadPercent>, <vertBelowPadPercent>)`
Parameters:	DOUBLE horizPadPercent DOUBLE vertAbovePadPercent DOUBLE vertBelowPadPercent
Return Value:	INT: • **0** – Success • **1** – Specified percentages are invalid.
VB Example:	`obj.ScrSetIcepakCabinetDimensions(35.0, 125.0, 55.0)`
IPY Example:	`oDoc.ScrSetIcepakCabinetDimensions(35.0, 125.0, 55.0)`

ScrSetIcepakComponentConfig

Selects a component settings file to use for Icepak convection simulations.	
UI Command:	None.
Syntax:	`obj.ScrSetIcepakComponentConfig(<fileName>)`
Parameters:	BSTR fileName (full file path)
Return Value:	BOOL • **0** – Failure • **1** – Success
VB Example:	`obj.ScrSetIcepakComponentConfig ("D:\Tests\IcepakScriptTest.pwrd")`
IPY Example:	`oDoc.ScrSetIcepakComponentConfig ('D:\Tests\IcepakScriptTest.pwrd')`

ScrSetIcepakMeshingDetail

Sets the meshing detail level for Icepak simulations.	
UI Command:	None.
Syntax:	`obj.ScrSetIcepakMeshingDetail(<meshLevel>)`
Parameters:	BSTR meshLevel
Return Value:	INT: • **0** – Success • **Else** – Failure
VB Example:	`obj.ScrSetIcepakMeshingDetail("basic")`
IPY Example:	`oDoc.ScrSetIcepakMeshingDetail('basic')`

ScrSetIcepakSimReportImageHeight

When generating Icepak reports or Icepak report data in a DC report, specifies the resolution (in pixels) of the smaller dimension of the images. For designs where the board is wider than it is tall, this corresponds to the y-resolution of the image.	
UI Command:	None.
Syntax:	`obj.ScrSetIcepakSimReportImageHeight(<imgHeight>)`
Parameters:	LONG imgHeight
Return Value:	BOOL • **0** – Failure (invalid height) • **1** – Success
VB Example:	`obj.ScrSetIcepakSimReportImageHeight(1024)`
IPY Example:	`oDoc.ScrSetIcepakSimReportImageHeight(1024)`

ScrSetIcepakTemperatureFile

Specifies an external Icepak solution file (*.sitemp) for use in specifying temperature for a DC simulation.	
UI Command:	**Simulation** > **Compute DC IR**. Select **Import temperature map from Icepak**. Select **External .sitemp file**. Click **Browse** and select file.
Syntax:	`obj.ScrSetIcepakTemperatureFile(<sitempFilename>)`
Parameters:	BSTR sitempFilename (full file path)
Return Value:	BOOL • **0** – Failure (file does not exist) • **1** – Success
VB Example:	`obj.ScrSetIcepakTemperatureFile("d:/abcd.sitemp")`
IPY Example:	`oDoc.ScrSetIcepakTemperatureFile('d:/abcd.sitemp')`

ScrSetIcepakThermalEnv

Sets the thermal environment settings to use for Icepak simulations.	
UI Command:	**Simulation** > **Icepak**. Select the desired settings.
Syntax:	`obj.ScrSetIcepakThermalEnv(<convection>, <forcedAir>, <topOrAmbientTempC>, <topOrOverallFlowDir>, <topOrOverallFlowSpeed>, <bottomTempC>, <bottomFlowDir>, <bottomFlowSpeed>, <gravVecX>, <gravVecY>, <gravVecZ>)`
Parameters:	BOOL convection (TRUE = convection; FALSE = conduction) BOOL forcedAir (TRUE = forced convection; FALSE = natural convection) DOUBLE topOrAmbientTempC (temperature above PCB, in Celsius) BSTR topOrOverallFlowDir (flow direction above PCB) DOUBLE topOrOverallFlowSpeed (flow speed above PCB) DOUBLE bottomTempC (temperature below PCB, in Celsius) BSTR bottomFlowDir (flow direction below PCB) DOUBLE bottomFlowSpeed (flow speed below PCB) DOUBLE gravVecX (gravity vector x for natural convection) DOUBLE gravVecY (gravity vector y for natural convection) DOUBLE gravVecZ (gravity vector z for natural convection)
Return Value:	INT: • **0** – Success • **Else** – Failure
VB Example:	`obj.ScrSetIcepakThermalEnv(TRUE, TRUE, 22.3, "+Y", 2.5, 0.0, "", 0.0, 0.0, 0.0, 0.0)`
IPY Example:	`oDoc.ScrSetIcepakThermalEnv(True, True, 22.3, '+Y', 2.5, 0.0, '', 0.0, 0.0, 0.0, 0.0)`

ScrSetIdealGroundNodeInDcSimulation

For a DC IR simulation, sets the Ideal Ground Node (the precise 0V). **IMPORTANT: This script should be called before running a DC solve using ScrRunDcSimulation or ScrRunSimulation with tag dc.**	
UI Command:	**Simulation** > **Compute DC IR**. Select an ideal ground node.
Syntax:	`obj.ScrSetIdealGroundNodeInDcSimulation(<circuit_element_name>, <node_type_id>)`
Parameters:	BSTR circuit_element_name (name of the circuit element in the design on which the ideal ground is) INT node_type_id (which terminal of the circuit element is ideal ground), where: • **0** – Neither Terminal • **1** – Negative Terminal • **2** – Positive Terminal
Return Value:	BOOL • **0** – Failure • **1** – Success
VB Example:	`obj.ScrSetIdealGroundNodeInDcSimulation("VU6", 1)`
IPY Example:	`oDoc.ScrSetIdealGroundNodeInDcSimulation('VU6', 1)`

ScrSetInducedVoltageMultipleIncidenceSpherical

For a Plane Wave Induced Voltage simulation with multiple incident waves specified in spherical system having a sweep of phi & theta angles and the polarization, specified as either having 1 on X or 1 on Y or both, on the orthogonal plane of every incidence vector. **IMPORTANT: This script and ScrAppendSweep should be called before running a Plane Wave Induced Voltage solve using ScrRunSimulation with tag iv.**	
UI Command:	**Simulation** > **Compute Induced Voltage**. Specify settings.
Syntax:	`obj.ScrSetInducedVoltageMultipleIncidenceSpherical (<phi_start>, <phi_stop>, <phi_step>, <theta_start>, <theta_stop>, <theta_step>, <phi_checked>, <theta_checked>, <save_for_all_angles>, <Magnitude>)`
Parameters:	DOUBLE phi_start (for sweep in phi) DOUBLE phi_stop (for sweep in phi) DOUBLE phi_step (for sweep in phi) DOUBLE theta_start (for sweep in theta) DOUBLE theta_stop (for sweep in theta) DOUBLE theta_step (for sweep in theta) DOUBLE e0_phi (polarization for phi sweep) DOUBLE e0_theta (polarization for theta sweep) INT save_for_all_angles (1 = save; 0 = do not save) DOUBLE magnitude
Return Value:	BOOL • **0** – Failure • **1** – Success

VB Example:	`obj.ScrClearAllSweeps "iv"`
	`obj.ScrAppendSweep "iv", 1500000, 3000000, 8, false`
	`obj.ScrAppendSweep "iv", 3000000, 5000000, 2, true`
	`obj.ScrSetInducedVoltageMultipleIncidenceSpherical 0, 20, 5, 90, 105, 5, 0, 0, 0, 1`
	`obj.ScrRunSimulation "iv", "tstmultiple"`

IPY Example:	`oDoc.ScrSetInducedVoltageMultipleIncidenceSpherical (0, 20, 5, 90, 105, 5, 0, 0, 0, 1)`

ScrSetInducedVoltageSingleIncidenceCartesian

For a Plane Wave Induced Voltage simulation with a single incident wave and polarization both specified in cartesian system on the XYZ plane. The incident wave vector and the polarization vector should be orthogonal. This is tested by verifying if the dot product of the vectors leads to zero. **IMPORTANT: This script and ScrAppendSweep should be called before running a Plane Wave Induced Voltage solve using ScrRunSimulation with tag iv.**	
UI Command:	**Simulation** > **Compute Induced Voltage**. Specify settings.
Syntax:	`obj.ScrSetInducedVoltageSingleIncidenceCartesian (<incidence_x>, <incidence_y>, <incidence_z>, <e0_x>, <e0_y>, <e0_z>, <e0_magnitude>)`
Parameters:	DOUBLE incidence_x (for incident vector) DOUBLE incidence_y (for incident vector) DOUBLE incidence_z (for incident vector) DOUBLE e0_x (for polarization vector) DOUBLE e0_y (for polarization vector) DOUBLE e0_z (for polarization vector) DOUBLE e0_magnitude
Return Value:	BOOL • **0** – Failure • **1** – Success

VB Example:	`obj.ScrClearAllSweeps "iv"`
	`obj.ScrAppendSweep "iv", 1500000, 3000000, 8, false`
	`obj.ScrAppendSweep "iv", 3000000, 5000000, 2, true`
	`obj.ScrSetInducedVoltageSingleIncidenceCartesian 1, 0, 0, 0, 1, 1, 1`
	`obj.ScrRunSimulation "iv", "tstcartesian"`

IPY Example:	`oDoc.ScrSetInducedVoltageSingleIncidenceCartesian (1, 0, 0, 0, 1, 1, 1)`

ScrSetInducedVoltageSingleIncidenceSpherical

For a Plane Wave Induced Voltage simulation with a single incident wave specified in spherical system (degrees) and polarization specified as a XY vector on the orthogonal plane of incidence vector. **IMPORTANT: This script and ScrAppendSweep should be called before running a Plane Wave Induced Voltage solve using ScrRunSimulation with tag iv.**	
UI Command:	**Simulation** > **Compute Induced Voltage**. Specify settings.
Syntax:	`obj.ScrSetInducedVoltageSingleIncidenceSpherical (<phi>, <theta>, <e0_phi>, <e0_theta>, <e0_magnitude>)`
Parameters:	DOUBLE phi (for incident vector) DOUBLE theta (for incident vector) DOUBLE e0_phi (for polarization vector) DOUBLE e0_theta (for polarization vector) DOUBLE e0_magnitude
Return Value:	BOOL • **0** – Failure • **1** – Success

VB Example:	`obj.ScrClearAllSweeps "iv"`
	`obj.ScrAppendSweep "iv", 1500000, 3000000, 8, false`
	`obj.ScrAppendSweep "iv", 3000000, 5000000, 2, true`
	`obj.ScrSetInducedVoltageSingleIncidenceSpherical 10, 45, 1, 3, 1`
	`obj.ScrRunSimulation "iv", "tstspherical"`

IPY Example:	`oDoc.ScrSetInducedVoltageSingleIncidenceSpherical (10, 45, 1, 3, 1)`

ScrSetInfiniteGroundPlaneLocation

Introduces an infinite ground plane the specified distance below the bottom layer.	
UI Command:	**Simulation** > **Options** > **SI/PI Advanced**. Select **Introduce infinite ground plane** and enter a value in mils.
Syntax:	`obj.ScrSetInfiniteGroundPlaneLocation(<elev>)`
Parameters:	DOUBLE elev
Return Value:	BOOL: • **0** – Failure • **1** – Success
VB Example:	`obj.ScrSetInfiniteGroundPlaneLocation "3.1"`
IPY Example:	`oDoc.ScrSetInfiniteGroundPlaneLocation('3.1')`

ScrSetLayerMaterial

Assigns a specified material to a specified layer.	
UI Command:	**Home** > **Layer stackup Editor**. Click **Edit Layer Properties**. Select a material from the **Material** drop-down menu.
Syntax:	`obj.ScrSetLayerMaterial(<layerNameBstr>, <layerMaterialBstr>)`
Parameters:	BSTR layerNameBstr BSTR layerMaterialBstr
Return Value:	BOOL • **0** – Failure • **1** – Success
VB Example:	`' outcome is TRUE on success, FALSE if specified layer or material could not be located` `outcome = obj.ScrSetLayerMaterial ( "SURFACE", "copper" )`
IPY Example:	`oDoc.ScrSetLayerMaterial('SURFACE', 'copper')`

ScrSetLayerThickness

Changes the thickness of a specified layer. The thickness value must be in the project's underlying length units.	
UI Command:	**Home** > **Layer stackup Editor**. Click **Edit Layer Properties**. Enter the thickness in the **Thickness** field.
Syntax:	`obj.ScrSetLayerThickness(<layerNameBstr>, <thickness>, <redraw>)`
Parameters:	BSTR layerNameBstr DOUBLE thickness BOOL redraw (TRUE = redraw; FALSE = do not redraw)
Return Value:	BOOL • **0** – Failure • **1** – Success

VB Example:	`' set the thickness of layer "SURFACE" to 35um (assuming the database units are mm)`
	`' outcome is TRUE on success, FALSE if specified layer could not be located`
	`outcome = obj.ScrSetLayerThickness( "SURFACE", 0.035, True )`

IPY Example:	`oDoc.ScrSetLayerThickness('SURFACE', 0.035, True)`

ScrSetLayerType

Sets a specified layer's type.	
UI Command:	**Home** > **Layer stackup Editor**. Select the layer you want to update. Then select a material type from the **Type** drop-down menu.
Syntax:	`obj.ScrSetLayerType(<layerName>, <layerTypeIndex>)`
Parameters:	BSTR layerName INT layerTypeIndex, where: • **0** – Dielectric • **1** – Metal • **2** – Wirebond
Return Value:	None.
VB Example:	`obj.ScrSetLayerType("layer name", 1)`
IPY Example:	`oDoc.ScrSetLayerType('layer name', 1)`

ScrSetLayerVisibility

Sets the visibility of the specified layer, or of specific object types on that layer.	
UI Command:	Select a layer in the **Layers Workspace**. Turn on visibility for the layer or for specific object types using the check boxes.
Syntax:	`obj.ScrSetLayerVisibility(<layerNameBstr>, <planeVis>, <traceVis>, <padVis>, <viaVis>, <cktElemVis>)`
Parameters:	BSTR layerNameBstr BOOL planeVis (TRUE = planes visible; FALSE = planes not visible) BOOL traceVis (TRUE = traces visible; FALSE = traces not visible) BOOL padVis (TRUE = pads visible; FALSE = pads not visible) BOOL viaVis (TRUE = vias visible; FALSE = vias not visible) BOOL cktElemVis (TRUE = circuit elements visible; FALSE = circuit elements not visible)
Return Value:	BOOL • **0** – Failure • **1** – Success

VB Example:	`'Make planes, traces, pads, and vias on layer L1 visible, but not circuit elements'`
	`outcome is TRUE on success, FALSE if specified layer could not be located.`
	`outcome = obj.ScrSetLayerVisibility( "L1", True, True, True, True, False)`

IPY Example:	`oDoc.ScrSetLayerVisibility('L1', True, True, True, True, False)`

ScrSetLayoutLengthUnit

Sets the layout length unit.	
UI Command:	None.
Syntax:	`obj.ScrSetLayoutLengthUnit <unitName>`
Parameters:	BSTR unitName ("microns", "um", "mils", "mm", "cm", "inches", or "meters")
Return Value:	BOOL • **0** – Failure • **1** – Success
VB Example:	`obj.ScrSetLayoutLengthUnit "um"`
IPY Example:	`oDoc.ScrSetLayoutLengthUnit('um')`

ScrSetLocalRefinementPercentInDcSimulation

For a DC IR simulation, sets the local refinement percentage.	
UI Command:	**Simulation** > **Options** > **DC Advanced**. Enter a value in the **Local Refinement** field.
Syntax:	`obj.ScrSetLocalRefinementPercentInDcSimulation (<localRefinePercent>)`
Parameters:	DOUBLE localRefinePercent
Return Value:	None.
VB Example:	`obj.ScrSetLocalRefinementPercentInDcSimulation 20`
IPY Example:	`oDoc.ScrSetLocalRefinementPercentInDcSimulation(20)`

ScrSetLogFreqPointDist

Sets the distribution type for frequency sweeps.	
UI Command:	None.
Syntax:	`obj.ScrSetLogFreqPointDist(<flag>)`
Parameters:	INT flag (1 = log-based frequency distribution; 0 = linear distribution)
Return Value:	None.
VB Example:	`obj.ScrSetLogFreqPointDist 1`
IPY Example:	`oDoc.ScrSetLogFreqPointDist(1)`

ScrSetLowBwProfile

Sets low bondwire profile to all bondwires of a specified model.	
UI Command:	None.
Syntax:	`obj.ScrSetLowBwProfile(<bwModelName>, <h1>, <h2>, <radius>, <alpha>, <beta>, <units>)`
Parameters:	BSTR bwModelName DOUBLE h1 DOUBLE h2 DOUBLE radius DOUBLE alpha DOUBLE beta BSTR units
Return Value:	BOOL • **0** – Failure • **1** – Success
VB Example:	`obj.ScrSetLowBwProfile "WB_PROFILE_1", 100, 200, 20, 85, 5, "mm"`
IPY Example:	`oDoc.ScrSetLowBwProfile('WB_PROFILE_1', 100, 200, 20, 85, 5, 'mm')`

ScrSetMaxRefinePassesInDcSimulation

Sets the maximum number of mesh refinement passes for DC IR drop simulations.	
UI Command:	**Simulation** > **Compute DC IR**. Click **Other solver options** button and choose the **DC Advanced** tab. In the **Adaptive Mesh Refinement Parameters** group, type a value in the **Maximum Number of Passes** box.
Syntax:	`obj.ScrSetMaxRefinePassesInDcSimulation (<maxPasses>)`
Parameters:	INT maxPasses
Return Value:	None.
VB Example:	`obj.ScrSetMaxRefinePassesInDcSimulation 5`
IPY Example:	`oDoc.ScrSetMaxRefinePassesInDcSimulation(5)`

ScrSetMeshBondwiresInDcSimulation

Sets the Mesh Bondwires setting.	
UI Command:	**Simulation** > **Options**. Click **DC Advanced** tab and select **Mesh Bondwires** check box.
Syntax:	`obj.ScrSetMeshBondwiresInDcSimulation(<meshBws>)`
Parameters:	BOOL meshBws (TRUE = mesh bondwires; FALSE = do not mesh bondwires)
Return Value:	None.
VB Example:	`obj.ScrSetMeshBondwiresInDcSimulation True`
IPY Example:	`oDoc.ScrSetMeshBondwiresInDcSimulation(1)`

ScrSetMeshViasInDcSimulation

Specifies whether or not to mesh vias during DC IR drop simulations.	
UI Command:	**Simulation** > **Compute DC IR**. Click **Other solver options** button and choose the **DC Advanced** tab. Select **Mesh Vias**.
Syntax:	`obj.ScrSetMeshViasInDcSimulation(<meshVias>)`
Parameters:	BOOL meshVias (TRUE = mesh vias; FALSE = do not mesh vias)
Return Value:	None.
VB Example:	`obj.ScrSetMeshViasInDcSimulation True`
IPY Example:	`oDoc.ScrSetMeshViasInDcSimulation(True)`

ScrSetMetalLayerFillerMaterial

Assigns a specified filler material to a specified metal layer.	
UI Command:	**Home** > **Layer Stackup Editor**. Select metal layer and change **Dielectric Fill**.
Syntax:	`obj.ScrSetMetalLayerFillerMaterial(<layerNameBstr>, <layerFillerMaterialBstr>)`
Parameters:	BSTR layerNameBstr BSTR layerFillerMaterialBstr
Return Value:	BOOL • **0** – Failure • **1** – Success
VB Example:	`' outcome is TRUE on success, FALSE if specified layer or material could not be located` `outcome = obj.ScrSetMetalLayerFillerMaterial ("SURFACE", "FR-4")`
IPY Example:	`oDoc.ScrSetMetalLayerFillerMaterial('SURFACE', 'FR-4')`

ScrSetMinCutoutArea

Controls the cutout defeaturing area threshold; all cutouts below this value are ignored during simulations. **This function is not applicable to DC IR drop simulations, during which all salient geometry is meshed and simulated (no cutouts are defeatured).**	
UI Command:	**Simulation** > **Options**. Enter the area threshold in the **Do not explicitly mesh any voids than** field.
Syntax:	`obj.ScrSetMinCutoutArea(<minVoidArea>, <unitsBstr>)`
Parameters:	DOUBLE minVoidArea BSTR unitsBstr
Return Value:	None.
VB Example:	`'defeature all cutouts less than 10mm`^{2} `during simulations.` `obj.ScrSetMinCutoutArea 10, mm`
IPY Example:	`oDoc.ScrSetMinCutoutArea(10, 'mm')`

ScrSetMinPadAreaToMesh

Sets the minimum pad area for meshing.	
UI Command:	**Simulation** > **Options** > **SI/PI Advanced**. Enter a value in the **Explicitly mesh pads larger than** field.
Syntax:	`obj.ScrSetMinPadAreaToMesh(<minPadAreaToMesh>)`
Parameters:	BSTR minPadAreaToMesh (including unit of measure)
Return Value:	None.
VB Example:	`obj.ScrSetMinPadAreaToMesh "112000mil2"`
IPY Example:	`oDoc.ScrSetMinPadAreaToMesh('112000mil2')`

ScrSetMinPlaneAreaToMesh

Sets the minimum plane area for meshing.	
UI Command:	**Simulation** > **Options** > **SI/PI Advanced**. Enter a value in the **Ignore planes smaller than** field.
Syntax:	`obj.ScrSetMinPlaneAreaToMesh(<minPlaneAreaToMesh>)`
Parameters:	BSTR minPlaneAreaToMesh (including unit of measure)
Return Value:	None.
VB Example:	`obj.ScrSetMinPlaneAreaToMesh "5769.2mil2"`
IPY Example:	`oDoc.ScrSetMinPlaneAreaToMesh('5769.2mil2')`

ScrSetMinRefinePassesInDcSimulation

Sets the minimum number of mesh refinement passes for DC IR drop simulations.	
UI Command:	**Simulation** > **Compute DC Current/Voltage**. In the **Adaptive Mesh Refinement Parameters** group, type a value in the **Minimum Number of Passes** field.
Syntax:	`obj.ScrSetMinRefinePassesInDcSimulation (<minPasses>)`
Parameters:	INT minPasses
Return Value:	None.
VB Example:	`obj.ScrSetMinRefinePassesInDcSimulation 1`
IPY Example:	`oDoc.ScrSetMinRefinePassesInDcSimulation(1)`

ScrSetNearFieldMeshingFrequencyDefault

For Near Field simulations, selects the Default meshing frequency.	
UI Command:	**Simulation** > **Compute Near Field**. For **Meshing Frequencies for the Observation Mesh**, select **Default**.
Syntax:	`obj.ScrSetNearFieldMeshingFrequencyDefault()`
Parameters:	None.
Return Value:	BOOL • **0** – Failure • **1** – Success
VB Example:	`obj.ScrSetNearFieldMeshingFrequencyDefault()`
IPY Example:	`oDoc.ScrSetNearFieldMeshingFrequencyDefault()`

ScrSetNearFieldMeshingFrequencyPoints

For Near Field simulations, selects the Points meshing frequency.	
UI Command:	**Simulation** > **Compute Near Field**. For **Meshing Frequencies for the Observation Mesh**, select **Points** and enter values.
Syntax:	`obj.ScrSetNearFieldMeshingFrequencyPoints (<freqPoints>)`
Parameters:	ARRAY freqPoints (contains strings *in ascending order*; uses Hz)
Return Value:	INT • **1** – Success • **2** – Array values aren't strings • **3** – One or more strings wasn't fully numeric • **4** – Duplicate value or bad order

VB Example:	`obj.ScrSetNearFieldMeshingFrequencyPoints freqPoints`
	`#Array of frequencies`
	`Dim freqPoints (3)`
	`freqPoints( 0 ) = "5e+10"`
	`freqPoints( 1 ) = "1e+11"`
	`freqPoints( 2 ) = "1.2e+11"`

IPY Example:	`oDoc.ScrSetNearFieldMeshingFrequencyPoints (['5e+10','1e+11','1.2e+11'])`

ScrSetNearFieldMeshingFrequencyRange

For Near Field simulations, selects the Range meshing frequency.	
UI Command:	**Simulation** > **Compute Near Field**. For **Meshing Frequencies for the Observation Mesh**, select **Points** and enter values.
Syntax:	`obj.ScrSetNearFieldMeshingFrequencyRange (<startFreq>,<stopFreq>)`
Parameters:	DOUBLE startFreq (in Hz) DOUBLE stopFreq (in Hz)
Return Value:	INT • **1** – Success • **Else** – Error
VB Example:	`obj.ScrSetNearFieldMeshingFrequencyRange (5e+10,1e+11)`
IPY Example:	`oDoc.ScrSetNearFieldMeshingFrequencyRange (5e+10,1e+11)`

ScrSetNearFieldSamplePointSpacing

Modifies the "Maximum Edge Length" option for Near Field simulations.	
UI Command:	**Simulation** > **Compute Near Field**. Set **Maximum Edge Length**.
Syntax:	`obj.ScrSetNearFieldSamplePointSpacing(<spacing>)`
Parameters:	DOUBLE spacing (in the project's native units)
Return Value:	BOOL: • **0** – Failure (spacing too small) • **1** – Success
VB Example:	`obj.ScrSetNearFieldSamplePointSpacing(132.034)`
IPY Example:	`oDoc.ScrSetNearFieldSamplePointSpacing(132.034)`

ScrSetNearFieldSolverOptions

For Near Field simulations, sets options for number of passes and error tolerance.	
UI Command:	**Simulation** > **Compute Near Field**. Under **Near Field Solver Options**, enter values for **Min. Adapt Passes**, **Max. Adapt Passes**, and **Global Error Tolerance**.
Syntax:	`obj.ScrSetNearFieldSolverOptions(<minAdaptPasses>, <maxAdaptPasses>, <gErrorTol>)`
Parameters:	INT minAdaptPasses INT maxAdaptPasses DOUBLE gErrorTol
Return Value:	BOOL: • **0** – Failure • **1** – Success
VB Example:	`obj.ScrSetNearFieldSolverOptions(1, 10, 0.5)`
IPY Example:	`oDoc.ScrSetNearFieldSolverOptions(1, 10, 0.5)`

ScrSetNearFieldSurfaceOffset

Controls the position of the surface over which near fields are computed. The positions are offset values in the design's length units with respect to the design's bounding cuboid.	
UI Command:	**Simulation** > **Compute Near Field**. Enter offset values in the **Cuboid Surface Positions** box.
Syntax:	`obj.ScrSetNearFieldSurfaceOffset(<px>, <nx>, <py>, <ny>, <pz>, <nz>)`
Parameters:	DOUBLE px (positive x) DOUBLE nx (negative x) DOUBLE py (positive y) DOUBLE ny (negative y) DOUBLE pz (positive z) DOUBLE nz (negative z)
Return Value:	BOOL • **0** – Failure • **1** – Success

VB Example:	`' places the near field surface 1mm above, below, front, behind, to the left and to the right of the design`
	`' (assuming the design's length units are mm)`
	`' outcome is TRUE on success, FALSE if any of the specified offset values are *NOT* positive`
	`outcome = obj.ScrSetNearFieldSurfaceOffset(1.0, 1.0, 1.0, 1.0, 1.0, 1.0)`

IPY Example:	`oDoc.ScrSetNearFieldSurfaceOffset(1.0, 1.0, 1.0, 1.0, 1.0, 1.0)`

ScrSetNumBondwireSidesInDcSimulation

For DC IR simulations, sets the number of Bondwire sides.	
UI Command:	**Simulation** > **Options**. On **DC Advanced** tab, set number for **Bondwire Discretization**.
Syntax:	`obj.ScrSetNumBondwireSidesInDcSimulation (<numBwSides>)`
Parameters:	INT numBwSides
Return Value:	None.
VB Example:	`obj.ScrSetNumBondwireSidesInDcSimulation(12)`
IPY Example:	`oDoc.ScrSetNumBondwireSidesInDcSimulation(12)`

ScrSetNumCpusToUse

Sets the number of CPUs (cores) to use during simulation.	
UI Command:	**Simulation** > **Options**. Click **Multiprocessing**. Select the **Number of CPUs to use when computing solution**.
Syntax:	`obj.ScrSetNumCpusToUse(<numCpus>)`
Parameters:	INT numCpus
Return Value:	None.
VB Example:	`obj.ScrSetNumCpusToUse 4`
IPY Example:	`oDoc.ScrSetNumCpusToUse(4)`

ScrSetNumModesToCompute

Sets the number of modes to solve for during resonant simulations.	
UI Command:	**Simulation** > **Compute Resonant Modes**. Enter a value in the **# of Modes to Compute** field.
Syntax:	`obj.ScrSetNumModesToCompute(<numModes>)`
Parameters:	INT numModes
Return Value:	None.
VB Example:	`obj.ScrSetNumModesToCompute 10`
IPY Example:	`oDoc.ScrSetNumModesToCompute(10)`

ScrSetNumViaSidesInDcSimulation

Sets the number of sides to use when generating polyhedral mesh approximations to cylindrical vias during DC IR drop simulations.	
UI Command:	**Simulation** > **Compute DC IR**. Click **Other solver options** button and choose the **DC Advanced** tab. Select the number of sides in the **Via Discretization** field.
Syntax:	`obj.ScrSetNumViaSidesInDcSimulation(<numViaSides>)`
Parameters:	INT numViaSides
Return Value:	None.
VB Example:	`' model vias as extruded regular octagons` `obj.ScrSetNumViaSidesInDcSimulation 8`
IPY Example:	`oDoc.ScrSetNumViaSidesInDcSimulation(8)`

ScrSetOptionsFor3DModelExport

Selects an options file for exporting a 3D model.	
UI Command:	**Export** > **3D Export Options**.
Syntax:	`obj.ScrSetOptionsFor3DModelExport(<filePath>)`
Parameters:	BSTR filePath
Return Value:	BOOL • **0** – Failure • **1** – Success
VB Example:	`obj.ScrSetOptionsFor3DModelExport "C:\Files\options.config"`
IPY Example:	`oDoc.ScrSetOptionsFor3DModelExport ('C:\Files\options.config')`

List of options and example values:

`NUM_PADS_FACET_COUNT 8`
`NUM_ANTIPAD_FACET_COUNT 8`
`DEFAULT_SOLDERBALL_FACET_COUNT 8`
`VIA_SEGMENTS 8`
`DEFAULT_BONDWIRE_FACET_COUNT 6`
`UNITE_NETS 0`

List of options and example values:
EXCLUDE_TERMINALS_FROM_UNITE 1
SIMPLY_VIA_MODEL 0
IGNORE_DIELECTRICS 0
SEPARATE_DIELECTRICS 1
UNITE_LAYERS_WITH_SAME_MATERIALS 1
IGNORE_UNCONNECTED_PADS 1
CLIP_TRACES 0
CUT_DIELECTRICS 0
CREATE_SHEET_BODIES 0
GENERATE_TERMINALS 0
IGNORE_PLANES_WITH_AREA_LESS_THAN_THRESOLD 1
IGNORE_FLOAT_BODIES 0
MIN_PLANE_AREA 0.358979
MIN_EDGE_LENGTH_PADS 1um
MIN_EDGE_LENGTH_PLANES 1um
MIN_EDGE_LENGTH_TRACES 1um
MIN_DIELECTRIC_EDGE_LENGTH 10um
DIELECTRIC_EXPANSION_FACTOR 0.100000
IGNORE_HOLES 1
MIN_HOLE_AREA 0.358979
REMOVE-PLATING_TAILS 0
SUBTRACT_METAL_FROM_SUBSTRATE 0
DISCRETIZE_ARCS 0
CHOP_TRACE_ENDS 0
AIRBOX_THICKNESS_FACTOR 1.100000
AIRBOX_PAD_AMOUNT_PLUS_Z 0.500000
AIRBOX_PAD_AMOUNT_MINUS_Z 0.500000

List of options and example values:
PORT_PAD_AMOUNT 0.500000
CREATE_PORTS_FOR_PWR_GND_NETS 0
PORTS_FOR_PWR_GND_NETS 0
LAUNCH_HFSS 1
USE_CAUSAL_MATERIALS 1
AUTO_DC_THICKNESS 1
HFSS_VERSION 2014
SOLVE_CAPACITANCE 1
SOLVE_DC_RESISTANCE 0
SOLVE_DC_INDUCTANCE_RESISTANCE 0
SOLVE_AC_INDUCTANCE_RESISTANCE 0
SOLVE_PROJECT 0
LAUNCH_Q3D 1
ASSIGN_SOLDER_BALLS_AS_SOURCES 0
Q3D_MERGE_SOURCES 0
Q3D_MERGE_SINKS 0
Q3D_VERSION 2014
ACIS_VERSION 0

ScrSetPadOnLayer

Changes an existing pad's shape or dimension; deletes an existing pad (by passing a shape of "None"); or adds a pad to the specified layer.	
UI Command:	**Home** > **Edit Padstacks**.
Syntax:	`obj.ScrSetPadOnLayer(<padstackName>, <layerName>, <shape>, <width>, <height>)`
Parameters:	BSTR padstackName BSTR layerName BSTR shape ("None", "Circle", "Oblong", or "Rectangle") BSTR width (including unit of measure) BSTR height (including unit of measure)
Return Value:	BOOL • **0** – Failure • **1** – Success
VB Example:	`obj.ScrSetPadOnLayer("VIA_M1_M2", "METAL-1", "Circle", "0.6mm", "0.6mm")`
IPY Example:	`oDoc.ScrSetPadOnLayer('VIA_M1_M2', 'METAL-1', 'Circle', '0.6mm', '0.6mm')`

ScrSetPadstackMaterial

Changes a padstack's material.	
UI Command:	**Home** > **Edit Padstacks**.
Syntax:	`obj.ScrSetPadstackMaterial(<padstackName>, <materialName>)`
Parameters:	BSTR padstackName BSTR materialName
Return Value:	BOOL • **0** – Failure • **1** – Success
VB Example:	`obj.ScrSetPadstackMaterial("VIA_M1_M2", "magnesium")`
IPY Example:	`oDoc.ScrSetPadstackMaterial('VIA_M1_M2', 'magnesium')`

ScrSetPadstackViaPlatingAbsolute

Changes a padstack's via plating absolute value.	
UI Command:	**Home** > **Edit Padstacks**. Select **Absolute** and set value.
Syntax:	`obj.ScrSetPadstackViaPlatingAbsolute (<padstackName>, <viaPlatingAbsolute>)`
Parameters:	BSTR padstackName BSTR viaPlatingAbsolute (including unit of measure; assumed to be meters if no unit specified)
Return Value:	BOOL • **0** – Failure • **1** – Success
VB Example:	`obj.ScrSetPadstackViaPlatingAbsolute("VIA_M1_M2", "0.1mm")`
IPY Example:	`oDoc.ScrSetPadstackViaPlatingAbsolute('VIA_M1_M2', '0.1mm')`

ScrSetPadstackViaPlatingRatio

Changes a padstack's via plating ratio.	
UI Command:	**Home** > **Edit Padstacks**. Select **Ratio** and select value.
Syntax:	`obj.ScrSetPadstackViaPlatingRatio(<padstackName>, <value>)`
Parameters:	BSTR padstackName DOUBLE value (percentage between 0 and 1; for example, 0.6 = 60%)
Return Value:	BOOL • **0** – Failure • **1** – Success
VB Example:	`obj.ScrSetPadstackViaPlatingRatio("VIA_M1_M2", 0.6)`
IPY Example:	`oDoc.ScrSetPadstackViaPlatingRatio('VIA_M1_M2', 0.6)`

ScrSetPlotAfterDcSimulation

Controls whether or not current/power/voltage plots are generated after DC IR drop simulations conclude.	
UI Command:	**Simulation** > **Compute DC IR**. Click **Other Solver Options** > **DC Advanced**. Select **Plot Current Density and Voltage Distribution**.
Syntax:	`obj.ScrSetPlotAfterDcSimulation(<plot>)`
Parameters:	BOOL plot (TRUE = plot; FALSE = do not plot)
Return Value:	None.
VB Example:	`obj.ScrSetPlotAfterDcSimulation True`
IPY Example:	`oDoc.ScrSetPlotAfterDcSimulation(True)`

ScrSetPlotLayers

Sets the layers specified for a Frequency Sweep.	
UI Command:	**Simulation** > **Compute Frequency Sweeps**. In the **Voltage Surface Plot Options** area, select layers.
Syntax:	`obj.ScrSetPlotLayers(<plotLayer>,<refLayer>)`
Parameters:	BSTR plotLayer BSTR refLayer (ground)
Return Value:	INT: • **0** – Success • **Else** – Failure
VB Example:	`obj.ScrSetPlotLayers "L2" "GND"`
IPY Example:	`oDoc.ScrSetPlotLayers('L2','GND')`

ScrSetPlotSyzMag

Selects "Plot Magnitude" for various simulations.	
UI Command:	**Results** > **[Simulation Type]** > **[Simulation Name]** > **Plot Magnitude**.
Syntax:	`obj.ScrSetPlotSyzMag(<flag>)`
Parameters:	BOOL flag (TRUE = plot magnitude; FALSE = do not plot magnitude)
Return Value:	None.
VB Example:	`obj.ScrSetPlotSyzMag True`
IPY Example:	`oDoc.ScrSetPlotSyzMag(1)`

ScrSetPlotSyzPhase

Selects "Plot Phase Animation" for various simulations.	
UI Command:	**Results** > **[Simulation Type]** > **[Simulation Name]** > **View Results** > **Phase Animation**.
Syntax:	`obj.ScrSetPlotSyzPhase(<flag>)`
Parameters:	BOOL flag (TRUE = plot phase animation; FALSE = do not plot phase animation)
Return Value:	None.
VB Example:	`obj.ScrSetPlotSyzPhase True`
IPY Example:	`oDoc.ScrSetPlotSyzPhase(1)`

ScrSetPortNamingConvention

Sets the port naming convention used in the following port generation commands: **ScrPlacePortsAtPinsOnSelectedNets, ScrPlacePortsAtPinsOnSelectedNetsPinNamesOut, ScrPlacePortsAtPinsOnSelectedNetsExcludePart**	
UI Command:	None.
Syntax:	`obj.ScrSetPortNamingConvention (<namingConvention>)`
Parameters:	BSTR namingConvention
Return Value:	BOOL • **0** – Failure • **1** – Success
VB Example:	`obj.ScrSetPortNamingConvention("TestPort_ $POSTERMINAL_$NETNAME_Test")`
IPY Example:	`oDoc.ScrSetPortNamingConvention('TestPort_ $POSTERMINAL_$NETNAME_Test')`

ScrSetPowerGroundNets

Selects nets to be designated power and ground nets.	
UI Command:	**Power/Ground Identification** in **Nets** window.
Syntax:	`obj.ScrSetPowerGroundNets(<netNames>, <appendToCurrentNetsSelected>)`
Parameters:	ARRAY netNames BOOL appendToCurrentNetsSelected (TRUE = nets already set to power/ground remain power/ground; FALSE = all nets not in the array are set to non-power/ground.)
Return Value:	BOOL: • **0** – Failure • **1** – Success

VB Example:	`dim netNames(3)`
	`netNames(0)="Net 1"`
	`netNames(1)="Net 2"`
	`netNames(2)="Net 3"`
	`outcome = obj.ScrSetPowerGroundNets(netNames, TRUE)`

IPY Example:	`oDoc.ScrSetPowerGroundNets (['Net1','Net2','Net3'],1)`

ScrSetPowerGroundNetsFromFile

Selects nets to be designated power and ground nets.	
UI Command:	**Power/Ground Identification** in **Nets** window.
Syntax:	`obj.ScrSetPowerGroundNetsFromFile(<filePath>, <appendToCurrentNetsSelected>)`
Parameters:	BSTR filePath BOOL appendToCurrentNetsSelected (TRUE = nets already set to power/ground remain power/ground; FALSE = all nets not in the file are set to non-power/ground.)
Return Value:	None.
VB Example:	`obj.ScrSetPowerGroundNetsFromFile("C:\Files\power_ground_nets.txt", TRUE)`
IPY Example:	`oDoc.ScrSetPowerGroundNetsFromFile('C:\Files\power_ground_nets.txt',1)`

ScrSetProjectModified

Determines whether you will be prompted to save before closing or loading a different project.	
UI Command:	None.
Syntax:	`obj.ScrSetProjectModified(<p>)`
Parameters:	BOOL p (TRUE = sets project as modified, will be prompted to save; FALSE = no save prompt)
Return Value:	None.
VB Example:	`obj.ScrSetProjectModified True`
IPY Example:	`oDoc.ScrSetProjectModified(1)`

ScrSetPsiOptionsFromFile

Sets general PSI simulation options from an XML configuration file.	
UI Command:	**Simulation** > **PSI Options** > **Import Settings**. Select file.
Syntax:	`obj.ScrSetPsiOptionsFromFile(<filename>)`
Parameters:	BSTR filename (full path)
Return Value:	BOOL • **0** – Failure • **1** – Success
VB Example:	`obj.ScrSetPsiOptionsFromFile("D:\Tests\Test.sps")`
IPY Example:	`oDoc.ScrSetPsiOptionsFromFile('D:\Tests\Test.sps')`

ScrSetPsiPortType

Sets the PSI port type for a specified port.	
UI Command:	None.
Syntax:	`obj.ScrSetPsiPortType(<portName>, <portType>)`
Parameters:	BSTR portName BSTR portType (See: SIwave-PSI Best Practices - Port Setup)
Return Value:	BOOL • **0** – Failure • **1** – Success
VB Example:	`obj.ScrSetPsiPortType("Port1","Lumped")`
IPY Example:	`oDoc.ScrSetPsiPortType('Port1','Lumped')`

ScrSetPsiSyzInterpOptions

Sets PSI SYZ Simulation options.	
UI Command:	**Simulation** > **PSI** > **Compute SYZ Parameters**.
Syntax:	`obj.ScrSetPsiSyzInterpOptions(<interp>, <fastsweep>, <adaptiveSamp>, <enforceDC>)`
Parameters:	INT interp (0 = Discrete Sweep; 1 = Interpolating Sweep) INT fastsweep (0 = Adaptive Sampling; 1 = Fast Sweep) INT adaptiveSamp (0 = Fast Sweep; 1 = Adaptive Sampling) INT enforceDC (for Adaptive Sampling, 0 = do not enforce DC point and causality; 1 = enforce DC point and causality)
Return Value:	BOOL • **0** – Failure • **1** – Success
VB Example:	`obj.ScrSetPsiSyzInterpOptions(1, 0, 1, 1)`
IPY Example:	`oDoc.ScrSetPsiSyzInterpOptions(1, 0, 1, 1)`

ScrSetRefineBondwiresInDcSimulation

For DC simulations, selects or deselects option to refine mesh along bondwires.	
UI Command:	**Simulation** > **Options**. On **DC Advanced** tab, select or deselect **Refine Mesh Along Bondwires**.
Syntax:	`obj.ScrSetRefineBondwiresInDcSimulation (<refineBws>)`
Parameters:	BOOL refineBws (TRUE = select; FALSE = deselect)
Return Value:	None.
VB Example:	`obj.ScrSetRefineBondwiresInDcSimulation True`
IPY Example:	`oDoc.ScrSetRefineBondwiresInDcSimulation(1)`

ScrSetRefineDcSimulation

Activates or deactivates adaptive mesh refinement during DC IR drop simulations.	
UI Command:	**Simulation** > **Compute DC IR**. Click **Other Solver Options** > **DC Advanced**. Select or deselect **Perform Adaptive Mesh Refinement**.
Syntax:	`obj.ScrSetRefineDcSimulation(<refine>)`
Parameters:	INT refine (0 = deselect refinement; 1 = select refinement)
Return Value:	None.
VB Example:	`obj.ScrSetRefineDcSimulation(1)`
IPY Example:	`oDoc.ScrSetRefineDcSimulation(1)`

ScrSetRefineViasInDcSimulation

For DC simulations, selects or deselects option to refine mesh along vias.	
UI Command:	**Simulation** > **Options**. On **DC Advanced** tab, select or deselect **Refine Mesh Along Vias**.
Syntax:	`obj.ScrSetRefineViasInDcSimulation(<refineVias>)`
Parameters:	BOOL refineBws (TRUE = select; FALSE = deselect)
Return Value:	None.
VB Example:	`obj.ScrSetRefineViasInDcSimulation True`
IPY Example:	`oDoc.ScrSetRefineViasInDcSimulation(1)`

ScrSetRemoveCutoutsByArea

Selects whether SIwave will always preserve cutouts whose area is greater than the minimum void area (controlled by the ScrSetMinCutoutArea function). By default, SIwave automatically decides which cutouts are to be defeatured (i.e., not meshed) during simulation.	
UI Command:	**Simulation** > **Options**. Click **SI/PI Advanced**. Select or deselect **Automatic** Mesh Refinement.
Syntax:	`obj.ScrSetRemoveCutoutsByArea <p>`
Parameters:	INT p (0 = automatic mesh refinement; 1 = no automatic mesh refinement)
Return Value:	None.
VB Example:	`obj.ScrSetRemoveCutoutsByArea(1)`
IPY Example:	`oDoc.ScrSetRemoveCutoutsByArea(1)`

ScrSetResonantModeMaxFreq

For Resonant Modes simulations, sets the maximum frequency for analysis.	
UI Command:	**Simulation** > **Compute Resonant Modes**. Enter a **Maximum Frequency** value, in Hz.
Syntax:	`obj.ScrSetResonantModeMaxFreq(<freq>)`
Parameters:	DOUBLE freq (in Hz)
Return Value:	None.
VB Example:	`obj.ScrSetResonantModeMaxFreq(2E+06)`
IPY Example:	`oDoc.ScrSetResonantModeMaxFreq(2E+06)`

ScrSetResonantModeMinFreq

For Resonant Modes simulations, sets the minimum frequency for analysis.	
UI Command:	**Simulation** > **Compute Resonant Modes**. Enter a **Minimum Frequency** value, in Hz.
Syntax:	`obj.ScrSetResonantModeMinFreq(<freq>)`
Parameters:	DOUBLE freq (in Hz)
Return Value:	None.
VB Example:	`obj.ScrSetResonantModeMinFreq(2E+06)`
IPY Example:	`oDoc.ScrSetResonantModeMinFreq(2E+06)`

ScrSetRLCValues

Changes RLC values for a specified RLC.	
UI Command:	None.
Syntax:	`obj.ScrSetRLCValues(<partName>, <r>, <l>, <c>)`
Parameters:	BSTR partName BSTR r (including unit of measure) BSTR l (including unit of measure) BSTR c (including unit of measure)
Return Value:	BOOL • **0** – Failure • **1** – Success
VB Example:	`obj.ScrSetRLCValues "RLC_XYZ_R", "1.5kohm", "1e-12h", "1uf"`
IPY Example:	`oDoc.ScrSetRLCValues('RLC_XYZ_R', '1.5kohm', '1e-12h', '1uf')`

ScrSetSignalNets

Selects nets to be designated signal nets.	
UI Command:	**Simulation** > **Signal Net Analyzer**. Select nets.
Syntax:	`obj.ScrSetSignalNets(<netNames>, <appendToCurrentNetsSelected>)`
Parameters:	ARRAY netNames BOOL appendToCurrentNetsSelected (TRUE = nets already set as signal nets remain so; FALSE = all nets not in the array are set to non-signal nets.)
Return Value:	BOOL: • **0** – Failure • **1** – Success

VB Example:	`dim netNames(3)`
	`netNames(0)="Net 1"`
	`netNames(1)="Net 2"`
	`netNames(2)="Net 3"`
	`outcome = obj.ScrSetSignalNets(netNames, TRUE)`

IPY Example:	`oDoc.ScrSetSignalNets(['Net1','Net2','Net3'],1)`

ScrSetSignalNetsFromFile

Selects nets to be designated signal nets.	
UI Command:	**Simulation** > **Signal Net Analyzer**. Select nets.
Syntax:	`obj.ScrSetSignalNetsFromFile(<filePath>, <appendToCurrentNetsSelected>)`
Parameters:	BSTR filePath BOOL appendToCurrentNetsSelected (TRUE = nets already set as signal nets remain so; FALSE = all nets not in the file are set to non-signal nets.)
Return Value:	BOOL: • **0** – Failure • **1** – Success
VB Example:	`obj.ScrSetSignalNetsFromFile("C:\Files\signal_nets.txt", TRUE)`
IPY Example:	`oDoc.ScrSetSignalNetsFromFile('C:\Files\signal_nets.txt',1)`

ScrSetSimulationName

Sets the simulation name for a specified simulation type.	
UI Command:	From the **Simulation** menu, click any option to compute a simulation and open the corresponding simulation window. Specify the simulation name.
Syntax:	`obj.ScrSetSimulationName(<simType>, <simName>)`
Parameters:	BSTR simType ("ac", "dc", "eigen", "ff", "nf", "syz" or "hfss_syz") BSTR simName
Return Value:	BOOL • **0** – Failure • **1** – Success
VB Example:	`' outcome is TRUE on success, FALSE if specified "simType" does not match one of the six strings above` `outcome = obj.ScrSetSimulationName ( "syz", "S-param no caps" )`
IPY Example:	`oDoc.ScrSetSimulationName('syz', 'S-param no caps')`

ScrSetSketchedBwProfile

Sets a specified sketched profile to all bondwires of a specified model.	
UI Command:	From the **Simulation** menu, click any option to compute a simulation and open the corresponding simulation window. Specify the simulation name.
Syntax:	`obj.ScrSetSketchedBwProfile(<bwModelName>, <filePath>, <radius>)`
Parameters:	BSTR bwModelName BSTR filePath DOUBLE radius
Return Value:	BOOL • **0** – Failure • **1** – Success
VB Example:	`obj.ScrSetSketchedBwProfile "WB_PROFILE_1", "SketchedProfile.bwp", 20`
IPY Example:	`oDoc.ScrSetSketchedBwProfile('WB_PROFILE_1', 'SketchedProfile.bwp', 20)`

Format of *.bwp:

UNITS microns

0.000000 400.000000

300.000000 700.000000

900.000000 700.000000

1200.000000 300.000000

ScrSetSketchedBwProfileFromArray

Sets a sketched profile (array of points) to all bondwires of a specified model.	
UI Command:	None.
Syntax:	`obj.ScrSetSketchedBwProfileFromArray(<modelName>, <unitName>, <bwPointArray>, <radius>)`
Parameters:	BSTR modelName BSTR unitName ARRAY bwPointArray (requires 8 doubles for 4 points) DOUBLE (radius)
Return Value:	BOOL: • **0** – Failure • **1** – Success

VB Example:	`obj.ScrSetSketchedBwProfileFromArray "BW_PROFILE_1", "micron", bwPointArray, 20`
	`Array of 4 point sketched profile`
	`Dim bwPoints (8)`
	`bwPoints( 0 ) = 0.0`
	`bwPoints( 1 ) = 400.0`
	`bwPoints( 2 ) = 300.0`
	`bwPoints( 3 ) = 700.0`
	`bwPoints( 4 ) = 900.0`
	`bwPoints( 5 ) = 700.0`
	`bwPoints( 6 ) = 1200.0`
	`bwPoints( 7 ) = 300.0`

IPY Example:	`oDoc.ScrSetSketchedBwProfileFromArray('BW_PROFILE_1', 'micron', [0.0,400.0,300.0,700.0,900.0,700.0,1200.0,300.0], 20)`

ScrSetSnapLengthThreshold

For DC simulations, sets the maximum length value for snapping vertices.	
UI Command:	**Simulation** > **Options**. On **DC Advanced** tab, enter a value in the **Snap vertices separated by less than** field.
Syntax:	`obj.ScrSetSnapLengthThreshold (<snapLengthThreshold>)`
Parameters:	BSTR snapLengthThreshold
Return Value:	None.
VB Example:	`obj.ScrSetSnapLengthThreshold("0.0734235mil")`
IPY Example:	`oDoc.ScrSetSnapLengthThreshold('0.0734235mil')`

ScrSetSolderballMaterial

Assigns a material to a specified solderball. **NOTE: You *must* assign a solderball profile before assigning a material.**	
UI Command:	**Home** > **Solderball Properties**. Assign a material.
Syntax:	`obj.ScrSetSolderballMaterial(<padstackName>, <material>)`
Parameters:	BSTR padstackName BSTR material
Return Value:	BOOL • **0** – Failure • **1** – Success
VB Example:	`obj.ScrSetSolderballMaterial("BP_BOT_500X600", "silver")`
IPY Example:	`oDoc.ScrSetSolderballMaterial('BP_BOT_500X600', 'silver')`

ScrSetSolderballParameters

Assigns solderballs/bumps of the specified dimensions (in the design's geometry units) to the specified padstack.	
UI Command:	**Home** > **Solderball Properties**. Select a padstack and assign solderballs/bumps.
Syntax:	`obj.ScrSetSolderballParameters(<padstackName>, <aboveStackup>, <height>, <radius>)`
Parameters:	BSTR padstackName BOOL/INT aboveStackup (TRUE/1 = above stackup; FALSE/0 = below stackup) DOUBLE height DOUBLE radius
Return Value:	BOOL • **0** – Failure • **1** – Success

VB Example:	`' creates solder bumps at the "BALL600" padstack that are 500um tall and have a radius of 200um`
	`' (assuming the design units are um)`
	`' outcome is TRUE on success, FALSE if a padstack with the specified name could not be located`
	`outcome = obj.ScrSetSolderballParameters ( "BALL600", False, 500, 200 )`
	`' creates solder bumps at the "BUMPPAD" padstack that are 100um tall and have a radius of 30um`
	`' (assuming the design units are um)`
	`outcome = obj.ScrSetSolderballParameters ( "BUMPPAD", True, 100, 30 )`

IPY Example:	`oDoc.ScrSetSolderballParameters('BUMPPAD', 1, 100, 30)`

ScrSetSourceMagnitude

Sets the current or voltage magnitude of a specified source.	
UI Command:	None.
Syntax:	`obj.ScrSetSourceMagnitude(<refDes>, <magnitude>)`
Parameters:	BSTR refDes BSTR magnitude (including unit)
Return Value:	BOOL • **0** – Failure • **1** – Success
VB Example:	`outcome = obj.ScrSetSourceMagnitude ("I_1", "2.5A")`
IPY Example:	`oDoc.ScrSetSourceMagnitude('I_1', '2.5A')`

ScrSetSparamModelSetup

Configures an N-Port S-parameter model for specified part(s).	
UI Command:	None.
Syntax:	`obj.ScrSetSparamModelSetup(<partName>, <activeRefDesList>, <fileName>, <modelName>, <refNet>, <pinOrder>)`
Parameters:	BSTR partName ARRAY activeRefDesList BSTR fileName (including path) BSTR modelName BSTR refNet ARRAY pinOrder
Return Value:	BOOL • **0** – Failure • **1** – Success

VB Example:	`Dim refdes1(0)`
	`refdes1(0) = "A1"`
	`Dim pinorder1()`
	`doc.ScrSetSparamModelSetup "TESTPART", refdes1, "D:\Tests\US142197\DLP11TB800UL2.s4p", "Model1", "NET_9", pinorder1`

IPY Example:	`refdes1 = ['A1']`
	`pinorder1 = []`
	`doc.ScrSetSparamModelSetup('TESTPART', refdes1, 'D:\Tests\US142197\DLP11TB800UL2.s4p', 'Model1', 'NET_9', pinorder1)`

ScrSetSpiceModelSetup

Configures an N-Port Spice model for specified part(s).	
UI Command:	None.
Syntax:	`obj.ScrSetSpiceModelSetup(<partName>, <activeRefDesList>, <fileName>, <modelName>, <pinOrder>)`
Parameters:	BSTR partName ARRAY activeRefDesList BSTR fileName (including path) BSTR modelName ARRAY pinOrder
Return Value:	BOOL • **0** – Failure • **1** – Success

VB Example:	`Dim refdes2(0)`
	`refdes2(0) = "A2"`
	`Dim pinorder2(3`
	`pinorder2(0) = "4"`
	`pinorder2(1) = "3"`
	`pinorder2(2) = "2"`
	`pinorder2(3) = "1"`
	`obj.ScrSetSpiceModelSetup "TESTPART", refdes2, "D:\Tests\US142197\testmod.sp", "Model2", pinorder2`

IPY Example:	`refdes2 = ['A2']`
	`pinorder2 = ['4', '3', '2', '1']`
	`oDoc.ScrSetSpiceModelSetup('TESTPART', refdes2, 'D:\Tests\US142197\testmod.sp', 'Model2', pinorder2)`

ScrSetSpiceSubcktFormat

Sets the desired output format for Full Wave Spice simulations.	
UI Command:	None.
Syntax:	`obj.ScrSetSpiceSubcktFormat(<format>)`
Parameters:	BSTR format (case-sensitive choices are: HSPICE, MSPICE, PSPICE, or Spectre)
Return Value:	None.
VB Example:	`obj.ScrSetSpiceSubcktFormat "HSPICE"`
IPY Example:	`oDoc.ScrSetSpiceSubcktFormat('PSPICE')`

ScrSetStackupLayerThickness

Sets the thickness of a specified layer.	
UI Command:	**Home** > **Layer Stackup Editor**. Edit a layer's **Thickness** value.
Syntax:	`obj.ScrSetStackupLayerThickness(<layerName>, <thickness>, <redraw>)`
Parameters:	BSTR layerName BSTR thickness (including unit) BOOL redraw (TRUE/1 = redraw; FALSE/0 = do not redraw)
Return Value:	BOOL: • **0** – Failure • **1** – Success
VB Example:	`obj.ScrSetStackupLayerThickness("LAYER-1", "0.79mils", True)`
IPY Example:	`oDoc.ScrSetStackupLayerThickness('LAYER-1', '0.79mils', 1)`

ScrSetStackupLayerThicknessUnit

Sets the layer thickness unit.	
UI Command:	None.
Syntax:	`obj.ScrSetStackupLayerThicknessUnit(<unit>)`
Parameters:	BSTR unit ("microns", "um", "mils", "mm", "cm", "inches", or "meters")
Return Value:	BOOL: • **0** – Failure • **1** – Success
VB Example:	`obj.ScrSetStackupLayerThicknessUnit "um"`
IPY Example:	`oDoc.ScrSetStackupLayerThicknessUnit('um')`

ScrSetSweepFreqRange

Sets the minimum and maximum frequencies for a frequency sweep.	
UI Command:	None.
Syntax:	`obj.ScrSetSweepFreqRange(<minFreq>, <maxFreq>)`
Parameters:	DOUBLE minFreq (in Hz; 1.0 = 1Hz, 1000.0 = 1kHz, 1e-6 = 1MHz; 1e-9 = 1GHz) DOUBLE maxFreq (in Hz; 1.0 = 1Hz, 1000.0 = 1kHz, 1e-6 = 1MHz; 1e-9 = 1GHz)
Return Value:	None.
VB Example:	`obj.ScrSetSweepFreqRange(5.0,5.0)`
IPY Example:	`oDoc.ScrSetSweepFreqRange(5.0,5.0)`

ScrSetSweepMaxFreq

Sets the maximum frequency for a frequency sweep.	
UI Command:	None.
Syntax:	`obj.ScrSetSweepMaxFreq(<maxFreq>)`
Parameters:	DOUBLE maxFreq (in Hz; 1.0 = 1Hz, 1000.0 = 1kHz, 1e-6 = 1MHz; 1e-9 = 1GHz)
Return Value:	None.
VB Example:	`obj.ScrSetSweepMaxFreq(5.0)`
IPY Example:	`oDoc.ScrSetSweepMaxFreq(5.0)`

ScrSetSweepMinFreq

Sets the minimum frequency for a frequency sweep.	
UI Command:	None.
Syntax:	`obj.ScrSetSweepMinFreq(<minFreq>)`
Parameters:	DOUBLE minFreq (in Hz; 1.0 = 1Hz, 1000.0 = 1kHz, 1e-6 = 1MHz; 1e-9 = 1GHz)
Return Value:	None.
VB Example:	`obj.ScrSetSweepMinFreq(5.0)`
IPY Example:	`oDoc.ScrSetSweepMinFreq(5.0)`

ScrSetSweepNumFreqPoints

Sets the number of frequency points for a frequency sweep.	
UI Command:	None.
Syntax:	`obj.ScrSetNumSweepFreqPoints(<numPoints>)`
Parameters:	INT numPoints
Return Value:	None.
VB Example:	`obj.ScrSetSweepNumFreqPoints(200)`
IPY Example:	`oDoc.ScrSetSweepNumFreqPoints(200)`

ScrSetSyzInterpSweep

Sets the SYZ-parameter sweep type to interpolating.	
UI Command:	**Simulation** > **Compute SYZ Parameters**. Select **Interpolating Sweep**.
Syntax:	`obj.ScrSetSyzInterpSweep(<p>)`
Parameters:	BOOL p (TRUE = Interpolating; FALSE = Discrete)
Return Value:	None.

VB Example:	`' switch to interpolating sweeps`
	`obj.ScrSetSyzInterpSweep True`
	`' switch back to discrete sweeps`
	`obj.ScrSetSyzInterpSweep False`

IPY Example:	`oDoc.ScrSetSyzInterpSweep(True)`

ScrSetSyzInterpSweepParams

Sets the convergence criterion (error tolerance) and maximum number of points to use during SYZ interpolating sweeps.	
UI Command:	**Simulation** > **Compute SYZ Parameters**. Select **Interpolating Sweep** and enter a value for **Error Tolerance**.
Syntax:	`obj.ScrSetSyzInterpSweepParams(<convergence>, <maxInterpPts>)`
Parameters:	DOUBLE convergence INT maxInterpPts
Return Value:	BOOL • **0** – Failure • **1** – Success
VB Example:	`' outcome is TRUE on success, FALSE if "maxInterpPts" exceeds the number of discrete sweep points` `outcome = obj.ScrSetSyzInterpSweepParams ( 0.005, 150)`
IPY Example:	`oDoc.ScrSetSyzInterpSweepParams(0.005, 150)`

ScrSetTDCrosstalkScanParameters

Specifies the parameters for running a time domain Crosstalk Scan via ScrRunSimulation.	
UI Command:	**Simulation** > **Crosstalk Scan** > **Time Domain**. Set values for solver options.
Syntax:	`obj.ScrSetTDCrosstalkScanParameters(<partNameList>, <refDesNameList>, <pinNameList>, <impedanceList>, <typeList>, <riseTimeList>, <voltageList>)`
Parameters:	ARRAY partNameList ARRAY refDesNameList ARRAY pinNameList ARRAY impedanceList ARRAY typeList ("1" represents Driver; "0" represents Receiver) ARRAY riseTimeList ARRAY voltageList **Each input parameter is an array of strings, and the number of elements in the array should be the same for all parameters. The first 3 are used to identify the pin and the rest are parameters to apply for the pin.
Return Value:	BOOL • **0** – Failure • **1** – Success
VB Example:	`obj.ScrSetTDCrosstalkScanParameters(parts, refdes, pins, impedances, types, risetimes, voltages)`

IPY Example:	`parts = ["G83568-001", "IPD031-201"]`
	`refdes = ["U1A1", "U2A5"]`
	`pins = ["K7", "9"]`
	`impedances = ["55ohms", "60"]`
	`types = ["0", "1"]`
	`risetimes = ["0", "6ns"]`
	`voltages = ["0", "1.2V"]`
	`oDoc.ScrSetTDCrosstalkScanParameters(parts, refdes, pins, impedances, types, risetimes, voltages)`

ScrSetThermalPadOnLayer

Changes, adds, or deletes thermal pads of a specified padstack. **If no thermal pad exists, script adds thermal pads to the specified padstack.** **If thermal pad exists, script modifies pad shape and/or dimensions.** **To delete a pad, choose "None" as the `shapeName`.**	
UI Command:	**Home** > **Edit Padstack**. Select **Thermal Relief Pag Properties**.
Syntax:	`obj.ScrSetThermalPadOnLayer(<padstackName>, <layerName>, <shapeName>, <widthString>, <heightString>)`
Parameters:	BSTR padstackName BSTR layerName BSTR shapeName ("None", "Circle", "Oblong", or "Rectangle") BSTR widthString (in mils if no unit specified; for circle pads, this is the radius) BSTR heightString (in mils if no unit specified)
Return Value:	BOOL • **0** – Failure • **1** – Success
VB Example:	`obj.ScrSetThermalPadOnLayer("VIA_M1_M2", "METAL-1", "Rectangle", "1.2mm", "1.2mm")`
IPY Example:	`oDoc.ScrSetThermalPadOnLayer('VIA_M1_M2', 'METAL-1', 'Rectangle', '1.2mm', '1.2mm')`

ScrSetTouchstoneExportFormatToDb

Sets the magnitude format in exported touchstone files.	
UI Command:	None.
Syntax:	`obj.ScrSetTouchstoneExportFormatToDb(<exportInDb>)`
Parameters:	BOOL exportInDb (TRUE = magnitude is specified in decibels [log of the magnitude]; FALSE = default settings)
Return Value:	None.
VB Example:	`obj.ScrSetTouchstoneExportFormatToDb True`
IPY Example:	`oDoc.ScrSetTouchstoneExportFormatToDb(1)`

ScrSetTouchstonePortOrder

Specifies port order in exported touchstone files. **NOTE: If any ports are not specified in the input, they will be sorted in alphabetical order and added to the end of the list.**	
UI Command:	None.
Syntax:	`obj.ScrSetTouchstonePortOrder(<portNamesList>)`
Parameters:	ARRAY portNamesList
Return Value:	BOOL • **0** – Failure • **1** – Success

VB Example:	`Dim ports(1)`
	`ports(0) = "port1"`
	`ports(1) = "port2"`
	`outcome = doc.ScrSetTouchstonePortOrder ports`

IPY Example:	`oDoc.ScrSetTouchstonePortOrder(['port1','port2'])`

ScrSetTouchstonePortRemapping

Sets the alternate port naming convention when name remapping is enabled in touchstone file exports.	
UI Command:	None.
Syntax:	`obj.ScrSetTouchstonePortRemapping(<portName>, <namingConv>)`
Parameters:	BSTR portName BSTR namingConv
Return Value:	BOOL • **0** – Failure • **1** – Success
VB Example:	`outcome = obj.ScrSetTouchstonePortRemapping "left", "foo"`
IPY Example:	`oDoc.ScrSetTouchstonePortRemapping('left', 'foo')`

ScrSetTraceCouplingDistance

Sets the XY coupling distance.	
UI Command:	**Simulation** > **Options**. On **CPA** tab, enter a value for **XY coupling distance**.
Syntax:	`obj.ScrSetTraceCouplingDistance (<traceCouplingDist>, <units>)`
Parameters:	DOUBLE traceCouplingDist BSTR units
Return Value:	None.
VB Example:	`obj.ScrSetTraceCouplingDistance("7.79", "mils")`
IPY Example:	`oDoc.ScrSetTraceCouplingDistance('7.79', 'mils')`

ScrSetUniformTemperature

Sets the uniform design temperature.	
UI Command:	**Simulation** > **Options**. On the **SI/PI** tab, enter a value in the **Set uniform design temperature to** field.
Syntax:	`obj.ScrSetUniformTemperature(<temp>)`
Parameters:	BSTR temp (temperature, including units)
Return Value:	None.
VB Example:	`obj.ScrSetUniformTemperature "100cel"`
IPY Example:	`oDoc.ScrSetUniformTemperature('100cel')`

ScrSetZ0ScanParameters

Specifies the parameters for running an Impedance Scan via ScrRunSimulation.	
UI Command:	**Simulation** > **Impedance Scan**. Set **Nominal Z0**, **Issue warning...** and **Issue violation...** values.
Syntax:	`obj.ScrSetZ0ScanParameters(<impedance>, <warningThreshold>, <violationThreshold>)`
Parameters:	DOUBLE impedance DOUBLE warningThreshold DOUBLE violationThreshold
Return Value:	BOOL • **0** – Failure • **1** – Success
VB Example:	`obj.ScrSetZ0ScanParameters(50,15,30)`
IPY Example:	`oDoc.ScrSetZ0ScanParameters(50,15,30)`

ScrSetZ0ScanReportImageHeight

When generating Impedance Scan reports, specifies the resolution (in pixels) of the smaller dimension of the images. For designs where the board is wider than it is tall, this corresponds to the y-resolution of the image.	
UI Command:	None.
Syntax:	`obj.ScrSetZ0ScanReportImageHeight(<imgHeight>)`
Parameters:	LONG imgHeight
Return Value:	BOOL • **0** – Failure (invalid height) • **1** – Success
VB Example:	`obj.ScrSetZ0ScanReportImageHeight(3000)`
IPY Example:	`oDoc.ScrSetZ0ScanReportImageHeight(3000)`

ScrShowSelectedNetsOnly

Determines whether SIwave displays all nets or selected nets only.	
UI Command:	None.
Syntax:	`obj.ScrShowSelectedNetsOnly(<selOnly>)`
Parameters:	BOOL selOnly (TRUE/1 = display only selected nets; FALSE/0 = display all nets)
Return Value:	None.
VB Example:	`obj.ScrShowSelectedNetsOnly(True)`
IPY Example:	`oDoc.ScrShowSelectedNetsOnly(1)`

ScrSIwaveEnable_3D_DDM

Enables or disables 3D DDM.	
UI Command:	None.
Syntax:	`obj.ScrSIwaveEnable_3D_DDM(<flag>)`
Parameters:	BOOL flag
Return Value:	None.
VB Example:	`obj.ScrSIwaveEnable_3D_DDM TRUE`
IPY Example:	`oDoc.ScrSIwaveEnable_3D_DDM(True)`

ScrSIwaveEnableHFSSRegions

Enables or disables HFSS Regions **Note: Region extents must exist in the layout for this setting to be meaningful.**	
UI Command:	None.
Syntax:	`obj.ScrSIwaveEnableHFSSRegions(<flag>)`
Parameters:	BOOL flag
Return Value:	None.
VB Example:	`obj.ScrSIwaveEnableHFSSRegions TRUE`
IPY Example:	`oDoc.ScrSIwaveEnableHFSSRegions(True)`

ScrSIwaveEnableReturnCurrentDistribution

Enables or disables tracing of return current distribution.	
UI Command:	**Simulation** > **Options**. On **SI/PI Advanced** tab, select **Trace return current distribution**.
Syntax:	`obj.ScrSIwaveEnableReturnCurrentDistribution (<flag>)`
Parameters:	BOOL flag (TRUE/1 = select; FALSE/0 = deselect)
Return Value:	None.
VB Example:	`obj.ScrSIwaveEnableReturnCurrentDistribution(True)`
IPY Example:	`oDoc.ScrSIwaveEnableReturnCurrentDistribution(1)`

ScrSIwaveIncludeSourceParasitics

For Resonance Sweep/SYZ simulations, enables or disables option to include source parasitics.	
UI Command:	**Simulation** > **Options**. On **SI/PI Advanced** tab, select **Include Voltage/Current Source Connections/Parasitics in Resonance/SYZ Simulations**.
Syntax:	`obj.ScrSIwaveIncludeSourceParasitics(<flag>)`
Parameters:	BOOL flag (TRUE/1 = include; FALSE/0 = do not include)
Return Value:	None.
VB Example:	`obj.ScrSIwaveIncludeSourceParasitics(True)`
IPY Example:	`oDoc.ScrSIwaveIncludeSourceParasitics(1)`

ScrSIwaveSyzComputeExactDcPoint

For SYZ simulations, selects or deselects the Compute Exact DC Point option.	
UI Command:	**Simulation** > **Compute SYZ Parameters**. Select **Compute Exact DC Point**.
Syntax:	`obj.ScrSIwaveSyzComputeExactDcPoint(<flag>)`
Parameters:	BOOL flag (TRUE = Compute exact DC point; FALSE = Do not compute exact DC point)
Return Value:	None.
VB Example:	`obj.ScrSIwaveSyzComputeExactDcPoint TRUE`
IPY Example:	`oDoc.ScrSIwaveSyzComputeExactDcPoint(True)`

Important:

The **Compute Exact DC Point** and **Enforce Causality** options are mutually exclusive. Selecting both generates an error.

ScrSIwaveSyzEnforceCausality

For SYZ simulations, selects or deselects the Enforce Causality option.	
UI Command:	**Simulation** > **Compute SYZ Parameters**. Select **Enforce Causality**.
Syntax:	`obj.ScrSIwaveSyzEnforceCausality(<flag>)`
Parameters:	BOOL flag (TRUE = Enforce; FALSE = Do not enforce)
Return Value:	None.
VB Example:	`obj.ScrSIwaveSyzEnforceCausality TRUE`
IPY Example:	`oDoc.ScrSIwaveSyzEnforceCausality(True)`

Important:

The **Enforce Causality** and **Compute Exact DC Point** options are mutually exclusive. Selecting both generates an error.

ScrSIwaveSyzEnforcePassivity

For SYZ simulations, selects or deselects the Enforce Passivity option.	
UI Command:	**Simulation** > **Compute SYZ Parameters**. Select **Enforce Passivity**.
Syntax:	`obj.ScrSIwaveSyzEnforcePassivity(<flag>)`
Parameters:	BOOL flag (TRUE = Enforce; FALSE = Do not enforce)
Return Value:	None.
VB Example:	`obj.ScrSIwaveSyzEnforcePassivity TRUE`
IPY Example:	`oDoc.ScrSIwaveSyzEnforcePassivity(True)`

ScrUnselectAll

Deselects all selected objects.	
UI Command:	Right-click in the Modeling workspace. Click **Unselect All**.
Syntax:	`obj.ScrUnselectAll()`
Parameters:	None.
Return Value:	BOOL • **0** – Failure • **1** – Success
VB Example:	`obj.ScrUnselectAll()`
IPY Example:	`oDoc.ScrUnselectAll()`

ScrUpdateComponentTree

For scripts run in graphical mode, updates the component tree. Scripted modifications will not display in the tree until this function is called. NOTE: This script is unnecessary for scripts run in non-graphical mode.	
UI Command:	None.
Syntax:	`obj.ScrUpdateComponentTree`
Parameters:	None.
Return Value:	None.
VB Example:	`obj.ScrUpdateComponentTree`
IPY Example:	`oDoc.ScrUpdateComponentTree()`

ScrUseIcepakTemperatureDataInDc

Enables or disables the use of Icepak thermal data in DC simulations.	
UI Command:	**Simulation** > **Compute DC IR**. Select Icepak data.
Syntax:	`obj.ScrUseIcepakTemperatureDataInDc(<use>)`
Parameters:	BOOL use (TRUE = Use thermal data; FALSE = Do not use thermal data)
Return Value:	None.
VB Example:	`obj.ScrUseIcepakTemperatureDataInDc True`
IPY Example:	`oDoc.ScrUseIcepakTemperatureDataInDc(True)`

ScrUseTouchstonePortRemapping

Enables or disables the use of port name remapping for touchstone file exports.	
UI Command:	None.
Syntax:	`obj.ScrUseTouchstonePortRemapping(remapNames)`
Parameters:	BOOL/INT remapNames (TRUE/1 = Enable; FALSE/0 = Disable)
Return Value:	BOOL • **0** – Failure • **1** – Success
VB Example:	`outcome = obj.ScrUseTouchstonePortRemapping (TRUE)`
IPY Example:	`oDoc.ScrUseTouchstonePortRemapping(1)`

Solve

Solves a specified simulation.	
UI Command:	**Launch** button in simulation settings window.
Syntax:	`obj.Solve(solutionName)`
Parameters:	BSTR solutionName
Return Value:	Simulation results in text format.
VB Example:	`obj.Solve("Far Field Sim 1")`
IPY Example:	`oDoc.Solve('Near Field Sim 1')`

StopSimLink

Aborts a specified simulation.	
UI Command:	**Process Monitor** > **Abort Simulation**.
Syntax:	`obj.StopSimLink(<simID>, <abort>)`
Parameters:	INT simID BOOL abort (TRUE/1 = abort; FALSE/0 = do not abort)
Return Value:	None.
VB Example:	`obj.StopSimLink(121, True)`
IPY Example:	`oDoc.StopSimLink(121, 1)`

SupportSParamLink

Returns whether design supports S-parameter links.	
UI Command:	None.
Syntax:	`obj.SupportSParamLink()`
Parameters:	None.
Return Value:	BOOL
VB Example:	`obj.SupportSParamLink()`
IPY Example:	`oDoc.SupportSParamLink()`

Index

L

R

S

T

V

W