OptiSystem

Visual Basic Reference

Optical Communication System Design Software

Version 11

OptiSystem

Visual Basic Reference

Optical Communication System Design Software

Disclaimer

Optiwave makes no representation or warranty with respect to the adequacy of this documentation or the programs which it describes for any particular purpose or with respect to its adequacy to produce any particular result. In no event shall Optiwave, its employees, its contractors, or the authors of this documentation be liable for special, direct, indirect, or consequential damages, losses, costs, charges, claims, demands, or claim for lost profits, fees, or expenses of any nature or kind.

Technical support

If you purchased Optiwave software from a distributor that is not listed here, please send technical questions to your distributor.

Optiwave — Canada/US

Tel	(613) 224-4700	**E-mail**	support@optiwave.com
Fax	(613) 224-4706	**URL**	www.optiwave.com

Cybernet Systems Co., Ltd. — Japan

Tel	+81 (03) 5297-3342	**E-mail**	owtech@cybernet.co.jp
Fax	+81 (03) 5297-3646	**URL**	www.cybernet.co.jp

Optiwave Europe — Europe

Tel	+33 (0) 494 08 27 97	**E-mail**	support@optiwave.com
Fax	+33 (0) 494 33 65 76	**URL**	www.optiwave.eu

Table of contents

Visual Basic Script

You can control many of the objects that you use when you create the layout using Visual Basic Script (VB Script). You can create components, assign parameters and results and run simulations. This powerful feature eliminates many lengthy, repetitive manual tasks.

This manual describes how to use VBScript with OptiSystem, and includes examples and tutorials.

Quick Visual Basic references

The following is a basic description of a selected part of the VB Script language.

For more information and a complete reference, refer to Microsoft online documentation at:

http://msdn.microsoft.com/scripting

In Visual Basic, every variable type is *variant*. Variants are tagged unions. A variant stores a value and information on the value type. Variants support the following types:

- 2-byte and 4-byte integer
- 4-byte and 8-byte floating points
- Strings
- Booleans
- HRESULT
- Pointers to IUnknown and IDispatch interfaces

You must be aware of the type of data required by the OptiSystem VB Script programming interface. For example, consider the number 80. The number 80 can be stored as a 2-byte integer, a 4-byte integer, a 4-byte float, an 8-byte float, or even as a string. When calling the application programming interface (API), you must ensure that you are passing the proper type. Otherwise, the API will not behave properly.

You must convert to the proper type.

Type conversions can be done as:

- **CInt(expression)**

  returns a variant of subtype 2-byte integer (referred to as just an integer)

- **CLng(expression)**

  returns a variant of subtype 4-byte integer (referred to as a long)

- **CSng(expression)**

  returns a variant of subtype 4-byte floating point (referred to as a single)

- **CDbl(expression)**

  returns a variant of subtype 8-byte floating point (referred to as a double)

- **CStr(expression)**

  returns a variable of subtype string (a system string of type BSTR or VB string)

Example

```
Dim MyDouble
Dim MyString
MyDouble = 437.324 'MyDouble is now a Variant of subtype Double.
MyString = CStr(MyDouble) MyString is now a Variant of 'subString
String and contains "437.324".
```

For loops are the main source of iteration control of multiple iteration simulations. **For** loops must have a count and the loop body must be encapsulated with a **Next** statement. **Step** is used as a way to specify the amount that the loop counter is to be changed in each iteration. This value can be negative, providing a decrementing loop if **start > end**.

```
For counter = start To end <Step step>
<statements>
Next
```

If structures provide flow control. For example:

```
If condition1 then
<statements>
ElseIf condition2 Then
<statements>
Else
<statements>
End If
```

VBScript also has many useful math functions. Methods for trigonometry, logarithms, exponential, square root, and randomize are all included.

For more information, see:

http://msdn.microsoft.com/scripting

CSDKOptiSystemDoc

RemoveAll(BOOL bDisplayMessage)

Purpose: This method removes all layouts except the first one and removes all components from the layout.

Arguments:

bDisplayMessage - Specifies to display message before RemovingAll if set to TRUE.

Return:

None

Example:

```
Document.RemoveAll TRUE
```

CalculateCurrentSweepIteration(BOOL bCalcOptimization,BOOL bStopOnError)

Purpose: This method calculates current sweep iteration for the current layout

Arguments:

bCalculateOptimization - Specifies to calculate Optimizations if set to TRUE.

bStopOnError - Specifies to stop on error during the calculation if set to TRUE.

Return:

None

Example:

```
Document.CalculateCurrentSweepIteration FALSE, FALSE
```

CalculateAllSweepIterations(BOOL bCalcOptimization, BOOL bStopOnError)

Purpose: This method calculates all sweep iterations for the current layout.

Arguments:

bCalculateOptimization - Specifies to calculate Optimizations if set to TRUE.

bStopOnError - Specifies to stop on error during the calculation if set to TRUE.

Return:

None

Example:

```
Document.CalculateAllSweepIterations FALSE, FALSE
```

CalculateProject(BOOL bCalcOptimization, BOOL bStopOnError)

Purpose: This method calculates the whole project

Arguments:

bCalculateOptimization - Specifies to calculate Optimizations if set to TRUE.

bStopOnError - Specifies to stop on error during the calculation if set to TRUE .

Return:

None

Example:

```
Document.CalculateProject FALSE, FALSE
```

`SaveAs()`

Purpose: This method displays the SaveAs dialog from which the user may save the document.

Remarks: If the user specifies a file that does not exist, then a new file will be created, otherwise the user will be asked whether they want to replace the existent file.

If the function is successful the document's Modified Flag will be set to FALSE.

Return:

None

Example:

```
Document.SaveAs
```

`GetSelection()`

Purpose: This method gets an array of SIDs of components in the selection.

Remarks: If the selection is empty, the function will still succeed, however the returned array will be empty. If the function succeeds the pArrSIDs will be allocated - both the descriptor and the data section will be allocated.

Return:

The array of SIDs of components. The SAFEARRAY part of this VARIANT is used here.

`RemoveFromSelection``( unsigned long nComponentSID)`

Purpose: This method removes the specified component from the selection.

Remarks: This function notifies the system manager of the component deselection.

Arguments:

nComponentSID - The SID of the component to remove from the selection.

Return:

None

`AddToSelection( unsigned long nComponentSID)`

Purpose: This method adds the specified component to the selection.

Remarks: The selection is a collection of selected components on the layout.

The document has a collection of layouts, however only one selection per document is possible - only the current layout may have the selection. The layout has a relationship with the selection, so that only components from the layout may be added to the selection, and the selection may only contain components from the layout.

The view has a relationship with the layout, so that only one layout may be displayed in the view, namely the current layout. When another layout is set as current, the selection is cleared.

Arguments:

nComponentSID - The SID of the component to add to the selection.

Return:

None

`ClearSelection()`

Purpose: This method clears the component selection.

Remarks: This function notifies the system manager of clearing the selection.

Return:

None

`Close()`

Purpose: This method closes the document. If the document was modified since it was last saved, the user will be given an opportunity to save the document.

Remarks: If the document is not saved, a message box will ask the user whether they want to save (Yes button), don't save (No button) or cancel (Cancel button) closing of the document. Answering No will simply close the document. Answering Cancel will not save, and will not close the document. Answering Yes will give the user a chance to save the document, they will be prompted with the SaveAs dialog where the user may save

(Save button), or cancel (Cancel button) saving. Canceling saving will also cancel closing of the document. This function notifies the system manager of the closed document.

Return:

None

`Save( BSTR sPathName)`

Purpose: This method saves the specified document.

Remarks: If a file with the specified name does not exist, then a new file will be created, otherwise the existent file will be replaced. If the function is successful the document's Modified Flag will be set to FALSE.

Arguments:

sPathName - A null terminating string specifying the path to the document's name as it should be saved. If this argument is empty, this function is equivalent to CSDKOptiSystemDoc::SaveAs() .

Return:

None

Example:

```
Document.Save "C:\Work\systems\OptiSystem\samples\Project5.osd"
```

`UpdateViewRect( long nLeft, long nTop, long nRight, long nBottom)`

Purpose: This method invalidates the specified portion of the view of this document.

Remarks: Together the four arguments form the bounding rectangle.

This rectangle specifes the area to be invalidated. This function is a generic version of CSDKOptiSystemDoc::UpdateView(). The extreme top-left corner coordinates are

(0,0), the horizontal coordinates increase towards the right, and vertical coordinates increase towards the bottom. If the arguments are not valid the entire view may be invalidated.

Arguments:

All arguments are in client coordinates.

nLeft - The leftmost area to invalidate. Value must be greater or equal to 0.

nTop - The topmost area to invalidate. Value must be greater or equal to 0.

nRight - The rightmost area to invalidate. Value must be greater then nLeft, and less then or equal to.

Return:

None

`UpdateView()`

Purpose: This method invalidates the entire view of this document.

Return:

None

Example:

```
Document.UpdateView
```

`get_SaveMonitorData()`

Purpose: This method returns TRUE if monitor data was set to be saved, FALSE otherwise.

Return:

A Boolean value whether monitor data should be saved. Valid values are TRUE and FALSE.

`put_SaveMonitorData(` `VARIANT_BOOL bSaveMonitorData` **`)`**

Purpose: This method sets Boolean that indicates whether monitor data should be saved.

Arguments:

bSaveMonitorData - Specifies to save monitor data if set to TRUE.

Valid values for this Boolean are TRUE and FALSE.

Return:

None

`GetLayoutMgr()`

Purpose: This method returns the layout manager object.

Return:

The Document's layout manager.

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
```

GetCSFramework()

Purpose: This method returns the calculation schedulers framework object.

Return:

The calculation scheduler's framework

Example:

```
Dim CS
Set CS = Document.GetCSFramework
```

GetLayoutParameter(BSTR sLayoutName, BSTR sParameterName)

Purpose: This method returns the parameter with name sParameterName from specified layout (sLayoutName). If sLayoutName is a empty string then retrieve from the current layout

Arguments:

sLayoutName - Specifies the name of the layout to retrieve.

sParameterName - Specifies the name of the parameter to retrieve

Return:

A parameter.

Example:

```
Dim par
Set par = Document.GetLayoutParameter( "", "Bit rate" )
```

GetLayoutParameterUnit(BSTR sLayoutName, long nSweepIteration, BSTR sParameterName)

Purpose: This method returns parameter's unit with name sParameterName from specified layout (sLayoutName). If sLayoutName is a empty string then retrieve from the current layout

Arguments:

sLayoutName - Specifies the name of the layout to retrieve.

nSweepIteration - Specifies the sweep iteration.

sParameterName - Specifies the name of the parameter to retrieve

Return:

The unit of the layout parameter.

Example:

```
Dim unit
unit = Document.GetLayoutParameterUnit( "", 1, "Bit rate" )
```

GetLayoutParameterValue(BSTR sLayoutName, long nSweepIteration, BSTR sParameterName)

Purpose: This method returns parameter's value with name sParameterName from specified layout (sLayoutName). If sLayoutName is a empty string then retrieve from the current layout

Arguments:

sLayoutName - Specifies the name of the layout to retrieve.

nSweepIteration - Specifies the sweep iteration.

sParameterName - Specifies the name of the parameter to retrieve

Return:

The value of the layout parameter.

Example:

```
Dim val
val = Document.GetLayoutParameterValue( "", 1, "Bit rate" )
```

GetLayoutMaxIterations(BSTR sLayoutName)

Purpose: This method returns the maximum sweep iterations for layout with name(sLayoutName). If sLayoutName is a empty string then retrieve from the current layout

Arguments:

sLayoutName - Specifies the name of the layout to retrieve.

Return:

The maximum sweep iterations of the layout .

Example:

```
Dim maxiter
maxiter = Document.GetLayoutMaxIterations( "" )
```

SetLayoutParameterValue(BSTR sLayoutName, long nSweepIteration, BSTR sParameterName, VARIANT newValue)

Purpose: This method sets parameter's value with name sParameterName from specified layout (sLayoutName). If sLayoutName is a empty string then sets from the current layout

Arguments:

sLayoutName - Specifies the name of the layout to retrieve.

nSweepIteration - Specifies the sweep iteration.

sParameterName - Specifies the name of the parameter.

newValue - The new value of the parameter.

Example:

Document.SetLayoutParameterValue "",1, "Reference bit rate", FALSE

GetComponentResultValue(BSTR sLayoutName, long nSweepIteration, BSTR sComponentName, BSTR sResultName)

Purpose: This method returns the results's value with name sResultName from specified layout (sLayoutName) and component (sComponentName). If sLayoutName is a empty string then retrieve from the current layout

Arguments:

sLayoutName - Specifies the name of the layout to retrieve.

nSweepIteration - Specifies the sweep iteration.

sComponentName - Specifies the component name.

sResultName - Specifies the name of the result to retrieve

Return:

The value of the component's result.

Example:

```
Dim resval
resval = Document.GetComponentResultValue( "",1, "WDM Analyzer",
"Max. Noise Power (dBm)")
```

GetComponentParameterUnit(BSTR sLayoutName, long nSweepIteration, BSTR sComponentName, BSTR sParameterName)

Purpose: This method returns the parameter's unit with name sParameterName from specified layout (sLayoutName) and component (sComponentName). If sLayoutName is a empty string then retrieve from the current layout.

Arguments:

sLayoutName - Specifies the name of the layout to retrieve.

nSweepIteration - Specifies the sweep iteration.

sComponentName - Specifies the component name.

sParameterName - Specifies the name of the parameter to retrieve

Return:

The unit of the component's parameter.

Example:

```
Dim parunit
parunit = Document.GetComponentParameterUnit( "",1, "CW Laser",
"Frequency")
```

GetComponentParameterValue(BSTR sLayoutName, long nSweepIteration, BSTR sComponentName, BSTR sParameterName)

Purpose: This method returns the parameter's value with name sParameterName from specified layout (sLayoutName) and component (sComponentName). If sLayoutName is a empty string then retrieve from the current layout.

Arguments:

sLayoutName - Specifies the name of the layout to retrieve.

nSweepIteration - Specifies the sweep iteration.

sComponentName - Specifies the component name.

sParameterName - Specifies the name of the parameter to set

Return:

The value of the component's parameter.

Example:

```
Dim parval
parval = Document.GetComponentParameterValue( "",1, "CW Laser",
"Frequency")
```

SetComponentParameterValue(BSTR sLayoutName, long nSweepIteration, BSTR sComponentName, BSTR sParameterName, VARIANT

newValue)

Purpose: This method sets the parameter's value with name sParameterName from specified layout (sLayoutName) and component (sComponentName). If sLayoutName is a empty string then retreave from the current layout.

Arguments:

sLayoutName - Specifies the name of the layout to retrieve.

nSweepIteration - Specifies the sweep iteration.

sComponentName - Specifies the component name.

sParameterName - Specifies the name of the parameter to retrieve.

newValue - The new value of the parameter.

Example:

```
Document.SetComponentParameterValue "",1, "CW Laser", "Frequency",
194.5
```

GetComponentParameter(BSTR sLayoutName, BSTR sComponentName, BSTR sParameterName)

Purpose: This method returns a parameter with name sParameterName from specified layout (sLayoutName) and component (sComponentName). If sLayoutName is a empty string then retreave from the current layout.

Arguments:

sLayoutName - Specifies the name of the layout to retrieve.

sComponentName - Specifies the component name.

sParameterName - Specifies the name of the parameter to retrieve.

Return:

A parameter.

Example:

```
Dim par
Set par = Document.GetComponentParameter( "","CW Laser",
"Frequency")
```

GetComponentResult(BSTR sLayoutName, BSTR sComponentName, BSTR sResultName)

Purpose: This method returns a result with name sResultName from specified layout (sLayoutName) and component (sComponentName). If sLayoutName is a empty string then retrieve from the current layout.

Arguments:

sLayoutName - Specifies the name of the layout to retrieve.

sComponentName - Specifies the component name.

sResultName - Specifies the name of the result to retrieve.

Return:

A result.

Example:

```
Dim res
Set res = Document.GetComponentResult( "","WDM Analyzer", "Max. OSNR
(dB)")
```

GetComponentGraph(BSTR sLayoutName, BSTR sComponentName, BSTR sGraphName)

Purpose: This method returns a graph with name sGraphName from specified layout (sLayoutName) and component (sComponentName). If sLayoutName is a empty string then retreave from the current layout.

Arguments:

sLayoutName - Specifies the name of the layout to retrieve.

sComponentName - Specifies the component name.

sGraphName - Specifies the name of the graph to retrieve.

Return:

A graph.

Example:

```
Dim graph
Set graph = Document.GetComponentGraph( "","WDM Analyzer", "Noise
spectrum")
```

CSDKCanvas

GetMainCount()

Purpose: This method retrieves the number of all layout objects for this canvas.

Return:

Long.

An integer value that represents the objects count.

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
Dim count
count = Canvas1.GetMainCount
MsgBox(count)
```

CreateComponent(BSTR sType, BSTR strCLSIDLibrary, long nPositionX, long nPositionY, long nWidth, long nHeight, BOOL bUpdateUI)

Purpose: This method creates a component.

Arguments:

sType - A null-terminated string that specifies the type of the component.

strCLSIDLibrary - A null-terminated string that specifies the CLSID of the component library.

nPositionX - Specifies the leftmost coordinate of the component. Value must be greater or equal to 0.

nPositionY - Specifies the topmost coordinate of the component. Value must be greater or equal to 0.

nWidth - Specifies the component width. Value must be greater than 0.

nHeight - Specifies the component height. Value must be greater than 0.

bUpdateUI - Specifies to update user interface if set to 1. Value must be 0 or 1.

Return:

An object of type Component.

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
Dim Canvas1_Component1
```


```
Set Canvas1_Component1 = Canvas1.CreateComponent("CW
Laser","{6DA31CEE-058F-11D4-93BD-0050DAB7C5D6}",300,190, 34, 34,0)
```

Reference

Here are the CLSIDs of all available libraries:

Waveguide Amplifier Library {00094873-B580-40c3-B2BA-0A66656F4C68}

Receivers Library {0B8011BF-3C6B-11D4-93EF-0050DAB7C5D6}

Raman Amplifiers Library {10A4FE75-E0B3-11d4-948D-0050DAB7C5D6}

Filters Library {161B94D1-3BA4-11D4-93EE-0050DAB7C5D6}

WDM Multiplexers Library {1747A24E-52A0-11D4-9403-0050DAB7C5D6}

Electrical Components Library {1CCD3D1F-8E82-4a3e-92B9-1C9C20572BB1}

Multimode Fibers Library {24797318-DD42-4f59-8B7A-D12D3BFC9B1B}

Amplifiers Library {255EDC8F-37E4-11D4-93EC-0050DAB7C5D6}

WDMSystemsComponentLibrary {2A9D9567-99DE-4c83-8F05-720F1480B57E}

SOA Component Library {3083FC6C-F741-4dd8-A1A9-B5851670A418}

Optiwave Software Library {33B1D7C9-2EE1-40c1-B640-96F158649FF0}

Optical Fibers Library {416EC6F1-529F-11D4-9403-0050DAB7C5D6}

CATV Component Library {4462750B-F858-42bb-A415-994DFE4D44BD}

Binary Components Library {498E720E-4E63-48ca-BF47-0A19891A9A5F}

MATLAB Library {6412335E-679E-4fc6-A4CE-216F51442F45}

Transmitters Library {6DA31CEE-058F-11D4-93BD-0050DAB7C5D6}

EDACosimulationLibrary {899442C9-5ADA-4921-BEAB-871473563AC9}

Dynamic OXC Library {8D228FBE-779C-49ff-B28E-0A63C55635A6}

Optical Components Library {A5BEC796-4CC4-4a8c-9F1B-345D023EE5CA}

Dispersion Compensation Library {DA6309D8-71BD-441f-BC04-A3908C37686C}

Tools Library {E138711F-3E0D-11D4-93F3-0050DAB7C5D6}

Passives Library {F11D0C07-3C7D-11D4-93F0-0050DAB7C5D6}

Network Library {F11D0C16-3C7D-11D4-93F0-0050DAB7C5D6}

Visualizers Library {F11D0C25-3C7D-11D4-93F0-0050DAB7C5D6}

`get_Height()`

Purpose: This method retrieves the height for this canvas.

Return:

Long.

A long representing the canvas height.

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
```

```
  Dim Layout1
  Set Layout1 = Lm.GetCurrentLayout
  Dim Canvas1
  Set Canvas1 = Layout1.GetCurrentCanvas
  Dim CanvasHeight
  CanvasHeight = Canvas1.Height
  MsgBox(CanvasHeight)
```

put_Height(long newVal)

Purpose: This method set the height for this canvas.

Arguments:

newVal - Specifies the height for this canvas as a long.

Example:

```
  Dim Lm
  Set Lm = Document.GetLayoutMgr
  Dim Layout1
  Set Layout1 = Lm.GetCurrentLayout
  Dim Canvas1
  Set Canvas1 = Layout1.GetCurrentCanvas
  Canvas.Height = 2000
```

get_Width()

Purpose: This method retrieves the width for this canvas.

Return:

Long.

A long representing the canvas width.

Example:

```
  Dim Lm
  Set Lm = Document.GetLayoutMgr
  Dim Layout1
  Set Layout1 = Lm.GetCurrentLayout
  Dim Canvas1
  Set Canvas1 = Layout1.GetCurrentCanvas
  Dim CanvasWidth
  CanvasWidth= Canvas1.Width
  MsgBox(CanvasWidth)
```

put_Width(long newVal)

Purpose: This method sets the width for this canvas as a long.

Arguments:

newVal - Specifies the width for this canvas as a long.

Return:

None

Example:

```
  Dim Lm
  Set Lm = Document.GetLayoutMgr
```


```
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
Canvas1.Width = 3000
```

UpdateAll()

Purpose: This method updates the user interface and Project Browser.

Return:

None

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
Canvas1.UpdateAll
```

RemoveAll()

Purpose: This method removes (Delete) all layout objects from this canvas.

Return:

None

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
Canvas1.RemoveAll
```

GetComponentByName(BSTR sName)

Purpose: This method returns the component with name sName.

Arguments:

sName - Specifies the name of the component to retrieve.

Return:

A component.

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Layout1.Name  = "Layout 1"
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
```

```
Dim Canvas1_Component1
Set Canvas1_Component1 = Canvas1.CreateComponent("CW
Laser","{6DA31CEE-058F-11D4-93BD-0050DAB7C5D6}",340,130, 34, 34,0)
Canvas1_Component1.Name  = "CW Laser1"
Dim comp
Set comp = Canvas1.GetComponentByName("CW Laser1")
Dim sname
sname = comp.Name
MsgBox(sname)
```

CreateCircle(long nPositionX, long nPositionY,long nWidth, long nHeight)

Purpose: This method creates a circle.

Arguments:

nPositionX - Specifies the leftmost coordinate of the rectangle where a circle is drawn. Value must be greater or equal to 0.

nPositionY - Specifies the topmost coordinate of the rectangle where a circle is drawn. Value must be greater or equal to 0.

nWidth - Specifies the rectangle width. Value must be greater than 0.

nHeight - Specifies the rectangle height. Value must be greater than 0.

Return:

None.

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
Canvas1.CreateCircle 89,346, 227, 151
```

CreateRectangle(long nPositionX, long nPositionY, long nWidth, long nHeight)

Purpose: This method creates a rectangle.

Arguments:

nPositionX - Specifies the leftmost coordinate of the rectangle. Value must be greater or equal to 0.

nPositionY - Specifies the topmost coordinate of the rectangle. Value must be greater or equal to 0.

nWidth - Specifies the rectangle width. Value must be greater than 0.

nHeight - Specifies the rectangle height. Value must be greater than 0.

Return:

None.

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
Canvas1.CreateRectangle 92,102, 423, 171
```

CreateLine(long nPositionX, long nPositionY, long nWidth, long nHeight, long nCurrentDirection, BOOL bArrowHead)

Purpose: This method creates a line.

Arguments:

nPositionX - Specifies the leftmost coordinate of the rectangle where a line is drawn. Value must be greater or equal to 0.

nPositionY- Specifies the topmost coordinate of the rectangle where a line is drawn. Value must be greater or equal to 0.

nWidth - Specifies the rectangle width. Value must be greater than 0.

nHeight - Specifies the rectangle height. Value must be greater than 0.

nCurrentDirection - Specifies the current direction of the line. Possible values are:

- 0 - right Up
- 1 - right Down
- 2 - left Up
- 3 - left Down
- 4 - Left
- 5 - Right
- 6 - Up
- 7 - Down

bArrowHead - Specifies that the line has arrow head if set to TRUE.

Return :

None.

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
Canvas1.CreateLine 605,130, 155, 212, 0, TRUE
```

CreateTextLabel(long nPositionX, long nPositionY, long nWidth, long nHeight,BSTR sText, BSTR sFaceName, long nFHeight, long nFWidth, long nFWeight, BOOL bItalic, BOOL

```
bUnderline)
```

Purpose: This method creates a text label.

Arguments:

nPositionX - Specifies the leftmost coordinate of the rectangle where a text label is drawn. Value must be greater or equal to 0.

nPositionY - Specifies the topmost coordinate of the rectangle where a text label is drawn. Value must be greater or equal to 0.

nWidth - Specifies the rectangle width. Value must be greater than 0.

nHeight - Specifies the rectangle height. Value must be greater than 0.

sText - Text to be drawn.

sFaceName - A null-terminated string that specifies the typeface name of the font. The length of this string must not exceed 32 characters, including the null terminator. The EnumFontFamilies function can be used to enumerate the typeface names of all currently available fonts. If sFaceName is an empty string, GDI uses the first font that matches the other specified attributes.

nFHeight - Specifies the height, in logical units, of the font's character cell or character. The character height value (also known as the em height) is the character cell height value minus the internal-leading value. The font mapper interprets the value specified in nFHeight in the following manner.

nFWidth - Specifies the average width, in logical units, of characters in the font. If nFWidth is zero, the aspect ratio of the device is matched against the digitization aspect ratio of the available fonts to find the closest match, determined by the absolute value of the difference.

nFWeight - Specifies the weight of the font in the range 0 through 1000. For example, 400 is normal and 700 is bold. If this value is zero, a default weight is used.

bItalic[in] - Specifies an italic font if set to TRUE.

bUnderline[in] - Specifies an underlined font if set to TRUE.

Return:

None.

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
Canvas1.CreateTextLabel 407,412, 52, 14,  "Text", "Arial", -11, 0,
400, FALSE, FALSE
CSDKComponentMgr functionalities
```

`GetPort``(int nX, int nY)`

Purpose: This method finds a port at a certain point on the screen.

Arguments:

nX - Specifies the X coordinates of point.

nY - Specifies the Y coordinates of the point.

Return:

A port.

Remarks: First found port will be returned.

GetComponent(int nX, int nY)

Purpose: This method finds a component at a certain point on the screen.

Arguments:

nX - Specifies the X coordinates of point.

nY - Specifies the Y coordinates of the point.

Return:

A component.

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Layout1.Name  = "Layout 1"
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
Dim Canvas1_Component1
Set Canvas1_Component1 = Canvas1.CreateComponent("CW
Laser","{6DA31CEE-058F-11D4-93BD-0050DAB7C5D6}",340,130, 34, 34,0)
Canvas1_Component1.Name  = "CW Laser1"
Dim comp
Set comp = Canvas1.GetComponent(340, 130)
Dim sName
sName = comp.Name
MsgBox(sName)
CSDKExtObjectMgr functionalities
```

GetObjectBySID(unsigned long nSID)

Purpose: This method returns an object with specified SID.

Arguments:

nSID - Specifies the SID of the object.

Return:

An object.

GetObjectByName(BSTR sName)

Purpose: This method returns an object with specified name.

Arguments:

sName - Specifies the name of the object to retrieve.

Return:

An object.

GetObject(unsigned long nLocalID)

Purpose: This method returns an object with specified local ID.

Arguments:

nLocalID - Specifies the local ID of the object.

Return:

An object.

DeleteObjectBySID(unsigned long nSID)

Purpose: This method deletes the specified object from this manager.

Arguments:

nSID - Specifies the SID of the object to delete.

Return:

None.

DeleteAll()

Purpose: This method deletes all objects from this manager.

Return:

None.

AddCategory(BSTR sName, OLE_HANDLE hIcon, BSTR shortDescription,BSTR longDescription)

Purpose: This method adds a category to this manager.

Remarks: Once the manager has a category, the objects in this manager may be grouped by categories by setting their categories using the method **AssignToCategory().**

Arguments:

sName - Specifies the name of the category to add. The name must be unique to this manager that is if a category with this name already exists in this manager then you may not reuse this name.

hIcon - A handle to an icon that will represent the category. If you do not wish to use this argument, set it to NULL.

shortDescription - A short description of the category. This may be used by tooltips. If you do not wish to use this argument, set it to NULL.

longDescription - A long description of the category. If you do not wish to use this argument, set it to NULL.

Return:

None.

MoveBackOnePosition(IDispatch* pDispObject)

Purpose: This method moves the specified object's position in the list one position towards the back.

Remarks: The tail of the list is the front, and the head of the list is the back.

Argument pSuccess will be set to VARIANT_FALSE if the specified object is already at the back.

Arguments:

pDispObject - A pointer to the IDispatch interface of the object to move.

Return:

BOOL.

TRUE if the object was moved towards the back, FALSE otherwise.

MoveForwardOnePosition(IDispatch* pDispObject)

Purpose: This method moves the specified object's position in the list one position towards the front.

Remarks: The tail of the list is the front, and the head of the list is the back.

Argument pSuccess will be set to VARIANT_FALSE if the specified object is already at the front.

Arguments:

pDispObject - A pointer to the IDispatch interface of the object to move.

Return:

BOOL.

TRUE if the object was moved towards the front, FALSE otherwise.

MoveToFront(IDispatch* pDispObject)

Purpose: This method moves the specified object's position in the list to the front.

Remarks: The tail of the list is the front, and the head of the list is the back.

Argument pSuccess will be set to VARIANT_FALSE if the specified object is already at the front.

Arguments:

pDispObject - A pointer to the IDispatch interface of the object to move.

Return:

BOOL.

TRUE if the object was moved to the front, FALSE otherwise.

MoveToBack(IDispatch* pDispObject,)

Purpose: This method moves the specified object's position in the list to the back.

Remarks: The tail of the list is the front, and the head of the list is the back.

Argument pSuccess will be set to VARIANT_FALSE if the specified object is already at the front.

Return:

BOOL.

TRUE if the object was moved to the front, FALSE otherwise.

CSDKExtObject functionalities

`get_Name()`

Purpose: This method returns the name of this object.

Return:

String.

The name of the object as a string.

`put_Name``( BSTR pName)`

Purpose: This method sets the name for this object.

Arguments:

pName - A null terminating string specifying the name of the component.

Return:

None.

`GetIteration()`

Purpose: This method returns the current iteration in the document at this time.

Return:

Long.

The current sweep iteration as a long.

`GetMaxIterations()`

Purpose: This method returns the maximum number of iterations the document supports at this time.

Return:

Long.

The max number of sweep iterations as a long.

`GetSID()`

Purpose: This method returns the system ID (SID) of this object.

Remarks: This value uniquely identifies this object in the document.

Return:

Unsigned long.

The SID of the object as an unsigned long.

CSDKLayout

GetCurrentCanvas()

Purpose: This method returns current layout's canvas.

Return:

The current layout's canvas

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
```

get_Description()

Purpose: This method returns description of the layout.

Return:

String.

The description of the canvas as a string.

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Layout1.Description = "This is Layout1"
```

put_Description(BSTR newVal)

Purpose: This method sets description of the layout.

Arguments:

newVal - A null terminating string specifuing the description.

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Layout1.Description = "This is Layout1"
Dim sDescription
sDescription =  Layout1.Description
MsgBox(sDescription)
```

GetDate()

Purpose: This method returns the date.

Return:

A variant of subtype DATE.

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Layout1.Name  = "Layout 1"
Dim sDate
sDate = Layout1.GetDate
MsgBox(sDate)
```

get_Author()

Purpose: This method returns the author of the layout.

Return:

String.

The author of the layout as a string.

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Layout1.Name  = "Layout 1"
Layout1.Author = "George"
Dim sAuthor
sAuthor = Layout1.Author
MsgBox(sAuthor)
```

put_Author(BSTR newVal)

Purpose: This method sets the author of the layout.

Arguments:

newVal - A null terminating string that specifies the autor of the layout.

Return:

None.

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Layout1.Name  = "Layout 1"
Layout1.Author = "George"
Dim sAuthor
sAuthor = Layout1.Author
MsgBox(sAuthor)
```

RemoveParameter(unsigned long nSID)

Purpose: This method removes a user defined parameter with SID (nSID)

Arguments:

nSID - A long that specifies the SID of the parameter to be removed.

Return:

None.

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Dim nSID
nSID = Layout1.AddParameter( "Param1", 2 , "Simulation", 0.000000,
100.000000, "12.5", "nm", "")
Layout1.RemoveParameter(nSID)
```

AddParameter(BSTR sName, long nType, BSTR sCategoryName, double dMin, double dMax, BSTR sValue, BSTR sUnit, BSTR sChoiceList)

Purpose: This method adds a parameter to the layout parameter manager.

Arguments:

sName - A null-terminated string that specifies the name of the added parameter. The name must be unique.

nType - Specifies the type of the added parameter. The following types are available:

- 0 - type BOOL
- 1 - type Choice
- 2 - type Double
- 3 - type Long
- 4 - type String
- 5 - type MxN

sCategoryName - A null-terminated string that specifies the name of the category to be added.

dMin - Specifies the min value of the parameter.

dMax - Specifies the max value of the parameter.

sValue - A null-terminated string that specifies the value of the added parameter.

sUnit - A null-terminated string that specifies the units of the added parameter.

sChoiceList - A null-terminated string that specifies the choice list of the added parameter. If the parameter is not a type Choice than the string is empty.

Return:

nSID - returns the SID of the added parameter.

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Layout1.AddParameter "Param1", 2 , "Simulation", 0.000000,
100.000000, "12.5", "nm", ""
```

GetParameterMgr()

Purpose: This method returns the layout's Parameter manager.

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Layout1.Name  = "Layout 1"
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
Dim Canvas1_Component6
Set Canvas1_Component6 = Canvas1.CreateComponent("CW
Laser","{6DA31CEE-058F-11D4-93BD-0050DAB7C5D6}",120,150, 34, 34,0)
Canvas1_Component6.Name  = "CW Laser"
Dim parMgr
Set parMgr = Layout1.GetParameterMgr
```

GetPathMgr()

Purpose: This method returns the layout's Path manager.

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Layout1.Name  = "Layout 1"
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
Dim Canvas1_Component6
Set Canvas1_Component6 = Canvas1.CreateComponent("CW
Laser","{6DA31CEE-058F-11D4-93BD-0050DAB7C5D6}",120,150, 34, 34,0)
Canvas1_Component6.Name  = "CW Laser"
Dim pathMgr
Set pathMgr = Layout1.GetPathMgr
```

GetResultMgr()

Purpose: This method returns the layout's Result manager.

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Layout1.Name  = "Layout 1"
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
Dim Canvas1_Component6
Set Canvas1_Component6 = Canvas1.CreateComponent("CW
Laser","{6DA31CEE-058F-11D4-93BD-0050DAB7C5D6}",120,150, 34, 34,0)
Canvas1_Component6.Name  = "CW Laser"
Dim resultMgr
```


```
Set resultMgr = Layout1.GetResultMgr
```

GetLayoutCost()

Purpose: This method returns the layout's cost.

Return:

Double.

The cost of the layout as a double.

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Layout1.Name  = "Layout 1"
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
Dim Canvas1_Component6
Set Canvas1_Component6 = Canvas1.CreateComponent("CW
Laser","{6DA31CEE-058F-11D4-93BD-0050DAB7C5D6}",120,150, 34, 34,0)
Canvas1_Component6.Name  = "CW Laser"
Dim Canvas1_Component7
Set Canvas1_Component7 = Canvas1.CreateComponent("CW
Laser","{6DA31CEE-058F-11D4-93BD-0050DAB7C5D6}",370,300, 34, 34,0)
Canvas1_Component7.Name  = "CW Laser_1"
Canvas1_Component6.Cost = 110.00
Canvas1_Component7.Cost = 220.00
Dim nCost
nCost = Layout1.GetLayoutCost()
MsgBox(nCost)
```

CSDKComponentMgr functionalities

GetPort(int nX, int nY)

Purpose: This method finds a port at a certain point on the screen.

Arguments:

nX - Specifies the X coordinates of point.

nY - Specifies the Y coordinates of the point.

Return:

A port.

Remarks: First found port will be returned.

GetComponent(int nX, int nY)

Purpose: This method finds a component at a certain point on the screen.

Arguments:

nX - Specifies the X coordinates of point.

nY - Specifies the Y coordinates of the point.

Return:

A component.

CSDKExtObjectMgr functionalities

GetObjectBySID(unsigned long nSID)

Purpose: This method returns an object with specified SID.

Arguments:

nSID - Specifies the SID of the object.

Return:

An object.

GetObjectByName(BSTR sName)

Purpose: This method returns an object with specified name.

Arguments:

sName - Specifies the name of the object to retrieve.

Return:

An object.

GetObject(unsigned long nLocalID)

Purpose: This method returns an object with specified local ID.

Arguments:

nLocalID - Specifies the local ID of the object.

Return:

An object.

DeleteObjectBySID(unsigned long nSID)

Purpose: This method deletes the specified object from this manager.

Arguments:

nSID - Specifies the SID of the object to be deleted.

Return:

None.

DeleteAll()

Purpose: This method deletes all objects from this manager.

Return:

None.

AddCategory(BSTR sName, OLE_HANDLE hIcon, BSTR shortDescription, BSTR longDescription)

Purpose: This method adds a category to this manager.

Remarks: Once the manager has a category, the objects in this manager may be grouped by categories by setting their categories using the method AssignToCategory().

Arguments:

sName - Specifies the name of the category to be added. The name must be unique to this manager that is if a category with this name already exists in this manager then you may not reuse this name.

hIcon - A handle to an icon that will represent the category. If you do not wish to use this argument, set it to NULL.

shortDescription - A short description of the category. This may be used by tooltips. If you do not wish to use this argument, set it to NULL.

longDescription - A long description of the category. If you do not wish to use this argument, set it to NULL.

Return:

None.

`MoveBackOnePosition``(IDispatch* pDispObject)`

Purpose: This method moves the specified object's position in the list one position towards the back.

Remarks: The tail of the list is the front, and the head of the list is the back.

Argument pSuccess will be set to VARIANT_FALSE if the specified object is already at the back.

Arguments:

pDispObject - A pointer to the IDispatch interface of the object to move.

Return:

BOOL.

TRUE if the object was moved towards the back, FALSE otherwise.

`MoveForwardOnePosition``(IDispatch* pDispObject)`

Purpose: This method moves the specified object's position in the list one position towards the front.

Remarks: The tail of the list is the front, and the head of the list is the back.

Argument pSuccess will be set to VARIANT_FALSE if the specified object is already at the front.

Arguments:

pDispObject - A pointer to the IDispatch interface of the object to move.

Return:

BOOL.

TRUE if the object was moved towards the front, FALSE otherwise.

`MoveToFront``(IDispatch* pDispObject)`

Purpose: This method moves the specified object's position in the list to the front.

Remarks: The tail of the list is the front, and the head of the list is the back.

Argument pSuccess will be set to VARIANT_FALSE if the specified object is already at the front.

Arguments:

pDispObject - A pointer to the IDispatch interface of the object to move.

Return:

BOOL.

TRUE if the object was moved to the front, FALSE otherwise.

MoveToBack(IDispatch* pDispObject,)

Purpose: This method moves the specified object's position in the list to the back.

Remarks: The tail of the list is the front, and the head of the list is the back.

Argument pSuccess will be set to VARIANT_FALSE if the specified object is already at the front.

Return:

BOOL.

TRUE if the object was moved to the front, FALSE otherwise.

CSDKExtObject functionalities

get_Name()

Purpose: This method returns the name of this object.

Return:

String.

The name of the object as a string.

put_Name(BSTR pName)

Purpose: This method sets the name of this object.

Arguments:

pName - A null terminating string specifying the name of the component.

Return:

None.

GetIteration()

Purpose: This method returns the current iteration in the document at this time.

Return:

Long.

The current sweep iteration as a long.

GetMaxIterations()

Purpose: This method returns the maximum number of iterations the document supports at this time.

Return:

Long.

The max number of sweep iterations as a long.

`GetSID()`

Purpose: This method returns the system ID (SID) of this object.

Remarks: This value uniquely identifies this object in the document.

Return:

Unsigned long.

The SID of the object as an unsigned long.

CSDKComponent

IsSecure()

Purpose: This method determines whether the component is secure or not.

Return:

BOOL.

TRUE if the component is secure, FALSE otherwise.

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
Dim Canvas1_Component1
Set Canvas1_Component1 = Canvas1.CreateComponent("CW
Laser","{6DA31CEE-058F-11D4-93BD-0050DAB7C5D6}",180,210, 34, 34,0)
Dim bSecure
bSecure = Canvas1_Component1.IsSecure
MsgBox(bSecure)
```

get_Cost()

Purpose: This method returns cost of the component.

Return:

Double.

The cost of the component as a double.

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
Dim Canvas1_Component1
Set Canvas1_Component1 = Canvas1.CreateComponent("CW
Laser","{6DA31CEE-058F-11D4-93BD-0050DAB7C5D6}",180,210, 34, 34,0)
Dim dCost
dCost = Canvas1_Component1.Cost
MsgBox(dCost)
```

put_Cost()

Purpose: This method sets component's cost.

Return:

None.

Example:

```
Dim Lm
```

```
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
Dim Canvas1_Component1
Set Canvas1_Component1 = Canvas1.CreateComponent("CW
Laser","{6DA31CEE-058F-11D4-93BD-0050DAB7C5D6}",180,210, 34, 34,0)
Canvas1_Component1.Cost = 100.5
Dim dCost
dCost = Canvas1_Component1.Cost
MsgBox(dCost)
```

GetComponentLength()

Purpose: This method retrieves the internal component length.

Return:

Double.

The length of the component as a double.

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
Dim Canvas1_Component1
Set Canvas1_Component1 = Canvas1.CreateComponent("CW
Laser","{6DA31CEE-058F-11D4-93BD-0050DAB7C5D6}",180,210, 34, 34,0)
Dim dLength
dLength = Canvas1_Component1.GetComponentLength
MsgBox(dLength)
```

GetOutputPort(long nNum)

Purpose: This method retrieves component's output port .

Arguments:

nNum - A long value indicating the output port sequence number.

Return:

The output port.

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
Dim Canvas1_Component1
```

```
Set Canvas1_Component1 = Canvas1.CreateComponent("CW
Laser","{6DA31CEE-058F-11D4-93BD-0050DAB7C5D6}",180,210, 34, 34,0)
Dim OutputPort
Set OutputPort  = Canvas1_Component1.GetOutputPort(1)
```

GetInputPort(long nNum)

Purpose: This method retrieves component's input port.

Arguments:

nNum - A long value indicating the input port sequence number.

Return:

The input port.

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
Dim Canvas1_Component2
Set Canvas1_Component2 = Canvas1.CreateComponent("EDFA","{255EDC8F-
37E4-11D4-93EC-0050DAB7C5D6}",260,190, 32, 32,0)
Dim InputPort
Set InputPort  = Canvas1_Component2.GetInputPort(1)
```

UnLockSecurity(BSTR strPassword)

Purpose: This method unlocks security component.

Arguments:

strPassword - A null terminating string that specifies the password to unlock the security for this component.

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Layout1.Name  = "Layout 1"
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
Dim Canvas1_Component15
Set Canvas1_Component15 = Canvas1.CreateComponent("CW
Laser","{6DA31CEE-058F-11D4-93BD-0050DAB7C5D6}",110,130, 34, 34,0)
Canvas1_Component15.Name  = "CW Laser"
Canvas1_Component15.LockSecurity("test")
Canvas1_Component15.UnLockSecurity("test")
```

LockSecurity(BSTR strPassword)

Purpose: This method locks security for component.

Arguments:

strPassword - A null terminating string that specifies the password to lock the security for this component.

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Layout1.Name  = "Layout 1"
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
Dim Canvas1_Component15
Set Canvas1_Component15 = Canvas1.CreateComponent("CW
Laser","{6DA31CEE-058F-11D4-93BD-0050DAB7C5D6}",110,130, 34, 34,0)
Canvas1_Component15.Name  = "CW Laser"
Canvas1_Component15.LockSecurity("test")
Canvas1_Component15.UnLockSecurity("test")
```

`GetPort``(long nX, long nY)`

Purpose: This method returns a port at a certain point on the screen.

Arguments:

nX - Specifies the X coordinates of point.

nY - Specifies the Y coordinates of the point.

Return:

The port.

`GetPortMgr()`

Purpose: This method returns component's Port manager.

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Layout1.Name  = "Layout 1"
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
Dim Canvas1_Component15
Set Canvas1_Component15 = Canvas1.CreateComponent("CW
Laser","{6DA31CEE-058F-11D4-93BD-0050DAB7C5D6}",110,130, 34, 34,0)
Dim portMgr
Set portMgr = Canvas1_Component15.GetPortMgr
```

`GetResultMgr()`

Purpose: This method returns component's Result manager.

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
```

```
Layout1.Name  = "Layout 1"
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
Dim Canvas1_Component15
Set Canvas1_Component15 = Canvas1.CreateComponent("CW
Laser","{6DA31CEE-058F-11D4-93BD-0050DAB7C5D6}",110,130, 34, 34,0)
Dim resultMgr
Set resultMgr = Canvas1_Component15.GetResultMgr
```

GetParameterMgr()

Purpose: This method returns component's Parameter manager.

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Layout1.Name  = "Layout 1"
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
Dim Canvas1_Component15
Set Canvas1_Component15 = Canvas1.CreateComponent("CW
Laser","{6DA31CEE-058F-11D4-93BD-0050DAB7C5D6}",110,130, 34, 34,0)
Dim parameterMgr
Set parameterMgr = Canvas1_Component15.GetParameterMgr
```

GetGraphMgr()

Purpose: This method returns component's Graph manager.

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Layout1.Name  = "Layout 1"
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
Dim Canvas1_Component15
Set Canvas1_Component15 = Canvas1.CreateComponent("CW
Laser","{6DA31CEE-058F-11D4-93BD-0050DAB7C5D6}",110,130, 34, 34,0)
Dim graphMgr
Set graphMgr = Canvas1_Component15.GetGraphMgr
```

CleanMonitorBuffer(long nIteration)

Purpose: This method purges component's monitor buffer for the current iteration.

Arguments:

nIteration - A long value indicating the current sweep iteration.

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
```


```
Set Layout1 = Lm.GetCurrentLayout
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
Dim Canvas1_Component1
Set Canvas1_Component1 = Canvas1.CreateComponent("CW
Laser","{6DA31CEE-058F-11D4-93BD-0050DAB7C5D6}",180,210, 34, 34,0)
Canvas1_Component1.CleanMonitorBuffer 1
```

SetParameterValue(BSTR sParName, VARIANT newValue)

Purpose: This method sets value for component's parameter.

Arguments:

sParName - A null terminating string specifying the name of the parameter.

newValue - Specifies parameter's new value. The following types are extracted from the VARIANT.

- VT_I2 - short
- VT_I4 - long
- VT_R4 - float
- VT_R8 - double
- VT_BSTR - BSTR
- VT_BOOL - BOOL

Return:

None.

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
Dim Canvas1_Component1
Set Canvas1_Component1 = Canvas1.CreateComponent("CW
Laser","{6DA31CEE-058F-11D4-93BD-0050DAB7C5D6}",180,210, 34, 34,0)
Canvas1_Component1.SetParameterValue "Frequency", 193.1
Canvas1_Component1.SetParameterValue "Enabled", TRUE
```

SetSweepParameterValue(BSTR sParName, long nIter, VARIANT newValue)

Purpose: This method sets value for component's parameter for the specified sweep iteration.

Arguments:

sParName - A null terminating string specifying the name of the parameter

nIter - specifies sweep iteration number.

newValue - Specifies parameter's new value. The following types are extracted from the VARIANT.

- VT_I2 - short

- VT_I4 - long
- VT_R4 - float
- VT_R8 - double
- VT_BSTR - BSTR
- VT_BOOL - BOOL

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Layout1.SetTotalSweepIterations(5)
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
Dim Canvas1_Component1
Set Canvas1_Component1 = Canvas1.CreateComponent("CW
Laser","{6DA31CEE-058F-11D4-93BD-0050DAB7C5D6}",180,210, 34, 34,0)
Canvas1_Component1.SetSweepParameterValue "Frequency",1,  193.1
Canvas1_Component1.SetSweepParameterValue "Frequency",2,  194.1
Canvas1_Component1.SetSweepParameterValue "Frequency",3,  195.1
Canvas1_Component1.SetSweepParameterValue "Frequency",4,  196.1
Canvas1_Component1.SetSweepParameterValue "Frequency",5,  197.1
```

SetParameterValueMxN(BSTR sParName, VARIANT arrMxN, VARIANT arrColumnTitles, long nYColumn)

Purpose: This method sets value for component's MxN parameter.

SetSweepParameterValueMxN(BSTR sParName, long nIteration, VARIANT arrMxN, VARIANT arrColumnTitles, long nYColumn);

Purpose: This method sets value for component's MxN parameter for the specified sweep iteration.

SetParameterMode(BSTR sParName, long nMode)

Purpose: This method sets component's parameter mode.

Arguments:

sParName - A null terminating string specifying the name of the parameter.

nMode - A long value specifying the mode of the parameter. The mode settings can be:

- 0 - Normal mode
- 1 - Optimized mode
- 2 - Iterated mode
- 3 - Scripted mode

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
```


```
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
Dim Canvas1_Component1
Set Canvas1_Component1 = Canvas1.CreateComponent("CW
Laser","{6DA31CEE-058F-11D4-93BD-0050DAB7C5D6}",180,210, 34, 34,0)
Canvas1_Component1.SetParameterMode "Power", 0
```

SetParameterScript(BSTR sParName, BSTR sScript)

Purpose: This method sets component's parameter script

Arguments:

sParName - A null terminating string specifying the name of the parameter.

sScript - A null terminating string specifying the script.

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
Dim Canvas1_Component1
Set Canvas1_Component1 = Canvas1.CreateComponent("CW
Laser","{6DA31CEE-058F-11D4-93BD-0050DAB7C5D6}",180,210, 34, 34,0)
Canvas1_Component1.SetParameterScript "Sample rate", "Sample rate"
```

GetGraph(BSTR sGraphName)

Purpose: This method returns a component's graph.

Arguments:

sName - A null terminating string specifying the name of the result.

Return:

A graph.

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
Dim Canvas1_Component2
Set Canvas1_Component2 = Canvas1.CreateComponent("Optical Spectrum
Analyzer","{F11D0C25-3C7D-11D4-93F0-0050DAB7C5D6}",300,280, 40,
34,0)
Canvas1_Component2.Name  = "Optical Spectrum Analyzer"
Dim graph
Set graph = Canvas1_Component2.GetGraph("Sampled signal spectrum")
Dim unit
unit = graph.GetUnitX
MsgBox(unit)
```

GetParameterValue(BSTR sName)

Purpose: This method returns the value of the parameter.

Arguments:

sName - A null terminating string that specifies the name of the parameter.

Return:

The parameter's value as a VARIANT.

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
Dim Canvas1_Component1
Set Canvas1_Component1 = Canvas1.CreateComponent("CW
Laser","{6DA31CEE-058F-11D4-93BD-0050DAB7C5D6}",180,210, 34, 34,0)
Canvas1_Component1.SetParameterMode "Frequency", 0
Canvas1_Component1.SetParameterUnit "Frequency", "THz"
Canvas1_Component1.SetParameterValue "Frequency", 193.1
Dim sValue
sValue = Canvas1_Component1.GetParameterValue("Frequency")
MsgBox(sValue)
```

GetSweepParameterValue(BSTR sName, long nIteration)

Purpose: This method returns the value of the parameter for specified iteration.

Arguments:

sName - A null terminating string specifying the name of the parameter.

nIteration - A long value specifying the sweep iteration.

Return:

The parameter's value as a VARIANT.

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Layout1.SetTotalSweepIterations(5)
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
Dim Canvas1_Component12
Set Canvas1_Component12 = Canvas1.CreateComponent("CW
Laser","{6DA31CEE-058F-11D4-93BD-0050DAB7C5D6}",310,210, 34, 34,0)
Canvas1_Component12.SetParameterMode "Frequency", 2
Canvas1_Component12.SetParameterUnit "Frequency", "THz"
Canvas1_Component12.SetSweepParameterValue "Frequency",1,  193.1
Canvas1_Component12.SetSweepParameterValue "Frequency",2,  194.1
Canvas1_Component12.SetSweepParameterValue "Frequency",3,  195.1
```


```
Canvas1_Component12.SetSweepParameterValue "Frequency",4,  196.1
Canvas1_Component12.SetSweepParameterValue "Frequency",5,  197.1
Dim sValue
sValue = Canvas1_Component12.GetSweepParameterValue("Frequency", 3)
MsgBox(sValue)
```

GetParameterScript(BSTR sName)

Purpose: This method returns script of the specified parameter.

Arguments:

sName - A null terminating string specifying the name of the parameter.

Return:

String.

The parameter's script.

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
Dim Canvas1_Component1
Set Canvas1_Component1 = Canvas1.CreateComponent("CW
Laser","{6DA31CEE-058F-11D4-93BD-0050DAB7C5D6}",180,210, 34, 34,0)
Canvas1_Component1.SetParameterMode "Iterations", 3
Canvas1_Component1.SetParameterScript "Iterations", "Iterations"
Dim sScript
sScript = Canvas1_Component1.GetParameterScript("Iterations")
MsgBox(sScript)
```

GetParameterUnit(BSTR sName)

Purpose: This method returns the unit of the specified parameter.

Arguments:

sName - A null terminating string that specifies the name of the parameter.

Return:

String.

The parameter's unit.

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
Dim Canvas1_Component1
Set Canvas1_Component1 = Canvas1.CreateComponent("CW
Laser","{6DA31CEE-058F-11D4-93BD-0050DAB7C5D6}",180,210, 34, 34,0)
```

```
Dim sUnit
sUnit = Canvas1_Component1.GetParameterUnit("Frequency")
MsgBox(sUnit)
```

GetParameterMode(BSTR sName)

Purpose: This method returns the mode of the specified parameter.

Arguments:

sName - A null terminating string that specifies the name of the parameter.

Return :

Long.

The parameter's mode as a long. Possible return values:

- 0 - Normal mode
- 1 - Optimized mode
- 2 - Iterated mode
- 3 - Scripted mode

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
Dim Canvas1_Component1
Set Canvas1_Component1 = Canvas1.CreateComponent("CW
Laser","{6DA31CEE-058F-11D4-93BD-0050DAB7C5D6}",180,210, 34, 34,0)
Dim nMode
nMode = Canvas1_Component1.GetParameterMode("Random seed index")
MsgBox(nMode)
```

GetResult(BSTR sName)

Purpose: This method returns the component's result.

Arguments:

sName - A null terminating string that specifies the name of the result.

Return:

The result.

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
Dim Canvas1_Component2
```


```
Set Canvas1_Component2 = Canvas1.CreateComponent("Optical Power
Meter Visualizer","{F11D0C25-3C7D-11D4-93F0-0050DAB7C5D6}",350,270,
34, 38,0)
Dim dResult
Set dResult = Canvas1_Component2.GetResult("Total Power (W)")
```

SetParameterUnit(BSTR sParName, BSTR sUnit)

Purpose: This method sets the component's parameter unit.

Arguments:

sParName - A null terminating string that specifies the name of the parameter.

sUnit - A null terminating string that specifies the unit.

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
Dim Canvas1_Component1
Set Canvas1_Component1 = Canvas1.CreateComponent("CW
Laser","{6DA31CEE-058F-11D4-93BD-0050DAB7C5D6}",180,210, 34, 34,0)
Canvas1_Component1.SetParameterUnit "Power", "dBm"
```

GetParameter(BSTR sParameterName)

Purpose: This method returns the component's parameter.

Arguments:

sParameterName - A null terminating string that specifies the name of the parameter.

Return:

The parameter.

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
Dim Canvas1_Component2
Set Canvas1_Component2 = Canvas1.CreateComponent("Optical Power
Meter Visualizer","{F11D0C25-3C7D-11D4-93F0-0050DAB7C5D6}",350,270,
34, 38,0)
Dim par
Set par = Canvas1_Component2.GetParameter("Minimum value")
```

GetParameterSID(BSTR sParameterName, unsigned long *pSID)

Purpose: This method returns the parameter's SID.

Arguments:

sParameterName - A null terminating string that specifies the name of the parameter.

Return:

The parameter's SID.

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
Dim Canvas1_Component2
Set Canvas1_Component2 = Canvas1.CreateComponent("Optical Power
Meter Visualizer","{F11D0C25-3C7D-11D4-93F0-0050DAB7C5D6}",350,270,
34, 38,0)
Dim parSID
parSID = Canvas1_Component2.GetParameterSID("Minimum value")
```

CSDKLayoutObject functionalities

`GetResizable()`

Purpose: This method returns the resize option of the component.

Return:

Long.

The possible return values are.

- 0 - No Resize
- 1 - Horizontal
- 2 - Vertical
- 3 - Both

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
Dim Canvas1_Component1
Set Canvas1_Component1 = Canvas1.CreateComponent("CW
Laser","{6DA31CEE-058F-11D4-93BD-0050DAB7C5D6}",180,210, 34, 34,0)
Dim resize1
resize1 = Canvas1_Component1.GetResizable
MsgBox(resize1)
Dim Canvas1_Component2
Set Canvas1_Component2 = Canvas1.CreateComponent("Ideal
Demux","{1747A24E-52A0-11D4-9403-0050DAB7C5D6}",270,160, 34, 86,0)
Dim resize2
resize2 = Canvas1_Component2.GetResizable
MsgBox(resize2)
```

SetPosition(long left, long top, long right, long bottom)

Purpose: This method sets the component position.

Arguments:

left - specifies the x-coordinate of the left edge of the rectangle.

top - specifies the y-coordinate of the top edge of the rectangle.

right - specifies the x-coordinate of the right edge of the rectangle.

bottom - specifies the y-coordinate of the bottom edge of the rectangle.

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
Dim Canvas1_Component1
Set Canvas1_Component1 = Canvas1.CreateComponent("CW
Laser","{6DA31CEE-058F-11D4-93BD-0050DAB7C5D6}",170,130, 34, 34,0)
Canvas1_Component1.SetPosition 182, 142, 216, 176
```

FlipObject()

Purpose: This method flips the component in the layout.

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
Dim Canvas1_Component1
Set Canvas1_Component1 = Canvas1.CreateComponent("CW
Laser","{6DA31CEE-058F-11D4-93BD-0050DAB7C5D6}",180,210, 34, 34,0)
Canvas1_Component1.FlipObject
```

IsFlipped()

Purpose: This method determines whether component is flipped or not.

Return:

BOOL.

Passible return values.

- 1 if component is flipped.
- 0 if component is not flipped.

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
```

```
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
Dim Canvas1_Component6
Set Canvas1_Component6 = Canvas1.CreateComponent("CW
Laser","{6DA31CEE-058F-11D4-93BD-0050DAB7C5D6}",184,100, -34, 34,0)
Canvas1_Component6.Name  = "CW Laser"
Canvas1_Component6.FlipObject
Dim bFlipped
bFlipped = Canvas1_Component6.IsFlipped
MsgBox(bFlipped)
```

CSDKExtObject functionalities

`get_Name()`

Purpose: This method returns the name of this object.

Return:

String.

The name of the object as a string.

`put_Name``( BSTR pName)`

Purpose: This method sets the name of this object.

Arguments:

pName - A null terminating string that specifies the name of the component.

Return:

None.

`GetIteration()`

Purpose: This method returns the current iteration in the document at this time.

Return:

Long.

The current sweep iteration as a long.

`GetMaxIterations();`

Purpose: This method returns the maximum number of iterations the document supports at this time.

Return:

Long.

The max number of sweep iterations as a long.

`GetSID()`

Purpose: This method returns the system ID (SID) of this object.

Remarks: This value uniquely identifies this object in the document.

Return:

Unsigned long.

The SID of the object as an unsigned long.

SetGraphYData(BSTR sParName, VARIANT arrYData)
Purpose: This method sets component's graph Y data.
Arguments:
sParName – A null terminating string specifying the name of the parameter.
arrYData - – A VARIANT array for component's graph Y data.

SetGraphXData(BSTR sParName,VARIANT arrXData)
Purpose: This method sets component's graph Y data.
Arguments:
sParName – A null terminating string specifying the name of the parameter.
arrYData – A VARIANT array for component's graph Y data.

GetSignalByPortName(BSTR sPortName, long nIndex)
Purpose: This method retrieves a signal from monitor's object buffers array based on port name for the current sweep iteration. There should be a monitor attached to this port in order to retreive the signal.
Arguments:
sPortName - A null terminating string specifying the name of the port.
nIndex – Specifies the index of monitor object buffers array. nIndex is zero based.
Example:

```
Dim pSignalO
nSize = Canvas1_Component7.GetSignalBufferSizeByPortName("Output")
Set pSignalO = Canvas1_Component7.GetSignalByPortName("Output", nSize - 1)
```

GetSignalByPortId(long nPort, long nIndex)
Purpose: This method retrieves a signal from monitor's object buffers array based on port ID for the current sweep iteration. There should be a monitor attached to this port in order to retreive the signal.
Arguments:
nPort - A long value indicating the output port sequence number.
nIndex – Specifies the index of monitor object buffers array. nIndex is zero based.
Example:

```
Dim pSignalO
Set pSignalO = Canvas1_Component7.GetSignalByPortId(1 , 0)
```

GetSweepSignalByPortName(BSTR sPortName, long nIndex, long nIteration)
Purpose: This method retrieves a signal from monitor's object buffers array based on port name for specified sweep iteration. There should be a monitor attached to this port in order to retreive the signal.
Arguments:
sPortName - A null terminating string specifying the name of the port.
nIndex – Specifies the index of monitor object buffers array. nIndex is zero based.
nIteration – Specifies sweep iteration number. nIteration is 1 based.
Example:

```
Dim pSignalO
Set pSignalO = Canvas1_Component7.GetSweepSignalByPortName("Output", 0, 1)
```

GetSweepSignalByPortId(long nPort, long nIndex, long nIteration)

Purpose: This method retrieves a signal from monitor's object buffers array based on port ID for specified sweep iteration. There should be a monitor attached to this port in order to retreive the signal.

Arguments:

nPort - A long value indicating the output port sequence number.

nIndex – Specifies the index of monitor object buffers array. nIndex is zero based.

nIteration – Specifies sweep iteration number. nIteration is 1 based.

Example:

```
Dim pSignalO
Set pSignalO = Canvas1_Component7.GetSweepSignalByPortId(1 , 0, 1)
```

CreateSignalByPortId(long nPortIndex)

Purpose: This method creates a signal of port's signal type and add to port's buffer array, based on port ID for the current sweep iteration. If monitor is attached to this port the signal will be duplicate and attache to the monitor's buffer array.

Arguments:

nPortIndex - A long value indicating the output port sequence number.

Example:

```
Dim pSignalO
Set pSignalO = Canvas1_Component7.CreateSignalByPortId(1)
```

CreateSignalByPortName(BSTR sPortName)

Purpose: This method creates a signal of port's signal type and add to port's buffer array, based on port name for the current sweep iteration. If monitor is attached to this port the signal will be duplicate and attache to the monitor's buffer array.

Arguments:

sPortName - A null terminating string specifying the name of the port.

Example:

```
Dim pSignalO
Set pSignalO = Canvas1_Component7.CreateSignalByPortName("Output")
```

CreateSweepSignalByPortId(long nPortIndex, long nIteration)

Purpose: This method creates a signal of port's signal type and add to port's buffer array, based on port ID for specified sweep iteration. If monitor is attached to this port the signal will be duplicate and attache to the monitor's buffer array.

Arguments:

nPortIndex - A long value indicating the output port sequence number.

nIteration – Specifies sweep iteration number. nIteration is 1 based.

Example:

```
Dim pSignalO
Set pSignalO = Canvas1_Component7.CreateSweepSignalByPortId(1,1)
```

CreateSweepSignalByPortName(BSTR sPortName, long nIteration)

Purpose: This method creates a signal of port's signal type and add to port's buffer array, based on port name for specified sweep iteration. If monitor is attached to this port the signal will be duplicate and attache to the monitor's buffer array.
Arguments:
sPortName - A null terminating string specifying the name of the port.
nIteration – Specifies sweep iteration number. nIteration is 1 based.
Example:

```
Dim pSignalO
Set pSignalO = Canvas1_Component7.CreateSweepSignalByPortName("Output", 1)
```

GetSignalBufferSizeByPortId(long nPortIndex)
Purpose: This method retrieves the size of monitor's object buffers array based on port ID for the current sweep iteration. There should be a monitor attached to this port in order to retreive the size of the array.
Arguments:
nPortIndex - A long value indicating the output port sequence number.
Example:

```
nSize = Canvas1_Component7.GetSignalBufferSizeByPortId(1)
```

GetSignalBufferSizeByPortName(BSTR sPortName)
Purpose: This method retrieves the size of monitor's object buffers array based on port name for the current sweep iteration. There should be a monitor attached to this port in order to retreive the size of the array.
Arguments:
sPortName - A null terminating string specifying the name of the port.
Example:

```
nSize = Canvas1_Component7.GetSignalBufferSizeByPortName("Output")
```

GetSweepSignalBufferSizeByPortId(long nPortIndex long nIteration)
Purpose: This method retrieves the size of monitor's object buffers array based on port ID for specified sweep iteration. There should be a monitor attached to this port in order to retreive the size of the array.
Arguments:
nPortIndex - A long value indicating the output port sequence number.
nIteration – Specifies sweep iteration number. nIteration is 1 based.
Example:

```
nSize = Canvas1_Component7.GetSweepSignalBufferSizeByPortId(1,1)
```

GetSweepSignalBufferSizeByPortName(BSTR sPortName, long nIteration)
Purpose: This method retrieves the size of monitor's object buffers array based on port name for specified sweep iteration. There should be a monitor attached to this port in order to retreive the size of the array.
Arguments:
sPortName - A null terminating string specifying the name of the port.
nIteration – Specifies sweep iteration number. nIteration is 1 based.
Example:

```
nSize = Canvas1_Component7.GetSweepSignalBufferSizeByPortName("Output",1)
```

CreateInitializeSignalByPortId(long nPortIndex, IDispatch* pDisIn)

Purpose: This method clones the signal (pDisIn) and add it to port's buffer array, based on port ID for the current sweep iteration. If monitor is attached to this port the signal will be duplicate and attache to the monitor's buffer array.

Arguments:

nPortIndex - A long value indicating the output port sequence number.

pDisIn – The signal that is used for a clone.

Example:

```
Dim pSignal0
Set pSignal0 = Canvas1_Component1.GetSignalByPortId(1, 0)
Dim pSignal02
Set pSignal02 = Canvas1_Component1.CreateInitializeSignalByPortId(1,pSignal0 )
```

CreateInitializeSignalByPortName(BSTR sPortName, IDispatch* pDisIn)

Purpose: This method clones the signal (pDisIn) and add it to port's buffer array, based on port name for the current sweep iteration. If monitor is attached to this port the signal will be duplicate and attache to the monitor's buffer array.

Arguments:

sPortName - A null terminating string specifying the name of the port.

pDisIn – The signal that is used for a clone.

Example:

```
Dim pSignal0
Set pSignal0 = Canvas1_Component1.GetSignalByPortName("Output", 0)
Dim pSignal02
Set pSignal02 = Canvas1_Component1.CreateInitializeSignalByPortName("Output",
pSignal0 )
```

CreateInitializeSweepSignalByPortId(long nPortIndex, long nIteration, IDispatch* pDisIn)

Purpose: This method clones the signal (pDisIn) and add it to port's buffer array, based on port ID for specified sweep iteration. If monitor is attached to this port the signal will be duplicate and attache to the monitor's buffer array.

Arguments:

nPortIndex - A long value indicating the output port sequence number.

pDisIn – The signal that is used for a clone.

Example:

```
Dim pSignal0
Set pSignal0 = Canvas1_Component1.GetSweepSignalByPortId(1,0,1)
Dim pSignal02
Set pSignal02 =
Canvas1_Component1.CreateInitializeSweepSignalByPortId(1,1,pSignal0)
```

CreateInitializeSweepSignalByPortName(BSTR sPortName, long nIteration, IDispatch* pDisIn)

Purpose: This method clones the signal (pDisIn) and add it to port's buffer array, based on port name for specified sweep iteration. If monitor is attached to this port the signal will be duplicate and attache to the monitor's buffer array.

Arguments:
sPortName - A null terminating string specifying the name of the port.
pDisIn – The signal that is used for a clone.
Example:

```
     Dim pSignalO
     Set pSignalO = Canvas1_Component1.GetSweepSignalByPortName(“Output”,0,1)
     Dim pSignalO2
     Set pSignalO2 =
Canvas1_Component1.CreateInitializeSweepSignalByPortName(“Output”,1,pSignalO)
```

SynchronizeSignalToMonitorByPortId(long nPortIndex, IDispatch* pDisIn)
Purpose: This method finds a signal from port’s monitor buffer array with the same signal ID as (pDisIn) and copy (synchronize) the signal, based on port Id for the current sweep iteration.
Arguments:
nPortIndex - A long value indicating the output port sequence number.
pDisIn – The signal that is used for a synchtronization.

SynchronizeSignalToMonitorByPortName(BSTR sPortName, IDispatch* pDisIn)
Purpose: This method finds a signal from port’s monitor buffer array with the same signal ID as (pDisIn) and copy (synchronize) the signal, based on port name for the current sweep iteration.
Arguments:
sPortName - A null terminating string specifying the name of the port.
pDisIn – The signal that is used for a synchtronization.
Example:

```
     Dim pSignalO
     Set pSignalO = Canvas1_Component1.GetSweepSignalByPortName(“Output”,0,1)
     Dim pSignalO2
     Set pSignalO2 =
Canvas1_Component1.CreateInitializeSweepSignalByPortName(“Output”,1,pSignalO)
     Call pSignalO2.GetNoiseBinSignalBandwidth( nNumNB - 1, nLF02, nUF02 )
     Call pSignalO2.SetNoiseBinSignalBandwidth( nNumNB - 1, nLF0*10, nUF0*10 )
     Call    Canvas1_Component1.SynchronizeSignalToMonitorByPortName("Output",
pSignalO2)
```

SynchronizeSweepSignalToMonitorByPortId(long nPortIndex,long nIteration, IDispatch* pDisIn)
Purpose: This method finds a signal from port’s monitor buffer array with the same signal ID as (pDisIn) and copy (synchronize) the signal, based on port Id for specified sweep iteration.
Arguments:
nPortIndex - A long value indicating the output port sequence number.
nIteration – Specifies sweep iteration number. nIteration is 1 based.
pDisIn – The signal that is used for a synchtronization.

SynchronizeSweepSignalToMonitorByPortName(BSTR sPortName,long nIteration, IDispatch* pDisIn)
Purpose: This method finds a signal from port’s monitor buffer array with the same signal ID as (pDisIn) and copy (synchronize) the signal, based on port name for specified sweep iteration.

```
Arguments:
sPortName - A null terminating string specifying the name of the
port.
nIteration – Specifies sweep iteration number. nIteration is 1 based.
pDisIn – The signal that is used for a synchtronization.
```


CSDKParameter

`IsStringFilename()`

Purpose: This function returns TRUE if parameter is a filename.

Return:

BOOL.

TRUE if parameter is a filename, FALSE otherwise.

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
Dim Canvas1_Component3
Set Canvas1_Component3 = Canvas1.CreateComponent("EDFA","{255EDC8F-
37E4-11D4-93EC-0050DAB7C5D6}",420,220, 32, 32,0)
Canvas1_Component3.Name  = "EDFA"
Dim parMgr
Set parMgr = Canvas1_Component3.GetParameterMgr
Dim param
Set param = parMgr.GetObjectByName("Cross section file name")
Dim bFilename
bFilename = param.IsStringFilename
MsgBox(bFilename)
```

`IsParameterLocked()`

Purpose: This method determines whether the parameter is locked or not.

Return:

BOOL.

TRUE if parameter is locked, FALSE otherwise.

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
Dim Canvas1_Component1
Set Canvas1_Component1 = Canvas1.CreateComponent("CW
Laser","{6DA31CEE-058F-11D4-93BD-0050DAB7C5D6}",180,210, 34, 34,0)
Dim parMgr
Set parMgr = Canvas1_Component1.GetParameterMgr
Dim param
Set param = parMgr.GetObjectByName("Iterations")
Dim bLocked
bLocked = param.IsParameterLocked
```

```
MsgBox(bLocked)
```

get_UIMode()

Purpose: This method returns parameter dependency style.

Return:

Long.

A long value indicating the dependency style. Possible values are:

- 0 - Enabled
- 1 - Disabled
- 2 - ReadOnly
- 3 - ReadOnlyButNotGrayed

put_UIMode(long newVal)

Purpose: This method sets parameter dependency style.

Arguments:

newVal - A long value indicating the dependency style. Possible values are:

- 0 - Enabled
- 1 - Disabled
- 2 - ReadOnly
- 3 - ReadOnlyButNotGrayed

get_Script()

Purpose: This method returns script of the parameter.

Return:

String.

A string that indicates the parameter's script.

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
Dim Canvas1_Component1
Set Canvas1_Component1 = Canvas1.CreateComponent("CW
Laser","{6DA31CEE-058F-11D4-93BD-0050DAB7C5D6}",180,210, 34, 34,0)
Dim parMgr
Set parMgr = Canvas1_Component1.GetParameterMgr
Dim param
Set param = parMgr.GetObjectByName("Iterations")
Dim script
script = param.Script
MsgBox(script)
```


put_Script(BSTR newVal)

Purpose: This method sets the parameter script.

Arguments:

newVal - A null terminating string that specifies the script.

Return:

None.

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
Dim Canvas1_Component1
Set Canvas1_Component1 = Canvas1.CreateComponent("CW
Laser","{6DA31CEE-058F-11D4-93BD-0050DAB7C5D6}",180,210, 34, 34,0)
Dim parMgr
Set parMgr = Canvas1_Component1.GetParameterMgr
Dim param
Set param = parMgr.GetObjectByName("Iterations")
param.Script = "2*Iterations"
```

get_Display()

Purpose: This method returns TRUE if parameter is set to be displayed.

Return:

BOOL.

TRUE if parameter is set to be displayed, FALSE otherwise.

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
Dim Canvas1_Component1
Set Canvas1_Component1 = Canvas1.CreateComponent("CW
Laser","{6DA31CEE-058F-11D4-93BD-0050DAB7C5D6}",180,210, 34, 34,0)
Dim parMgr
Set parMgr = Canvas1_Component1.GetParameterMgr
Dim param
Set param = parMgr.GetObjectByName("Frequency")
Dim bDisplay
bDisplay = param.Display
MsgBox(bDisplay)
```

put_Display(VARIANT_BOOL newVal)

Purpose: This method sets parameter to be displayed or not.

Arguments:

newVal - A BOOL specifying parameter to be displayed or not.

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
Dim Canvas1_Component1
Set Canvas1_Component1 = Canvas1.CreateComponent("CW
Laser","{6DA31CEE-058F-11D4-93BD-0050DAB7C5D6}",180,210, 34, 34,0)
Dim parMgr
Set parMgr = Canvas1_Component1.GetParameterMgr
Dim param
Set param = parMgr.GetObjectByName("Frequency")
param.Display = FALSE
```

`GetValue``( long nIteration)`

Purpose: This method returns the value of the parameter.

Arguments:

nIteration - A long value that specifies the sweep iteration.

Return:

The parameter's value as a variant.

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
Dim Canvas1_Component1
Set Canvas1_Component1 = Canvas1.CreateComponent("CW
Laser","{6DA31CEE-058F-11D4-93BD-0050DAB7C5D6}",180,210, 34, 34,0)
Canvas1_Component1.SetParameterMode "Frequency", 0
Canvas1_Component1.SetParameterUnit "Frequency", "THz"
Canvas1_Component1.SetParameterValue "Frequency", 193.1
Dim parMgr
Set parMgr = Canvas1_Component1.GetParameterMgr
Dim param
Set param = parMgr.GetObjectByName("Frequency")
Dim value_
value_ = param.GetValue(1)
MsgBox(value_)
```

`GetType()`

Purpose: This method returns the parameter's type.

Return:

Long.

A long value indicating the parameter's type. Possible return values are:

- 0 - BOOL
- 1 - Choice
- 2 - Double
- 3 - Long
- 4 - String
- 5 - MxN

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
Dim Canvas1_Component1
Set Canvas1_Component1 = Canvas1.CreateComponent("CW
Laser","{6DA31CEE-058F-11D4-93BD-0050DAB7C5D6}",180,210, 34, 34,0)
Canvas1_Component1.SetParameterMode "Frequency", 0
Canvas1_Component1.SetParameterUnit "Frequency", "THz"
Canvas1_Component1.SetParameterValue "Frequency", 193.1
Dim parMgr
Set parMgr = Canvas1_Component1.GetParameterMgr
Dim param
Set param = parMgr.GetObjectByName("Frequency")
Dim type_
type_ = param.GetType
MsgBox(type_)
```

`SetMode``( long nMode)`

Purpose: This method sets the parameter mode.

Arguments:

nMode - A long value specifying the mode of the parameter. The mode settings can be:

- 0 - Normal mode
- 1 - Optimized mode
- 2 - Iterated mode
- 3 - Scripted mode

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
```

```
Dim Canvas1_Component1
Set Canvas1_Component1 = Canvas1.CreateComponent("CW
Laser","{6DA31CEE-058F-11D4-93BD-0050DAB7C5D6}",180,210, 34, 34,0)
Canvas1_Component1.SetParameterUnit "Frequency", "THz"
Canvas1_Component1.SetParameterValue "Frequency", 193.1
Dim parMgr
Set parMgr = Canvas1_Component1.GetParameterMgr
Dim param
Set param = parMgr.GetObjectByName("Frequency")
param.SetMode(2)
```

GetMode()

Purpose: This method returns the mode of the parameter.

Return:

Long.

A long value that indicates the parameter's mode. Possible return values:

- 0 - Normal mode
- 1 - Optimized mode
- 2 - Iterated mode
- 3 - Scripted mode

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
Dim Canvas1_Component1
Set Canvas1_Component1 = Canvas1.CreateComponent("CW
Laser","{6DA31CEE-058F-11D4-93BD-0050DAB7C5D6}",180,210, 34, 34,0)
Canvas1_Component1.SetParameterMode "Frequency", 0
Canvas1_Component1.SetParameterUnit "Frequency", "THz"
Canvas1_Component1.SetParameterValue "Frequency", 193.1
Dim parMgr
Set parMgr = Canvas1_Component1.GetParameterMgr
Dim param
Set param = parMgr.GetObjectByName("Frequency")
Dim mode
mode = param.GetMode
MsgBox(mode)
```

GetValueString(long nIteration)

Purpose: This method returns the value of the parameter.

Arguments:

nIteration - A long value indicating the sweep iteration.

Return:

String.

A null terminating string that represents the parameter value.

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
Dim Canvas1_Component1
Set Canvas1_Component1 = Canvas1.CreateComponent("CW
Laser","{6DA31CEE-058F-11D4-93BD-0050DAB7C5D6}",180,210, 34, 34,0)
Canvas1_Component1.SetParameterMode "Frequency", 0
Canvas1_Component1.SetParameterUnit "Frequency", "THz"
Canvas1_Component1.SetParameterValue "Frequency", 193.1
Dim parMgr
Set parMgr = Canvas1_Component1.GetParameterMgr
Dim param
Set param = parMgr.GetObjectByName("Frequency")
Dim value
value = param.GetValueString(1)
MsgBox(value)
```

`GetUnit()`

Purpose: This method returns the unit of the parameter.

Return:

String.

The parameter's unit as a string value.

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
Dim Canvas1_Component1
Set Canvas1_Component1 = Canvas1.CreateComponent("CW
Laser","{6DA31CEE-058F-11D4-93BD-0050DAB7C5D6}",180,210, 34, 34,0)
Canvas1_Component1.SetParameterMode "Frequency", 0
Canvas1_Component1.SetParameterUnit "Frequency", "THz"
Canvas1_Component1.SetParameterValue "Frequency", 193.1
Dim parMgr
Set parMgr = Canvas1_Component1.GetParameterMgr
Dim param
Set param = parMgr.GetObjectByName("Frequency")
Dim unit
unit = param.GetUnit
MsgBox(unit)
```

GetMax()

Purpose: This method returns the max limit of the parameter.

Return:

Double:

The max limit of the parameter as a double value.

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
Dim Canvas1_Component1
Set Canvas1_Component1 = Canvas1.CreateComponent("CW
Laser","{6DA31CEE-058F-11D4-93BD-0050DAB7C5D6}",180,210, 34, 34,0)
Canvas1_Component1.SetParameterMode "Frequency", 0
Canvas1_Component1.SetParameterUnit "Frequency", "THz"
Canvas1_Component1.SetParameterValue "Frequency", 193.1
Dim parMgr
Set parMgr = Canvas1_Component1.GetParameterMgr
Dim param
Set param = parMgr.GetObjectByName("Frequency")
Dim max
max = param.GetMax
MsgBox(max)
```

GetMin()

Purpose: This method returns the min limit of the parameter.

Return:

Double:

The min limit of the parameter as a double value.

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
Dim Canvas1_Component1
Set Canvas1_Component1 = Canvas1.CreateComponent("CW
Laser","{6DA31CEE-058F-11D4-93BD-0050DAB7C5D6}",180,210, 34, 34,0)
Canvas1_Component1.SetParameterMode "Frequency", 0
Canvas1_Component1.SetParameterUnit "Frequency", "THz"
Canvas1_Component1.SetParameterValue "Frequency", 193.1
Dim parMgr
Set parMgr = Canvas1_Component1.GetParameterMgr
Dim param
```


```
Set param = parMgr.GetObjectByName("Frequency")
Dim min
min = param.GetMin
MsgBox(min)
```

get_Precision()

Purpose: This function returns the number of decimal places for floating-point values.

Return:

Long.

The number of decimal places for floating-point values as a long value.

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
Dim Canvas1_Component1
Set Canvas1_Component1 = Canvas1.CreateComponent("CW
Laser","{6DA31CEE-058F-11D4-93BD-0050DAB7C5D6}",180,210, 34, 34,0)
Dim parMgr
Set parMgr = Canvas1_Component1.GetParameterMgr
Dim param
Set param = parMgr.GetObjectByName("Frequency")
Dim nPrecision
nPrecision = param.Precision
MsgBox(nPrecision)
```

put_Precision(long newVal)

Purpose: This function specifies the number of decimal places for floating-point values.

Arguments:

newVal - A long value that indicates the number of decimal places for floating-point values.

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
Dim Canvas1_Component1
Set Canvas1_Component1 = Canvas1.CreateComponent("CW
Laser","{6DA31CEE-058F-11D4-93BD-0050DAB7C5D6}",180,210, 34, 34,0)
Dim parMgr
Set parMgr = Canvas1_Component1.GetParameterMgr
Dim param
Set param = parMgr.GetObjectByName("Frequency")
```

```
param.Precision = 4
```

CSDKExtObject functionalities

`get_Name()`

Purpose: This method returns the name of this object.

Return:

String.

The name of the object as a string.

`put_Name``( BSTR pName)`

Purpose: This method sets the name of this object.

Arguments:

pName - A null terminating string that specifies the name of the component.

Return:

None.

`GetIteration()`

Purpose: This method returns the current iteration in the document at this time.

Return:

Long.

The current sweep iteration as a long.

`GetMaxIterations()`

Purpose: This method returns the maximum number of iterations the document supports at this time.

Return:

Long.

The max number of sweep iterations as a long.

`GetSID()`

Purpose: This method returns the system ID (SID) of this object.

Remarks: This value uniquely identifies this object in the document.

Return:

Unsigned long.

The SID of the object as an unsigned long.

CSDKResult

GetUnit()

Purpose: This method returns the unit of the result.

Return:

String.

The unit of the result as a string value.

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
Dim Canvas1_Component2
Set Canvas1_Component2 = Canvas1.CreateComponent("Optical Power
Meter Visualizer","{F11D0C25-3C7D-11D4-93F0-0050DAB7C5D6}",330,210,
34, 38,0)
Dim resMgr
Set resMgr = Canvas1_Component2.GetResultMgr
Dim res
Set res = resMgr.GetObjectByName("Noise Power (dBm)")
Dim sUnit
sUnit = res.GetUnit
MsgBox(sUnit)
```

GetValue(long nIteration)

Purpose: This method returns the value of the result.

Arguments:

nIteration - A long value that specifies the sweep iteration.

Return:

Double.

The result's value as a double.

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
Dim Canvas1_Component1
Set Canvas1_Component1 = Canvas1.CreateComponent("CW
Laser","{6DA31CEE-058F-11D4-93BD-0050DAB7C5D6}",230,90, 34, 34,0)
Dim Canvas1_Component2
Set Canvas1_Component2 = Canvas1.CreateComponent("Optical Power
Meter Visualizer","{F11D0C25-3C7D-11D4-93F0-0050DAB7C5D6}",330,210,
34, 38,0)
```

```
Canvas1_Component1.GetOutputPort(1).CreateMonitor
Canvas1_Component1.GetOutputPort(1).ConnectVisualizer(Canvas1_Comp
onent2.GetInputPort(1))
Document.CalculateProject TRUE, TRUE
Dim resMgr
Set resMgr = Canvas1_Component2.GetResultMgr
Dim res
Set res = resMgr.GetObjectByName("Noise Power (dBm)")
Dim value_
value_ = res.GetValue(1)
MsgBox(value_)
```

CSDKExtObject functionalities

`get_Name()`

Purpose: This method returns the name of this object.

Return:

String.

The name of the object as a string.

`put_Name``( BSTR pName)`

Purpose: This method sets the name of this object.

Arguments:

pName - A null terminating string that specifies the name of the component.

Return:

None.

`GetIteration()`

Purpose: This method returns the current iteration in the document at this time.

Return:

Long.

The current sweep iteration as a long.

`GetMaxIterations()`

Purpose: This method returns the maximum number of iterations the document supports at this time.

Return:

Long.

The max number of sweep iterations as a long.

`GetSID()`

Purpose: This method returns the system ID (SID) of this object.

Remarks: This value uniquely identifies this object in the document.

Return:

Unsigned long.

The SID of the object as an unsigned long.

CSDKPort

`GetSignalType()`

Purpose: Gets the type of the signal that passes through this port.

Return:

String.

The type of the signal that passes through this port as a string.

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
Dim Canvas1_Component1
Set Canvas1_Component1 = Canvas1.CreateComponent("CW
Laser","{6DA31CEE-058F-11D4-93BD-0050DAB7C5D6}",180,210, 34, 34,0)
Canvas1_Component1.GetOutputPort(1).CreateMonitor
Dim sSigType
sSigType = Canvas1_Component1.GetOutputPort(1).GetSignalType
MsgBox(sSigType)
```

`Handshake``( IDispatch* pDispPort)`

Purpose: This method tests whether two ports have compatible signal types.

Arguments:

pDispPort - The tested port object .

Return:

BOOL.

A BOOL which indicates whether the test is positive or not. The value is TRUE if both ports have compatible signal types and FALSE otherwise.

`get_Position()`

Purpose: This method gets the position (along the edge) where this port resides.

Return:

Double.

A double value that indicates the position. The value of the position should be greater then or equal to zero. For a port on a vertical edge, the value of the position should be less then or equal to the height of the component this port belongs to. For a port on a horizontal edge, the value of the position should be less then or equal to the width of the component this port belongs to. Consider a component that is bounded by a rectangle. The ports are placed along the edges of the rectangle. The value of the position increases from left to right and from top to bottom along the edge. The topmost value is 0, and the leftmost value is zero. The caller is responsible allocating/deallocating this argument.

put_Position(double dPosition)

Purpose: Sets the position (along the edge) where this port resides.

Arguments:

dPosition - A double value that indicates the position of the port. The value of the position should be greater then or equal to zero. For a port on a vertical edge, the value of the position should be less then or equal to the height of the component this port belongs to. For a port on a horizontal edge, the value of the position should be less then or equal to the width of the component this port belongs to. Consider a component that is bounded by a rectangle. The ports are placed along the edges of the rectangle. The value of the position increases from left to right and from top to bottom along the edge. The topmost value is 0, and the leftmost value is zero.

Return:

None.

get_Edge()

Purpose: This method gets the edge on which this port resides.

Return:

Long.

pEdge - A long value that indicates the edge. Valid values for this argument are:

- 0 - Left edge
- 1 - Top edge
- 2 - Right edge
- 3 - Bottom edge

put_Edge(long nEdge)

Purpose: This method sets the edge on which this port resides.

Arguments:

pEdge - A long value that indicates the edge on which the port resides. Valid values for this argument are:

- 0 - Left edge
- 1 - Top edge
- 2 - Right edge
- 3 - Bottom edge

Return:

None.

GetConnection()

Purpose: This method gets the port that is connected to this port.

Return:

The port that is connected to this port.

Connect(IDispatch* pDispPort)

Purpose: This method connects the specified port to this port.

Arguments:

pDispPort - A port object to connect to this port.

Return:

None

`Disconnect()`

Purpose: This method disconnects the connected port from this port.

Return:

None

`IsConnected()`

Purpose: This method determines whether a port is connected to this port.

Return:

BOOL.

A BOOL which indicates whether the port is connected. The value is TRUE if the port is connected and FALSE otherwise.

CSDKExtObject functionalities

`get_Name()`

Purpose: This method returns the name of this object.

Return:

String.

The name of the object as a string.

`put_Name``( BSTR pName)`

Purpose: This method sets the name of this object.

Arguments:

pName - A null terminating string that specifies the name of the component.

Return:

None.

`GetIteration()`

Purpose: This method returns the current iteration in the document at this time.

Return:

Long.

The current sweep iteration as a long.

`GetMaxIterations()`

Purpose: This method returns the maximum number of iterations the document supports at this time.

Return:

Long.

The max number of sweep iterations as a long.

`GetSID()`

Purpose: This method returns the system ID (SID) of this object.

Remarks: This value uniquely identifies this object in the document.

Return:

Unsigned long.

The SID of the object as an unsigned long.

CSDKInputPort

GetSignalType()

Purpose: Gets the type of the signal that passes through this port.

Return:

String.

A string that holds a guid of the signal. Each signal has a unique guid.

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Layout1.Name  = "Layout 1"
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
Dim Canvas1_Component5
Set Canvas1_Component5 = Canvas1.CreateComponent("EDFA","{255EDC8F-
37E4-11D4-93EC-0050DAB7C5D6}",300,270, 32, 32,0)
Canvas1_Component5.Name  = "EDFA"
Dim Canvas1_Component6
Set Canvas1_Component6 = Canvas1.CreateComponent("CW
Laser","{6DA31CEE-058F-11D4-93BD-0050DAB7C5D6}",150,180, 34, 34,0)
Canvas1_Component6.Name  = "CW Laser"
Canvas1_Component6.GetOutputPort(1).Connect(Canvas1_Component5.Get
InputPort(1))
Dim sSigType
sSigType = Canvas1_Component5.GetInputPort(1).GetSignalType
MsgBox(sSigType)
```

Handshake(IDispatch* pDispPort)

Purpose: This method tests whether two ports have compatible signal types.

Arguments:

pDispPort - The tested port object .

Return:

BOOL.

A BOOL which indicates whether the test is positive or not. The value is TRUE if both ports have compatible signal types and FALSE otherwise.

get_Position();

Purpose: This method gets the position (along the edge) where this port resides.

Return:

Double.

A double value that indicates the position. The value of the position should be greater then or equal to zero. For a port on a vertical edge, the value of the position should be less then or equal to the height of the component this port belongs to. For a port on a horizontal edge, the value of the position should be less then or equal to the width

of the component this port belongs to. Consider a component that is bounded by a rectangle. The ports are placed along the edges of the rectangle. The value of the position increases from left to right and from top to bottom along the edge. The topmost value is 0, and the leftmost value is zero. The caller is responsible allocating/deallocating this argument.

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Layout1.Name  = "Layout 1"
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
Dim Canvas1_Component5
Set Canvas1_Component5 = Canvas1.CreateComponent("EDFA","{255EDC8F-
37E4-11D4-93EC-0050DAB7C5D6}",300,270, 32, 32,0)
Canvas1_Component5.Name  = "EDFA"
Dim Canvas1_Component6
Set Canvas1_Component6 = Canvas1.CreateComponent("CW
Laser","{6DA31CEE-058F-11D4-93BD-0050DAB7C5D6}",150,180, 34, 34,0)
Canvas1_Component6.Name  = "CW Laser"
Canvas1_Component6.GetOutputPort(1).Connect(Canvas1_Component5.Get
InputPort(1))
Dim dPosition
dPosition = Canvas1_Component5.GetInputPort(1).Position
MsgBox(dPosition)
```

`put_Position( double dPosition)`

Purpose: Sets the position (along the edge) where this port resides.

Arguments:

dPosition - A double value that indicates the position of the port. The value of the position should be greater then or equal to zero. For a port on a vertical edge, the value of the position should be less then or equal to the height of the component this port belongs to. For a port on a horizontal edge, the value of the position should be less then or equal to the width of the component this port belongs to. Consider a component that is bounded by a rectangle. The ports are placed along the edges of the rectangle. The value of the position increases from left to right and from top to bottom along the edge. The topmost value is 0, and the leftmost value is zero.

Return:

None.

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Layout1.Name  = "Layout 1"
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
Dim Canvas1_Component5
```

```
Set Canvas1_Component5 = Canvas1.CreateComponent("Subsystem
1.0","{C83C8C01-53FD-11D4-9407-0050DAB7C5D6}",220,290, 162, 120,0)
Canvas1_Component5.Name  = "Subsystem"
Canvas1_Component5.AddPort "Output", 3, 2, 0.414286
Canvas1_Component5.AddPort "Input", 2, 0, 0.507143
Canvas1_Component5.GetInputPort(1).Position = 0.7
Dim dPosition
dPosition = Canvas1_Component5.GetInputPort(1).Position
MsgBox(dPosition)
```

`get_Edge();`

Purpose: This method gets the edge on which this port resides.

Return:

Long.

pEdge - A long value that indicates the edge. Valid values for this argument are:

- 0 - Left edge
- 1 - Top edge
- 2 - Right edge
- 3 - Bottom edge

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Layout1.Name  = "Layout 1"
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
Dim Canvas1_Component5
Set Canvas1_Component5 = Canvas1.CreateComponent("Subsystem
1.0","{C83C8C01-53FD-11D4-9407-0050DAB7C5D6}",220,290, 162, 120,0)
Canvas1_Component5.Name  = "Subsystem"
Canvas1_Component5.AddPort "Output", 3, 2, 0.414286
Canvas1_Component5.AddPort "Input", 2, 0, 0.507143
Canvas1_Component5.GetInputPort(1).Position = 0.7
Canvas1_Component5.GetInputPort(1).Edge = 1
Dim nEdge
nEdge = Canvas1_Component5.GetInputPort(1).Edge
MsgBox(nEdge)
```

`put_Edge``( long nEdge)`

Purpose: This method sets the edge on which this port resides.

Arguments:

pEdge - A long value that indicates the edge on which the port resides. Valid values for this argument are:

- 0 - Left edge
- 1 - Top edge

- 2 - Right edge
- 3 - Bottom edge

Return:

None.

Example:

```
See get_Edge()
```

`GetConnection()`

Purpose: This method gets the port that is connected to this port.

Return:

The port that is connected to this port.

`Connect``( IDispatch* pDispPort)`

Purpose: This method connects the specified port to this port.

Arguments:

pDispPort - A port object to connect to this port.

Return:

None

`Disconnect()`

Purpose: This method disconnects the connected port from this port.

Return:

None

`IsConnected()`

Purpose: This method determines whether a port is connected to this port.

Return:

BOOL.

A BOOL which indicates whether the port is connected. The value is TRUE if the port is connected and FALSE otherwise.

CSDKOutputPort

RemoveMonitor()

Purpose: This method removes a monitor from the output port.

Return:

None.

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
Dim Canvas1_Component1
Set Canvas1_Component1 = Canvas1.CreateComponent("CW
Laser","{6DA31CEE-058F-11D4-93BD-0050DAB7C5D6}",180,210, 34, 34,0)
Canvas1_Component1.GetOutputPort(1).CreateMonitor
Canvas1_Component1.GetOutputPort(1).RemoveMonitor
```

DisconnectVisualizer(IDispatch* pVisualizerPort)

Purpose: This method disconnects a visualizer from the output port.

Arguments:

pVisualizerPort - A Visualizer port object.

Return:

None.

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
Dim Canvas1_Component1
Set Canvas1_Component1 = Canvas1.CreateComponent("CW
Laser","{6DA31CEE-058F-11D4-93BD-0050DAB7C5D6}",180,210, 34, 34,0)
Dim Canvas1_Component2
Set Canvas1_Component2 = Canvas1.CreateComponent("Optical Spectrum
Analyzer","{F11D0C25-3C7D-11D4-93F0-0050DAB7C5D6}",270,230, 40,
34,0)
Canvas1_Component2.Name  = "Optical Spectrum Analyzer"
Canvas1_Component1.GetOutputPort(1).CreateMonitor
Canvas1_Component1.GetOutputPort(1).ConnectVisualizer(Canvas1_Comp
onent2.GetInputPort(1))
Canvas1_Component1.GetOutputPort(1).DisconnectVisualizer(Canvas1_C
omponent2.GetInputPort(1))
```

ConnectVisualizer(IDispatch* pVisualizerPort)

Purpose: This method connects a visualizer to the output port.

Arguments:

pVisualizerPort - A Visualizer port object.

Return:

None.

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
Dim Canvas1_Component1
Set Canvas1_Component1 = Canvas1.CreateComponent("CW
Laser","{6DA31CEE-058F-11D4-93BD-0050DAB7C5D6}",180,210, 34, 34,0)
Dim Canvas1_Component2
Set Canvas1_Component2 = Canvas1.CreateComponent("Optical Spectrum
Analyzer","{F11D0C25-3C7D-11D4-93F0-0050DAB7C5D6}",270,230, 40,
34,0)
Canvas1_Component2.Name  = "Optical Spectrum Analyzer"
Canvas1_Component1.GetOutputPort(1).CreateMonitor
Canvas1_Component1.GetOutputPort(1).ConnectVisualizer(Canvas1_Comp
onent2.GetInputPort(1))
```

`IsAnyVisualizerConnected()`

Purpose: This method indicates whether any visualisers are connected to the output port.

Return:

BOOL.

TRUE if any visualisers are connected to the output port, FALSE otherwise.

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
Dim Canvas1_Component1
Set Canvas1_Component1 = Canvas1.CreateComponent("CW
Laser","{6DA31CEE-058F-11D4-93BD-0050DAB7C5D6}",180,210, 34, 34,0)
Dim Canvas1_Component2
Set Canvas1_Component2 = Canvas1.CreateComponent("Optical Spectrum
Analyzer","{F11D0C25-3C7D-11D4-93F0-0050DAB7C5D6}",270,230, 40,
34,0)
Canvas1_Component2.Name  = "Optical Spectrum Analyzer"
Canvas1_Component1.GetOutputPort(1).CreateMonitor
Canvas1_Component1.GetOutputPort(1).ConnectVisualizer(Canvas1_Comp
onent2.GetInputPort(1))
Dim bV
```

```
bV = Canvas1_Component1.GetOutputPort(1).IsAnyVisualizerConnected
MsgBox(bV)
```

`CreateMonitor()`

Purpose: This method attaches a monitor to the output port.

Return:

None.

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
Dim Canvas1_Component1
Set Canvas1_Component1 = Canvas1.CreateComponent("CW
Laser","{6DA31CEE-058F-11D4-93BD-0050DAB7C5D6}",180,210, 34, 34,0)
Canvas1_Component1.GetOutputPort(1).CreateMonitor
```

`HasMonitor()`

Purpose: This method gets a value indicating whether the output port has a monitor attached.

Return:

BOOL

TRUE if the output port has a monitor attached to it, FALSE otherwise.

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
Dim Canvas1_Component1
Set Canvas1_Component1 = Canvas1.CreateComponent("CW
Laser","{6DA31CEE-058F-11D4-93BD-0050DAB7C5D6}",180,210, 34, 34,0)
Canvas1_Component1.GetOutputPort(1).CreateMonitor
Dim bMonitor
bMonitor =  Canvas1_Component1.GetOutputPort(1).HasMonitor
MsgBox(bMonitor)
```


CSDKPort functionalities

`GetSignalType()`

Purpose: Gets the type of the signal that passes through this port.

Return:

String.

The type of the signal that passes through this port as a string.

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Layout1.Name  = "Layout 1"
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
Dim Canvas1_Component5
Set Canvas1_Component5 = Canvas1.CreateComponent("EDFA","{255EDC8F-
37E4-11D4-93EC-0050DAB7C5D6}",300,270, 32, 32,0)
Canvas1_Component5.Name  = "EDFA"
Dim Canvas1_Component6
Set Canvas1_Component6 = Canvas1.CreateComponent("CW
Laser","{6DA31CEE-058F-11D4-93BD-0050DAB7C5D6}",150,180, 34, 34,0)
Canvas1_Component6.Name  = "CW Laser"
Canvas1_Component6.GetOutputPort(1).Connect(Canvas1_Component5.Get
InputPort(1))
Dim sSigType
sSigType = Canvas1_Component5.GetInputPort(1).GetSignalType
MsgBox(sSigType)
```

`Handshake``( IDispatch* pDispPort)`

Purpose: This method tests whether two ports have compatible signal types.

Arguments:

pDispPort - The tested port object .

Return:

BOOL.

A BOOL which indicates whether the test is positive or not. The value is TRUE if both ports have compatible signal types and FALSE otherwise.

`get_Position();`

Purpose: This method gets the position (along the edge) where this port resides.

Return:

Double.

A double value that indicates the position. The value of the position should be greater then or equal to zero. For a port on a vertical edge, the value of the position should be less then or equal to the height of the component this port belongs to. For a port on a horizontal edge, the value of the position should be less then or equal to the width

of the component this port belongs to. Consider a component that is bounded by a rectangle. The ports are placed along the edges of the rectangle. The value of the position increases from left to right and from top to bottom along the edge. The topmost value is 0, and the leftmost value is zero. The caller is responsible for allocating/de-allocating this argument.

put_Position(double dPosition)

Purpose: Sets the position (along the edge) where this port resides.

Arguments:

dPosition - A double value indicating the position of the port. The value of the position should be greater then or equal to zero. For a port on a vertical edge, the value of the position should be less then or equal to the height of the component this port belongs to. For a port on a horizontal edge, the value of the position should be less then or equal to the width of the component this port belongs to. Consider a component that is bounded by a rectangle. The ports are placed along the edges of the rectangle. The value of the position increases from left to right and from top to bottom along the edge. The topmost value is 0, and the leftmost value is zero.

Return:

None.

get_Edge()

Purpose: This method gets the edge on which this port resides.

Return:

Long.

pEdge - A long value indicating the edge. Valid values for this argument are:

- 0 - Left edge
- 1 - Top edge
- 2 - Right edge
- 3 - Bottom edge

put_Edge(long nEdge)

Purpose: This method sets the edge on which this port resides.

Arguments:

pEdge - A long value that indicates the edge on which the port resides. Valid values for this argument are:

- 0 - Left edge
- 1 - Top edge
- 2 - Right edge
- 3 - Bottom edge

Return:

None.

GetConnection()

Purpose: This method gets the port that is connected to this port.

Return:

The port that is connected to this port.

Connect(IDispatch* pDispPort)

Purpose: This method connects the specified port to this port.

Arguments:

pDispPort - A port object to connect to this port.

Return:

None

Disconnect()

Purpose: This method disconnects the connected port from this port.

Return:

None

IsConnected()

Purpose: This method determines whether a port is connected to this port.

Return:

BOOL.

A BOOL which indicates whether the port is connected. The value is TRUE if the port is connected and FALSE otherwise.

CSDKExtObject functionalities

get_Name()

Purpose: This method returns the name of this object.

Return:

String.

The name of the object as a string.

put_Name(BSTR pName)

Purpose: This method sets the name of this object.

Arguments:

pName - A null terminating string that specifies the name of the component.

Return:

None.

GetIteration()

Purpose: This method returns the current iteration in the document at this time.

Return:

Long.

The current sweep iteration as a long.

`GetMaxIterations()`

Purpose: This method returns the maximum number of iterations the document supports at this time.

Return:

Long.

The max number of sweep iterations as a long.

`GetSID()`

Purpose: This method returns the system ID (SID) of this object.

Remarks: This value uniquely identifies this object in the document.

Return:

Unsigned long.

The SID of the object as an unsigned long.

CSDKVisualizer

`HideDialog()`

Purpose: This method hides the visualizer's dialog.

Return:

None.

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
Dim Canvas1_Component1
Set Canvas1_Component1 = Canvas1.CreateComponent("CW
Laser","{6DA31CEE-058F-11D4-93BD-0050DAB7C5D6}",130,110, 34, 34,0)
Canvas1_Component1.Name  = "CW Laser"
Dim Canvas1_Component2
Set Canvas1_Component2 = Canvas1.CreateComponent("Optical Spectrum
Analyzer","{F11D0C25-3C7D-11D4-93F0-0050DAB7C5D6}",250,210, 40,
34,0)
Canvas1_Component2.Name  = "Optical Spectrum Analyzer"
Canvas1_Component1.GetOutputPort(1).CreateMonitor
Canvas1_Component1.GetOutputPort(1).ConnectVisualizer(Canvas1_Comp
onent2.GetInputPort(1))
Document.CalculateProject TRUE, TRUE
Canvas1_Component2.ShowDialog
Canvas1_Component2.HideDialog
```

`ShowDialog()`

Purpose: This method shows the visualizer's dialog.

Return:

None.

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
Dim Canvas1_Component1
Set Canvas1_Component1 = Canvas1.CreateComponent("CW
Laser","{6DA31CEE-058F-11D4-93BD-0050DAB7C5D6}",130,110, 34, 34,0)
Canvas1_Component1.Name  = "CW Laser"
Dim Canvas1_Component2
Set Canvas1_Component2 = Canvas1.CreateComponent("Optical Spectrum
Analyzer","{F11D0C25-3C7D-11D4-93F0-0050DAB7C5D6}",250,210, 40,
34,0)
Canvas1_Component2.Name  = "Optical Spectrum Analyzer"
```

```
Canvas1_Component1.GetOutputPort(1).CreateMonitor
Canvas1_Component1.GetOutputPort(1).ConnectVisualizer(Canvas1_Comp
onent2.GetInputPort(1))
Document.CalculateProject TRUE, TRUE
Canvas1_Component2.ShowDialog
```

`Update()`

Purpose: This method updates the visualizer's dialog.

Return:

None.

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
Dim Canvas1_Component1
Set Canvas1_Component1 = Canvas1.CreateComponent("CW
Laser","{6DA31CEE-058F-11D4-93BD-0050DAB7C5D6}",130,110, 34, 34,0)
Canvas1_Component1.Name  = "CW Laser"
Dim Canvas1_Component2
Set Canvas1_Component2 = Canvas1.CreateComponent("Optical Spectrum
Analyzer","{F11D0C25-3C7D-11D4-93F0-0050DAB7C5D6}",250,210, 40,
34,0)
Canvas1_Component2.Name  = "Optical Spectrum Analyzer"
Canvas1_Component1.GetOutputPort(1).CreateMonitor
Canvas1_Component1.GetOutputPort(1).ConnectVisualizer(Canvas1_Comp
onent2.GetInputPort(1))
Document.CalculateProject TRUE, TRUE
Canvas1_Component2.Update
```


CSDK2DGraph

GetXDataAt(int nIndex, long nIteration)

Purpose: This method returns the X graph data for given index and sweep iteration.

Arguments:

nIndex - An integer value between 0 and (number of graph points - 1)

nIteration - A long value indicating the sweep iteration.

Return:

Double.

The X graph data for given index and sweep iteration.

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
Dim Canvas1_Component1
Set Canvas1_Component1 = Canvas1.CreateComponent("CW
Laser","{6DA31CEE-058F-11D4-93BD-0050DAB7C5D6}",130,110, 34, 34,0)
Canvas1_Component1.Name  = "CW Laser"
Dim Canvas1_Component2
Set Canvas1_Component2 = Canvas1.CreateComponent("Optical Spectrum
Analyzer","{F11D0C25-3C7D-11D4-93F0-0050DAB7C5D6}",250,210, 40,
34,0)
Canvas1_Component2.Name  = "Optical Spectrum Analyzer"
Canvas1_Component1.GetOutputPort(1).CreateMonitor
Canvas1_Component1.GetOutputPort(1).ConnectVisualizer(Canvas1_Comp
onent2.GetInputPort(1))
Document.CalculateProject TRUE, TRUE
Dim graph
Set graph = Canvas1_Component2.GetGraph("Sampled signal spectrum")
Dim nNumPoints
nNumPoints = graph.GetNrOfPoints
Dim nData
nData = graph.GetXDataAt( 0, 1 )
MsgBox(nData)
nData = graph.GetXDataAt( nNumPoints - 1, 1 )
MsgBox(nData)
```

GetYDataAt(int nIndex, long nIteration)

Purpose: This method returns the Y data for given index and sweep iteration.

Arguments:

nIndex - An integer value between 0 and (number of graph points - 1)

nIteration - A long value that indicates the sweep iteration.

Return:

Double.

The Y graph data for given index and sweep iteration.

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
Dim Canvas1_Component1
Set Canvas1_Component1 = Canvas1.CreateComponent("CW
Laser","{6DA31CEE-058F-11D4-93BD-0050DAB7C5D6}",130,110, 34, 34,0)
Canvas1_Component1.Name  = "CW Laser"
Dim Canvas1_Component2
Set Canvas1_Component2 = Canvas1.CreateComponent("Optical Spectrum
Analyzer","{F11D0C25-3C7D-11D4-93F0-0050DAB7C5D6}",250,210, 40,
34,0)
Canvas1_Component2.Name  = "Optical Spectrum Analyzer"
Canvas1_Component1.GetOutputPort(1).CreateMonitor
Canvas1_Component1.GetOutputPort(1).ConnectVisualizer(Canvas1_Comp
onent2.GetInputPort(1))
Document.CalculateProject TRUE, TRUE
Dim graph
Set graph = Canvas1_Component2.GetGraph("Sampled signal spectrum")
Dim nNumPoints
nNumPoints = graph.GetNrOfPoints
Dim nData
nData = graph.GetXDataAt( 0, 1 )
MsgBox(nData)
nData = graph.GetXDataAt( nNumPoints - 1, 1 )
MsgBox(nData)
```

`GetSize``(long nIter)`

Purpose: This method returns the size of the data array for the specified sweep iteration.

Arguments:

nIter - A long value that indicates the sweep iteration.

Return:

Long.

The size of the data array as a long.

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
Dim Canvas1_Component1
```


```
Set Canvas1_Component1 = Canvas1.CreateComponent("CW
Laser","{6DA31CEE-058F-11D4-93BD-0050DAB7C5D6}",130,110, 34, 34,0)
Canvas1_Component1.Name  = "CW Laser"
Dim Canvas1_Component2
Set Canvas1_Component2 = Canvas1.CreateComponent("Optical Spectrum
Analyzer","{F11D0C25-3C7D-11D4-93F0-0050DAB7C5D6}",250,210, 40,
34,0)
Canvas1_Component2.Name  = "Optical Spectrum Analyzer"
Canvas1_Component1.GetOutputPort(1).CreateMonitor
Canvas1_Component1.GetOutputPort(1).ConnectVisualizer(Canvas1_Comp
onent2.GetInputPort(1))
Document.CalculateProject TRUE, TRUE
Dim graph
Set graph = Canvas1_Component2.GetGraph("Sampled signal spectrum")
Dim nSize
nSize = graph.GetSize( 1 )
MsgBox(nSize)
```

GetYData(long nIter)

Purpose: This method returns the Y data array for the specified sweep iteration.

Arguments:

nIter - A long value that indicates the sweep iteration.

Return:

Array of doubles.

The Y graph data as an array of doubles.

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
Dim Canvas1_Component1
Set Canvas1_Component1 = Canvas1.CreateComponent("CW
Laser","{6DA31CEE-058F-11D4-93BD-0050DAB7C5D6}",130,110, 34, 34,0)
Canvas1_Component1.Name  = "CW Laser"
Dim Canvas1_Component2
Set Canvas1_Component2 = Canvas1.CreateComponent("Optical Spectrum
Analyzer","{F11D0C25-3C7D-11D4-93F0-0050DAB7C5D6}",250,210, 40,
34,0)
Canvas1_Component2.Name  = "Optical Spectrum Analyzer"
Canvas1_Component1.GetOutputPort(1).CreateMonitor
Canvas1_Component1.GetOutputPort(1).ConnectVisualizer(Canvas1_Comp
onent2.GetInputPort(1))
Document.CalculateProject TRUE, TRUE
Dim graph
Set graph = Canvas1_Component2.GetGraph("Sampled signal spectrum")
Dim arrYData
arrYData = graph.GetYData( 1 )
```

```
Dim bA
bA = IsArray(arrYData)
MsgBox(bA)
Dim nUBound
nUBound = UBound(arrYData)
MsgBox(nUBound)
```

GetXData(long nIter)

Purpose: This method returns the X data array for the specified sweep iteration.

Arguments:

nIter - A long value that indicates the sweep iteration.

Return:

Array of doubles.

The X graph data as an array of doubles.

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
Dim Canvas1_Component1
Set Canvas1_Component1 = Canvas1.CreateComponent("CW
Laser","{6DA31CEE-058F-11D4-93BD-0050DAB7C5D6}",130,110, 34, 34,0)
Canvas1_Component1.Name  = "CW Laser"
Dim Canvas1_Component2
Set Canvas1_Component2 = Canvas1.CreateComponent("Optical Spectrum
Analyzer","{F11D0C25-3C7D-11D4-93F0-0050DAB7C5D6}",250,210, 40,
34,0)
Canvas1_Component2.Name  = "Optical Spectrum Analyzer"
Canvas1_Component1.GetOutputPort(1).CreateMonitor
Canvas1_Component1.GetOutputPort(1).ConnectVisualizer(Canvas1_Comp
onent2.GetInputPort(1))
Document.CalculateProject TRUE, TRUE
Dim graph
Set graph = Canvas1_Component2.GetGraph("Sampled signal spectrum")
Dim arrXData
arrXData = graph.GetXData( 1 )
Dim bA
bA = IsArray(arrXData)
MsgBox(bA)
Dim nUBound
nUBound = UBound( arrXData)
MsgBox(nUBound)
```

GetNrOfPoints()

Purpose: This method returns the graph's number of points.

Return:

Long.

The number of points in the graph as a long.

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
Dim Canvas1_Component1
Set Canvas1_Component1 = Canvas1.CreateComponent("CW
Laser","{6DA31CEE-058F-11D4-93BD-0050DAB7C5D6}",130,110, 34, 34,0)
Canvas1_Component1.Name  = "CW Laser"
Dim Canvas1_Component2
Set Canvas1_Component2 = Canvas1.CreateComponent("Optical Spectrum
Analyzer","{F11D0C25-3C7D-11D4-93F0-0050DAB7C5D6}",250,210, 40,
34,0)
Canvas1_Component2.Name  = "Optical Spectrum Analyzer"
Canvas1_Component1.GetOutputPort(1).CreateMonitor
Canvas1_Component1.GetOutputPort(1).ConnectVisualizer(Canvas1_Comp
onent2.GetInputPort(1))
Document.CalculateProject TRUE, TRUE
Dim graph
Set graph = Canvas1_Component2.GetGraph("Sampled signal spectrum")
Dim nNumPoints
nNumPoints = graph.GetNrOfPoints
MsgBox(nNumPoints)
```

`get_TitleY()`

Purpose: This method returns the title of Y coordinates.

Return.

String.

The title of Y coordinates as a string.

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
Dim Canvas1_Component1
Set Canvas1_Component1 = Canvas1.CreateComponent("CW
Laser","{6DA31CEE-058F-11D4-93BD-0050DAB7C5D6}",130,110, 34, 34,0)
Canvas1_Component1.Name  = "CW Laser"
Dim Canvas1_Component2
Set Canvas1_Component2 = Canvas1.CreateComponent("Optical Spectrum
Analyzer","{F11D0C25-3C7D-11D4-93F0-0050DAB7C5D6}",250,210, 40,
34,0)
Canvas1_Component2.Name  = "Optical Spectrum Analyzer"
```

```
Canvas1_Component1.GetOutputPort(1).CreateMonitor
Canvas1_Component1.GetOutputPort(1).ConnectVisualizer(Canvas1_Comp
onent2.GetInputPort(1))
Document.CalculateProject TRUE, TRUE
Dim graph
Set graph = Canvas1_Component2.GetGraph("Sampled signal spectrum")
Dim sTitle
sTitle = graph.TitleY
MsgBox(sTitle)
```

get_TitleX()

Purpose: This method returns the title of X coordinates.

Return.

String.

The title of X coordinates as a string.

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
Dim Canvas1_Component1
Set Canvas1_Component1 = Canvas1.CreateComponent("CW
Laser","{6DA31CEE-058F-11D4-93BD-0050DAB7C5D6}",130,110, 34, 34,0)
Canvas1_Component1.Name  = "CW Laser"
Dim Canvas1_Component2
Set Canvas1_Component2 = Canvas1.CreateComponent("Optical Spectrum
Analyzer","{F11D0C25-3C7D-11D4-93F0-0050DAB7C5D6}",250,210, 40,
34,0)
Canvas1_Component2.Name  = "Optical Spectrum Analyzer"
Canvas1_Component1.GetOutputPort(1).CreateMonitor
Canvas1_Component1.GetOutputPort(1).ConnectVisualizer(Canvas1_Comp
onent2.GetInputPort(1))
Document.CalculateProject TRUE, TRUE
Dim graph
Set graph = Canvas1_Component2.GetGraph("Sampled signal spectrum")
Dim sTitle
sTitle = graph.TitleX
MsgBox(sTitle)
```

GetUnitX()

Purpose: This method returns the units of X coordinates.

Return.

String.

The units of X coordinate as a string.

Example:

```
Dim Lm
```


```
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
Dim Canvas1_Component1
Set Canvas1_Component1 = Canvas1.CreateComponent("CW
Laser","{6DA31CEE-058F-11D4-93BD-0050DAB7C5D6}",130,110, 34, 34,0)
Canvas1_Component1.Name  = "CW Laser"
Dim Canvas1_Component2
Set Canvas1_Component2 = Canvas1.CreateComponent("Optical Spectrum
Analyzer","{F11D0C25-3C7D-11D4-93F0-0050DAB7C5D6}",250,210, 40,
34,0)
Canvas1_Component2.Name  = "Optical Spectrum Analyzer"
Canvas1_Component1.GetOutputPort(1).CreateMonitor
Canvas1_Component1.GetOutputPort(1).ConnectVisualizer(Canvas1_Comp
onent2.GetInputPort(1))
Document.CalculateProject TRUE, TRUE
Dim graph
Set graph = Canvas1_Component2.GetGraph("Sampled signal spectrum")
Dim sUnit
sUnit = graph.GetUnitX
MsgBox(sUnit)
```

`GetUnitY()`

Purpose: This method returns the units of Y coordinates.

Return.

String.

The units of Y coordinate as a string.

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
Dim Canvas1_Component1
Set Canvas1_Component1 = Canvas1.CreateComponent("CW
Laser","{6DA31CEE-058F-11D4-93BD-0050DAB7C5D6}",130,110, 34, 34,0)
Canvas1_Component1.Name  = "CW Laser"
Dim Canvas1_Component2
Set Canvas1_Component2 = Canvas1.CreateComponent("Optical Spectrum
Analyzer","{F11D0C25-3C7D-11D4-93F0-0050DAB7C5D6}",250,210, 40,
34,0)
Canvas1_Component2.Name  = "Optical Spectrum Analyzer"
Canvas1_Component1.GetOutputPort(1).CreateMonitor
Canvas1_Component1.GetOutputPort(1).ConnectVisualizer(Canvas1_Comp
onent2.GetInputPort(1))
Document.CalculateProject TRUE, TRUE
Dim graph
```

```
Set graph = Canvas1_Component2.GetGraph("Sampled signal spectrum")
Dim sUnit
sUnit = graph.GetUnitY
MsgBox(sUnit)
```

CSDK3DGraph

Base Example:

```
'Get Layout Manager.
Dim Lm
Set Lm = Document.GetLayoutMgr
'SCRIPT for Layout 1
'Get Current Layout.
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Layout1.Name  = "Layout 1"
 'Set Total Sweep Iterations
Layout1.SetTotalSweepIterations(3)
 'Set Current Sweep Iteration
Layout1.SetCurrentSweepIteration(3)
'Get Current Canvas.
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
'SCRIPT for Layout global parameters.
Layout1.SetParameterMode "Simulation window", 0
Layout1.SetParameterValue "Simulation window", "Set bit rate"
Layout1.SetParameterMode "Reference bit rate", 0
Layout1.SetParameterValue "Reference bit rate", TRUE
Layout1.SetParameterMode "Bit rate", 0
Layout1.SetParameterValue "Bit rate", 4e+010
Layout1.SetParameterMode "Time window", 0
Layout1.SetParameterValue "Time window", 2e-010
Layout1.SetParameterMode "Sample rate", 0
Layout1.SetParameterValue "Sample rate", 5.12e+012
Layout1.SetParameterMode "Sequence length", 0
Layout1.SetParameterValue "Sequence length", 8
Layout1.SetParameterMode "Samples per bit", 0
Layout1.SetParameterValue "Samples per bit", 128
Layout1.SetParameterMode "Number of samples", 0
Layout1.SetParameterValue "Number of samples", 1024
Layout1.SetParameterMode "Iterations", 0
Layout1.SetParameterValue "Iterations", 1
Layout1.SetParameterMode "Parameterized", 0
Layout1.SetParameterValue "Parameterized", FALSE
Layout1.SetParameterMode "Convert noise bins", 0
Layout1.SetParameterValue "Convert noise bins", FALSE
Layout1.SetParameterMode "Calculate signal tracing", 0
Layout1.SetParameterValue "Calculate signal tracing", TRUE
Layout1.SetParameterMode "Power unit", 0
Layout1.SetParameterValue "Power unit", "dBm"
Layout1.SetParameterMode "Frequency unit", 0
Layout1.SetParameterValue "Frequency unit", "THz"
Layout1.SetParameterMode "Decimal places", 0
Layout1.SetParameterValue "Decimal places", 4
Layout1.SetParameterMode "Sensitivity", 0
Layout1.SetParameterValue "Sensitivity", -100
```

```
Layout1.SetParameterMode "Resolution", 0
Layout1.SetParameterValue "Resolution", 0.1
Layout1.SetParameterMode "Calculate noise floor", 0
Layout1.SetParameterValue "Calculate noise floor", FALSE
Layout1.SetParameterMode "Interpolation offset", 0
Layout1.SetParameterValue "Interpolation offset", 0.5
'SCRIPT for each component in the Layout.
'SCRIPT for component Optical Fiber.
Dim Canvas1_Component2
Set Canvas1_Component2 = Canvas1.CreateComponent("Optical
Fiber","{416EC6F1-529F-11D4-9403-0050DAB7C5D6}",350,170, 32, 32,0)
Canvas1_Component2.Name  = "Optical Fiber"
'Set Optical Fiber parameters.
Canvas1_Component2.SetParameterMode "User defined reference
wavelength", 0
Canvas1_Component2.SetParameterValue "User defined reference
wavelength", FALSE
Canvas1_Component2.SetParameterMode "Reference wavelength", 0
Canvas1_Component2.SetParameterValue "Reference wavelength", 1550
Canvas1_Component2.SetParameterMode "Length", 0
Canvas1_Component2.SetParameterValue "Length", 3.9482
Canvas1_Component2.SetParameterMode "Attenuation effect", 0
Canvas1_Component2.SetParameterValue "Attenuation effect", FALSE
Canvas1_Component2.SetParameterMode "Attenuation data type", 0
Canvas1_Component2.SetParameterValue "Attenuation data type",
"Constant"
Canvas1_Component2.SetParameterMode "Attenuation", 0
Canvas1_Component2.SetParameterValue "Attenuation", 0.2
Canvas1_Component2.SetParameterMode "Attenuation vs. wavelength", 0
Canvas1_Component2.SetParameterValue "Attenuation vs. wavelength",
"Attenuation.dat"
Canvas1_Component2.SetParameterMode "Group velocity dispersion", 0
Canvas1_Component2.SetParameterValue "Group velocity dispersion",
TRUE
Canvas1_Component2.SetParameterMode "Third-order dispersion", 0
Canvas1_Component2.SetParameterValue "Third-order dispersion",
FALSE
Canvas1_Component2.SetParameterMode "Dispersion data type", 0
Canvas1_Component2.SetParameterValue "Dispersion data type",
"Constant"
Canvas1_Component2.SetParameterMode "Frequency domain parameters",
0
Canvas1_Component2.SetParameterValue "Frequency domain parameters",
TRUE
Canvas1_Component2.SetParameterMode "Dispersion", 0
Canvas1_Component2.SetParameterValue "Dispersion", 16.75
Canvas1_Component2.SetParameterMode "Dispersion slope", 0
Canvas1_Component2.SetParameterValue "Dispersion slope", 0.075
Canvas1_Component2.SetParameterMode "Beta 2", 0
Canvas1_Component2.SetParameterValue "Beta 2", -20
Canvas1_Component2.SetParameterMode "Beta 3", 0
Canvas1_Component2.SetParameterValue "Beta 3", 0
```


```
Canvas1_Component2.SetParameterMode "Dispersion file format", 0
Canvas1_Component2.SetParameterValue "Dispersion file format",
"Dispersion vs. wavelength"
Canvas1_Component2.SetParameterMode "Dispersion file name", 0
Canvas1_Component2.SetParameterValue "Dispersion file name",
"Dispersion.dat"
Canvas1_Component2.SetParameterMode "Birefringence type", 0
Canvas1_Component2.SetParameterValue "Birefringence type",
"Deterministic"
Canvas1_Component2.SetParameterMode "Differential group delay", 0
Canvas1_Component2.SetParameterValue "Differential group delay", 0
Canvas1_Component2.SetParameterMode "PMD coefficient", 0
Canvas1_Component2.SetParameterValue "PMD coefficient", 0.5
Canvas1_Component2.SetParameterMode "Mean scattering section
length", 0
Canvas1_Component2.SetParameterValue "Mean scattering section
length", 2000
Canvas1_Component2.SetParameterMode "Scattering section
dispersion", 0
Canvas1_Component2.SetParameterValue "Scattering section
dispersion", 400
Canvas1_Component2.SetParameterMode "Self-phase modulation", 0
Canvas1_Component2.SetParameterValue "Self-phase modulation", TRUE
Canvas1_Component2.SetParameterMode "Effective area data type", 0
Canvas1_Component2.SetParameterValue "Effective area data type",
"Constant"
Canvas1_Component2.SetParameterMode "Effective area", 0
Canvas1_Component2.SetParameterValue "Effective area", 80
Canvas1_Component2.SetParameterMode "Effective area vs.
wavelength", 0
Canvas1_Component2.SetParameterValue "Effective area vs.
wavelength", "EffectiveAra.dat"
Canvas1_Component2.SetParameterMode "n2 data type", 0
Canvas1_Component2.SetParameterValue "n2 data type", "Constant"
Canvas1_Component2.SetParameterMode "n2", 0
Canvas1_Component2.SetParameterValue "n2", 2.6e-020
Canvas1_Component2.SetParameterMode "n2 vs. wavelength", 0
Canvas1_Component2.SetParameterValue "n2 vs. wavelength", "n2.dat"
Canvas1_Component2.SetParameterMode "Self-steepening", 0
Canvas1_Component2.SetParameterValue "Self-steepening", FALSE
Canvas1_Component2.SetParameterMode "Full Raman Response", 0
Canvas1_Component2.SetParameterValue "Full Raman Response", FALSE
Canvas1_Component2.SetParameterMode "Intrapulse Raman Scatt.", 0
Canvas1_Component2.SetParameterValue "Intrapulse Raman Scatt.",
FALSE
Canvas1_Component2.SetParameterMode "Raman self-shift time1", 0
Canvas1_Component2.SetParameterValue "Raman self-shift time1", 14.2
Canvas1_Component2.SetParameterMode "Raman self-shift time2", 0
Canvas1_Component2.SetParameterValue "Raman self-shift time2", 3
Canvas1_Component2.SetParameterMode "Fract. Raman contribution", 0
Canvas1_Component2.SetParameterValue "Fract. Raman contribution",
0.18
```

```
Canvas1_Component2.SetParameterMode "Orthogonal Raman factor", 0
Canvas1_Component2.SetParameterValue "Orthogonal Raman factor",
0.75
Canvas1_Component2.SetParameterMode "Model type", 0
Canvas1_Component2.SetParameterValue "Model type", "Scalar"
Canvas1_Component2.SetParameterMode "Propagator type", 0
Canvas1_Component2.SetParameterValue "Propagator type",
"Exponential"
Canvas1_Component2.SetParameterMode "Calculation type", 0
Canvas1_Component2.SetParameterValue "Calculation type",
"Noniterative"
Canvas1_Component2.SetParameterMode "Number of iterations", 0
Canvas1_Component2.SetParameterValue "Number of iterations", 2
Canvas1_Component2.SetParameterMode "Step size", 2
Canvas1_Component2.SetSweepParameterValue "Step size",  1,
"Constant"
Canvas1_Component2.SetSweepParameterValue "Step size",  2,
"Variable"
Canvas1_Component2.SetSweepParameterValue "Step size",  3,
"Variable"
Canvas1_Component2.SetParameterMode "Max. nonlinear phase shift", 2
Canvas1_Component2.SetSweepParameterValue "Max. nonlinear phase
shift",1,  20
Canvas1_Component2.SetSweepParameterValue "Max. nonlinear phase
shift",2,  20
Canvas1_Component2.SetSweepParameterValue "Max. nonlinear phase
shift",3,  10
Canvas1_Component2.SetParameterMode "Boundary conditions", 0
Canvas1_Component2.SetParameterValue "Boundary conditions",
"Periodic"
Canvas1_Component2.SetParameterMode "Filter steepness", 0
Canvas1_Component2.SetParameterValue "Filter steepness", 0.005
Canvas1_Component2.SetParameterMode "Lower calculation limit", 0
Canvas1_Component2.SetParameterValue "Lower calculation limit",
1000
Canvas1_Component2.SetParameterMode "Upper calculation limit", 0
Canvas1_Component2.SetParameterValue "Upper calculation limit",
2000
Canvas1_Component2.SetParameterMode "Calculate graphs", 0
Canvas1_Component2.SetParameterValue "Calculate graphs", TRUE
Canvas1_Component2.SetParameterMode "Number of distance steps", 0
Canvas1_Component2.SetParameterValue "Number of distance steps", 100
Canvas1_Component2.SetParameterMode "Number of wavelength/time
steps", 0
Canvas1_Component2.SetParameterValue "Number of wavelength/time
steps", 1024
Canvas1_Component2.SetParameterMode "Linear scale", 0
Canvas1_Component2.SetParameterValue "Linear scale", TRUE
Canvas1_Component2.SetParameterMode "Minimum value", 0
Canvas1_Component2.SetParameterValue "Minimum value", -100
Canvas1_Component2.SetParameterMode "Spectrum (total power) graph",
0
```


```
Canvas1_Component2.SetParameterValue "Spectrum (total power)
graph", FALSE
Canvas1_Component2.SetParameterMode "Spectrum (X component) graph",
0
Canvas1_Component2.SetParameterValue "Spectrum (X component)
graph", FALSE
Canvas1_Component2.SetParameterMode "Spectrum (Y component) graph",
0
Canvas1_Component2.SetParameterValue "Spectrum (Y component)
graph", FALSE
Canvas1_Component2.SetParameterMode "Waveform (total power) graph",
0
Canvas1_Component2.SetParameterValue "Waveform (total power)
graph", TRUE
Canvas1_Component2.SetParameterMode "Waveform (X component) graph",
0
Canvas1_Component2.SetParameterValue "Waveform (X component)
graph", FALSE
Canvas1_Component2.SetParameterMode "Waveform (Y component) graph",
0
Canvas1_Component2.SetParameterValue "Waveform (Y component)
graph", FALSE
Canvas1_Component2.SetParameterMode "Enabled", 0
Canvas1_Component2.SetParameterValue "Enabled", TRUE
Canvas1_Component2.SetParameterMode "Convert noise bins", 0
Canvas1_Component2.SetParameterValue "Convert noise bins", FALSE
Canvas1_Component2.SetParameterMode "Generate random seed", 0
Canvas1_Component2.SetParameterValue "Generate random seed", TRUE
Canvas1_Component2.SetParameterMode "Random seed index", 0
Canvas1_Component2.SetParameterValue "Random seed index", 0
'SCRIPT for component Optical Spectrum Analyzer.
Dim Canvas1_Component3
Set Canvas1_Component3 = Canvas1.CreateComponent("Optical Spectrum
Analyzer","{F11D0C25-3C7D-11D4-93F0-0050DAB7C5D6}",300,60, 40,
34,0)
Canvas1_Component3.Name  = "Optical Spectrum Analyzer"
'Set Optical Spectrum Analyzer parameters.
Canvas1_Component3.SetParameterMode "Resolution bandwidth", 0
Canvas1_Component3.SetParameterValue "Resolution bandwidth", "Off"
Canvas1_Component3.SetParameterMode "Filter type", 0
Canvas1_Component3.SetParameterValue "Filter type", "Rectangle"
Canvas1_Component3.SetParameterMode "Bandwidth", 0
Canvas1_Component3.SetParameterValue "Bandwidth", 0.01
Canvas1_Component3.SetParameterMode "Power unit", 0
Canvas1_Component3.SetParameterValue "Power unit", "dBm"
Canvas1_Component3.SetParameterMode "Minimum value", 0
Canvas1_Component3.SetParameterValue "Minimum value", -100
Canvas1_Component3.SetParameterMode "Frequency unit", 0
Canvas1_Component3.SetParameterValue "Frequency unit", "Hz"
Canvas1_Component3.SetParameterMode "Limit number of points", 0
Canvas1_Component3.SetParameterValue "Limit number of points", TRUE
Canvas1_Component3.SetParameterMode "Max. number of points", 0
```

```
Canvas1_Component3.SetParameterValue "Max. number of points", 128000
Canvas1_Component3.SetParameterMode "Enabled", 0
Canvas1_Component3.SetParameterValue "Enabled", TRUE
Canvas1_Component3.SetParameterMode "Dynamic update", 0
Canvas1_Component3.SetParameterValue "Dynamic update", TRUE
'SCRIPT for component Optical Time Domain Visualizer_1.
Dim Canvas1_Component4
Set Canvas1_Component4 = Canvas1.CreateComponent("Optical Time
Domain Visualizer","{F11D0C25-3C7D-11D4-93F0-
0050DAB7C5D6}",290,270, 40, 34,0)
Canvas1_Component4.Name  = "Optical Time Domain Visualizer_1"
'Set Optical Time Domain Visualizer_1 parameters.
Canvas1_Component4.SetParameterMode "Time unit", 0
Canvas1_Component4.SetParameterValue "Time unit", "s"
Canvas1_Component4.SetParameterMode "Reference bit rate", 3
Canvas1_Component4.SetParameterUnit "Reference bit rate", "Bits/s"
Canvas1_Component4.SetParameterScript "Reference bit rate", "Bit
rate"
Canvas1_Component4.SetParameterMode "Phase unit", 0
Canvas1_Component4.SetParameterValue "Phase unit", "deg"
Canvas1_Component4.SetParameterMode "Unwrap phase", 0
Canvas1_Component4.SetParameterValue "Unwrap phase", TRUE
Canvas1_Component4.SetParameterMode "Power unit", 0
Canvas1_Component4.SetParameterValue "Power unit", "W"
Canvas1_Component4.SetParameterMode "Minimum value", 0
Canvas1_Component4.SetParameterValue "Minimum value", -100
Canvas1_Component4.SetParameterMode "Limit number of points", 0
Canvas1_Component4.SetParameterValue "Limit number of points", TRUE
Canvas1_Component4.SetParameterMode "Max. number of points", 0
Canvas1_Component4.SetParameterValue "Max. number of points", 128000
Canvas1_Component4.SetParameterMode "Centered at max power", 0
Canvas1_Component4.SetParameterValue "Centered at max power", TRUE
Canvas1_Component4.SetParameterMode "Center frequency", 0
Canvas1_Component4.SetParameterUnit "Center frequency", "THz"
Canvas1_Component4.SetParameterValue "Center frequency", 193.1
Canvas1_Component4.SetParameterMode "Sample rate", 3
Canvas1_Component4.SetParameterUnit "Sample rate", "Hz"
Canvas1_Component4.SetParameterScript "Sample rate", "5 * ( Sample
rate )"
Canvas1_Component4.SetParameterMode "Enabled", 0
Canvas1_Component4.SetParameterValue "Enabled", TRUE
Canvas1_Component4.SetParameterMode "Dynamic update", 0
Canvas1_Component4.SetParameterValue "Dynamic update", TRUE
Canvas1_Component4.SetParameterMode "Generate random seed", 0
Canvas1_Component4.SetParameterValue "Generate random seed", TRUE
Canvas1_Component4.SetParameterMode "Random seed index", 0
Canvas1_Component4.SetParameterValue "Random seed index", 0
'SCRIPT for component Optical Spectrum Analyzer_1.
Dim Canvas1_Component5
```


```
Set Canvas1_Component5 = Canvas1.CreateComponent("Optical Spectrum
Analyzer","{F11D0C25-3C7D-11D4-93F0-0050DAB7C5D6}",470,60, 40,
34,0)
Canvas1_Component5.Name  = "Optical Spectrum Analyzer_1"
'Set Optical Spectrum Analyzer_1 parameters.
Canvas1_Component5.SetParameterMode "Resolution bandwidth", 0
Canvas1_Component5.SetParameterValue "Resolution bandwidth", "Off"
Canvas1_Component5.SetParameterMode "Filter type", 0
Canvas1_Component5.SetParameterValue "Filter type", "Rectangle"
Canvas1_Component5.SetParameterMode "Bandwidth", 0
Canvas1_Component5.SetParameterValue "Bandwidth", 0.01
Canvas1_Component5.SetParameterMode "Power unit", 0
Canvas1_Component5.SetParameterValue "Power unit", "dBm"
Canvas1_Component5.SetParameterMode "Minimum value", 0
Canvas1_Component5.SetParameterValue "Minimum value", -100
Canvas1_Component5.SetParameterMode "Frequency unit", 0
Canvas1_Component5.SetParameterValue "Frequency unit", "Hz"
Canvas1_Component5.SetParameterMode "Limit number of points", 0
Canvas1_Component5.SetParameterValue "Limit number of points", TRUE
Canvas1_Component5.SetParameterMode "Max. number of points", 0
Canvas1_Component5.SetParameterValue "Max. number of points", 128000
Canvas1_Component5.SetParameterMode "Enabled", 0
Canvas1_Component5.SetParameterValue "Enabled", TRUE
Canvas1_Component5.SetParameterMode "Dynamic update", 0
Canvas1_Component5.SetParameterValue "Dynamic update", TRUE
'SCRIPT for component Optical Time Domain Visualizer.
Dim Canvas1_Component7
Set Canvas1_Component7 = Canvas1.CreateComponent("Optical Time
Domain Visualizer","{F11D0C25-3C7D-11D4-93F0-
0050DAB7C5D6}",490,250, 40, 34,0)
Canvas1_Component7.Name  = "Optical Time Domain Visualizer"
'Set Optical Time Domain Visualizer parameters.
Canvas1_Component7.SetParameterMode "Time unit", 0
Canvas1_Component7.SetParameterValue "Time unit", "s"
Canvas1_Component7.SetParameterMode "Reference bit rate", 3
Canvas1_Component7.SetParameterUnit "Reference bit rate", "Bits/s"
Canvas1_Component7.SetParameterScript "Reference bit rate", "Bit
rate"
Canvas1_Component7.SetParameterMode "Phase unit", 0
Canvas1_Component7.SetParameterValue "Phase unit", "deg"
Canvas1_Component7.SetParameterMode "Unwrap phase", 0
Canvas1_Component7.SetParameterValue "Unwrap phase", TRUE
Canvas1_Component7.SetParameterMode "Power unit", 0
Canvas1_Component7.SetParameterValue "Power unit", "W"
Canvas1_Component7.SetParameterMode "Minimum value", 0
Canvas1_Component7.SetParameterValue "Minimum value", -100
Canvas1_Component7.SetParameterMode "Limit number of points", 0
Canvas1_Component7.SetParameterValue "Limit number of points", TRUE
Canvas1_Component7.SetParameterMode "Max. number of points", 0
Canvas1_Component7.SetParameterValue "Max. number of points", 128000
Canvas1_Component7.SetParameterMode "Centered at max power", 0
```

```
Canvas1_Component7.SetParameterValue "Centered at max power", TRUE
Canvas1_Component7.SetParameterMode "Center frequency", 0
Canvas1_Component7.SetParameterUnit "Center frequency", "THz"
Canvas1_Component7.SetParameterValue "Center frequency", 193.1
Canvas1_Component7.SetParameterMode "Sample rate", 3
Canvas1_Component7.SetParameterUnit "Sample rate", "Hz"
Canvas1_Component7.SetParameterScript "Sample rate", "5 * ( Sample
rate )"
Canvas1_Component7.SetParameterMode "Enabled", 0
Canvas1_Component7.SetParameterValue "Enabled", TRUE
Canvas1_Component7.SetParameterMode "Dynamic update", 0
Canvas1_Component7.SetParameterValue "Dynamic update", TRUE
Canvas1_Component7.SetParameterMode "Generate random seed", 0
Canvas1_Component7.SetParameterValue "Generate random seed", TRUE
Canvas1_Component7.SetParameterMode "Random seed index", 0
Canvas1_Component7.SetParameterValue "Random seed index", 0
'SCRIPT for component User Defined Bit Sequence Generator_1.
Dim Canvas1_Component8
Set Canvas1_Component8 = Canvas1.CreateComponent("User Defined Bit
Sequence Generator","{6DA31CEE-058F-11D4-93BD-
0050DAB7C5D6}",50,120, 34, 34,0)
Canvas1_Component8.Name  = "User Defined Bit Sequence Generator_1"
'Set User Defined Bit Sequence Generator_1 parameters.
Canvas1_Component8.SetParameterMode "Bit rate", 3
Canvas1_Component8.SetParameterUnit "Bit rate", "Bits/s"
Canvas1_Component8.SetParameterScript "Bit rate", "Bit rate"
Canvas1_Component8.SetParameterMode "Load from file", 0
Canvas1_Component8.SetParameterValue "Load from file", FALSE
Canvas1_Component8.SetParameterMode "Filename", 0
Canvas1_Component8.SetParameterValue "Filename", "Sequence.dat"
Canvas1_Component8.SetParameterMode "Bit sequence", 0
Canvas1_Component8.SetParameterValue "Bit sequence", "00010000"
Canvas1_Component8.SetParameterMode "Number of leading zeros", 3
Canvas1_Component8.SetParameterScript "Number of leading zeros",
"(Time window * 3 / 100 ) * Bit rate"
Canvas1_Component8.SetParameterMode "Number of trailing zeros", 3
Canvas1_Component8.SetParameterScript "Number of trailing zeros",
"(Time window * 3 / 100 ) * Bit rate"
Canvas1_Component8.SetParameterMode "Enabled", 0
Canvas1_Component8.SetParameterValue "Enabled", TRUE
Canvas1_Component8.SetParameterMode "Iterations", 3
Canvas1_Component8.SetParameterScript "Iterations", "Iterations"
'SCRIPT for component Optical Sech Pulse Generator_1.
Dim Canvas1_Component9
Set Canvas1_Component9 = Canvas1.CreateComponent("Optical Sech Pulse
Generator","{6DA31CEE-058F-11D4-93BD-0050DAB7C5D6}",180,180, 34,
34,0)
Canvas1_Component9.Name  = "Optical Sech Pulse Generator_1"
'Set Optical Sech Pulse Generator_1 parameters.
Canvas1_Component9.SetParameterMode "Frequency", 0
Canvas1_Component9.SetParameterUnit "Frequency", "THz"
```


```
Canvas1_Component9.SetParameterValue "Frequency", 193.1
Canvas1_Component9.SetParameterMode "Power", 2
Canvas1_Component9.SetParameterUnit "Power", "W"
Canvas1_Component9.SetSweepParameterValue "Power",1,  0.30208
Canvas1_Component9.SetSweepParameterValue "Power",2,  1.20832
Canvas1_Component9.SetSweepParameterValue "Power",3,  2.71872
Canvas1_Component9.SetParameterMode "Bias", 0
Canvas1_Component9.SetParameterUnit "Bias", "dBm"
Canvas1_Component9.SetParameterValue "Bias", -100
Canvas1_Component9.SetParameterMode "Width", 0
Canvas1_Component9.SetParameterValue "Width", 0.5
Canvas1_Component9.SetParameterMode "Position", 0
Canvas1_Component9.SetParameterValue "Position", 0.5
Canvas1_Component9.SetParameterMode "Truncated", 0
Canvas1_Component9.SetParameterValue "Truncated", FALSE
Canvas1_Component9.SetParameterMode "Chirp definition", 0
Canvas1_Component9.SetParameterValue "Chirp definition", "Linear"
Canvas1_Component9.SetParameterMode "Chirp factor", 0
Canvas1_Component9.SetParameterValue "Chirp factor", 0
Canvas1_Component9.SetParameterMode "Alpha parameter", 0
Canvas1_Component9.SetParameterValue "Alpha parameter", 0
Canvas1_Component9.SetParameterMode "Adiabatic chirp", 0
Canvas1_Component9.SetParameterValue "Adiabatic chirp", 0
Canvas1_Component9.SetParameterMode "Azimuth", 0
Canvas1_Component9.SetParameterValue "Azimuth", 0
Canvas1_Component9.SetParameterMode "Ellipticity", 0
Canvas1_Component9.SetParameterValue "Ellipticity", 0
Canvas1_Component9.SetParameterMode "Enabled", 0
Canvas1_Component9.SetParameterValue "Enabled", TRUE
Canvas1_Component9.SetParameterMode "Parameterized", 3
Canvas1_Component9.SetParameterScript "Parameterized",
"Parameterized"
Canvas1_Component9.SetParameterMode "Sample rate", 3
Canvas1_Component9.SetParameterUnit "Sample rate", "Hz"
Canvas1_Component9.SetParameterScript "Sample rate", "Sample rate"
 'Set Total Sweep Iterations
Layout1.SetTotalSweepIterations(3)
 'Set Current Sweep Iteration
Layout1.SetCurrentSweepIteration(3)
'Attach Monitors.
Canvas1_Component2.GetOutputPort(1).CreateMonitor
Canvas1_Component9.GetOutputPort(1).CreateMonitor
'Connecting components.
Canvas1_Component9.GetOutputPort(1).ConnectVisualizer(Canvas1_Comp
onent3.GetInputPort(1))
Canvas1_Component9.GetOutputPort(1).ConnectVisualizer(Canvas1_Comp
onent4.GetInputPort(1))
Canvas1_Component2.GetOutputPort(1).ConnectVisualizer(Canvas1_Comp
onent5.GetInputPort(1))
Canvas1_Component2.GetOutputPort(1).ConnectVisualizer(Canvas1_Comp
onent7.GetInputPort(1))
```

```
Canvas1_Component8.GetOutputPort(1).Connect(Canvas1_Component9.Get
InputPort(1))
Canvas1_Component9.GetOutputPort(1).Connect(Canvas1_Component2.Get
InputPort(1))
```

GetSweepValue(long iter, long nX, long nY);

Purpose: This method returns the representing Z value for particular X and Y point and certain sweep iteration.

Arguments:

iter - A long value that indicates the sweep iteration.

nX - A long value that indicates the X point.

nY - A long value that indicates the Y point.

Return:

Double.

The Z value as a double.

Example: (add to the basic example)

```
Document.CalculateProject TRUE, TRUE
Dim graph
Set graph = Canvas1_Component2.GetGraph("Waveform (total power)")
Dim dValue
dValue = graph.GetSweepValue(1, 1, 1)
MsgBox(dValue)
```

GetSweepYStartValue(long iter)

Purpose: This method returns the starting value on Y coordinate for the specified sweep iteration.

Return:

Double.

The Y starting value as a double. It is the same as X starting value.

Example: (add to the basic example)

```
Document.CalculateProject TRUE, TRUE
Dim graph
Set graph = Canvas1_Component2.GetGraph("Waveform (total power)")
Dim nDim
nDim = graph. GetSweepYStartValue (1)
MsgBox(nDim)
```

GetSweepXStartValue(long iter)

Purpose: This method returns the starting value on X coordinate for the specified sweep iteration.

Return:

Double.

The X starting value as a double. It is the same as Y starting value.

Example: (add to the basic example)

```
Document.CalculateProject TRUE, TRUE
```


```
Dim graph
Set graph = Canvas1_Component2.GetGraph("Waveform (total power)")
Dim nDim
nDim = graph. GetSweepXStartValue (1)
MsgBox(nDim)
```

GetSweepYDim(long iter)

Purpose: This method returns the number of points on Y coordinate for the specified sweep iteration.

Arguments:

iter - A long value that indicates the sweep iteration.

Return:

Long.

The number of Y points as a long.

Example: (add to the basic example)

```
Document.CalculateProject TRUE, TRUE
Dim graph
Set graph = Canvas1_Component2.GetGraph("Waveform (total power)")
Dim nDim
nDim = graph.GetSweepYDim(1)
MsgBox(nDim)
```

GetSweepXDim(long iter)

Purpose: This method returns the number of points on X coordinate for the specified sweep iteration.

Arguments:

iter - A long value that indicates the sweep iteration

Return:

Long.

The number of X points as a long.

Example: (add to the basic example)

```
Document.CalculateProject TRUE, TRUE
Dim graph
Set graph = Canvas1_Component2.GetGraph("Waveform (total power)")
Dim nDim
nDim = graph.GetSweepXDim(1)
MsgBox(nDim)
```

get_YDim()

Purpose: This method returns the number of points on Y coordinate.

Return:

Long.

The number of Y points as a long.

Example: (add to the basic example)

```
Document.CalculateProject TRUE, TRUE
Dim graph
Set graph = Canvas1_Component2.GetGraph("Waveform (total power)")
Dim nSize
nSize = graph.YDim
MsgBox(nSize)
```

`get_XDim()`

Purpose: This method returns the number of points on X coordinate.

Return:

Long.

The number of X points as a long.

Example: (add to the basic example)

```
Document.CalculateProject TRUE, TRUE
Dim graph
Set graph = Canvas1_Component2.GetGraph("Waveform (total power)")
Dim nSize
nSize = graph.XDim
MsgBox(nSize)
```

`GetValue``(long nX, long nY)`

Purpose: This method returns the representing Z value for particular X and Y point.

Arguments:

nX - A long value that indicates the X point.

nY - A long value that indicates the Y point.

Return:

Double.

The Z value as a double.

Example: (add to the basic example)

```
Document.CalculateProject TRUE, TRUE
Dim graph
Set graph = Canvas1_Component2.GetGraph("Waveform (total power)")
Dim dValue
dValue = graph.GetSweepValue(50, 500)
MsgBox(dValue)
```

`get_YSpacing()`

Purpose: This method returns the distance between each point on Y coordinate.

Return:

Double.

The distances between each point on the Y coordinate.

Example: (add to the basic example)

```
Dim graph
Set graph = Canvas1_Component2.GetGraph("Waveform (total power)")
Dim sTitle
```


```
sTitle = graph.YSpacing
MsgBox(sTitle)
```

get_XSpacing()

Purpose: This method returns the distance between each point on X coordinate.

Return:

Double.

The distances between each point on the X coordinate.

Example: (add to the basic example)

```
Dim graph
Set graph = Canvas1_Component2.GetGraph("Waveform (total power)")
Dim sTitle
sTitle = graph.XSpacing
MsgBox(sTitle)
```

get_YStartValue()

Purpose: This method returns the starting value on Y coordinate.

Return:

Double.

The Y starting value as a double. It is the same as X starting value.

Example: (add to the basic example)

```
Dim graph
Set graph = Canvas1_Component2.GetGraph("Waveform (total power)")
Dim sTitle
sTitle = graph.YStartValue
MsgBox(sTitle)
```

get_XStartValue()

Purpose: This method returns the X starting value.

Return:

Double.

The X starting value as a double. It is the same as Y starting value.

Example: (add to the basic example)

```
Dim graph
Set graph = Canvas1_Component2.GetGraph("Waveform (total power)")
Dim sTitle
sTitle = graph.XStartValue
MsgBox(sTitle)
```

get_TitleZ()

Purpose: This method returns the title of the Z coordinate.

Return:

String.

The title of the Z coordinate.

Example: (add to the basic example)

```
Dim graph
Set graph = Canvas1_Component2.GetGraph("Waveform (total power)")
Dim sTitle
sTitle = graph.TitleZ
MsgBox(sTitle)
```

`GetUnitZ()`

Purpose: This method returns the units of the Z coordinate.

Return:

String.

The units of the Z coordinate as a string.

Example: (add to the basic example)

```
Dim graph
Set graph = Canvas1_Component2.GetGraph("Waveform (total power)")
Dim sUnit
sUnit = graph.GetUnitZ
MsgBox(sUnit)
```


CSDKPortMgr

`DeleteAllMonitors()`

Purpose: This method deletes all monitors attached to the ports in the port manager.

Return:

None.

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Layout1.Name  = "Layout 1"
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
Dim Canvas1_Component1
Set Canvas1_Component1 = Canvas1.CreateComponent("CW
Laser","{6DA31CEE-058F-11D4-93BD-0050DAB7C5D6}",200,80, 34, 34,0)
Canvas1_Component1.Name  = "CW Laser"
Dim Canvas1_Component2
Set Canvas1_Component2 = Canvas1.CreateComponent("Optical Spectrum
Analyzer","{F11D0C25-3C7D-11D4-93F0-0050DAB7C5D6}",300,130, 40,
34,0)
Canvas1_Component2.Name  = "Optical Spectrum Analyzer"
Canvas1_Component1.GetOutputPort(1).CreateMonitor
Canvas1_Component1.GetOutputPort(1).ConnectVisualizer(Canvas1_Comp
onent2.GetInputPort(1))
Dim portMgr
Set portMgr = Canvas1_Component1.GetPortMgr
portMgr.DeleteAllMonitors
```

`DisconnectAllPorts()`

Purpose: This method disconnects all ports in the port manager.

Return:

None.

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Layout1.Name  = "Layout 1"
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
Dim Canvas1_Component1
Set Canvas1_Component1 = Canvas1.CreateComponent("CW
Laser","{6DA31CEE-058F-11D4-93BD-0050DAB7C5D6}",200,80, 34, 34,0)
Canvas1_Component1.Name  = "CW Laser"
Dim Canvas1_Component2
```

```
Set Canvas1_Component2 = Canvas1.CreateComponent("Optical Spectrum
Analyzer","{F11D0C25-3C7D-11D4-93F0-0050DAB7C5D6}",300,130, 40,
34,0)
Canvas1_Component2.Name  = "Optical Spectrum Analyzer"
Canvas1_Component1.GetOutputPort(1).CreateMonitor
Canvas1_Component1.GetOutputPort(1).ConnectVisualizer(Canvas1_Comp
onent2.GetInputPort(1))
Dim portMgr
Set portMgr = Canvas1_Component1.GetPortMgr
portMgr.DisconnectAllPorts
```

IsAnyPortConnected()

Purpose: This method returns TRUE if any of the ports is connected.

Return:

BOOL.

TRUE if any of the ports is connected, FALSE otherwise.

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Layout1.Name  = "Layout 1"
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
Dim Canvas1_Component1
Set Canvas1_Component1 = Canvas1.CreateComponent("CW
Laser","{6DA31CEE-058F-11D4-93BD-0050DAB7C5D6}",200,80, 34, 34,0)
Canvas1_Component1.Name  = "CW Laser"
Dim Canvas1_Component2
Set Canvas1_Component2 = Canvas1.CreateComponent("Optical Spectrum
Analyzer","{F11D0C25-3C7D-11D4-93F0-0050DAB7C5D6}",300,130, 40,
34,0)
Canvas1_Component2.Name  = "Optical Spectrum Analyzer"
Canvas1_Component1.GetOutputPort(1).CreateMonitor
Canvas1_Component1.GetOutputPort(1).ConnectVisualizer(Canvas1_Comp
onent2.GetInputPort(1))
Dim portMgr
Set portMgr = Canvas1_Component1.GetPortMgr
Dim bV
bV = portMgr.IsAnyPortConnected
MsgBox(bV)
```


CSDKExtObjectMgr functionalities

GetObjectBySID(unsigned long nSID)

Purpose: This method returns an object with specified SID.

Arguments:

nSID - Specifies the SID of the object.

Return:

An object.

GetObjectByName(BSTR sName)

Purpose: This method returns an object with specified name.

Arguments:

sName - Specifies the name of the object to retrieve.

Return:

An object.

GetObject(unsigned long nLocalID)

Purpose: This method returns an object with specified local ID.

Arguments:

nLocalID - Specifies the local ID of the object.

Return:

An object.

DeleteObjectBySID(unsigned long nSID)

Purpose: This method deletes the specified object from this manager.

Arguments:

nSID - Specifies the SID of the object to delete.

Return:

None.

DeleteAll()

Purpose: This method deletes all objects from this manager.

Return:

None.

CSDKLayoutMgr

GetCurrentLayout()

Purpose: This method retrieves the current layout.

Return:

The current layout.

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
```

GetCurrentLayoutIndex()

Purpose: This method retreives the index of the current layout.

Return:

Long.

The index of the current layout as a long.

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim nIndex
nIndex = Lm.GetCurrentLayoutIndex
MsgBox(nIndex)
```

SetCurrentLayout(long nLayoutIndex)

Purpose: This method sets the current layout by index.

Return:

None.

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Lm.AddLayout "Layout 2"
Lm.AddLayout "Layout 3"
Lm.SetCurrentLayout 2
```

GetLayoutCount()

Purpose: This method returns the number of layouts.

Return:

Long.

The number of layouts as a long.

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim nCount
nCount = Lm.GetLayoutCount
```


```
MsgBox(nCount)
```

DuplicateCurrentLayout()

Purpose: This method duplicates the current layout.

Return:

None.

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Lm.AddLayout "Layout 2"
Lm.AddLayout "Layout 3"
Lm.DuplicateCurrentLayout
```

DeleteCurrentLayout()

Purpose: This method deletes the current layout.

Return:

None.

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Lm.AddLayout "Layout 2"
Lm.AddLayout "Layout 3"
Lm.DeleteCurrentLayout
```

AddLayout`( BSTR sName)`

Purpose: This method adds a layout.

Arguments:

sName - A null terminating string that indicates the name of the layout.

Return:

None.

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Lm.AddLayout "Layout 2"
Lm.AddLayout "Layout 3"
```

CSDKExtObjectMgr functionalities

GetObjectBySID(unsigned long nSID);

Purpose: This method returns an object with specified SID.

Arguments:

nSID - Specifies the SID of the object.

Return:

An object.

GetObjectByName(BSTR sName)

Purpose: This method returns an object with specified name.

Arguments:

sName - Specifies the name of the object to retrieve.

Return:

An object.

GetObject(unsigned long nLocalID)

Purpose: This method returns an object with specified local ID.

Arguments:

nLocalID - Specifies the local ID of the object.

Return:

An object.

DeleteObjectBySID(unsigned long nSID)

Purpose: This method deletes the specified object from this manager.

Arguments:

nSID - Specifies the SID of the object to delete.

Return:

None.

DeleteAll()

Purpose: This method deletes all objects from this manager.

Return:

None.

CSDKPath

get_Color()

Purpose: This method gets the path color.

Return:

OLE_COLOR

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
Dim Canvas1_Component1
Set Canvas1_Component1 = Canvas1.CreateComponent("CW
Laser","{6DA31CEE-058F-11D4-93BD-0050DAB7C5D6}",180,110, 34, 34,0)
Canvas1_Component1.Name  = "CW Laser"
Dim Canvas1_Component2
Set Canvas1_Component2 = Canvas1.CreateComponent("EDFA","{255EDC8F-
37E4-11D4-93EC-0050DAB7C5D6}",320,160, 32, 32,0)
Canvas1_Component2.Name  = "EDFA"
Dim Canvas1_Component3
Set Canvas1_Component3 = Canvas1.CreateComponent("Optical Spectrum
Analyzer","{F11D0C25-3C7D-11D4-93F0-0050DAB7C5D6}",500,200, 40,
34,0)
Canvas1_Component3.Name  = "Optical Spectrum Analyzer"
Canvas1_Component2.GetOutputPort(1).CreateMonitor
Canvas1_Component1.GetOutputPort(1).Connect(Canvas1_Component2.Get
InputPort(1))
Canvas1_Component2.GetOutputPort(1).ConnectVisualizer(Canvas1_Comp
onent3.GetInputPort(1))
Dim path
Set path = Layout1.GetPathMgr.CreatePath( "Path1",
Canvas1_Component1.GetOutputPort(1).GetSID,
Canvas1_Component2.GetOutputPort(1).GetSID )
path.Color = RGB(250,50,50)
Dim color1
color1 = path.Color
MsgBox(color1)
```

put_Color(OLE_COLOR newVal)

Purpose: This method sets the path color.

Return:

None.

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
```

```
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
Dim Canvas1_Component1
Set Canvas1_Component1 = Canvas1.CreateComponent("CW
Laser","{6DA31CEE-058F-11D4-93BD-0050DAB7C5D6}",180,110, 34, 34,0)
Canvas1_Component1.Name  = "CW Laser"
Dim Canvas1_Component2
Set Canvas1_Component2 = Canvas1.CreateComponent("EDFA","{255EDC8F-
37E4-11D4-93EC-0050DAB7C5D6}",320,160, 32, 32,0)
Canvas1_Component2.Name  = "EDFA"
Dim Canvas1_Component3
Set Canvas1_Component3 = Canvas1.CreateComponent("Optical Spectrum
Analyzer","{F11D0C25-3C7D-11D4-93F0-0050DAB7C5D6}",500,200, 40,
34,0)
Canvas1_Component3.Name  = "Optical Spectrum Analyzer"
Canvas1_Component2.GetOutputPort(1).CreateMonitor
Canvas1_Component1.GetOutputPort(1).Connect(Canvas1_Component2.Get
InputPort(1))
Canvas1_Component2.GetOutputPort(1).ConnectVisualizer(Canvas1_Comp
onent3.GetInputPort(1))
Dim path
Set path = Layout1.GetPathMgr.CreatePath( "Path1",
Canvas1_Component1.GetOutputPort(1).GetSID,
Canvas1_Component2.GetOutputPort(1).GetSID )
path.Color = RGB(250,50,50)
```

`get_Visible()`

Purpose: This method returns TRUE if path was set to be visible.

Return:

BOOL

TRUE if path was set to be visible, FALSE otherwise.

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
Dim Canvas1_Component1
Set Canvas1_Component1 = Canvas1.CreateComponent("CW
Laser","{6DA31CEE-058F-11D4-93BD-0050DAB7C5D6}",180,110, 34, 34,0)
Canvas1_Component1.Name  = "CW Laser"
Dim Canvas1_Component2
Set Canvas1_Component2 = Canvas1.CreateComponent("EDFA","{255EDC8F-
37E4-11D4-93EC-0050DAB7C5D6}",320,160, 32, 32,0)
Canvas1_Component2.Name  = "EDFA"
Dim Canvas1_Component3
Set Canvas1_Component3 = Canvas1.CreateComponent("Optical Spectrum
Analyzer","{F11D0C25-3C7D-11D4-93F0-0050DAB7C5D6}",500,200, 40,
34,0)
```


```
Canvas1_Component3.Name  = "Optical Spectrum Analyzer"
Canvas1_Component2.GetOutputPort(1).CreateMonitor
Canvas1_Component1.GetOutputPort(1).Connect(Canvas1_Component2.Get
InputPort(1))
Canvas1_Component2.GetOutputPort(1).ConnectVisualizer(Canvas1_Comp
onent3.GetInputPort(1))
Dim path
Set path = Layout1.GetPathMgr.CreatePath( "Path1",
Canvas1_Component1.GetOutputPort(1).GetSID,
Canvas1_Component2.GetOutputPort(1).GetSID )
Dim bVisible
bVisible = path.Visible
MsgBox(bVisible)
```

put_Visible(BOOL newVal)

Purpose: This method sets the path to visible or not.

Arguments:

newVal - TRUE to set a path to be visible, FALSE otherwise.

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
Dim Canvas1_Component1
Set Canvas1_Component1 = Canvas1.CreateComponent("CW
Laser","{6DA31CEE-058F-11D4-93BD-0050DAB7C5D6}",180,110, 34, 34,0)
Canvas1_Component1.Name  = "CW Laser"
Dim Canvas1_Component2
Set Canvas1_Component2 = Canvas1.CreateComponent("EDFA","{255EDC8F-
37E4-11D4-93EC-0050DAB7C5D6}",320,160, 32, 32,0)
Canvas1_Component2.Name  = "EDFA"
Dim Canvas1_Component3
Set Canvas1_Component3 = Canvas1.CreateComponent("Optical Spectrum
Analyzer","{F11D0C25-3C7D-11D4-93F0-0050DAB7C5D6}",500,200, 40,
34,0)
Canvas1_Component3.Name  = "Optical Spectrum Analyzer"
Canvas1_Component2.GetOutputPort(1).CreateMonitor
Canvas1_Component1.GetOutputPort(1).Connect(Canvas1_Component2.Get
InputPort(1))
Canvas1_Component2.GetOutputPort(1).ConnectVisualizer(Canvas1_Comp
onent3.GetInputPort(1))
Dim path
Set path = Layout1.GetPathMgr.CreatePath( "Path1",
Canvas1_Component1.GetOutputPort(1).GetSID,
Canvas1_Component2.GetOutputPort(1).GetSID )
path.Visible = FALSE
```

GetComponentList()

Purpose: This method retrieves the list of all components in the path.

Return:

Array of componets in the path.

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
Dim Canvas1_Component1
Set Canvas1_Component1 = Canvas1.CreateComponent("CW
Laser","{6DA31CEE-058F-11D4-93BD-0050DAB7C5D6}",180,110, 34, 34,0)
Canvas1_Component1.Name  = "CW Laser"
Dim Canvas1_Component2
Set Canvas1_Component2 = Canvas1.CreateComponent("EDFA","{255EDC8F-
37E4-11D4-93EC-0050DAB7C5D6}",320,160, 32, 32,0)
Canvas1_Component2.Name  = "EDFA"
Dim Canvas1_Component3
Set Canvas1_Component3 = Canvas1.CreateComponent("Optical Spectrum
Analyzer","{F11D0C25-3C7D-11D4-93F0-0050DAB7C5D6}",500,200, 40,
34,0)
Canvas1_Component3.Name  = "Optical Spectrum Analyzer"
Canvas1_Component2.GetOutputPort(1).CreateMonitor
Canvas1_Component1.GetOutputPort(1).Connect(Canvas1_Component2.Get
InputPort(1))
Canvas1_Component2.GetOutputPort(1).ConnectVisualizer(Canvas1_Comp
onent3.GetInputPort(1))
Dim path
Set path = Layout1.GetPathMgr.CreatePath( "Path1",
Canvas1_Component1.GetOutputPort(1).GetSID,
Canvas1_Component2.GetOutputPort(1).GetSID )
Dim list1
list1 = path.GetComponentList
Dim bV
bV = IsArray(list1)
MsgBox(bV)
```

`GetPortList()`

Purpose: This method retrieves the list of all ports in the path.

Return:

Array of ports in the path.

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
Dim Canvas1_Component1
```


```
Set Canvas1_Component1 = Canvas1.CreateComponent("CW
Laser","{6DA31CEE-058F-11D4-93BD-0050DAB7C5D6}",180,110, 34, 34,0)
Canvas1_Component1.Name  = "CW Laser"
Dim Canvas1_Component2
Set Canvas1_Component2 = Canvas1.CreateComponent("EDFA","{255EDC8F-
37E4-11D4-93EC-0050DAB7C5D6}",320,160, 32, 32,0)
Canvas1_Component2.Name  = "EDFA"
Dim Canvas1_Component3
Set Canvas1_Component3 = Canvas1.CreateComponent("Optical Spectrum
Analyzer","{F11D0C25-3C7D-11D4-93F0-0050DAB7C5D6}",500,200, 40,
34,0)
Canvas1_Component3.Name  = "Optical Spectrum Analyzer"
Canvas1_Component2.GetOutputPort(1).CreateMonitor
Canvas1_Component1.GetOutputPort(1).Connect(Canvas1_Component2.Get
InputPort(1))
Canvas1_Component2.GetOutputPort(1).ConnectVisualizer(Canvas1_Comp
onent3.GetInputPort(1))
Dim path
Set path = Layout1.GetPathMgr.CreatePath( "Path1",
Canvas1_Component1.GetOutputPort(1).GetSID,
Canvas1_Component2.GetOutputPort(1).GetSID )
Dim list1
list1 = path.GetPortList
Dim bV
bV = IsArray(list1)
MsgBox(bV)
```

CSDKPathMgr

CreatePath(BSTR strPathName, long startSID, long endSID, IDispatch **pDis)

Purpose: This method creates a path.

Arguments:

strPathName - A null terminating string that specifies the name of the path.

startSID - A long that specifies the SID of the starting port.

endSID - A long that specifies the SID of the end port.

Return:

A path.

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
Dim Canvas1_Component1
Set Canvas1_Component1 = Canvas1.CreateComponent("CW
Laser","{6DA31CEE-058F-11D4-93BD-0050DAB7C5D6}",180,110, 34, 34,0)
Canvas1_Component1.Name  = "CW Laser"
Dim Canvas1_Component2
Set Canvas1_Component2 = Canvas1.CreateComponent("EDFA","{255EDC8F-
37E4-11D4-93EC-0050DAB7C5D6}",320,160, 32, 32,0)
Canvas1_Component2.Name  = "EDFA"
Dim Canvas1_Component3
Set Canvas1_Component3 = Canvas1.CreateComponent("Optical Spectrum
Analyzer","{F11D0C25-3C7D-11D4-93F0-0050DAB7C5D6}",500,200, 40,
34,0)
Canvas1_Component3.Name  = "Optical Spectrum Analyzer"
Canvas1_Component2.GetOutputPort(1).CreateMonitor
Canvas1_Component1.GetOutputPort(1).Connect(Canvas1_Component2.Get
InputPort(1))
Canvas1_Component2.GetOutputPort(1).ConnectVisualizer(Canvas1_Comp
onent3.GetInputPort(1))
Layout1.GetPathMgr.CreatePath "Path1",
Canvas1_Component1.GetOutputPort(1).GetSID,
Canvas1_Component2.GetOutputPort(1).GetSID
```

CSDKExtObjectMgr functionalities

GetObjectBySID(unsigned long nSID)

Purpose: This method returns an object with specified SID.

Arguments:

nSID - Specifies the SID of the object.

Return:

An object.

GetObjectByName(BSTR sName)

Purpose: This method returns an object with specified name.

Arguments:

sName - Specifies the name of the object to retrieve.

Return:

An object.

GetObject(unsigned long nLocalID)

Purpose: This method returns an object with specified local ID.

Arguments:

nLocalID - Specifies the local ID of the object.

Return:

An object.

DeleteObjectBySID(unsigned long nSID)

Purpose: This method deletes the specified object from this manager.

Arguments:

nSID - Specifies the SID of the object to delete.

Return:

None.

DeleteAll();

Purpose: This method deletes all objects from this manager.

Return:

None.

AddCategory(BSTR sName, OLE_HANDLE hIcon, BSTR shortDescription,BSTR longDescription)

Purpose: This method adds a category to this manager.

Remarks: Once the manager has a category, the objects in this manager may be grouped by categories by setting their categories using the method AssignToCategory().

Arguments:

sName - Specifies the name of the category to add. The name must be unique to this manager that is if a category with this name already exists in this manager then you may not reuse this name.

hIcon - A handle to an icon that will represent the category. If you do not wish to use this argument, set it to NULL.

shortDescription - A short description of the category. This may be used by tooltips. If you do not wish to use this argument, set it to NULL.

longDescription - A long description of the category. If you do not wish to use this argument, set it to NULL.

Return:

None.

MoveBackOnePosition(IDispatch* pDispObject)

Purpose: This method moves the specified object's position in the list one position towards the back.

Remarks: The tail of the list is the front, and the head of the list is the back.

Argument pSuccess will be set to VARIANT_FALSE if the specified object is already at the back.

Arguments:

pDispObject - A pointer to the IDispatch interface of the object to move.

Return:

BOOL.

TRUE if the object was moved towards the back, FALSE otherwise.

MoveForwardOnePosition(IDispatch* pDispObject)

Purpose: This method moves the specified object's position in the list one position towards the front.

Remarks: The tail of the list is the front, and the head of the list is the back.

Argument pSuccess will be set to VARIANT_FALSE if the specified object is already at the front.

Arguments:

pDispObject - A pointer to the IDispatch interface of the object to move.

Return:

BOOL.

TRUE if the object was moved towards the front, FALSE otherwise.

MoveToFront(IDispatch* pDispObject)

Purpose: This method moves the specified object's position in the list to the front.

Remarks: The tail of the list is the front, and the head of the list is the back.

Argument pSuccess will be set to VARIANT_FALSE if the specified object is already at the front.

Arguments:

pDispObject - A pointer to the IDispatch interface of the object to move.

Return:

BOOL.

TRUE if the object was moved to the front, FALSE otherwise.

MoveToBack(IDispatch* pDispObject,)

Purpose: This method moves the specified object's position in the list to the back.

Remarks: The tail of the list is the front, and the head of the list is the back. Argument pSuccess will be set to VARIANT_FALSE if the specified object is already at the front.

Return:

BOOL.

TRUE if the object was moved to the front, FALSE otherwise.

CSDKSubsystem

RemoveParameterBySID(unsigned long nSID)

Purpose: This method removes a user defined parameter with SID (nSID)

Arguments:

nSID - A long that specifies the SID of the parameter to be removed.

Return:

None.

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
Dim Canvas1_Component1
Set Canvas1_Component1 = Canvas1.CreateComponent("Subsystem
1.0","{C83C8C01-53FD-11D4-9407-0050DAB7C5D6}",240,250, 162, 120,0)
Canvas1_Component1.Name  = "Subsystem"
Dim nSID
nSID = Canvas1_Component1.AddParameter( "Param1", 2 , "Image",
0.000000, 100.000000, "10.9", "Hz", "")
Canvas1_Component1.SetParameterMode "Param1", 0
Canvas1_Component1.SetParameterValue "Param1", 10.9
Canvas1_Component1.RemoveParameterBySID nSID
```

RemoveParameter(BSTR sName)

Purpose: This method removes a user defined parameter by name (sName).

Arguments:

sName - A null terminating string that specifies the name of the parameter to be removed.

Return:

None.

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
Dim Canvas1_Component1
Set Canvas1_Component1 = Canvas1.CreateComponent("Subsystem
1.0","{C83C8C01-53FD-11D4-9407-0050DAB7C5D6}",240,250, 162, 120,0)
Canvas1_Component1.Name  = "Subsystem"
Canvas1_Component1.AddParameter "Param1", 2 , "Image", 0.000000,
100.000000, "10.9", "Hz", ""
Canvas1_Component1.SetParameterMode "Param1", 0
```

```
Canvas1_Component1.SetParameterValue "Param1", 10.9
Canvas1_Component1.RemoveParameter "Param1"
```

AddParameter(BSTR sName, long nType, BSTR sCategoryName, double dMin, double dMax, BSTR sValue, BSTR sUnit, BSTR sChoiceList)

Purpose: This method adds a parameter to the subsystem.

Arguments:

sName - A null-terminated string that specifies the name of the added parameter. The name must be unique.

nType - A long value that specifies the type of the added parameter. The following types are available:

- 0 - type BOOL
- 1 - type Choice
- 2 - type Double
- 3 - type Long
- 4 - type String
- 5 - type MxN

sCategoryName - A null-terminated string that specifies the name of the category to add.

dMin - A double value that specifies the min value of the parameter.

dMax - A double value that specifies the max value of the parameter.

sValue - A null-terminated string that specifies the value of the added parameter.

sUnit - A null-terminated string that specifies the units of the added parameter.

sChoiceList - A null-terminated string that specifies the choice list of the added parameter. If the parameter is not a type Choice than the string is empty.

Return:

Unsigned long.

nSID - returns the SID of the added parameter

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
Dim Canvas1_Component1
Set Canvas1_Component1 = Canvas1.CreateComponent("Subsystem
1.0","{C83C8C01-53FD-11D4-9407-0050DAB7C5D6}",240,250, 162, 120,0)
Canvas1_Component1.Name  = "Subsystem"
Canvas1_Component1.AddParameter "Param1", 2 , "Image", 0.000000,
100.000000, "10.9", "Hz", ""
Canvas1_Component1.SetParameterMode "Param1", 0
Canvas1_Component1.SetParameterValue "Param1", 10.9
```

RemovePortBySID(unsigned long nSID)

Purpose: This method removes a port with SID (nSID)

Arguments:

nSID - A long that specifies the SID of the port to be removed.

Return:

None.

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
Dim Canvas1_Component1
Set Canvas1_Component1 = Canvas1.CreateComponent("Subsystem
1.0","{C83C8C01-53FD-11D4-9407-0050DAB7C5D6}",200,170, 162, 120,0)
Canvas1_Component1.Name  = "Subsystem"
Canvas1_Component1.AddPort "Output", 3, 2, 0.428571
Dim nSID
nSID = Canvas1_Component1.AddPort("Input", 2, 0, 0.635714)
Canvas1_Component1.RemovePortBySID nSID
```

RemovePort(BSTR sName)

Purpose: This method removes a port by name (sName)

Return:

None.

Arguments:

sName - A null terminating string that specifies the name of the port to be removed.

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
Dim Canvas1_Component1
Set Canvas1_Component1 = Canvas1.CreateComponent("Subsystem
1.0","{C83C8C01-53FD-11D4-9407-0050DAB7C5D6}",200,170, 162, 120,0)
Canvas1_Component1.Name  = "Subsystem"
Canvas1_Component1.AddPort "Output", 3, 2, 0.428571
Canvas1_Component1.AddPort "Input", 2, 0, 0.635714
Canvas1_Component1.RemovePort "Input"
```

AddPort(BSTR sName, long nType, long nEdge, double dPosition)

Purpose: This method adds a port to the subsystem.

Arguments:

sName - A null terminating string indicating the name of the port.

nType - A long value that specifies the port type. There are four possible types:

- 0 - Input port
- 1 - Output port
- 2 - RelayInput port
- 3 - RelayOutput port

nEdge - The edge of subsystem on which the port will reside:

- 0 - Left
- 1 - Top
- 2 - Right
- 3 - Bottom

dPosition - A double value that specifies the position on the subsystem's edge where the port will reside. The double value must be between 0.0 - 1.0

Return:

Unsigned long.

nSID - SID of the port.

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
Dim Canvas1_Component1
Set Canvas1_Component1 = Canvas1.CreateComponent("Subsystem
1.0","{C83C8C01-53FD-11D4-9407-0050DAB7C5D6}",220,280, 162, 120,0)
Canvas1_Component1.Name  = "Subsystem"
Canvas1_Component1.AddPort "Output", 3, 2, 0.485714
Dim dSID
dSID = Canvas1_Component1.AddPort("Input", 2, 0, 0.485714)
MsgBox(dSID)
```

`GetCanvas()`

Purpose: This method returns the subsystem canvas.

Return:

The subsystem canvas.

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
```

```
Dim Canvas1_Component1
Set Canvas1_Component1 = Canvas1.CreateComponent("Subsystem
1.0","{C83C8C01-53FD-11D4-9407-0050DAB7C5D6}",240,250, 162, 120,0)
Canvas1_Component1.Name  = "Subsystem"
Canvas1_Component1.AddParameter "Param1", 2 , "Image", 0.000000,
100.000000, "10.9", "Hz", ""
Canvas1_Component1.SetParameterMode "Param1", 0
Canvas1_Component1.SetParameterValue "Param1", 10.9
Dim Canvas1_1
Set Canvas1_1 = Canvas1_Component1.GetCanvas
Canvas1_1.Height = 140
Canvas1_1.Width = 140
```

`GetGlassBoxBitmap()`

Purpose: This method returns the subsystem glass box bitmap.

Return:

The subsystem glass box bitmap.

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
Dim Canvas1_Component1
Set Canvas1_Component1 = Canvas1.CreateComponent("Subsystem
1.0","{C83C8C01-53FD-11D4-9407-0050DAB7C5D6}",240,250, 162, 120,0)
Canvas1_Component1.Name  = "Subsystem"
Canvas1_Component1.AddParameter "Param1", 2 , "Image", 0.000000,
100.000000, "10.9", "Hz", ""
Canvas1_Component1.SetParameterMode "Param1", 0
Canvas1_Component1.SetParameterValue "Param1", 10.9
Dim Canvas1_1
Set Canvas1_1 = Canvas1_Component1.GetCanvas
Canvas1_1.Height = 140
Canvas1_1.Width = 140
Dim nBitmap
nBitmap = Canvas1_Component1.GetGlassBoxBitmap
```


CSDKRelayInputPort

`DisconnectRelay()`

Purpose: This method disconnects the connection between this relay input port and input port (or relay output port) inside subsystem.

Return:

BOOL

TRUE if the relay input port is disconnected from another port inside subsystem, FALSE otherwise.

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
Dim Canvas1_Component1
Set Canvas1_Component1 = Canvas1.CreateComponent("CW
Laser","{6DA31CEE-058F-11D4-93BD-0050DAB7C5D6}",90,130, 34, 34,0)
Dim Canvas1_Component3
Set Canvas1_Component3 = Canvas1.CreateComponent("EDFA","{255EDC8F-
37E4-11D4-93EC-0050DAB7C5D6}",500,200, 32, 32,0)
Dim Canvas1_Component4
Set Canvas1_Component4 = Canvas1.CreateComponent("Subsystem
1.0","{C83C8C01-53FD-11D4-9407-0050DAB7C5D6}",230,140, 162, 120,0)
Dim Canvas1_4
Set Canvas1_4 = Canvas1_Component4.GetCanvas
Canvas1_4.Height = 172
Canvas1_4.Width = 172
Canvas1_Component4.SetPosition 260, 150, 422, 270
Dim Canvas1_4_Component2
Set Canvas1_4_Component2 = Canvas1_4.CreateComponent("Linear
Multimode Fiber","{24797318-DD42-4F59-8B7A-D12D3BFC9B1B}",70,70,
32, 32,0)
Canvas1_Component4.AddPort "Input", 2, 0, 0.656250
Canvas1_Component4.AddPort "Output", 3, 2, 0.666250
Canvas1_Component1.GetOutputPort(1).Connect(Canvas1_Component4.Get
InputPort(1))
Canvas1_Component4.GetOutputPort(1).Connect(Canvas1_Component3.Get
InputPort(1))
Canvas1_Component4.GetInputPort(1).Connect(Canvas1_4_Component2.Ge
tInputPort(1))
Canvas1_Component4.GetOutputPort(1).Connect(Canvas1_4_Component2.G
etOutputPort(1))
Canvas1_Component4.GetInputPort(1).DisconnectRelay
```

`ConnectRelay``( IDispatch* pPort)`

Purpose: This method connects the relay input port to relay output port.

Arguments:

pPort - The relay output port object.

Return:

BOOL

TRUE if connection is successful and FALSE otherwise.

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
Dim Canvas1_Component1
Set Canvas1_Component1 = Canvas1.CreateComponent("Subsystem
1.0","{C83C8C01-53FD-11D4-9407-0050DAB7C5D6}",230,180, 162, 120,0)
Canvas1_Component1.Name  = "Subsystem"
Canvas1_Component1.AddPort "Input", 2, 0, 0.492857
Canvas1_Component1.AddPort "Output", 3, 2, 0.457143
Dim Canvas1_1
Set Canvas1_1 = Canvas1_Component1.GetCanvas
Canvas1_1.Height = 140
Canvas1_1.Width = 140
Canvas1_Component1.SetPosition 230, 180, 392, 300
Canvas1_Component1.GetInputPort(1).ConnectRelay(Canvas1_Component1
.GetOutputPort(1))
```

`IsRelayConnected()`

Purpose: This method indicates whether the relay input port is connected to another port inside subsystem.

Return:

BOOL

TRUE if the relay input port is connected to another port inside subsystem, FALSE otherwise.

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
Dim Canvas1_Component34
Set Canvas1_Component34 = Canvas1.CreateComponent("Subsystem
1.0","{C83C8C01-53FD-11D4-9407-0050DAB7C5D6}",250,240, 162, 120,0)
Canvas1_Component34.Name  = "Subsystem"
Canvas1_Component34.AddPort "Output", 3, 2, 0.414286
Canvas1_Component34.AddPort "Input", 2, 0, 0.457143
Dim Canvas1_34
Set Canvas1_34 = Canvas1_Component34.GetCanvas
Canvas1_34.Height = 140
```


```
Canvas1_34.Width = 140
Canvas1_Component34.SetPosition 250, 240, 412, 360
Canvas1_Component34.GetOutputPort(1).ConnectRelay(Canvas1_Componen
t34.GetInputPort(1))
Dim bResult
bResult = Canvas1_Component34.GetInputPort(1).IsRelayConnected
MsgBox(bResult)
```

GetRelayConnection()

Purpose: This method returns the port connected to this port.

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
Dim Canvas1_Component1
Set Canvas1_Component1 = Canvas1.CreateComponent("Subsystem
1.0","{C83C8C01-53FD-11D4-9407-0050DAB7C5D6}",280,250, 162, 120,0)
Canvas1_Component1.AddPort "Output", 3, 2, 0.442857
Canvas1_Component1.AddPort "Input", 2, 0, 0.492857
Dim Canvas1_1
Set Canvas1_1 = Canvas1_Component1.GetCanvas
Canvas1_1.Height = 140
Canvas1_1.Width = 140
Canvas1_Component1.SetPosition 280, 250, 442, 370
Canvas1_Component1.GetOutputPort(1).ConnectRelay(Canvas1_Component
1.GetInputPort(1))
Dim nPort
Set nPort = Canvas1_Component1.GetInputPort(1).GetRelayConnection
Dim sName
sName = nPort.Name
MsgBox(sName)
```

CSDKPort functionalities

`GetSignalType()`

Purpose: Gets the type of the signal that passes through this port.

Return:

String.

The type of the signal that passes through this port as a string.

`Handshake``( IDispatch* pDispPort)`

Purpose: This method tests whether two ports have compatible signal types.

Arguments:

pDispPort - The tested port object .

Return:

BOOL.

A BOOL which indicates whether the test is positive or not. The value is TRUE if both ports have compatible signal types and FALSE otherwise.

`get_Position()`

Purpose: This method gets the position (along the edge) where this port resides.

Return:

Double.

A double value that indicates the port position. The value of the position should be greater then or equal to zero. For a port on a vertical edge, the value of the position should be less then or equal to the height of the component this port belongs to. For a port on a horizontal edge, the value of the position should be less then or equal to the width of the component this port belongs to. Consider a component that is bounded by a rectangle. The ports are placed along the edges of the rectangle. The value of the position increases from left to right and from top to bottom along the edge. The topmost value is 0, and the leftmost value is zero. The caller is responsible allocating/deallocating this argument.

`put_Position``( double dPosition)`

Purpose: Sets the position (along the edge) where this port resides.

Arguments:

dPosition - A double value that indicates the position of the port. The value of the position should be greater then or equal to zero. For a port on a vertical edge, the value of the position should be less then or equal to the height of the component this port belongs to. For a port on a horizontal edge, the value of the position should be less then or equal to the width of the component this port belongs to. Consider a component that is bounded by a rectangle. The ports are placed along the edges of the rectangle. The value of the position increases from left to right and from top to bottom along the edge. The topmost value is 0, and the leftmost value is zero.

Return:

None.

`get_Edge()`

Purpose: This method gets the edge on which this port resides.

Return:

Long.

pEdge - A long value that indicates the edge. Valid values for this argument are:

- 0 - Left edge
- 1 - Top edge
- 2 - Right edge
- 3 - Bottom edge

`put_Edge``( long nEdge)`

Purpose: This method sets the edge on which this port resides.

Arguments:

pEdge - A long value that indicates the edge on which the port resides. Valid values for this argument are:

- 0 - Left edge
- 1 - Top edge
- 2 - Right edge
- 3 - Bottom edge

Return:

None.

`GetConnection()`

Purpose: This method gets the port that is connected to this port.

Return:

The port that is connected to this port.

`Connect``( IDispatch* pDispPort)`

Purpose: This method connects the specified port to this port.

Arguments:

pDispPort - A port object to connect to this port.

Return:

None

`Disconnect()`

Purpose: This method disconnects the connected port from this port.

Return:

None

`IsConnected()`

Purpose: This method determines whether a port is connected to this port.

Return:

BOOL.

A BOOL which indicates whether the port is connected. The value is TRUE if the port is connected and FALSE otherwise.

CSDKExtObject functionalities

`get_Name()`

Purpose: This method returns the name of this object.

Return:

String.

The name of the object as a string.

`put_Name``( BSTR pName)`

Purpose: This method sets the name of this object.

Arguments:

pName - A null terminating string that specifies the name of the component.

Return:

None.

`GetIteration()`

Purpose: This method returns the current iteration in the document at this time.

Return:

Long.

The current sweep iteration as a long.

`GetMaxIterations()`

Purpose: This method returns the maximum number of iterations the document supports at this time.

Return:

Long.

The max number of sweep iterations as a long.

`GetSID()`

Purpose: This method returns the system ID (SID) of this object.

Remarks: This value uniquely identifies this object in the document.

Return:

Unsigned long.

The SID of the object as an unsigned long.

CSDKRelayOutputPort

`DisconnectRelay()`

Purpose: This method disconnects the connection between this relay output port and output port (or relay input port) inside subsystem.

Return:

BOOL

TRUE if ports are disconnected, FALSE otherwise.

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
Dim Canvas1_Component1
Set Canvas1_Component1 = Canvas1.CreateComponent("CW
Laser","{6DA31CEE-058F-11D4-93BD-0050DAB7C5D6}",90,130, 34, 34,0)
Dim Canvas1_Component3
Set Canvas1_Component3 = Canvas1.CreateComponent("EDFA","{255EDC8F-
37E4-11D4-93EC-0050DAB7C5D6}",500,200, 32, 32,0)
Dim Canvas1_Component4
Set Canvas1_Component4 = Canvas1.CreateComponent("Subsystem
1.0","{C83C8C01-53FD-11D4-9407-0050DAB7C5D6}",230,140, 162, 120,0)
Dim Canvas1_4
Set Canvas1_4 = Canvas1_Component4.GetCanvas
Canvas1_4.Height = 172
Canvas1_4.Width = 172
Canvas1_Component4.SetPosition 260, 150, 422, 270
Dim Canvas1_4_Component2
Set Canvas1_4_Component2 = Canvas1_4.CreateComponent("Linear
Multimode Fiber","{24797318-DD42-4F59-8B7A-D12D3BFC9B1B}",70,70,
32, 32,0)
Canvas1_Component4.AddPort "Input", 2, 0, 0.656250
Canvas1_Component4.AddPort "Output", 3, 2, 0.666250
Canvas1_Component1.GetOutputPort(1).Connect(Canvas1_Component4.Get
InputPort(1))
Canvas1_Component4.GetOutputPort(1).Connect(Canvas1_Component3.Get
InputPort(1))
Canvas1_Component4.GetInputPort(1).Connect(Canvas1_4_Component2.Ge
tInputPort(1))
Canvas1_Component4.GetOutputPort(1).Connect(Canvas1_4_Component2.G
etOutputPort(1))
Canvas1_Component4.GetOutputPort(1).DisconnectRelay
```

`ConnectRelay``( IDispatch* pPort)`

Purpose: This method connects the relay output port to relay input port.

Arguments:

pPort - The relay input port object.

Return:

TRUE if connection is successful and FALSE otherwise.

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
Dim Canvas1_Component34
Set Canvas1_Component34 = Canvas1.CreateComponent("Subsystem
1.0","{C83C8C01-53FD-11D4-9407-0050DAB7C5D6}",250,240, 162, 120,0)
Canvas1_Component34.Name  = "Subsystem"
Canvas1_Component34.AddPort "Output", 3, 2, 0.414286
Canvas1_Component34.AddPort "Input", 2, 0, 0.457143
Dim Canvas1_34
Set Canvas1_34 = Canvas1_Component34.GetCanvas
Canvas1_34.Height = 140
Canvas1_34.Width = 140
Canvas1_Component34.SetPosition 250, 240, 412, 360
Canvas1_Component34.GetOutputPort(1).ConnectRelay(Canvas1_Componen
t34.GetInputPort(1))
```

`IsRelayConnected()`

Purpose: This method indicates whether the relay output port is connected to another port inside subsystem.

Return:

BOOL

TRUE if the relay output port is connected to another port inside subsystem, FALSE otherwise.

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
Dim Canvas1_Component34
Set Canvas1_Component34 = Canvas1.CreateComponent("Subsystem
1.0","{C83C8C01-53FD-11D4-9407-0050DAB7C5D6}",250,240, 162, 120,0)
Canvas1_Component34.Name  = "Subsystem"
Canvas1_Component34.AddPort "Output", 3, 2, 0.414286
Canvas1_Component34.AddPort "Input", 2, 0, 0.457143
Dim Canvas1_34
Set Canvas1_34 = Canvas1_Component34.GetCanvas
Canvas1_34.Height = 140
Canvas1_34.Width = 140
Canvas1_Component34.SetPosition 250, 240, 412, 360
```


```
Canvas1_Component34.GetOutputPort(1).ConnectRelay(Canvas1_Componen
t34.GetInputPort(1))
Dim bResult
bResult = Canvas1_Component1.GetOutputPort(1).IsRelayConnected
MsgBox(bResult)
```

GetRelayConnection()

Purpose: This method returns the IDispatch pointer of the port connected to thes port.

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
Dim Canvas1_Component1
Set Canvas1_Component1 = Canvas1.CreateComponent("Subsystem
1.0","{C83C8C01-53FD-11D4-9407-0050DAB7C5D6}",280,250, 162, 120,0)
Canvas1_Component1.AddPort "Output", 3, 2, 0.442857
Canvas1_Component1.AddPort "Input", 2, 0, 0.492857
Dim Canvas1_1
Set Canvas1_1 = Canvas1_Component1.GetCanvas
Canvas1_1.Height = 140
Canvas1_1.Width = 140
Canvas1_Component1.SetPosition 280, 250, 442, 370
Canvas1_Component1.GetOutputPort(1).ConnectRelay(Canvas1_Component
1.GetInputPort(1))
Dim nPort
Set nPort = Canvas1_Component1.GetOutputPort(1).GetRelayConnection
Dim sName
sName = nPort.Name
MsgBox(sName)
```

CSDKOutputPort functionalities

RemoveMonitor()

Purpose: This method removes a monitor from the output port.

Return:

None.

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
Dim Canvas1_Component1
Set Canvas1_Component1 = Canvas1.CreateComponent("CW
Laser","{6DA31CEE-058F-11D4-93BD-0050DAB7C5D6}",180,210, 34, 34,0)
Canvas1_Component1.GetOutputPort(1).CreateMonitor
```

```
Canvas1_Component1.GetOutputPort(1).RemoveMonitor
```

DisconnectVisualizer(IDispatch* pVisualizerPort)

Purpose: This method disconnects a visualizer from the output port.

Arguments:

pVisualizerPort - A Visualizer port object.

Return:

None.

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
Dim Canvas1_Component1
Set Canvas1_Component1 = Canvas1.CreateComponent("CW
Laser","{6DA31CEE-058F-11D4-93BD-0050DAB7C5D6}",180,210, 34, 34,0)
Dim Canvas1_Component2
Set Canvas1_Component2 = Canvas1.CreateComponent("Optical Spectrum
Analyzer","{F11D0C25-3C7D-11D4-93F0-0050DAB7C5D6}",270,230, 40,
34,0)
Canvas1_Component2.Name  = "Optical Spectrum Analyzer"
Canvas1_Component1.GetOutputPort(1).CreateMonitor
Canvas1_Component1.GetOutputPort(1).ConnectVisualizer(Canvas1_Comp
onent2.GetInputPort(1))
Canvas1_Component1.GetOutputPort(1).DisconnectVisualizer(Canvas1_C
omponent2.GetInputPort(1))
```

ConnectVisualizer(IDispatch* pVisualizerPort)

Purpose: This method connects a visualizer to the output port.

Arguments:

pVisualizerPort - A Visualizer port object t.

Return:

None.

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
Dim Canvas1_Component1
Set Canvas1_Component1 = Canvas1.CreateComponent("CW
Laser","{6DA31CEE-058F-11D4-93BD-0050DAB7C5D6}",180,210, 34, 34,0)
Dim Canvas1_Component2
```


```
Set Canvas1_Component2 = Canvas1.CreateComponent("Optical Spectrum
Analyzer","{F11D0C25-3C7D-11D4-93F0-0050DAB7C5D6}",270,230, 40,
34,0)
Canvas1_Component2.Name  = "Optical Spectrum Analyzer"
Canvas1_Component1.GetOutputPort(1).CreateMonitor
Canvas1_Component1.GetOutputPort(1).ConnectVisualizer(Canvas1_Comp
onent2.GetInputPort(1))
```

`IsAnyVisualizerConnected()`

Purpose: This method indicates whether any visualisers are connected to the output port.

Return:

BOOL.

TRUE if any visualisers are connected to the output port, FALSE otherwise.

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
Dim Canvas1_Component1
Set Canvas1_Component1 = Canvas1.CreateComponent("CW
Laser","{6DA31CEE-058F-11D4-93BD-0050DAB7C5D6}",180,210, 34, 34,0)
Dim Canvas1_Component2
Set Canvas1_Component2 = Canvas1.CreateComponent("Optical Spectrum
Analyzer","{F11D0C25-3C7D-11D4-93F0-0050DAB7C5D6}",270,230, 40,
34,0)
Canvas1_Component2.Name  = "Optical Spectrum Analyzer"
Canvas1_Component1.GetOutputPort(1).CreateMonitor
Canvas1_Component1.GetOutputPort(1).ConnectVisualizer(Canvas1_Comp
onent2.GetInputPort(1))
Dim bV
bV = Canvas1_Component1.GetOutputPort(1).IsAnyVisualizerConnected
MsgBox(bV)
```

`CreateMonitor()`

Purpose: This method attaches a monitor to the output port.

Return:

None.

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
Dim Canvas1_Component1
```

```
Set Canvas1_Component1 = Canvas1.CreateComponent("CW
Laser","{6DA31CEE-058F-11D4-93BD-0050DAB7C5D6}",180,210, 34, 34,0)
Canvas1_Component1.GetOutputPort(1).CreateMonitor
```

HasMonitor()

Purpose: This method gets a value indicating whether the output port has a monitor attached.

Return:

BOOL

TRUE if the output port has a monitor attached to it, FALSE otherwise.

Example:

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
Dim Canvas1_Component1
Set Canvas1_Component1 = Canvas1.CreateComponent("CW
Laser","{6DA31CEE-058F-11D4-93BD-0050DAB7C5D6}",180,210, 34, 34,0)
Canvas1_Component1.GetOutputPort(1).CreateMonitor
Dim bMonitor
bMonitor =  Canvas1_Component1.GetOutputPort(1).HasMonitor
MsgBox(bMonitor)
```

CSDKExtObject functionalities

get_Name()

Purpose: This method returns the name of this object.

Return:

String.

The name of the object as a string.

put_Name(BSTR pName)

Purpose: This method sets the name of this object.

Arguments:

pName - A null terminating string specifying the name of the component.

Return:

None.

GetIteration()

Purpose: This method returns the current iteration in the document at this time.

Return:

Long.

The current sweep iteration as a long.

GetMaxIterations()

Purpose: This method returns the maximum number of iterations the document supports at this time.

Return:

Long.

The max number of sweep iterations as a long.

GetSID()

Purpose: This method returns the system ID (SID) of this object.

Remarks: This value uniquely identifies this object in the document.

Return:

Unsigned long.

The SID of the object as an unsigned long.

SDKDefaultSignalLibrary

SDKBinarySignal

GetSignalName()
Purpose: This method returns the Signal name (“Binary Signal”)
Return: The name of the Signal as a String.
Example:

```
Dim pSignal
Set pSignal = Canvas1_Component1.GetSignalByPortName("Bit Sequence", 0)
sName = pSignal.GetSignalName
```

GetSignalID()
Purpose: This method returns the Signal ID
Return: The Signal ID as a Long.

GetTimeStamp()
Purpose: This method returns the Signal Time Stamp.
Return: the Signal Time Stamp as a Long.

GetSize()
Purpose: This method returns the size of the Signal Bit Array.
Return: The size of the Signal Bit array as a Long.
Example:

```
Dim pSignal
Set pSignal = Canvas1_Component1.GetSignalByPortName("Bit Sequence", 0)
nSize = pSignal.GetSize
```

IsNull()
Purpose: This method determines whether the Signal is NULL or not.

```
Return: BOOL.
Example:
     bNULL = pSignal.IsNull

IsValid(BOOL* bRet)
Purpose: This method determines whether the Signal is Valid or not.
Return: BOOL.
Example:
     bNULL = pSignal. IsValid

RemoveAll()
Purpose: This method remove all elements from the Signal Bit array
and Call Initialize.

SetValue(double newVal)
Purpose: This method sets new value to each element of the Signal
Bit array.
Arguments::
newVal – Specifies the new value.
GetBitRate()
Purpose: This method retruns the Signal bit rate
Example:
     dBitRate = pSignal.GetBitRate

SetBitRate(double dBitRate)
Purpose: This method sets value for the Signal bit rate.
Arguments:
dBitRate – specifies the new Signal bit rate.
Example:
     pSignal.SetBitRate dBitRate*2

CopyFrom(VARIANT arrData);
Purpose: This method copy the elements of arrData to the elements of
Signal Bit array.
Arguments:
arrData – A VARIANT_ARRAY
GetBits()
Purpose: This method returns a Signal Bit array elements.
Return: The array of Signal Bit elements. The SAFEARRAY part of thes
VARIANT is used here.
Example:
     XData = pSignal.GetBits
     MyCheck = IsArray(XData)
     sizeL = LBound( XData )
     sizeU = UBound( XData )
     For i = sizeL to sizeU
        Excel.ActiveWorkbook.Worksheets("sheet1").Cells(i+1,1) =
XData(i)
     Next
```


SDKElectricalSignal

GetSignalName()
Purpose: This method returns the Signal name (“Electrical Signal”)
Return: The name of the Signal as a String.
Example:

```
Dim pSignalE
Set pSignalE = Canvas1_Component3.GetSignalByPortName("Output",
0)
sNameE = pSignalE.GetSignalName
```

GetSignalID()
Purpose: This method returns the Signal ID
Return: The Signal ID as a Long.

GetTimeStamp()
Purpose: This method returns the Signal Time Stamp.
Return: the Signal Time Stamp as a Long.

IsNull()
Purpose: This method determines whether the ElectricalSignal is NULL or not.
Return: BOOL.
Example:

```
bNULL = pSignal.IsNull
```

IsValid()
Purpose: This method determines whether the Signal is Valid or not.
Return: BOOL.
Example:

```
bNULL = pSignal. IsValid
```

RemoveAll()
Purpose: This method remove all elements from ElectricalSampledSignal, ElectricalIndividualSample and ElectricalSampledNoise.

SetValue(double newVal)
Purpose: This method sets new value to each element of the ElectricalSampledSignal, ElectricalIndividualSample and ElectricalSampledNoise.
Arguments:
newVal – Specifies the new value.

IsSampledNoise();
Purpose: This method determines whether the ElectricalSampledSignal (SampledNoise) is NULL or not.
Return: BOOL.

IsNoise();

Purpose: This method determines whether the ElectricalSampledNoise is NULL or not.
Return: BOOL.

IsSampledSignal()
Purpose: This method determines whether the ElectricalSampledSignal is NULL or not.
Return: BOOL.

IsIndividualSample()
Purpose: This method determines whether the ElectricalIndividualSample is NULL or not.
Return: BOOL.

GetFrequencyGridSpacing()
Purpose: This method returns the Electrical Signal Frequency Grid Spacing.
Return: Double
Example:

```
dFGridSpacing = pSignalE.GetFrequencyGridSpacing
```

SetFrequencyGridSpacing(double nGridSpacing)
Purpose: This method sets the Electrical Signal Frequency Grid Spacing.
Arguments:
nGridSpacing – Specifies frequency grid spacing
Example:

```
pSignalE.SetFrequencyGridSpacing(dFGridSpacing + 1.0)
```

GetSampledSignalSampleRate()
Purpose: This method returns the SElectricalSampledSignal SampleRate.
Return: Double
Example:

```
dSS_SampleRate = pSignalE.GetSampledSignalSampleRate
```

SetSampledSignalSampleRate(double dSampleRate)
Purpose: This method sets the SElectricalSampledSignal SampleRate.
Arguments:
dSampleRate – Specifies the SampleRate.
Example:

```
pSignalE.SetSampledSignalSampleRate( dSS_SampleRate + 1.0 )
```

GetSampledSignalAmplitude(VARIANT* arrAmplitudeReal, VARIANT* arrAmplitudeImg)
Purpose: This method returns arrays for SElectricalSampledSignal's Amplitude real and imaginary part.
Arguments:
arrAmplitudeReal – A VARIANT array SElectricalSampledSignal's Amplitude real part.

arrAmplitudeImg – A VARIANT array SElectricalSampledSignal's Amplitude imaginary part.

Example:

```
Call pSignalE2.GetSampledSignalAmplitude(XDataER2, XDataEI2)
sizeLE2 = LBound( XDataER2 )
sizeUE2 = UBound( XDataER2 )
For i = sizeLE2 to sizeUE2
    Excel.ActiveWorkbook.Worksheets("sheet1").Cells(i+1,2) = XDataER2(i)
Next
```

SetSampledSignalAmplitude(VARIANT arrAmplitudeReal,VARIANT arrAmplitudeImg)

Purpose: This method sets SElectricalSampledSignal's Amplitude real and imaginary part.

Arguments:

arrAmplitudeReal – A VARIANT array SElectricalSampledSignal's Amplitude real part.

arrAmplitudeImg – A VARIANT array SElectricalSampledSignal's Amplitude imaginary part.

GetIndividualSampleSampleRate()

Purpose: This method returns the SElectricalIndividualSample SampleRate.

Return: Double

Example:

```
dSIS_SampleRate = pSignalE.GetIndividualSampleSampleRate
```

SetIndividualSampleSampleRate(double dSampleRate)

Purpose: This method sets the SElectricalIndividualSample SampleRate.

Arguments:

dSampleRate – Specifies the SampleRate.

Example:

```
pSignalE.SetIndividualSampleSampleRate( dSIS_SampleRate + 1.0)
```

GetIndividualSampleAmplitude(VARIANT* pAmplitudeReal,VARIANT* pAmplitudeImg)

Purpose: This method returns SElectricalIndividualSample's Amplitude real and imaginary part.

Arguments:

arrAmplitudeReal – A VARIANT SElectricalIndividualSample's Amplitude real part.

arrAmplitudeImg – A VARIANT SElectricalIndividualSample's Amplitude imaginary part.

SetIndividualSampleAmplitude(double dAmplitudeReal, double dAmplitudeImg)

Purpose: This method sets SElectricalIndividualSample's Amplitude real and imaginary part.

Arguments:

dAmplitudeReal – A double SElectricalIndividualSample's Amplitude real part.

dAmplitudeImg – A double SElectricalIndividualSample's Amplitude imaginary part.

GetSampledNoiseSampleRate()
Purpose: This method returns the SElectricalSampledSignal SampleRate.
Return: Double
Example:

```
dSIS_SampleRate = pSignalE.GetSampledNoiseSampleRate
```

SetSampledNoiseSampleRate(double dSampleRate);
Purpose: This method sets the SElectricalSampledSignal SampleRate.
Arguments:
dSampleRate – Specifies the SampleRate.
Example:

```
pSignalE.SetSampleNoiseSampleRate( dSIS_SampleRate + 1.0)
```

GetSampledNoiseAmplitude(VARIANT* arrAmplitudeReal,VARIANT* arrAmplitudeImg)
Purpose: This method returns arrays for SElectricalSampledSignal's Amplitude real and imaginary part.
Arguments:
arrAmplitudeReal – A VARIANT array SElectricalSampledSignal's Amplitude real part.
arrAmplitudeImg – A VARIANT array SElectricalSampledSignal's Amplitude imaginary part.
Example:

```
Call pSignalE2.GetSampledNoiseAmplitude(XDataER2, XDataEI2)
sizeLE2 = LBound( XDataER2 )
sizeUE2 = UBound( XDataER2 )
For i = sizeLE2 to sizeUE2
    Excel.ActiveWorkbook.Worksheets("sheet1").Cells(i+1,2) = XDataER2(i)
Next
```

SetSampledNoiseAmplitude(VARIANT arrAmplitudeReal,VARIANT arrAmplitudeImg)
Purpose: This method sets SElectricalSampledSignal's Amplitude real and imaginary part.
Arguments:
arrAmplitudeReal – A VARIANT array SElectricalSampledSignal's Amplitude real part.
arrAmplitudeImg – A VARIANT array SElectricalSampledSignal's Amplitude imaginary part.

SDKMarySignal

GetSignalName()
Purpose: This method returns the Signal name ("Mary Signal")
Return: The name of the Signal as a String.
Example:

```
Dim pSignalM
Set pSignalM = Canvas1_Component1.GetSignalByPortName("PAM Sequence", 0)
sNameM = pSignalM.GetSignalName
```


GetSignalID()
Purpose: This method returns the Signal ID
Return: The Signal ID as a Long.

GetTimeStamp()
Purpose: This method returns the Signal Time Stamp.
Return: the Signal Time Stamp as a Long.

IsNull()
Purpose: This method determines whether the Signal is NULL or not.
Return: BOOL.
Example:

```
bNULL = pSignal.IsNull
```

IsValid()
Purpose: This method determines whether the Signal is Valid or not.
Return: BOOL.
Example:

```
bNULL = pSignal. IsValid
```

RemoveAll()
Purpose: This method remove all elements from the Signal Bit array.

SetValue(double newVal)
Purpose: This method sets new value to each element of the Signal Bit array.
Arguments::
newVal – Specifies the new value.

SetBitRate(double dBitRate);
CopyFrom(VARIANT arrData);
GetBits(VARIANT* arrBits);

GetBitRate()
Purpose: This method retruns the Signal bit rate.
Return: double
Example:

```
bBitRate = pSignalM.GetBitRate
```

SetBitRate(double dBitRate)
Purpose: This method sets value for the Signal bit rate.
Arguments:
dBitRate – Specifies the new Signal bit rate.
Example:

```
pSignal.SetBitRate dBitRate*2
```

CopyFrom(VARIANT arrData);
Purpose: This method copy the elements of arrData to the elements of Signal Bit array.
Arguments:

arrData – A VARIANT_ARRAY

GetBits()
Purpose: This method returns a Signal Bit array elements.
Return: The array of Signal Bit elements. The SAFEARRAY part of thes VARIANT is used here.
Example:

```
XData = pSignal.GetBits
MyCheck = IsArray(XData)
sizeL = LBound( XData )
sizeU = UBound( XData )
For i = sizeL to sizeU
    Excel.ActiveWorkbook.Worksheets("sheet1").Cells(i+1,1) = XData(i)
Next
```

SDKOpticalSignal

GetSignalName()
Purpose: This method returns the Signal name (“Optical Signal”)
Return: The name of the Signal as a String.
Example:

```
Dim pSignalO
Set pSignalO = Canvas1_Component5.GetSignalByPortName("Output", 0)
sNameO = pSignalO.GetSignalName
```

GetSignalID()
Purpose: This method returns the Signal ID
Return: The Signal ID as a Long.

GetTimeStamp()
Purpose: This method returns the Signal Time Stamp.
Return: the Signal Time Stamp as a Long.

IsNull()
Purpose: This method determines whether the Optical Signal is NULL or not.
Return: BOOL.
Example:

```
bNULL = pSignal.IsNull
```

IsValid()
Purpose: This method determines whether the Signal is Valid or not.
Return: BOOL.
Example:

```
bNULL = pSignal. IsValid
```

RemoveAll()
Purpose: This method remove all elements from OpticalNoiseBins, OpticalParameterizedSignals, OpticalSampledSignals and OpticalIndividualSamples.

SetValue(double newVal)

Purpose: This method sets new value to each element of OpticalNoiseBins, OpticalParameterizedSignals, OpticalSampledSignals, and OpticalIndividualSamples arrays.
Arguments:
newVal – Specifies the new value.

IsParameterized()
Purpose: This method determines whether the OpticalParameterizedSignals array of data is NULL or not.
Return: BOOL.

IsNoiseBins()
Purpose: This method determines whether the OpticalNoiseBins array of data is NULL or not.
Return: BOOL.

IsSampledSignals()
Purpose: This method determines whether the OpticalSampledSignals array of data is NULL or not.
Return: BOOL.

IsIndividualSamples()
Purpose: This method determines whether the OpticalIndividualSamples array of data is NULL or not.
Return: BOOL.

GetChannels(VARIANT* arrChanels)
Purpose: This method returns array of frequency channels.
Return: A VARIANT array.

GetFrequencyGridSpacing()
Purpose: This method returns a Frequency Grid Spacing.
Return: Double

GetGridSpacingX()
Purpose: This method returns Optical Signal Grid Spacing X.
Return: Double

GetGridSpacingY()
Purpose: This method returns Optical Signal Grid Spacing Y.
Return: Double

GetNumberOfNoiseBins()
Purpose: This method returns the size of OpticalNoiseBinsArray.
Return: Long.
Example:

```
nNumNB = pSignalO.GetNumberOfNoiseBins
```

GetNumberOfParameterizedSignals()
Purpose: This method returns the size of OpticalParameterizedSignalsArray.

Return: Long.
Example:

```
nNumParSig = pSignalO.GetNumberOfParameterizedSignals
```

GetNumberOfSampledSignals()
Purpose: This method returns the size of COpticalSampledSignalsArray.
Return: Long.
Example:

```
nNumSampleSig = pSignalO.GetNumberOfSampledSignals
```

GetNumberOfIndividualSamples()
Purpose: This method returns the size of COpticalIndividualSamplesArray.
Return: Long.
Example:

```
nNumIndSig = pSignalO.GetNumberOfIndividualSamples
```

GetNoiseBinSignalBandwidth(long nIndex,VARIANT* pLowerFrequency,VARIANT* pUpperFrequency)
Purpose: This method returns OpticalNoiseBin Bandwidth Lower and Upper Frequency for specified index of OpticalNoiseBinsArray.
Arguments:
nIndex – Specifies the index of OpticalNoiseBinsArray. nIndex is zero based.
pLowerFrequency – A VARIANT OpticalNoiseBins Bandwidth Lower Frequency.
pUpperFrequency – A VARIANT OpticalNoiseBins Bandwidth Upper Frequency.
Example:

```
nNumNB = pSignalO.GetNumberOfNoiseBins
Call pSignalO.GetNoiseBinSignalBandwidth( nNumNB - 1, nLF0, nUF0 )
```

GetNoiseBinOpticalSignalPolarizationPower(long nIndex)
Purpose:This method returns OpticalNoiseBin Polarization Power for specified index of OpticalNoiseBinsArray.
Arguments:
nIndex – Specifies the index of OpticalNoiseBinsArray. Index is zero based.
Return: Double.
Example:

```
nPP = pSignalO.GetNoiseBinOpticalSignalPolarizationPower(0)
```

GetNoiseBinOpticalSignalPolarizationSplittingRatio(long nIndex)
Purpose:This method returns OpticalNoiseBin Polarization SplittingRatio for specified index of OpticalNoiseBinsArray.
Arguments:
nIndex – Specifies the index of OpticalNoiseBinsArray. nIndex is zero based.
Return: Double.
Example:

```
nPSR = pSignalO.GetNoiseBinOpticalSignalPolarizationSplittingRatio(0)
```


GetNoiseBinOpticalSignalPolarizationPhaseDifference(long nIndex)
Purpose:This method returns OpticalNoiseBin Polarization PhaseDifference for specified index of OpticalNoiseBinsArray.
Arguments:
nIndex – Specifies the index of OpticalNoiseBinsArray. nIndex is zero based.
Return: Double.
Example:

```
nPPD = pSignalO.GetNoiseBinOpticalSignalPolarizationPhaseDifference(0)
```

GetNoiseBinOpticalSignalPolarization(long nIndex,VARIANT* pS0, VARIANT* pS1, VARIANT* pS2, VARIANT* pS3)
Purpose:This method returns OpticalNoiseBin Polarization Stokes vector parameters for specified index of OpticalNoiseBinsArray.
Arguments:
nIndex – Specifies the index of OpticalNoiseBinsArray. Index is zero based.
pS0 – A VARIANT OpticalNoiseBin Polarization Stokes vector parameter.
pS1 – A VARIANT OpticalNoiseBin Polarization Stokes vector parameter.
pS2 – A VARIANT OpticalNoiseBin Polarization Stokes vector parameter.
pS3 – A VARIANT OpticalNoiseBin Polarization Stokes vector parameter.
Example:

```
Call pSignalO.GetNoiseBinOpticalSignalPolarization(0, s0, s1, s2, s3)
```

GetParameterizedSignalChannelID(long nIndex)
Purpose:This method returns OpticalParameterizedSignal ChannelID for specified index of OpticalParameterizedSignalsArray.
Arguments:
nIndex – Specifies the index of OpticalParameterizedSignalsArray. nIndex is zero based.
Return: Double.
Example:

```
nChanelID = pSignalO.GetParameterizedSignalChannelID(0)
```

GetParameterizedSignalSignalBandwidth(long nIndex,VARIANT* pLowerFrequency, VARIANT* pUpperFrequency)
Purpose: This method returns OpticalParameterizedSignal Bandwidth Lower and Upper Frequency for specified index of OpticalParameterizedSignalsArray.
Arguments:
nIndex – Specifies the index of OpticalParameterizedSignalsArray. nIndex is zero based.
pLowerFrequency – A VARIANT OpticalParameterizedSignal Bandwidth Lower Frequency.
pUpperFrequency – A VARIANT OpticalParameterizedSignal Bandwidth Upper Frequency.
Example:

```
nNumParSig = pSignalO.GetNumberOfParameterizedSignals
Call pSignalO.GetParameterizedSignalSignalBandwidth(nNumParSig 1,nLF0,nUF0)
```

GetParameterizedSignalOpticalSignalPolarizationPower(long nIndex)
Purpose:This method returns OpticalParameterizedSignal Polarization Power for specified index of OpticalParameterizedSignalsArray.
Arguments:
nIndex – Specifies the index of OpticalParameterizedSignalsArray. nIndex is zero based.
Return: Double.
Example:

```
dPolPower = pSignalO.GetParameterizedSignalOpticalSignalPolarizationPower( 0 )
```

GetParameterizedSignalOpticalSignalPolarizationSplittingRatio(long nIndex)
Purpose:This method returns OpticalParameterizedSignal Polarization SplittingRatio for specified index of OpticalParameterizedSignalsArray.
Arguments:
nIndex – A long index of OpticalParameterizedSignalsArray. nIndex is zero based.
Return: Double.
Example:

```
nSpRat = pSignalO.GetParameterizedSignalOpticalSignalPolarizationSplittingRatio(0)
```

GetParameterizedSignalOpticalSignalPolarizationPhaseDifference(lon g nIndex)
Purpose:This method returns OpticalParameterizedSignal Polarization PhaseDifference for specified index of OpticalParameterizedSignalsArray.
Arguments:
nIndex – Specifies the index of OpticalParameterizedSignalsArray. nIndex is zero based.
Return: Double.
Example:

```
nPhDiff = pSignalO.GetParameterizedSignalOpticalSignalPolarizationPhaseDifference(0)
```

GetParameterizedSignalOpticalSignalPolarization(long nIndex,VARIANT* pS0, VARIANT* pS1, VARIANT* pS2, VARIANT* pS3)
Purpose:This method returns OpticalParameterizedSignal Polarization Stokes vector parameters for specified index of OpticalParameterizedSignalsArray.
Arguments:
nIndex – A long index of OpticalParameterizedSignalsArray. nIndex is zero based.
pS0 – A VARIANT OpticalNoiseBin Polarization Stokes vector parameter.
pS1 – A VARIANT OpticalNoiseBin Polarization Stokes vector parameter.
pS2 – A VARIANT OpticalNoiseBin Polarization Stokes vector parameter.
pS3 – A VARIANT OpticalNoiseBin Polarization Stokes vector parameter.
Example:

```
Call pSignalO.GetParameterizedSignalOpticalSignalPolarization(0, ps0, ps1, ps2, ps3)
```


GetParameterizedSignalStatisticalSignalParameter(long nIndex,BSTR pName, VARIANT* pValue)

Purpose:This method returns OpticalParameterizedSignal Parameter for specified index of OpticalParameterizedSignalsArray based on Parameter name.

Arguments:

nIndex – Specifies the index of OpticalParameterizedSignalsArray. nIndex is zero based.

pName – Specifies the parameter name.

Note: Valid parameter names:

"BitRate"
"ModulationType"
"FECT"
"PulseWidth"
"ExtinctionRatio"
"ChirpPeakValue"
"ChirpDuration"
"Dispersion"
"TimeDelay"
"Linewidth"
"PMD"
"RIN"

Return: VARIANT the parameter value.

Example:

```
      nParBitRate = pSignalO.GetParameterizedSignalStatisticalSignalParameter(0,
"BitRate")
      nMT = pSignalO.GetParameterizedSignalStatisticalSignalParameter(0,
"ModulationType")
      nFECT = pSignalO.GetParameterizedSignalStatisticalSignalParameter(0, "FECT")
      nER = pSignalO.GetParameterizedSignalStatisticalSignalParameter(0,
"ExtinctionRatio")
      nCPV = pSignalO.GetParameterizedSignalStatisticalSignalParameter(0,
"ChirpPeakValue")
      nCD = pSignalO.GetParameterizedSignalStatisticalSignalParameter(0,
"ChirpDuration")
      nDisp = pSignalO.GetParameterizedSignalStatisticalSignalParameter(0,
"Dispersion")
      nTDelay = pSignalO.GetParameterizedSignalStatisticalSignalParameter(0,
"TimeDelay")
      nL = pSignalO.GetParameterizedSignalStatisticalSignalParameter(0, "Linewidth")
      nPMD = pSignalO.GetParameterizedSignalStatisticalSignalParameter(0, "PMD")
      nRIN = pSignalO.GetParameterizedSignalStatisticalSignalParameter(0, "RIN")
```

GetParameterizedSignalCommonSignalChannelsArraySize(long nIndex, long* nSize)

Purpose:This method returns OpticalParameterizedSignal Common Channels array size for specified index of OpticalParameterizedSignalsArray.

Arguments:

nIndex – Specifies the index of OpticalParameterizedSignalsArray. nIndex is zero based.

Return: Long.

Example:

```
      nNPS = pSignalO.GetNumberOfParameterizedSignals
      nAS = pSignalO.GetParameterizedSignalCommonSignalChannelsArraySize(nNPS - 1)
```

GetParameterizedSignalCommonSignalChannelChannelID(long nIndex, long nIndex2)

Purpose:This method returns OpticalParameterizedSignal Common Channels ChannelID for specified nIndex2 of CommonChannelsArray and for specified nIndex of OpticalParameterizedSignalsArray.

Arguments:

nIndex – Specifies the index of OpticalParameterizedSignalsArray. nIndex is zero based.

nIndex2 – Specifies the index of CommonChannelsArray. nIdex2 is zero based.

Return: Long.

Example:

```
nNPS = pSignalO.GetNumberOfParameterizedSignals
nAS = pSignalO.GetParameterizedSignalCommonSignalChannelsArraySize(nNPS - 1)
nChID = pSignalO.GetParameterizedSignalCommonSignalChannelChannelID(nNPS - 1, nAS - 1)
```

GetParameterizedSignalCommonSignalChannelPower(long nIndex, long nIndex2)

Purpose:This method returns OpticalParameterizedSignal Common Channels Power for specified nIndex2 of CommonChannelsArray and for specified nIndex of OpticalParameterizedSignalsArray.

Arguments:

nIndex – Specifies the index of OpticalParameterizedSignalsArray. nIndex is zero based.

nIndex2 – Specifies the index of CommonChannelsArray. nIdex2 is zero based.

Return: Double.

Example:

```
nNPS = pSignalO.GetNumberOfParameterizedSignals
nAS = pSignalO.GetParameterizedSignalCommonSignalChannelsArraySize(nNPS - 1)
dChPower = pSignalO.GetParameterizedSignalCommonSignalChannelPower(nNPS - 1, nAS - 1)
```

GetIndividualSampleSignalBandwidth(long nIndex,VARIANT* pLowerFrequency, VARIANT* pUpperFrequency)

Purpose: This method returns OpticalIndividualSample Bandwidth Lower and Upper Frequency for specified index of OpticalIndividualSamplesArray.

Arguments:

nIndex – Specifies the index of OpticalIndividualSamplesArray. nIndex is zero based.

pLowerFrequency – A VARIANT OpticalIndividualSample Bandwidth Lower Frequency.

pUpperFrequency – A VARIANT OpticalIndividualSample Bandwidth Upper Frequency.

Example:

```
nNIS = pSignalO.GetNumberOfIndividualSamples
Call pSignalO.GetIndividualSampleSignalBandwidth(nNIS - 1 , nLF0, nUF0)
```

GetIndividualSampleAmplitudeX(long nIndex,VARIANT *AmplitudeXReal, VARIANT *AmplitudeXImg)

Purpose: This method returns OpticalIndividualSample’s AmplitudeX real and imaginary part for specified index of OpticalIndividualSamplesArray.

Arguments:
nIndex, – A long index of OpticalIndividualSamplesArray. nIndex is zero based.
AmplitudeXReal – A VARIANT OpticalIndividualSample's AmplitudeX real part.
AmplitudeXImg – A VARIANT OpticalIndividualSample's AmplitudeX imaginary part.
Example:

```
Call pSignalO.GetIndividualSampleAmplitudeX( 0, a, b)
```

GetIndividualSampleAmplitudeY(long nIndex,VARIANT *AmplitudeYReal,/VARIANT *AmplitudeYImg);
Purpose: This method returns OpticalIndividualSample's AmplitudeY real and imaginary part for specified index of OpticalIndividualSamplesArray.

Arguments:
nIndex – Specifies the index of OpticalIndividualSamplesArray. nIndex is zero based.
AmplitudeXReal – A VARIANT OpticalIndividualSample's AmplitudeY real part.
AmplitudeXImg – A VARIANT OpticalIndividualSample's AmplitudeY imaginary part.
Example:

```
Call pSignalO.GetIndividualSampleAmplitudeY( 0, c, d)
```

GetSampledSignalSignalBandwidth(long nIndex,VARIANT* pLowerFrequency, VARIANT* pUpperFrequency)
Purpose: This method returns OpticalSampledSignal Bandwidth Lower and Upper Frequency for specified index of OpticalSampledSignalsArray.
Arguments:
nIndex – Specifies the index of OpticalSampledSignalsArray. Index is zero based.
pLowerFrequency – A VARIANT OpticalSampledSignal Bandwidth Lower Frequency.
pUpperFrequency – A VARIANT OpticalSampledSignal Bandwidth Upper Frequency.

GetSampledSignalOpticalSignalPolarizationPower(long nIndex)
Purpose:This method returns OpticalSampledSignal Polarization Power for specified index of OpticalSampledSignalsArray.
Arguments:
nIndex – Specifies the index of OpticalSampledSignalsArray. nIndex is zero based.
Return: Double.
Example:

```
dPolPower = pSignalO.GetSampledSignalOpticalSignalPolarizationPower( 0 )
```

GetSampledSignalOpticalSignalPolarizationSplittingRatio(long nIndex)

Purpose:This method returns OpticalSampledSignal Polarization Splitting Ratio for specified index of OpticalSampledSignalsArray.
Arguments:
nIndex – Specifies the index of OpticalSampledSignalsArray. nIndex is zero based.
Return: Double.
Example:

```
nSpRat = pSignalO.GetSampledSignalOpticalSignalPolarizationSplittingRatio(0)
```

GetSampledSignalOpticalSignalPolarizationPhaseDifference(long nIndex)
Purpose:This method returns OpticalSampledSignal Polarization Phase Difference for specified index of OpticalSampledSignalsArray.
Arguments:
nIndex – Specifies the index of OpticalSampledSignalsArray. nIndex is zero based.
Return: Double.

Example:

```
nPhDiff = pSignalO.GetSampledSignalOpticalSignalPolarizationPhaseDifference(0)
```

GetSampledSignalOpticalSignalPolarization(long nIndex, VARIANT* pS0, VARIANT* pS1, VARIANT* pS2, VARIANT* pS3)
Purpose:This method returns OpticalSampledSignal Polarization Stokes vector parameters for specified nIndex of OpticalSampledSignalsArray.
Arguments:
nIndex – A long index of OpticalParameterizedSignalsArray. nIndex is zero based.
pS0 – A VARIANT OpticalSampledSignal Polarization Stokes vector parameter.
pS1 – A VARIANT OpticalSampledSignal Polarization Stokes vector parameter.
pS2 – A VARIANT OpticalSampledSignal Polarization Stokes vector parameter.
pS3 – A VARIANT OpticalSampledSignal Polarization Stokes vector parameter.
Example:

```
Call pSignalO.GetSampledSignalOpticalSignalPolarization(0, ps0, ps1, ps2, ps3)
```

GetSampledSignalArrAmplitudeX(long nIndex,VARIANT *arrAmplitudeXReal, VARIANT *arrAmplitudeXImg)
Purpose: This method returns arrays for OpticalSampledSignal's AmplitudeX real and imaginary part for specified nIndex of OpticalSampledSignalsArray.
Arguments:
nIndex – Specifies the index of OpticalParameterizedSignalsArray. nIndex is zero based.
arrAmplitudeReal – A VARIANT array OpticalSampledSignal's AmplitudeX real part.
arrAmplitudeImg – A VARIANT array OpticalSampledSignal's AmplitudeX imaginary part.
Example:

```
nNSS = pSignalO.GetNumberOfSampledSignals
```


```
Call pSignalO.GetSampledSignalArrAmplitudeX(nNSS - 1, dataR, dataI)
sizeLM = LBound( dataR )
sizeUM = UBound( dataR )
For i = sizeLM to sizeUM
    Excel.ActiveWorkbook.Worksheets("sheet1").Cells(i+1,1) = dataR(i)
    Excel.ActiveWorkbook.Worksheets("sheet1").Cells(i+1,2) = dataI(i)
Next
```

GetSampledSignalArrAmplitudeY(long nIndex,VARIANT *arrAmplitudeYReal, VARIANT *arrAmplitudeYImg)

Purpose: This method returns arrays for OpticalSampledSignal's AmplitudeY real and imaginary part for specified nIndex of OpticalSampledSignalsArray.

Arguments:

nIndex – Specifies the index of OpticalParameterizedSignalsArray. nIndex is zero based.

arrAmplitudeReal – A VARIANT array OpticalSampledSignal's AmplitudeY real part.

arrAmplitudeImg – A VARIANT array OpticalSampledSignal's AmplitudeY imaginary part.

Example:

```
nNSS = pSignalO.GetNumberOfSampledSignals
Call pSignalO.GetSampledSignalArrAmplitudeY(nNSS - 1, dataR, dataI)
sizeLM = LBound( dataR )
sizeUM = UBound( dataR )
For i = sizeLM to sizeUM
    Excel.ActiveWorkbook.Worksheets("sheet1").Cells(i+1,1) = dataR(i)
    Excel.ActiveWorkbook.Worksheets("sheet1").Cells(i+1,2) = dataI(i)
Next
```

GetSampledSignalArrModeX(long nIndex,VARIANT *arrModeXReal, VARIANT *arrModeXImg)

Purpose: This method returns arrays for OpticalSampledSignal's ModeX real and imaginary part for specified nIndex of OpticalSampledSignalsArray.

Arguments:

nIndex – Specifies the index of OpticalParameterizedSignalsArray. nIndex is zero based.

arrModeXReal – A VARIANT array OpticalSampledSignal's ModeX real part.

arrModeXImg – A VARIANT array OpticalSampledSignal's ModeX imaginary part.

GetSampledSignalArrModeY(long nIndex,VARIANT *arrModeYReal, VARIANT *arrModeYImg)

Purpose: This method returns arrays for OpticalSampledSignal's ModeX real and imaginary part for specified nIndex of OpticalSampledSignalsArray.

Arguments:

nIndex – Specifies the index of OpticalParameterizedSignalsArray. nIndex is zero based.

arrModeXReal – A VARIANT array OpticalSampledSignal's ModeX real part.

arrModeXImg – A VARIANT array OpticalSampledSignal's ModeX imaginary part.

GetSampledSignalTransverseModeMeshX(long nIndex, VARIANT * varSizeX, VARIANT * varSizeY)
Purpose: This method returns OpticalSampledSignal's MeshX sizeX and sizeY for specified nIndex of OpticalSampledSignalsArray.
Arguments:
nIndex – Specifies the index of OpticalParameterizedSignalsArray. nIndex is zero based.
varSizeX – A VARIANT OpticalSampledSignal's MeshX sizeX.
varSizeY – A VARIANT OpticalSampledSignal's MeshX sizeY.

GetSampledSignalTransverseModeMeshY(long nIndex,VARIANT * varSizeX, VARIANT * varSizeY)
Purpose: This method returns OpticalSampledSignal's MeshY sizeX and sizeY for specified nIndex of OpticalSampledSignalsArray.
Arguments:
nIndex – Specifies the index of OpticalParameterizedSignalsArray. nIndex is zero based.
varSizeX – A VARIANT OpticalSampledSignal's MeshY sizeX.
varSizeY – A VARIANT OpticalSampledSignal's MeshY sizeY.

GetSampledSignalModePropertiesX(long nIndex)
Purpose:This method returns OpticalSampledSignal ModePropertiesX for specified index of OpticalSampledSignalsArray.
Arguments:
nIndex – Specifies the index of OpticalSampledSignalsArray. nIndex is zero based.
Return: String.
Example:

```
nSpMP = pSignalO.GetSampledSignalModePropertiesX(0)
```

GetSampledSignalModePropertiesY(long nIndex)
Purpose:This method returns OpticalSampledSignal ModePropertiesY for specified index of OpticalSampledSignalsArray.
Arguments:
nIndex – Specifies the index of OpticalSampledSignalsArray. nIndex is zero based.
Return: String.
Example:

```
nSpMP = pSignalO.GetSampledSignalModePropertiesY(0)
```

SetFrequencyGridSpacing(double dGridSpacing)
Purpose: This method sets Optical Signal Frequency Grid Spacing.
Arguments:
dGridSpacing – Specifies Frequency Grid Spacing.
SetGridSpacingX(double dGridSpacingX)
Purpose: This method sets Optical Signal Grid SpacingX.
Arguments:
dGridSpacingX – Specifies Frequency Grid SpacingX.

SetGridSpacingY(double dGridSpacingY)
Purpose: This method sets Optical Signal Grid SpacingY.
Arguments:

dGridSpacingX – Specifies Frequency Grid SpacingY.

SetNoiseBinSignalBandwidth(long nIndex,double dLowerFrequency, double dUpperFrequency)

Purpose: This method sets OpticalNoiseBin Bandwidth Lower and Upper Frequency for specified index of OpticalNoiseBinsArray.

Arguments:

nIndex – Specifies the index of OpticalNoiseBinsArray. nIndex is zero based.

pLowerFrequency – Specifies OpticalNoiseBins Bandwidth Lower Frequency.

pUpperFrequency – Specifies OpticalNoiseBins Bandwidth Upper Frequency.

Example:

```
nNumNB = pSignalO.GetNumberOfNoiseBins
Call pSignalO.GetNoiseBinSignalBandwidth( nNumNB - 1, nLF0, nUF0 )
Call pSignalO.SetNoiseBinSignalBandwidth( nNumNB - 1, nLF0*5, nUF0*5 )
```

SetNoiseBinOpticalSignalPolarizationParameters(long nIndex, double dS0, double dS1, double dS2, double dS3)

Purpose:This method sets OpticalNoiseBin Polarization Stokes vector parameters for specified index of OpticalNoiseBinsArray.

Arguments:

nIndex – Specifies the index of OpticalNoiseBinsArray. Index is zero based.

pS0 – Specifies OpticalNoiseBin Polarization Stokes vector parameter.

pS1 – Specifies OpticalNoiseBin Polarization Stokes vector parameter.

pS2 – Specifies OpticalNoiseBin Polarization Stokes vector parameter.

pS3 – Specifies OpticalNoiseBin Polarization Stokes vector parameter.

Example:

```
Call pSignalO.GetNoiseBinOpticalSignalPolarization(0, s0, s1, s2, s3)
Call pSignalO.SetNoiseBinOpticalSignalPolarizationParameters(0, s0+ 5, s1 + 5, s2
+ 5, s3 + 5)
```

SetNoiseBinOpticalSignalPolarization(long nIndex,double dPower, double dSplittingRatio, double dPhaseDifference)

Purpose:This method sets OpticalNoiseBin Polarization parameters for specified index of OpticalNoiseBinsArray.

Arguments:

nIndex – Specifies the index of OpticalNoiseBinsArray. Index is zero based.

dPower – Specifies OpticalNoiseBin Polarization Power.

dSplittingRatio – Specifies OpticalNoiseBin Polarization Splitting Ratio.

dPhaseDifference – Specifies OpticalNoiseBin Polarization Phase Difference.

Example:

```
nNumNB = pSignalO.GetNumberOfNoiseBins
nPP = pSignalO.GetNoiseBinOpticalSignalPolarizationPower(0)
nPSR = pSignalO.GetNoiseBinOpticalSignalPolarizationSplittingRatio(0)
nPPD = pSignalO.GetNoiseBinOpticalSignalPolarizationPhaseDifference(0)
```

```
     Call pSignalO.SetNoiseBinOpticalSignalPolarization(nNumNB - 1, nPP*5, nPSR*5,
nPPD)
```

SetParameterizedSignalChannelID(long nIndex, long nChannelID)
Purpose:This method sets OpticalParameterizedSignal ChannelID for specified index of OpticalParameterizedSignalsArray.
Arguments:
nIndex – Specifies the index of OpticalParameterizedSignalsArray. nIndex is zero based.
nChannelID – Specifies OpticalParameterizedSignal ChannelID.

SetParameterizedSignalSignalBandwidth(long nIndex, double dLowerFrequency, double dUpperFrequency)
Purpose: This method sets OpticalParameterizedSignal Bandwidth Lower and Upper Frequency for specified index of OpticalParameterizedSignalsArray.
Arguments:
nIndex – Specifies the index of OpticalParameterizedSignalsArray. nIndex is zero based.
pLowerFrequency – Specifies OpticalParameterizedSignal Bandwidth Lower Frequency.
pUpperFrequency – Specifies OpticalParameterizedSignal Bandwidth Upper Frequency.
Example:

```
     nNumParSig = pSignalO.GetNumberOfParameterizedSignals
     Call pSignalO.GetParameterizedSignalSignalBandwidth( nNumParSig - 1 , nLF0, nUF0)
     Call pSignalO.GetParameterizedSignalSignalBandwidth( nNumParSig - 1 , nLF0*5,
nUF0*5)
```

SetParameterizedSignalOpticalSignalPolarizationParameters(long nIndex,double dS0, double dS1, double dS2, double dS3)
Purpose:This method sets OpticalParameterizedSignal Polarization Stokes vector parameters for specified index of OpticalParameterizedSignalsArray.
Arguments:
nIndex – Specifies the index of OpticalParameterizedSignalsArray. Index is zero based.
pS0 – Specifies OpticalParameterizedSignal Polarization Stokes vector parameter.
pS1 – Specifies OpticalParameterizedSignal Polarization Stokes vector parameter.
pS2 – Specifies OpticalParameterizedSignal Polarization Stokes vector parameter.
pS3 – Specifies OpticalParameterizedSignal Polarization Stokes vector parameter.
Example:

```
     Call pSignalO. GetParameterizedSignalOpticalSignalPolarization(0, s0, s1, s2, s3)
     Call pSignalO.SetParameterizedSignalOpticalSignalPolarizationParameters(0, s0+
5, s1 + 5, s2 + 5, s3 + 5)
```

SetParameterizedSignalOpticalSignalPolarization(long nIndex,double dPower, double dSplittingRatio, double dPhaseDifference)
Purpose:This method sets OpticalParameterizedSignal Polarization parameters for specified index of OpticalParameterizedSignalsArray.
Arguments:

nIndex – Specifies the index of OpticalParameterizedSignalsArray. Index is zero based.
dPower – Specifies OpticalParameterizedSignal Polarization Power.
dSplittingRatio – Specifies OpticalParameterizedSignal Polarization Splitting Ratio.
dPhaseDifference – Specifies OpticalParameterizedSignal Polarization Phase

SetParameterizedSignalStatisticalSignalParameter(long nIndex,BSTR pName, VARIANT pValue)
Purpose:This method sets OpticalParameterizedSignal Parameter for specified index of OpticalParameterizedSignalsArray based on Parameter name.
Arguments:
nIndex – Specifies index of OpticalParameterizedSignalsArray. nIndex is zero based.
pName – Specifies the parameter name.
Note: Valid parameter names:
"BitRate"
"ModulationType"
"FECT"
"PulseWidth"
"ExtinctionRatio"
"ChirpPeakValue"
"ChirpDuration"
"Dispersion"
"TimeDelay"
"Linewidth"
"PMD"
"RIN"
pValue - Specifies the parameter value.
Example:

```
     nParBitRate = pSignalO.GetParameterizedSignalStatisticalSignalParameter(0,
     "BitRate")
     Call pSignalO.SetParameterizedSignalStatisticalSignalParameter(0, "BitRate",
     nParBitRate + 1.0)
     nModulationType = pSignalO.GetParameterizedSignalStatisticalSignalParameter(0,
     "ModulationType")
     Call pSignalO.SetParameterizedSignalStatisticalSignalParameter(0,
     "ModulationType", "NRZ")
     nFECT = pSignalO.GetParameterizedSignalStatisticalSignalParameter(0, "FECT")
     Call pSignalO.SetParameterizedSignalStatisticalSignalParameter(0, "FECT",
     "Super")
     nExtinctionRatio = pSignalO.GetParameterizedSignalStatisticalSignalParameter(0,
     "ExtinctionRatio")
     Call pSignalO.SetParameterizedSignalStatisticalSignalParameter(0,
     "ExtinctionRatio", nExtinctionRatio + 1.0)
nChirpPeakValue = pSignalO.GetParameterizedSignalStatisticalSignalParameter(0,
     "ChirpPeakValue")
     Call pSignalO.SetParameterizedSignalStatisticalSignalParameter(0,
     "ChirpPeakValue", nChirpPeakValue + 1.0)
nChirpDuration = pSignalO.GetParameterizedSignalStatisticalSignalParameter(0,
     "ChirpDuration")
Call pSignalO.SetParameterizedSignalStatisticalSignalParameter(0, "ChirpDuration",
     nChirpDuration + 1.0)
nDispersion = pSignalO.GetParameterizedSignalStatisticalSignalParameter(0,
     "Dispersion")
Call pSignalO.SetParameterizedSignalStatisticalSignalParameter(0, "Dispersion",
     nDispersion + 1.0)
nTimeDelay = pSignalO.GetParameterizedSignalStatisticalSignalParameter(0, "TimeDelay")
```

```
Call pSignalO.SetParameterizedSignalStatisticalSignalParameter(0, "TimeDelay",
      nTimeDelay + 1.0)
nLinewidth = pSignalO.GetParameterizedSignalStatisticalSignalParameter(0, "Linewidth")
Call pSignalO.SetParameterizedSignalStatisticalSignalParameter(0, "Linewidth",
      nLinewidth + 1.0)
nPMD = pSignalO.GetParameterizedSignalStatisticalSignalParameter(0, "PMD")
Call pSignalO.SetParameterizedSignalStatisticalSignalParameter(0, "PMD", nPMD + 1.0)
nRIN = pSignalO.GetParameterizedSignalStatisticalSignalParameter(0, "RIN")
Call pSignalO.SetParameterizedSignalStatisticalSignalParameter(0, "RIN", nRIN + 1.0)
```

SetParameterizedSignalCommonSignalChannelsArraySize(long nIndex, long nSize)

Purpose:This method sets OpticalParameterizedSignal Common Channels array size for specified index of OpticalParameterizedSignalsArray.

Arguments:

nIndex – Specifies the index of OpticalParameterizedSignalsArray. nIndex is zero based.

nSize – Specifies the size of the array.

SetParameterizedSignalCommonSignalChannelChannelID(long nIndex, long nIndex2, long nChannelID)

Purpose:This method sets OpticalParameterizedSignal Common Channels ChannelID for specified nIndex2 of CommonChannelsArray and for specified nIndex of OpticalParameterizedSignalsArray.

Arguments:

nIndex – Specifies the index of OpticalParameterizedSignalsArray. nIndex is zero based.

nIndex2 – Specifies the index of CommonChannelsArray. nIdex2 is zero based.

nChannelID – Specifies sets OpticalParameterizedSignal Common Channels ChannelID.

Example:

```
      nNPS = pSignalO.GetNumberOfParameterizedSignals
      nAS = pSignalO.GetParameterizedSignalCommonSignalChannelsArraySize(nNPS - 1)
      nChID = pSignalO.GetParameterizedSignalCommonSignalChannelChannelID(nNPS - 1, nAS
- 1)
      Call pSignalO.SetParameterizedSignalCommonSignalChannelChannelID(nNPS - 1, nAS -
1, nChID)
```

SetParameterizedSignalCommonSignalChannelPower(long nIndex, long nIndex2, double dPower)

Purpose:This method sets OpticalParameterizedSignal Common Channels Power for specified nIndex2 of CommonChannelsArray and for specified nIndex of OpticalParameterizedSignalsArray.

Arguments:

nIndex – Specifies the index of OpticalParameterizedSignalsArray. nIndex is zero based.

nIndex2 – Specifies the index of CommonChannelsArray. nIdex2 is zero based.

dPower – Specifies OpticalParameterizedSignal Common Channels Power.

Example:

```
      nNPS = pSignalO.GetNumberOfParameterizedSignals
      nAS = pSignalO.GetParameterizedSignalCommonSignalChannelsArraySize(nNPS - 1)
      dChPower = pSignalO.GetParameterizedSignalCommonSignalChannelPower(nNPS - 1, nAS
- 1)
      Call pSignalO.SetParameterizedSignalCommonSignalChannelPower(nNPS - 1, nAS - 1,
dChPower + 1.0)
```


SetIndividualSampleSignalBandwidth(long nIndex, double dLowerFrequency, double dUpperFrequency)

Purpose: This method sets OpticalIndividualSample Bandwidth Lower and Upper Frequency for specified index of OpticalIndividualSamplesArray.

Arguments:

nIndex – Specifies the index of OpticalIndividualSamplesArray. nIndex is zero based.

pLowerFrequency – Specifies OpticalIndividualSample Bandwidth Lower Frequency.

pUpperFrequency – Specifies OpticalIndividualSample Bandwidth Upper Frequency.

Example:

```
nNIS = pSignal0.GetNumberOfIndividualSamples
Call pSignal0.GetIndividualSampleSignalBandwidth(nNIS - 1 , nLF0, nUF0)
Call pSignal0.SetIndividualSampleSignalBandwidth(nNumIndSig - 1, nLF0*500.0, nUF0*500.0)
```

SetIndividualSampleAmplitudeX(long nIndex,double AmplitudeXReal, double AmplitudeXImg)

Purpose: This method sets OpticalIndividualSample's AmplitudeX real and imaginary part for specified index of OpticalIndividualSamplesArray.

Arguments:

nIndex – Specifies the index of OpticalIndividualSamplesArray. nIndex is zero based.

AmplitudeXReal – Specifies OpticalIndividualSample's AmplitudeX real part.

AmplitudeXImg – Specifies OpticalIndividualSample's AmplitudeX imaginary part.

Example:

```
Call pSignal0.GetIndividualSampleAmplitudeX( 0, a, b)
Call pSignal0.SetIndividualSampleAmplitudeX(0, a*500.0, b*500.0)
```

SetIndividualSampleAmplitudeY(long nIndex, double AmplitudeYReal, double AmplitudeYImg)

Purpose: This method sets OpticalIndividualSample's AmplitudeY real and imaginary part for specified index of OpticalIndividualSamplesArray.

Arguments:

nIndex – Specifies the index of OpticalIndividualSamplesArray. nIndex is zero based.

AmplitudeXReal – Specifies OpticalIndividualSample's AmplitudeY real part.

AmplitudeXImg – Specifies OpticalIndividualSample's AmplitudeY imaginary part.

Example:

```
Call pSignal0.GetIndividualSampleAmplitudeY( 0, c, d)
Call pSignal0.SetIndividualSampleAmplitudeY(0, c*500.0, d*500.0)
```

SetSampledSignalSignalBandwidth(long nIndex, double dLowerFrequency, double dUpperFrequency)

Purpose: This method sets OpticalSampledSignal Bandwidth Lower and Upper Frequency for specified index of OpticalSampledSignalsArray.

Arguments:

nIndex – Specifies the index of OpticalSampledSignalsArray. Index is zero based.
pLowerFrequency – Specifies OpticalSampledSignal Bandwidth Lower Frequency.
pUpperFrequency – Specifies OpticalSampledSignal Bandwidth Upper Frequency.

SetSampledSignalOpticalSignalPolarizationParameters(long nIndex,double dS0, double dS1, double dS2, double dS3)
Purpose:This method sets OpticalSampledSignal Polarization Stokes vector parameters for specified index of OpticalSampledSignalArray.
Arguments:
nIndex – Specifies the index of OpticalSampledSignalArray. Index is zero based.
pS0 – Specifies OpticalSampledSignalArray Polarization Stokes vector parameter.
pS1 – Specifies OpticalSampledSignalArray Polarization Stokes vector parameter.
pS2 – Specifies OpticalSampledSignalArray Polarization Stokes vector parameter.
pS3 – Specifies OpticalSampledSignalArray Polarization Stokes vector parameter.
Example:

```
Call pSignalO. GetSampledSignalOpticalSignalPolarization(0, s0, s1, s2, s3)
Call pSignalO.SetSampledSignalOpticalSignalPolarizationParameters(0, s0+ 5, s1 + 5, s2 + 5, s3 + 5)
```

SetSampledSignalOpticalSignalPolarization(long nIndex,double dPower, double dSplittingRatio, double dPhaseDifference)
Purpose:This method sets OpticalSampledSignal Polarization parameters for specified index of OpticalSampledSignalsArray.
Arguments:
nIndex – Specifies the index of OpticalSampledSignalsArray. Index is zero based.
dPower – Specifies OpticalSampledSignal Polarization Power.
dSplittingRatio – Specifies OpticalSampledSignal Polarization Splitting Ratio.
dPhaseDifference – Specifies OpticalSampledSignal Polarization Phase.

SetSampledSignalArrAmplitudeX(long nIndex,VARIANT arrAmplitudeReal,VARIANT arrAmplitudeImg)
Purpose: This method sets arrays for OpticalSampledSignal's AmplitudeX real and imaginary part for specified nIndex of OpticalSampledSignalsArray.
Arguments:
nIndex – Specifies the index of OpticalParameterizedSignalsArray. nIndex is zero based.
arrAmplitudeReal – A VARIANT array OpticalSampledSignal's AmplitudeX real part.
arrAmplitudeImg – A VARIANT array OpticalSampledSignal's AmplitudeX imaginary part.
Example:

```
nNSS = pSignalO.GetNumberOfSampledSignals
```

```
Call pSignalO.GetSampledSignalArrAmplitudeX(nNSS - 1, dataR, dataI)
sizeLM = LBound( dataR )
sizeUM = UBound( dataR )
For i = sizeLM to sizeUM
    dataR(i) =  dataR(i) + 2.0;
    dataI(i) =  dataI(i) + 2.0;
Next
Call pSignalO.SetSampledSignalArrAmplitudeX(nNSS - 1, dataR, dataI)
```

SetSampledSignalArrAmplitudeY(long nIndex,VARIANT arrAmplitudeReal, VARIANT arrAmplitudeImg)

Purpose: This method sets arrays for OpticalSampledSignal's AmplitudeY real and imaginary part for specified nIndex of OpticalSampledSignalsArray.

Arguments:

nIndex – Specifies the index of OpticalParameterizedSignalsArray. nIndex is zero based.

arrAmplitudeReal – A VARIANT array OpticalSampledSignal's AmplitudeY real part.

arrAmplitudeImg – A VARIANT array OpticalSampledSignal's AmplitudeY imaginary part.

Example:

```
nNSS = pSignalO.GetNumberOfSampledSignals
Call pSignalO.GetSampledSignalArrAmplitudeY(nNSS - 1, dataR, dataI)
sizeLM = LBound( dataR )
sizeUM = UBound( dataR )
For i = sizeLM to sizeUM
    dataR(i) =  dataR(i) + 2.0;
    dataI(i) =  dataI(i) + 2.0;
Next
Call pSignalO.SetSampledSignalArrAmplitudeY(nNSS - 1, dataR, dataI)
```

SetSampledSignalArrModeX(long nIndex,VARIANT arrModeReal, VARIANT arrModeImg)

Purpose: This method sets arrays for OpticalSampledSignal's ModeX real and imaginary part for specified nIndex of OpticalSampledSignalsArray.

Arguments:

nIndex – Specifies the index of OpticalParameterizedSignalsArray. nIndex is zero based.

arrModeXReal – A VARIANT array OpticalSampledSignal's ModeX real part.

arrModeXImg – A VARIANT array OpticalSampledSignal's ModeX imaginary part.

SetSampledSignalArrModeY(long nIndex,VARIANT arrModeReal,VARIANT arrModeImg)

Purpose: This method sets arrays for OpticalSampledSignal's ModeY real and imaginary part for specified nIndex of OpticalSampledSignalsArray.

Arguments:

nIndex – Specifies the index of OpticalParameterizedSignalsArray. nIndex is zero based.

arrModeXReal – A VARIANT array OpticalSampledSignal's ModeY real part.

arrModeXImg – A VARIANT array OpticalSampledSignal's ModeY imaginary part.

SetSampledSignalTransverseModeMeshX(long nIndex,long dSizeX,long dSizeY);
Purpose: This method sets OpticalSampledSignal's MeshX sizeX and sizeY for specified nIndex of OpticalSampledSignalsArray.
Arguments:
nIndex – Specifies the index of OpticalParameterizedSignalsArray. nIndex is zero based.
varSizeX – Specifies OpticalSampledSignal's MeshX sizeX.
varSizeY – Specifies OpticalSampledSignal's MeshX sizeY.

SetSampledSignalTransverseModeMeshY(long nIndex,long dSizeX, long dSizeY)
Purpose: This method sets OpticalSampledSignal's MeshY sizeX and sizeY for specified nIndex of OpticalSampledSignalsArray.
Arguments:
nIndex – Specifies the index of OpticalParameterizedSignalsArray. nIndex is zero based.
varSizeX – Specifies OpticalSampledSignal's MeshY sizeX.
varSizeY – Specifies OpticalSampledSignal's MeshY sizeY.

SetSampledSignalModePropertiesX(long nIndex,BSTR sModePropertiesX)
Purpose:This method sets OpticalSampledSignal ModePropertiesX for specified index of OpticalSampledSignalsArray.
Arguments:
nIndex – Specifies the index of OpticalSampledSignalsArray. nIndex is zero based.
sModePropertiesX – Specifies OpticalSampledSignal's ModePropertiesX.

SetSampledSignalModePropertiesY(long nIndex,BSTR sModePropertiesY)
Purpose:This method sets OpticalSampledSignal ModePropertiesY for specified index of OpticalSampledSignalsArray.
Arguments:
nIndex – Specifies the index of OpticalSampledSignalsArray. nIndex is zero based.
sModePropertiesX – Specifies OpticalSampledSignal's ModePropertiesY.

SetNumberOfNoiseBins(long nNumberOfNoiseBins)
Purpose: This method sets the size of OpticalNoiseBinsArray.
Arguments:
nNumberOfNoiseBins – Specifies the size of OpticalNoiseBinsArray.

SetNumberOfParameterizedSignals(long nNumberOfParameterizedSignals)
Purpose: This method sets the size of OpticalParameterizedSignalsArray.
nNumberOfParameterizedSignals - Specifies the size of OpticalParameterizedSignalsArray.

SetNumberOfSampledSignals(long nNumberOfSampledSignals)

Purpose: This method sets the size of COpticalSampledSignalsArray.
nNumberOfSampledSignals - Specifies the size of
COpticalSampledSignalsArray.

SetNumberOfIndividualSamples(long nNumberOfIndividualSamples)
Purpose: This method returns the size of
COpticalIndividualSamplesArray.
nNumberOfIndividualSamples - Specifies the size of
COpticalIndividualSamplesArray.

VB Script Tutorials

Tutorial 1

The tutorial demonstrates the following topics:

- The document as a top level item.
- How to get to the Layout manager.
- How to add a layout.
- How to set the Current layout and getting the layout by index.
- How to set the Layout parameters.
- How to set the Layout canvas.
- How to set the Total sweep iterations and current sweep iteration.

Example

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Layout1.Name  = "Layout 1"
Layout1.SetTotalSweepIterations(3)
Layout1.SetCurrentSweepIteration(1)
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
Layout1.SetParameterMode "Bit rate", 0
Layout1.SetParameterValue "Bit rate", 2.5e+009
Layout1.AddParameter "Param1", 2 , "Simulation", 0.000000,
100.000000, "7.7", "", ""
Layout1.SetParameterMode "Param1", 0
Layout1.SetParameterValue "Param1", 7.7
Lm.AddLayout "Layout 2"
Dim Layout2
Set Layout2 = Lm.GetCurrentLayout
Layout2.Name  = "Layout 2"
Layout2.SetTotalSweepIterations(4)
Layout2.SetCurrentSweepIteration(1)
Dim Canvas2
Set Canvas2 = Layout2.GetCurrentCanvas
Layout2.SetParameterMode "Reference bit rate", 0
Layout2.SetParameterValue "Reference bit rate", TRUE
```

Analysis

Dim is a VB (Visual Basic) keyword for **dimension**. It allocates space for a variable.

Set is a VB keyword for assignment of an object variable. **Document** is a top-level object. This means that the document already exists at scripttime and doesn't need to

be created. You access the document through the OptiSystem defined string "**Document**".

> ***Note:*** It is a good idea to delete everything at the beginning of the script, especially if the script has a lot of variables that can be changed (for example, make a change, run the script, see the results, make another change).

Call **Document.RemoveAll TRUE** to delete all components in the layout. This call is very useful at the beginning of a complicated script because it clears the layout (possibly from the objects created by running the same script earlier). If **RemoveAll** was not included and the script was run previously, all the objects from the previous script or layouts will stay.

The document contains a Layout Manager which can be obtained by calling **Document.GetLayoutMgr**. The Layout manager keeps track of all Layouts in the Document. The user can obtain the current Layout by calling **Lm.GetCurrentLayout. LmAddLayout "Layout 2"** adds a new Layout to the Layout manager thus increasing the number of layouts to two. The user can set any layout to be a current for example by calling the method **Lm.SetCurrentLayout 2**.

You can set the maximum number of sweep iterations by using the method **Layout2.SetTotalSweepIterations(4).** The method **Layout2.SetCurrentSweepIteration(1)** sets the current sweep iteration for the Layout2. Every layout has global parameters which attributes can be set. For example **Layout2.SetParameterMode** "**Reference bit rate"**, 0 sets parameter mode to normal and **Layout2.SetParameterValue** "**Reference bit rate**", TRUE set the parameter value to TRUE. The user can add new parameter to the layout and later set its attributes. In our example this is illustrated with the method **Layout1.AddParameter** "**Param1**", **2** , "**Simulation**", **0.000000**, **100.000000**, "**7**.**7**", **"", ""**.

Tutorial 2

The tutorial demonstrates the following topics:

- How to create a Component.
- How to set the name of the Component.
- How to set the Parameter mode.
- How to set the Parameter'unit.
- How to set the Parameter value.
- How to set the Parameter script.

Example

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Layout1.Name  = "Layout 1"
Layout1.SetTotalSweepIterations(3)
Layout1.SetCurrentSweepIteration(1)
Dim Canvas1_Component7
Set Canvas1_Component7 = Layout1.CreateComponent("CW
Laser","{6DA31CEE-058F-11D4-93BD-0050DAB7C5D6}",230,60, 34, 34,0)
Canvas1_Component7.Name  = "Laser01"
Canvas1_Component7.SetParameterMode "Frequency", 2
Canvas1_Component7.SetParameterUnit "Frequency", "THz"
Canvas1_Component7.SetSweepParameterValue "Frequency",1,  193.1
Canvas1_Component7.SetSweepParameterValue "Frequency",2,  194.1
Canvas1_Component7.SetSweepParameterValue "Frequency",3,  195.1
Canvas1_Component7.SetParameterMode "Iterations", 3
Canvas1_Component7.SetParameterScript "Iterations", "Iterations"
```

Analysis

The user can create a component in the layout by calling the method **CreateComponent Layout1.CreateComponent**("**CW Laser**","**{6DA31CEE-058F-11D4-93BD-0050DAB7C5D6}",230,60, 34, 34,0**)

"**CW Laser**" - A string that specifies the type of the component.

"**{6DA31CEE-058F-11D4-93BD-0050DAB7C5D6}**" - A string that specifies the CLSID of the component library.

230 - Specifies the leftmost coordinate of the component.

60 - Specifies the topmost coordinate of the component.

34 - Specifies the component width.

34 - Specifies the component height.

0 - Specifies not to update the user interface.

The statement **Canvas1_Component7.Name = "Laser01"** set the component name as "**Laser01**".

Once the component has been created the user can set the component's parameters attributes.

Note: If you don't set the parameters attributes then the default settings are in place.

Canvas1_Component7.SetParameterMode "Frequency", 2 - sets parameter "Frequency" mode to iterate.

Canvas1_Component7.SetParameterUnit "Frequency", "THz" - sets parameter "Frequency" unit to "THz".

Canvas1_Component7.SetSweepParameterValue "Frequency",1, 193.1 - sets parameter "Frequency" value for the first sweep iteration to 193.1.

Canvas1_Component7.SetParameterMode "Iterations", 3 - sets parameter "Iterations" mode to script.

Canvas1_Component7.SetParameterScript "Iterations", "Iterations" - sets parameter "Iterations" script to "Iterations".

Tutorial 3

This tutorial demonstrates the following topics:

- How to create a Subsystem.
- How to set the dimmentions of a subsystem.
- How to add a parameter to a subsystem.
- How to add a port to a subsystem.

Example

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Layout1.Name  = "Layout 1"
Layout1.SetTotalSweepIterations(3)
Layout1.SetCurrentSweepIteration(1)
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
Dim Canvas1_Component7
Set Canvas1_Component7 = Canvas1.CreateComponent("Subsystem
1.0","{C83C8C01-53FD-11D4-9407-0050DAB7C5D6}",250,250, 162, 120,0)
Canvas1_Component7.Name  = "MySubsystem"
Canvas1_Component7.AddPort "Output", 3, 2, 0.385714
Canvas1_Component7.AddPort "Input", 2, 0, 0.421429
Canvas1_Component7.SetParameterMode "Subsystem Representation", 0
Canvas1_Component7.SetParameterValue "Subsystem Representation",
"GlassBox Image"
Canvas1_Component7.SetParameterMode "Image Filename", 0
Canvas1_Component7.SetParameterValue "Image Filename", ""
Canvas1_Component7.SetParameterMode "Stretch Image", 0
Canvas1_Component7.SetParameterValue "Stretch Image", FALSE
Canvas1_Component7.AddParameter "Param1", 2 , "category1", 0.000000,
100.000000, "5.5", "Hz", ""
Canvas1_Component7.SetParameterMode "Param1", 0
Canvas1_Component7.SetParameterValue "Param1", 5.5
Dim Canvas1_7
Set Canvas1_7 = Canvas1_Component7.GetCanvas
Canvas1_7.Height = 449
Canvas1_7.Width = 499
Canvas1_Component7.SetPosition 250, 250, 412, 370
```

Analysis

Let's first look at the characteristics of one subsystem.

- Subsystem is a component.
- Subsystem is a collection of components and or subsystems.

The subsystem can be created the same way is creating the component, using the Layout or Canvas method **CreateComponent**. Thus, the **Set Canvas1_Component7 = Canvas1.CreateComponent("Subsystem**

1.0","{C83C8C01-53FD-11D4-9407-0050DAB7C5D6}",250,250, 162, 120,0) statement creates a subsystem.

"**Subsystem 1.0**" - Specifies the subsystem type of the component.

"{**C83C8C01-53FD-11D4-9407-0050DAB7C5D6**}" - The CLSID for any subsystem.

250 - Specifies the leftmost coordinate of the subsystem.

250 - Specifies the topmost coordinate of the subsystem.

162 - Specifies the subsystem width.

120 - Specifies the subsystem height.

0 - Specifies not to update the user interface.

The user can add components to the subsystem by using the subsystem's canvas and method **CreateComponent**. The following code adds a component to the subsystem "**MySubsystem**".

```
Dim subComponent1
Set subComponent1  = Canvas1_7.CreateComponent("CW
Laser","{6DA31CEE-058F-11D4-93BD-0050DAB7C5D6}",70,70, 34, 34,0)
subComponent1  .Name  = "CW Laser1"
```

The method **Canvas1_Component7.AddPort "Output", 3, 2, 0.385714** adds an output port to the subsystem.

"**Output**"- A string indicating the name of the port.

3 - A long value that specifies the port type (RelayOutput port)

2 - The edge of subsystem on which the port will reside (right):

0.385714 - A double value that specifies the position on the subsystem's edge where the port will reside.

The method **Canvas1_Component7.AddParameter "Param1", 2 , "category1", 0.000000, 100.000000, "5.5", "Hz", ""** , adds a parameter "**Param1**" to the subsystem.

"**Param1**" - A string that specifies the name of the added parameter.

2 - Specifies the type of the added parameter (type Double):

"**category1**"- A string that specifies the name of the category to add.

0.0 - Specifies the min value of the parameter.

100.000000 - Specifies the max value of the parameter.

"5.5"- A string that specifies the value of the added parameter.

"Hz" - A string that specifies the units of the added parameter.

"" - A string that specifies the choice list of the added parameter. If the parameter is not a type **Choice** then the string is empty.

Tutorial 4

This tutorial demonstrates the following topics:

- How to create a path.
- How to connect components.
- How to connect subsystems.

Example

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Layout1.Name  = "Layout 1"
Layout1.SetTotalSweepIterations(1)
Layout1.SetCurrentSweepIteration(1)
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
Dim Canvas1_Component1
Set Canvas1_Component1 = Canvas1.CreateComponent("CW
Laser","{6DA31CEE-058F-11D4-93BD-0050DAB7C5D6}",110,130, 34, 34,0)
Canvas1_Component1.Name  = "CW Laser"
Dim Canvas1_Component2
Set Canvas1_Component2 = Canvas1.CreateComponent("EDFA","{255EDC8F-
37E4-11D4-93EC-0050DAB7C5D6}",270,160, 32, 32,0)
Canvas1_Component2.Name  = "EDFA"
Canvas1_Component1.GetOutputPort(1).Connect(Canvas1_Component2.Get
InputPort(1))
Layout1.GetPathMgr.CreatePath "Path 1",
Canvas1_Component1.GetOutputPort(1).GetSID,
Canvas1_Component2.GetOutputPort(1).GetSID
```

Analysis

The regular connection between two components is atcieved by connecting the Output port of one of the component and the Input port of the other componet.

In order to connect two components the user should do the following:

- Specify the component's output port. (**Canvas1_Component1.GetOutputPort(1)**).
- Use the output port's **Connect** method.
- Specify the other componet's Input port (**Canvas1_Component2.GetInputPort(1)**).

A path is created from Path manager using the method **CreatePath**. In order to create a path the user should do the following:

- Get the Layout's Path manager (**Layout1.GetPathMgr**).
- Use the Path manager's **CreatePath** method.
- Specify the name of the path ("**Path 1**") as a string.
- Specify the output port's SID of the first component of the path **(Canvas1_Component1.GetOutputPort(1).GetSID)**

- Specify the output port's SID of the second component of the path **(Canvas1_Component2.GetOutputPort(1).GetSID)**

Connecting Subsystems

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Layout1.Name  = "Layout 1"
Layout1.SetTotalSweepIterations(1)
Layout1.SetCurrentSweepIteration(1)
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
Dim Canvas1_Component2
Set Canvas1_Component2 = Canvas1.CreateComponent("CW
Laser","{6DA31CEE-058F-11D4-93BD-0050DAB7C5D6}",40,300, 34, 34,0)
Canvas1_Component2.Name  = "CW Laser"
Dim Canvas1_Component4
Set Canvas1_Component4 = Canvas1.CreateComponent("EDFA","{255EDC8F-
37E4-11D4-93EC-0050DAB7C5D6}",510,310, 32, 32,0)
Canvas1_Component4.Name  = "EDFA_1"
Dim Canvas1_Component5
Set Canvas1_Component5 = Canvas1.CreateComponent("Subsystem
1.0","{C83C8C01-53FD-11D4-9407-0050DAB7C5D6}",200,180, 162, 120,0)
Canvas1_Component5.Name  = "Subsystem"
Canvas1_Component5.AddPort "Input", 2, 0, 0.468750
Canvas1_Component5.AddPort "Output", 3, 2, 0.478750
Canvas1_Component5.AddPort "Output", 3, 2, 0.244186
Canvas1_Component5.AddPort "Input", 2, 0, 0.244186
Dim Canvas1_5
Set Canvas1_5 = Canvas1_Component5.GetCanvas
Canvas1_5.Height = 172
Canvas1_5.Width = 172
Canvas1_Component5.SetPosition 200, 180, 362, 300
Dim Canvas1_5_Component3
Set Canvas1_5_Component3 =
Canvas1_5.CreateComponent("EDFA","{255EDC8F-37E4-11D4-93EC-
0050DAB7C5D6}",70,70, 32, 32,0)
Canvas1_5_Component3.Name  = "EDFA"
Canvas1_Component2.GetOutputPort(1).Connect(Canvas1_Component5.Get
InputPort(1))
Canvas1_Component5.GetOutputPort(1).Connect(Canvas1_Component4.Get
InputPort(1))
Canvas1_Component5.GetInputPort(1).Connect(Canvas1_5_Component3.Ge
tInputPort(1))
Canvas1_Component5.GetOutputPort(2).ConnectRelay(Canvas1_Component
5.GetInputPort(2))
Canvas1_Component5.GetOutputPort(1).Connect(Canvas1_5_Component3.G
etOutputPort(1))
```

Analysis

Connecting Relay ports inside subsystems is different from connecting regular ports among components.

Connecting Relay Output port with Relay Input port can be done by using the Relay Output port's method **ConnectRelay**. Thus the statement **Canvas1_Component5.GetOutputPort(2).ConnectRelay(Canvas1_Component5.GetInputPort(2))** connects Subsystem's Relay Output port (2) to Subsystem's Relay Input port (2) inside the Subsystem.

The user must use the same method **Connect** when connecting Relay Input port to Input port inside Subsystem. Thus the statement **Canvas1_Component5.GetInputPort(1).Connect(Canvas1_5_Component3.GetInputPort(1))** connects Subsystem's Relay Input port to the other component's Input port.

The same approach is valid when connecting Relay Output port to other component's output port inside the subsystem.

Tutorial 5

This tutorial demonstrates the following topic:

- How to calculate a project.

Example

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Layout1.Name  = "Layout 1"
Layout1.SetTotalSweepIterations(1)
Layout1.SetCurrentSweepIteration(1)
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
Dim Canvas1_Component1
Set Canvas1_Component1 = Canvas1.CreateComponent("CW
Laser","{6DA31CEE-058F-11D4-93BD-0050DAB7C5D6}",110,110, 34, 34,0)
Canvas1_Component1.Name  = "CW Laser"
Dim Canvas1_Component2
Set Canvas1_Component2 = Canvas1.CreateComponent("Optical Spectrum
Analyzer","{F11D0C25-3C7D-11D4-93F0-0050DAB7C5D6}",220,190, 40,
34,0)
Canvas1_Component2.Name  = "Optical Spectrum Analyzer"
Layout1.SetTotalSweepIterations(1)
Layout1.SetCurrentSweepIteration(1)
Canvas1_Component1.GetOutputPort(1).CreateMonitor
Canvas1_Component1.GetOutputPort(1).ConnectVisualizer(Canvas1_Comp
onent2.GetInputPort(1))
Document.CalculateProject TRUE, TRUE
```

Running scripts

You can create scripts manually by typing script in the Scripting window, or automatically by using a generate script option. You can automatically generate part of the script and then manually modify the script according to your needs.

To run the script, from the **Tool** bar, select **Run Script**. This feature executes the script in the Script window. It is common to create a layout with a script and not run the calculations from the script. For example, you can create a complex component system with scripting and run the calculation from the layout by selecting the Calculate button from the Tool bar. If you want to run the calculation from the script, the statement **Document.CalculateProject** must be made somewhere in the script. This actually calculates the whole project.

The other options for calculating are: **Document.CalculateAllSweepIterations TRUE, TRUE** and **Document.CalculateCurrentSweepIteration TRUE, TRUE**.

Additional projects

Exporting graphs to Excel

Project **Exporting graphs to Excel Excel Graph.osd** demonstrates how to export graphs from a visualizer to a Excel spreadsheet and to a graph object. The OptiSystem script is provided bellow.

```
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Layout1.Name  = "Layout 1"
Layout1.SetTotalSweepIterations(1)
Layout1.SetCurrentSweepIteration(1)
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
Layout1.SetParameterMode "Simulation window", 0
Layout1.SetParameterValue "Simulation window", "Set bit rate"
Layout1.SetParameterMode "Reference bit rate", 0
Layout1.SetParameterValue "Reference bit rate", TRUE
Layout1.SetParameterMode "Bit rate", 0
Layout1.SetParameterValue "Bit rate", 2.5e+009
Layout1.SetParameterMode "Time window", 0
Layout1.SetParameterValue "Time window", 6.4e-009
Layout1.SetParameterMode "Sample rate", 0
Layout1.SetParameterValue "Sample rate", 4e+010
Layout1.SetParameterMode "Sequence length", 0
Layout1.SetParameterValue "Sequence length", 16
Layout1.SetParameterMode "Samples per bit", 0
Layout1.SetParameterValue "Samples per bit", 16
Layout1.SetParameterMode "Number of samples", 0
Layout1.SetParameterValue "Number of samples", 256
Layout1.SetParameterMode "Iterations", 0
Layout1.SetParameterValue "Iterations", 1
Layout1.SetParameterMode "Parameterized", 0
Layout1.SetParameterValue "Parameterized", FALSE
Layout1.SetParameterMode "Convert noise bins", 0
Layout1.SetParameterValue "Convert noise bins", FALSE
Layout1.SetParameterMode "Calculate signal tracing", 0
Layout1.SetParameterValue "Calculate signal tracing", TRUE
Layout1.SetParameterMode "Power unit", 0
Layout1.SetParameterValue "Power unit", "dBm"
Layout1.SetParameterMode "Frequency unit", 0
Layout1.SetParameterValue "Frequency unit", "THz"
Layout1.SetParameterMode "Decimal places", 0
Layout1.SetParameterValue "Decimal places", 4
Layout1.SetParameterMode "Sensitivity", 0
Layout1.SetParameterValue "Sensitivity", -100
```


```
Layout1.SetParameterMode "Resolution", 0
Layout1.SetParameterValue "Resolution", 0.1
Layout1.SetParameterMode "Calculate noise floor", 0
Layout1.SetParameterValue "Calculate noise floor", FALSE
Layout1.SetParameterMode "Interpolation offset", 0
Layout1.SetParameterValue "Interpolation offset", 0.5
'SCRIPT for each component in the Layout.
'SCRIPT for component WDM Transmitter.
Dim Canvas1_Component1
Set Canvas1_Component1 = Canvas1.CreateComponent("WDM
Transmitter","{2A9D9567-99DE-4C83-8F05-720F1480B57E}",60,140, 34,
60,0)
Canvas1_Component1.Name  = "WDM Transmitter"
'Set WDM Transmitter parameters.
Canvas1_Component1.SetParameterMode "Number of output ports", 0
Canvas1_Component1.SetParameterValue "Number of output ports", 1
Canvas1_Component1.SetParameterMode "Frequency", 0
Canvas1_Component1.SetParameterUnit "Frequency", "THz"
Canvas1_Component1.SetParameterValue "Frequency", 193.1
Canvas1_Component1.SetParameterMode "Frequency spacing", 0
Canvas1_Component1.SetParameterUnit "Frequency spacing", "GHz"
Canvas1_Component1.SetParameterValue "Frequency spacing", 100
Canvas1_Component1.SetParameterMode "Power", 0
Canvas1_Component1.SetParameterUnit "Power", "dBm"
Canvas1_Component1.SetParameterValue "Power", 0
Canvas1_Component1.SetParameterMode "Extinction ratio", 0
Canvas1_Component1.SetParameterValue "Extinction ratio", 10
Canvas1_Component1.SetParameterMode "Linewidth", 0
Canvas1_Component1.SetParameterValue "Linewidth", 10
Canvas1_Component1.SetParameterMode "Initial phase", 0
Canvas1_Component1.SetParameterValue "Initial phase", 0
Canvas1_Component1.SetParameterMode "Bit rate", 3
Canvas1_Component1.SetParameterUnit "Bit rate", "Bits/s"
Canvas1_Component1.SetParameterScript "Bit rate", "Bit rate"
Canvas1_Component1.SetParameterMode "Order", 3
Canvas1_Component1.SetParameterScript "Order", "log(Sequence
length)/log(2)"
Canvas1_Component1.SetParameterMode "Number of leading zeros", 0
Canvas1_Component1.SetParameterValue "Number of leading zeros", 1
Canvas1_Component1.SetParameterMode "Number of trailing zeros", 0
Canvas1_Component1.SetParameterValue "Number of trailing zeros", 1
Canvas1_Component1.SetParameterMode "Modulation type", 0
Canvas1_Component1.SetParameterValue "Modulation type", "NRZ"
Canvas1_Component1.SetParameterMode "Duty cycle", 0
Canvas1_Component1.SetParameterValue "Duty cycle", 0.5
Canvas1_Component1.SetParameterMode "Position", 0
Canvas1_Component1.SetParameterValue "Position", 0
Canvas1_Component1.SetParameterMode "Rise time", 3
Canvas1_Component1.SetParameterUnit "Rise time", "s"
Canvas1_Component1.SetParameterScript "Rise time", "1 / (Bit rate )
* 0.05"
```

```
Canvas1_Component1.SetParameterMode "Fall time", 3
Canvas1_Component1.SetParameterUnit "Fall time", "s"
Canvas1_Component1.SetParameterScript "Fall time", "1 / (Bit rate )
* 0.05"
Canvas1_Component1.SetParameterMode "Transmitter type", 0
Canvas1_Component1.SetParameterValue "Transmitter type", "EML"
Canvas1_Component1.SetParameterMode "Overshoot", 0
Canvas1_Component1.SetParameterValue "Overshoot", 30
Canvas1_Component1.SetParameterMode "Undershoot", 0
Canvas1_Component1.SetParameterValue "Undershoot", 30
Canvas1_Component1.SetParameterMode "Damping time leading edge", 3
Canvas1_Component1.SetParameterUnit "Damping time leading edge", "s"
Canvas1_Component1.SetParameterScript "Damping time leading edge",
"1 / (Bit rate ) * 0.5"
Canvas1_Component1.SetParameterMode "Damping time trailing edge", 3
Canvas1_Component1.SetParameterUnit "Damping time trailing edge",
"s"
Canvas1_Component1.SetParameterScript "Damping time trailing edge",
"1 / (Bit rate ) * 0.5"
Canvas1_Component1.SetParameterMode "Resonant frequency leading
edge", 3
Canvas1_Component1.SetParameterUnit "Resonant frequency leading
edge", "Hz"
Canvas1_Component1.SetParameterScript "Resonant frequency leading
edge", "(Bit rate ) * 5"
Canvas1_Component1.SetParameterMode "Resonant frequency trailing
edge", 3
Canvas1_Component1.SetParameterUnit "Resonant frequency trailing
edge", "Hz"
Canvas1_Component1.SetParameterScript "Resonant frequency trailing
edge", "(Bit rate ) * 5"
Canvas1_Component1.SetParameterMode "Calculate side mode", 0
Canvas1_Component1.SetParameterValue "Calculate side mode", FALSE
Canvas1_Component1.SetParameterMode "Separation", 0
Canvas1_Component1.SetParameterUnit "Separation", "GHz"
Canvas1_Component1.SetParameterValue "Separation", 75
Canvas1_Component1.SetParameterMode "Suppression ratio", 0
Canvas1_Component1.SetParameterValue "Suppression ratio", 30
Canvas1_Component1.SetParameterMode "Include RIN", 0
Canvas1_Component1.SetParameterValue "Include RIN", FALSE
Canvas1_Component1.SetParameterMode "RIN", 0
Canvas1_Component1.SetParameterValue "RIN", -130
Canvas1_Component1.SetParameterMode "Measured power", 0
Canvas1_Component1.SetParameterUnit "Measured power", "dBm"
Canvas1_Component1.SetParameterValue "Measured power", 10
Canvas1_Component1.SetParameterMode "Alpha parameter", 0
Canvas1_Component1.SetParameterValue "Alpha parameter", 0
Canvas1_Component1.SetParameterMode "Adiabatic chirp", 0
Canvas1_Component1.SetParameterValue "Adiabatic chirp", 0
Canvas1_Component1.SetParameterMode "Azimuth", 0
Canvas1_Component1.SetParameterValue "Azimuth", 0
Canvas1_Component1.SetParameterMode "Ellipticity", 0
```


```
Canvas1_Component1.SetParameterValue "Ellipticity", 0
Canvas1_Component1.SetParameterMode "Polarization filter", 0
Canvas1_Component1.SetParameterValue "Polarization filter", "None"
Canvas1_Component1.SetParameterMode "Enabled", 0
Canvas1_Component1.SetParameterValue "Enabled", TRUE
Canvas1_Component1.SetParameterMode "Iterations", 3
Canvas1_Component1.SetParameterScript "Iterations", "Iterations"
Canvas1_Component1.SetParameterMode "Parameterized", 3
Canvas1_Component1.SetParameterScript "Parameterized",
"Parameterized"
Canvas1_Component1.SetParameterMode "Sample rate", 3
Canvas1_Component1.SetParameterUnit "Sample rate", "Hz"
Canvas1_Component1.SetParameterScript "Sample rate", "Sample rate"
Canvas1_Component1.SetParameterMode "Noise bandwidth", 3
Canvas1_Component1.SetParameterUnit "Noise bandwidth", "Hz"
Canvas1_Component1.SetParameterScript "Noise bandwidth", "Sample
rate"
Canvas1_Component1.SetParameterMode "Noise bins spacing", 3
Canvas1_Component1.SetParameterUnit "Noise bins spacing", "Hz"
Canvas1_Component1.SetParameterScript "Noise bins spacing", "Sample
rate"
Canvas1_Component1.SetParameterMode "Convert noise bins", 3
Canvas1_Component1.SetParameterScript "Convert noise bins",
"Convert noise bins"
Canvas1_Component1.SetParameterMode "Generate random seed", 0
Canvas1_Component1.SetParameterValue "Generate random seed", TRUE
Canvas1_Component1.SetParameterMode "Random seed index", 0
Canvas1_Component1.SetParameterValue "Random seed index", 0
'SCRIPT for component Optical Spectrum Analyzer.
Dim Canvas1_Component2
Set Canvas1_Component2 = Canvas1.CreateComponent("Optical Spectrum
Analyzer","{F11D0C25-3C7D-11D4-93F0-0050DAB7C5D6}",200,150, 40,
34,0)
Canvas1_Component2.Name  = "Optical Spectrum Analyzer"
'Set Optical Spectrum Analyzer parameters.
Canvas1_Component2.SetParameterMode "Resolution bandwidth", 0
Canvas1_Component2.SetParameterValue "Resolution bandwidth", "Off"
Canvas1_Component2.SetParameterMode "Filter type", 0
Canvas1_Component2.SetParameterValue "Filter type", "Rectangle"
Canvas1_Component2.SetParameterMode "Bandwidth", 0
Canvas1_Component2.SetParameterValue "Bandwidth", 0.01
Canvas1_Component2.SetParameterMode "Power unit", 0
Canvas1_Component2.SetParameterValue "Power unit", "dBm"
Canvas1_Component2.SetParameterMode "Minimum value", 0
Canvas1_Component2.SetParameterValue "Minimum value", -100
Canvas1_Component2.SetParameterMode "Frequency unit", 0
Canvas1_Component2.SetParameterValue "Frequency unit", "m"
Canvas1_Component2.SetParameterMode "Limit number of points", 0
Canvas1_Component2.SetParameterValue "Limit number of points", TRUE
Canvas1_Component2.SetParameterMode "Max. number of points", 0
Canvas1_Component2.SetParameterValue "Max. number of points", 128000
```

```
Canvas1_Component2.SetParameterMode "Enabled", 0
Canvas1_Component2.SetParameterValue "Enabled", TRUE
Canvas1_Component2.SetParameterMode "Dynamic update", 0
Canvas1_Component2.SetParameterValue "Dynamic update", TRUE
 'Set Total Sweep Iterations
Layout1.SetTotalSweepIterations(1)
 'Set Current Sweep Iteration
Layout1.SetCurrentSweepIteration(1)
'Attach Monitors.
Canvas1_Component1.GetOutputPort(1).CreateMonitor
'Connecting components.
Canvas1_Component1.GetOutputPort(1).ConnectVisualizer(Canvas1_Comp
onent2.GetInputPort(1))
Document.CalculateProject TRUE, TRUE
'Get graphs from OSA
Dim OSA
Set OSA = Canvas1.GetComponentByName("Optical Spectrum Analyzer")
Dim GraphSSP
Set GraphSSP = OSA.GetGraph("Sampled signal spectrum")
Dim NumberOfPoints
NumberOfPoints = GraphSSP.GetNrOfPoints
Dim XData()
Dim YData()
Redim XData(NumberOfPoints)
Redim YData(NumberOfPoints)
For i = 0 to NumberOfPoints - 1
    XData(i) = GraphSSP.GetXDataAt(i,1)
    YData(i) = GraphSSP.GetYDataAt(i,1)
Next
Title = GraphSSP.GraphTitle
TitleX = GraphSSP.TitleX
TitleY = GraphSSP.TitleY
'Create Excel application
Dim Excel
Set Excel = CreateObject("Excel.Application")
Excel.Visible = true
Excel.Workbooks.Add
' populate the cells
For i = 0 to UBound(XData)-1
    Excel.ActiveWorkbook.Worksheets("sheet1").Cells(i+1,1) =
XData(i)
    Excel.ActiveWorkbook.Worksheets("sheet1").Cells(i+1,2) =
YData(i)
Next
' Add the chart
Excel.Charts.Add
' Format the chart, set type of chart, shape of the bars, show title,
get the data for the chart, show datatable, show legend
Excel.activechart.ChartType = 73
Excel.activechart.SetSourceData
Excel.ActiveWorkbook.Worksheets("sheet1").Range("A1:B250"),2
```


```
Excel.activechart.HasTitle = True
Excel.activechart.ChartTitle.Characters.Text = Title
Excel.activechart.Axes(1,1).HasTitle = True
Excel.activechart.Axes(1,1).AxisTitle.Characters.Text = TitleX
Excel.activechart.Axes(2,1).HasTitle = True
Excel.activechart.Axes(2,1).AxisTitle.Characters.Text = TitleY
```

Exporting sweeps to file

Project **Exporting sweeps to file Nested Sweep Export Text.osd** demonstrates how to export parameters and results from sweeps to a file. The OptiSystem script is provided bellow.

```
'Get Layout Manager.
Dim Lm
Set Lm = Document.GetLayoutMgr
'SCRIPT for Layout 1
'Get Current Layout.
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Layout1.Name  = "Layout 1"
 'Set Total Sweep Iterations
Layout1.SetTotalSweepIterations(1)
 'Set Current Sweep Iteration
Layout1.SetCurrentSweepIteration(1)
'Get Current Canvas.
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
'SCRIPT for Layout global parameters.
Layout1.SetParameterMode "Simulation window", 0
Layout1.SetParameterValue "Simulation window", "Set bit rate"
Layout1.SetParameterMode "Reference bit rate", 0
Layout1.SetParameterValue "Reference bit rate", TRUE
Layout1.SetParameterMode "Bit rate", 0
Layout1.SetParameterValue "Bit rate", 2.5e+009
Layout1.SetParameterMode "Time window", 0
Layout1.SetParameterValue "Time window", 5.12e-008
Layout1.SetParameterMode "Sample rate", 0
Layout1.SetParameterValue "Sample rate", 1.6e+011
Layout1.SetParameterMode "Sequence length", 0
Layout1.SetParameterValue "Sequence length", 128
Layout1.SetParameterMode "Samples per bit", 0
Layout1.SetParameterValue "Samples per bit", 64
Layout1.SetParameterMode "Number of samples", 0
Layout1.SetParameterValue "Number of samples", 8192
Layout1.SetParameterMode "Iterations", 0
Layout1.SetParameterValue "Iterations", 1
Layout1.SetParameterMode "Parameterized", 0
Layout1.SetParameterValue "Parameterized", FALSE
Layout1.SetParameterMode "Convert noise bins", 0
Layout1.SetParameterValue "Convert noise bins", FALSE
Layout1.SetParameterMode "Calculate signal tracing", 0
Layout1.SetParameterValue "Calculate signal tracing", TRUE
Layout1.SetParameterMode "Power unit", 0
Layout1.SetParameterValue "Power unit", "dBm"
Layout1.SetParameterMode "Frequency unit", 0
```


```
Layout1.SetParameterValue "Frequency unit", "THz"
Layout1.SetParameterMode "Decimal places", 0
Layout1.SetParameterValue "Decimal places", 4
Layout1.SetParameterMode "Sensitivity", 0
Layout1.SetParameterValue "Sensitivity", -100
Layout1.SetParameterMode "Resolution", 0
Layout1.SetParameterValue "Resolution", 0.1
Layout1.SetParameterMode "Calculate noise floor", 0
Layout1.SetParameterValue "Calculate noise floor", FALSE
Layout1.SetParameterMode "Interpolation offset", 0
Layout1.SetParameterValue "Interpolation offset", 0.5
'SCRIPT for each component in the Layout.
'SCRIPT for component CW Laser.
Dim Canvas1_Component1
Set Canvas1_Component1 = Canvas1.CreateComponent("CW
Laser","{6DA31CEE-058F-11D4-93BD-0050DAB7C5D6}",60,90, 34, 34,0)
Canvas1_Component1.Name  = "CW Laser"
'Set CW Laser parameters.
Canvas1_Component1.SetParameterMode "Frequency", 0
Canvas1_Component1.SetParameterUnit "Frequency", "THz"
Canvas1_Component1.SetParameterValue "Frequency", 193
Canvas1_Component1.SetParameterMode "Power", 0
Canvas1_Component1.SetParameterUnit "Power", "dBm"
Canvas1_Component1.SetParameterValue "Power", 10
Canvas1_Component1.SetParameterMode "Linewidth", 0
Canvas1_Component1.SetParameterValue "Linewidth", 10
Canvas1_Component1.SetParameterMode "Initial phase", 0
Canvas1_Component1.SetParameterValue "Initial phase", 0
Canvas1_Component1.SetParameterMode "Azimuth", 0
Canvas1_Component1.SetParameterValue "Azimuth", 0
Canvas1_Component1.SetParameterMode "Ellipticity", 0
Canvas1_Component1.SetParameterValue "Ellipticity", 0
Canvas1_Component1.SetParameterMode "Enabled", 0
Canvas1_Component1.SetParameterValue "Enabled", TRUE
Canvas1_Component1.SetParameterMode "Iterations", 3
Canvas1_Component1.SetParameterScript "Iterations", "Iterations"
Canvas1_Component1.SetParameterMode "Parameterized", 3
Canvas1_Component1.SetParameterScript "Parameterized",
"Parameterized"
Canvas1_Component1.SetParameterMode "Sample rate", 3
Canvas1_Component1.SetParameterUnit "Sample rate", "Hz"
Canvas1_Component1.SetParameterScript "Sample rate", "Sample rate"
Canvas1_Component1.SetParameterMode "Noise bandwidth", 0
Canvas1_Component1.SetParameterUnit "Noise bandwidth", "THz"
Canvas1_Component1.SetParameterValue "Noise bandwidth", 0
Canvas1_Component1.SetParameterMode "Noise threshold", 0
Canvas1_Component1.SetParameterValue "Noise threshold", -100
Canvas1_Component1.SetParameterMode "Noise dynamic", 0
Canvas1_Component1.SetParameterValue "Noise dynamic", 3
Canvas1_Component1.SetParameterMode "Generate random seed", 0
Canvas1_Component1.SetParameterValue "Generate random seed", TRUE
```

```
Canvas1_Component1.SetParameterMode "Random seed index", 0
Canvas1_Component1.SetParameterValue "Random seed index", 0
'SCRIPT for component EDFA.
Dim Canvas1_Component2
Set Canvas1_Component2 = Canvas1.CreateComponent("EDFA","{255EDC8F-
37E4-11D4-93EC-0050DAB7C5D6}",210,170, 32, 32,0)
Canvas1_Component2.Name  = "EDFA"
'Set EDFA parameters.
Canvas1_Component2.SetParameterMode "Core radius", 0
Canvas1_Component2.SetParameterValue "Core radius", 2.2
Canvas1_Component2.SetParameterMode "Er doping radius", 0
Canvas1_Component2.SetParameterValue "Er doping radius", 2.2
Canvas1_Component2.SetParameterMode "Er metastable lifetime", 0
Canvas1_Component2.SetParameterValue "Er metastable lifetime", 10
Canvas1_Component2.SetParameterMode "Numerical aperture", 0
Canvas1_Component2.SetParameterValue "Numerical aperture", 0.24
Canvas1_Component2.SetParameterMode "Er ion density", 0
Canvas1_Component2.SetParameterUnit "Er ion density", "m^-3"
Canvas1_Component2.SetParameterValue "Er ion density", 1e+025
Canvas1_Component2.SetParameterMode "Loss at 1550 nm", 0
Canvas1_Component2.SetParameterValue "Loss at 1550 nm", 0.1
Canvas1_Component2.SetParameterMode "Loss at 980 nm", 0
Canvas1_Component2.SetParameterValue "Loss at 980 nm", 0.15
Canvas1_Component2.SetParameterMode "Length", 0
Canvas1_Component2.SetParameterValue "Length", 20
Canvas1_Component2.SetParameterMode "Forward pump power", 0
Canvas1_Component2.SetParameterUnit "Forward pump power", "mW"
Canvas1_Component2.SetParameterValue "Forward pump power", 100
Canvas1_Component2.SetParameterMode "Backward pump power", 0
Canvas1_Component2.SetParameterUnit "Backward pump power", "mW"
Canvas1_Component2.SetParameterValue "Backward pump power", 0
Canvas1_Component2.SetParameterMode "Forward pump wavelength", 0
Canvas1_Component2.SetParameterValue "Forward pump wavelength", 980
Canvas1_Component2.SetParameterMode "Backward pump wavelength", 0
Canvas1_Component2.SetParameterValue "Backward pump wavelength",
980
Canvas1_Component2.SetParameterMode "File frequency unit", 0
Canvas1_Component2.SetParameterValue "File frequency unit", "nm"
Canvas1_Component2.SetParameterMode "EDFA_Design format", 0
Canvas1_Component2.SetParameterValue "EDFA_Design format", FALSE
Canvas1_Component2.SetParameterMode "Cross section file name", 0
Canvas1_Component2.SetParameterValue "Cross section file name",
"Erbium.dat"
Canvas1_Component2.SetParameterMode "Relative error", 0
Canvas1_Component2.SetParameterValue "Relative error", 0.1
Canvas1_Component2.SetParameterMode "Max. number of iterations", 0
Canvas1_Component2.SetParameterValue "Max. number of iterations", 50
Canvas1_Component2.SetParameterMode "Longitudinal steps", 0
Canvas1_Component2.SetParameterValue "Longitudinal steps", 50
Canvas1_Component2.SetParameterMode "Polarization filter", 0
Canvas1_Component2.SetParameterValue "Polarization filter", "None"
```


```
Canvas1_Component2.SetParameterMode "Enabled", 0
Canvas1_Component2.SetParameterValue "Enabled", TRUE
Canvas1_Component2.SetParameterMode "Noise center frequency", 0
Canvas1_Component2.SetParameterUnit "Noise center frequency", "THz"
Canvas1_Component2.SetParameterValue "Noise center frequency",
193.4
Canvas1_Component2.SetParameterMode "Noise bandwidth", 0
Canvas1_Component2.SetParameterUnit "Noise bandwidth", "THz"
Canvas1_Component2.SetParameterValue "Noise bandwidth", 13
Canvas1_Component2.SetParameterMode "Noise bins spacing", 0
Canvas1_Component2.SetParameterUnit "Noise bins spacing", "GHz"
Canvas1_Component2.SetParameterValue "Noise bins spacing", 125
Canvas1_Component2.SetParameterMode "Noise threshold", 0
Canvas1_Component2.SetParameterValue "Noise threshold", -100
Canvas1_Component2.SetParameterMode "Noise dynamic", 0
Canvas1_Component2.SetParameterValue "Noise dynamic", 3
Canvas1_Component2.SetParameterMode "Convert noise bins", 3
Canvas1_Component2.SetParameterScript "Convert noise bins",
"Convert noise bins"
Canvas1_Component2.SetParameterMode "Generate random seed", 0
Canvas1_Component2.SetParameterValue "Generate random seed", TRUE
Canvas1_Component2.SetParameterMode "Random seed index", 0
Canvas1_Component2.SetParameterValue "Random seed index", 0
'SCRIPT for component Dual Port WDM Analyzer.
Dim Canvas1_Component3
Set Canvas1_Component3 = Canvas1.CreateComponent("Dual Port WDM
Analyzer","{F11D0C25-3C7D-11D4-93F0-0050DAB7C5D6}",200,50, 40,
34,0)
Canvas1_Component3.Name  = "Dual Port WDM Analyzer"
'Set Dual Port WDM Analyzer parameters.
Canvas1_Component3.SetParameterMode "Lower frequency limit", 0
Canvas1_Component3.SetParameterUnit "Lower frequency limit", "THz"
Canvas1_Component3.SetParameterValue "Lower frequency limit", 185
Canvas1_Component3.SetParameterMode "Upper frequency limit", 0
Canvas1_Component3.SetParameterUnit "Upper frequency limit", "THz"
Canvas1_Component3.SetParameterValue "Upper frequency limit", 200
Canvas1_Component3.SetParameterMode "Resolution bandwidth", 0
Canvas1_Component3.SetParameterValue "Resolution bandwidth", 0.1
Canvas1_Component3.SetParameterMode "Minimum value", 0
Canvas1_Component3.SetParameterValue "Minimum value", -100
Canvas1_Component3.SetParameterMode "Noise interpolation", 0
Canvas1_Component3.SetParameterValue "Noise interpolation", "Off"
Canvas1_Component3.SetParameterMode "Interpolation offset", 0
Canvas1_Component3.SetParameterValue "Interpolation offset", 0.1
Canvas1_Component3.SetParameterMode "Frequency unit", 0
Canvas1_Component3.SetParameterValue "Frequency unit", "nm"
Canvas1_Component3.SetParameterMode "Enabled", 0
Canvas1_Component3.SetParameterValue "Enabled", TRUE
Canvas1_Component3.SetParameterMode "Dynamic update", 0
Canvas1_Component3.SetParameterValue "Dynamic update", TRUE
'Set Total Sweep Iterations
```

```
Layout1.SetTotalSweepIterations(1)
'Set Current Sweep Iteration
Layout1.SetCurrentSweepIteration(1)
'Attach Monitors.
Canvas1_Component1.GetOutputPort(1).CreateMonitor
Canvas1_Component2.GetOutputPort(1).CreateMonitor
'Connecting components.
Canvas1_Component1.GetOutputPort(1).Connect(Canvas1_Component2.Get
InputPort(1))
Canvas1_Component1.GetOutputPort(1).ConnectVisualizer(Canvas1_Comp
onent3.GetInputPort(1))
Canvas1_Component2.GetOutputPort(1).ConnectVisualizer(Canvas1_Comp
onent3.GetInputPort(2))
'-------------------------------------------------------------------
-------------------'
'OptiSystem Script: Nested loops with 2 parameters and 3 results
'                     Parameters and Results can be exported to Text
'-------------------------------------------------------------------
-------------------'
FileName = "C:\Results.txt"
'-------------------------------------------------------------------
-------------------'
'Global parameters - Parameter and Result names
'-------------------------------------------------------------------
-------------------'
ComponentName1 = "CW Laser"
ComponentName2 = "EDFA"
ParameterName1 = "Power"
ParameterName2 = "Length"
VisualizerName1 = "Dual Port WDM Analyzer"
VisualizerName2 = "Dual Port WDM Analyzer"
VisualizerName3 = "Dual Port WDM Analyzer"
ResultName1 = "Max. Gain (dB)"
ResultName2 = "Min. Noise Figure (dB)"
ResultName3 = "Output : Max. OSNR (dB)"
'-------------------------------------------------------------------
-------------------'
'Global parameters - Sweep ranges for parameters
'-------------------------------------------------------------------
-------------------'
NumberOfSweepIterations1 = 5
ParameterStart1 = -20
ParameterEnd1 = 10
NumberOfSweepIterations2 = 5
ParameterStart2 = 1
ParameterEnd2 = 20
'-------------------------------------------------------------------
-------------------'
'Internal parameters - step for each sweep
'-------------------------------------------------------------------
-------------------'
```


```
ParameterStep1 = ( ParameterEnd1 - ParameterStart1 ) / (
NumberOfSweepIterations1 - 1 )
ParameterStep2 = ( ParameterEnd2 - ParameterStart2 ) / (
NumberOfSweepIterations2 - 1 )
'-------------------------------------------------------------------
------------------'
'Create FileSystemObject
'-------------------------------------------------------------------
------------------'
Dim FileSystemObject
Dim OutF
Set FileSystemObject = CreateObject("Scripting.FileSystemObject")
Set OutF = FileSystemObject.CreateTextFile(FileName, True)
'-------------------------------------------------------------------
------------------'
' OptiSystem SDK specifics - access components and visualizers
'-------------------------------------------------------------------
------------------'
Dim Component1
Set Component1 = Canvas1.GetComponentByName(ComponentName1)
Dim Component2
Set Component2 = Canvas1.GetComponentByName(ComponentName2)
Dim Visualizer1
Set Visualizer1 = Canvas1.GetComponentByName(VisualizerName1)
Dim Visualizer2
Set Visualizer2 = Canvas1.GetComponentByName(VisualizerName2)
Dim Visualizer3
Set Visualizer3 = Canvas1.GetComponentByName(VisualizerName3)
'-------------------------------------------------------------------
------------------'
' Calculation loop - access parameters and results
'-------------------------------------------------------------------
------------------'
For i = 0 to NumberOfSweepIterations1 - 1
    For j = 0 to NumberOfSweepIterations2 - 1
    'Parameter Values
    ParameterValue1 = ( ParameterStart1 + i * ParameterStep1 )
    ParameterValue2 = ( ParameterStart2 + j * ParameterStep2 )
    'Set component parameters
    Component1.SetParameterValue ParameterName1, ParameterValue1 *
1.0
    Component2.SetParameterValue ParameterName2, ParameterValue2 *
1.0
    'Calculate
    Document.CalculateProject TRUE , TRUE
    'Access visualizer results
    Set Result1 = Visualizer1.GetResult( ResultName1 )
    Set Result2 = Visualizer2.GetResult( ResultName2 )
    Set Result3 = Visualizer3.GetResult( ResultName3 )
    'Access result values
    ResultValue1 = Result1.GetValue( 1 )
    ResultValue2 = Result2.GetValue( 1 )
```

```
    ResultValue3 = Result3.GetValue( 1 )
    'Send parameters and results to file
    TextLine = ""
    TextLine = TextLine + CStr( ParameterValue1 ) + " "
    TextLine = TextLine + CStr( ParameterValue2 ) + " "
    TextLine = TextLine + CStr( ResultValue1 ) + " "
    TextLine = TextLine + CStr( ResultValue2 ) + " "
    TextLine = TextLine + CStr( ResultValue3 )

    OutF.WriteLine( TextLine )
    Next
Next
'-------------------------------------------------------------------
-------------------'
' Close file
'-------------------------------------------------------------------
-------------------'
OutF.Close
```


Nested sweeps using Excel

Project **Nested sweeps using Excel Nested Sweep Excel.osd** demonstrates how to create nested sweeps and export parameters and results to Excel. The OptiSystem script is provided bellow.

```
'Get Layout Manager.
Dim Lm
Set Lm = Document.GetLayoutMgr
'SCRIPT for Layout 1
'Get Current Layout.
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Layout1.Name  = "Layout 1"
 'Set Total Sweep Iterations
Layout1.SetTotalSweepIterations(1)
 'Set Current Sweep Iteration
Layout1.SetCurrentSweepIteration(1)
'Get Current Canvas.
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
'SCRIPT for Layout global parameters.
Layout1.SetParameterMode "Simulation window", 0
Layout1.SetParameterValue "Simulation window", "Set bit rate"
Layout1.SetParameterMode "Reference bit rate", 0
Layout1.SetParameterValue "Reference bit rate", TRUE
Layout1.SetParameterMode "Bit rate", 0
Layout1.SetParameterValue "Bit rate", 2.5e+009
Layout1.SetParameterMode "Time window", 0
Layout1.SetParameterValue "Time window", 5.12e-008
Layout1.SetParameterMode "Sample rate", 0
Layout1.SetParameterValue "Sample rate", 1.6e+011
Layout1.SetParameterMode "Sequence length", 0
Layout1.SetParameterValue "Sequence length", 128
Layout1.SetParameterMode "Samples per bit", 0
Layout1.SetParameterValue "Samples per bit", 64
Layout1.SetParameterMode "Number of samples", 0
Layout1.SetParameterValue "Number of samples", 8192
Layout1.SetParameterMode "Iterations", 0
Layout1.SetParameterValue "Iterations", 1
Layout1.SetParameterMode "Parameterized", 0
Layout1.SetParameterValue "Parameterized", FALSE
Layout1.SetParameterMode "Convert noise bins", 0
Layout1.SetParameterValue "Convert noise bins", FALSE
Layout1.SetParameterMode "Calculate signal tracing", 0
Layout1.SetParameterValue "Calculate signal tracing", TRUE
Layout1.SetParameterMode "Power unit", 0
Layout1.SetParameterValue "Power unit", "dBm"
Layout1.SetParameterMode "Frequency unit", 0
```

```
Layout1.SetParameterValue "Frequency unit", "THz"
Layout1.SetParameterMode "Decimal places", 0
Layout1.SetParameterValue "Decimal places", 4
Layout1.SetParameterMode "Sensitivity", 0
Layout1.SetParameterValue "Sensitivity", -100
Layout1.SetParameterMode "Resolution", 0
Layout1.SetParameterValue "Resolution", 0.1
Layout1.SetParameterMode "Calculate noise floor", 0
Layout1.SetParameterValue "Calculate noise floor", FALSE
Layout1.SetParameterMode "Interpolation offset", 0
Layout1.SetParameterValue "Interpolation offset", 0.5
'SCRIPT for each component in the Layout.
'SCRIPT for component CW Laser.
Dim Canvas1_Component1
Set Canvas1_Component1 = Canvas1.CreateComponent("CW
Laser","{6DA31CEE-058F-11D4-93BD-0050DAB7C5D6}",60,90, 34, 34,0)
Canvas1_Component1.Name  = "CW Laser"
'Set CW Laser parameters.
Canvas1_Component1.SetParameterMode "Frequency", 0
Canvas1_Component1.SetParameterUnit "Frequency", "THz"
Canvas1_Component1.SetParameterValue "Frequency", 193
Canvas1_Component1.SetParameterMode "Power", 0
Canvas1_Component1.SetParameterUnit "Power", "dBm"
Canvas1_Component1.SetParameterValue "Power", 10
Canvas1_Component1.SetParameterMode "Linewidth", 0
Canvas1_Component1.SetParameterValue "Linewidth", 10
Canvas1_Component1.SetParameterMode "Initial phase", 0
Canvas1_Component1.SetParameterValue "Initial phase", 0
Canvas1_Component1.SetParameterMode "Azimuth", 0
Canvas1_Component1.SetParameterValue "Azimuth", 0
Canvas1_Component1.SetParameterMode "Ellipticity", 0
Canvas1_Component1.SetParameterValue "Ellipticity", 0
Canvas1_Component1.SetParameterMode "Enabled", 0
Canvas1_Component1.SetParameterValue "Enabled", TRUE
Canvas1_Component1.SetParameterMode "Iterations", 3
Canvas1_Component1.SetParameterScript "Iterations", "Iterations"
Canvas1_Component1.SetParameterMode "Parameterized", 3
Canvas1_Component1.SetParameterScript "Parameterized",
"Parameterized"
Canvas1_Component1.SetParameterMode "Sample rate", 3
Canvas1_Component1.SetParameterUnit "Sample rate", "Hz"
Canvas1_Component1.SetParameterScript "Sample rate", "Sample rate"
Canvas1_Component1.SetParameterMode "Noise bandwidth", 0
Canvas1_Component1.SetParameterUnit "Noise bandwidth", "THz"
Canvas1_Component1.SetParameterValue "Noise bandwidth", 0
Canvas1_Component1.SetParameterMode "Noise threshold", 0
Canvas1_Component1.SetParameterValue "Noise threshold", -100
Canvas1_Component1.SetParameterMode "Noise dynamic", 0
Canvas1_Component1.SetParameterValue "Noise dynamic", 3
Canvas1_Component1.SetParameterMode "Generate random seed", 0
Canvas1_Component1.SetParameterValue "Generate random seed", TRUE
```


```
Canvas1_Component1.SetParameterMode "Random seed index", 0
Canvas1_Component1.SetParameterValue "Random seed index", 0
'SCRIPT for component EDFA.
Dim Canvas1_Component2
Set Canvas1_Component2 = Canvas1.CreateComponent("EDFA","{255EDC8F-
37E4-11D4-93EC-0050DAB7C5D6}",210,170, 32, 32,0)
Canvas1_Component2.Name  = "EDFA"
'Set EDFA parameters.
Canvas1_Component2.SetParameterMode "Core radius", 0
Canvas1_Component2.SetParameterValue "Core radius", 2.2
Canvas1_Component2.SetParameterMode "Er doping radius", 0
Canvas1_Component2.SetParameterValue "Er doping radius", 2.2
Canvas1_Component2.SetParameterMode "Er metastable lifetime", 0
Canvas1_Component2.SetParameterValue "Er metastable lifetime", 10
Canvas1_Component2.SetParameterMode "Numerical aperture", 0
Canvas1_Component2.SetParameterValue "Numerical aperture", 0.24
Canvas1_Component2.SetParameterMode "Er ion density", 0
Canvas1_Component2.SetParameterUnit "Er ion density", "m^-3"
Canvas1_Component2.SetParameterValue "Er ion density", 1e+025
Canvas1_Component2.SetParameterMode "Loss at 1550 nm", 0
Canvas1_Component2.SetParameterValue "Loss at 1550 nm", 0.1
Canvas1_Component2.SetParameterMode "Loss at 980 nm", 0
Canvas1_Component2.SetParameterValue "Loss at 980 nm", 0.15
Canvas1_Component2.SetParameterMode "Length", 0
Canvas1_Component2.SetParameterValue "Length", 20
Canvas1_Component2.SetParameterMode "Forward pump power", 0
Canvas1_Component2.SetParameterUnit "Forward pump power", "mW"
Canvas1_Component2.SetParameterValue "Forward pump power", 100
Canvas1_Component2.SetParameterMode "Backward pump power", 0
Canvas1_Component2.SetParameterUnit "Backward pump power", "mW"
Canvas1_Component2.SetParameterValue "Backward pump power", 0
Canvas1_Component2.SetParameterMode "Forward pump wavelength", 0
Canvas1_Component2.SetParameterValue "Forward pump wavelength", 980
Canvas1_Component2.SetParameterMode "Backward pump wavelength", 0
Canvas1_Component2.SetParameterValue "Backward pump wavelength",
980
Canvas1_Component2.SetParameterMode "File frequency unit", 0
Canvas1_Component2.SetParameterValue "File frequency unit", "nm"
Canvas1_Component2.SetParameterMode "EDFA_Design format", 0
Canvas1_Component2.SetParameterValue "EDFA_Design format", FALSE
Canvas1_Component2.SetParameterMode "Cross section file name", 0
Canvas1_Component2.SetParameterValue "Cross section file name",
"Erbium.dat"
Canvas1_Component2.SetParameterMode "Relative error", 0
Canvas1_Component2.SetParameterValue "Relative error", 0.1
Canvas1_Component2.SetParameterMode "Max. number of iterations", 0
Canvas1_Component2.SetParameterValue "Max. number of iterations", 50
Canvas1_Component2.SetParameterMode "Longitudinal steps", 0
Canvas1_Component2.SetParameterValue "Longitudinal steps", 50
Canvas1_Component2.SetParameterMode "Polarization filter", 0
Canvas1_Component2.SetParameterValue "Polarization filter", "None"
```

```
Canvas1_Component2.SetParameterMode "Enabled", 0
Canvas1_Component2.SetParameterValue "Enabled", TRUE
Canvas1_Component2.SetParameterMode "Noise center frequency", 0
Canvas1_Component2.SetParameterUnit "Noise center frequency", "THz"
Canvas1_Component2.SetParameterValue "Noise center frequency",
193.4
Canvas1_Component2.SetParameterMode "Noise bandwidth", 0
Canvas1_Component2.SetParameterUnit "Noise bandwidth", "THz"
Canvas1_Component2.SetParameterValue "Noise bandwidth", 13
Canvas1_Component2.SetParameterMode "Noise bins spacing", 0
Canvas1_Component2.SetParameterUnit "Noise bins spacing", "GHz"
Canvas1_Component2.SetParameterValue "Noise bins spacing", 125
Canvas1_Component2.SetParameterMode "Noise threshold", 0
Canvas1_Component2.SetParameterValue "Noise threshold", -100
Canvas1_Component2.SetParameterMode "Noise dynamic", 0
Canvas1_Component2.SetParameterValue "Noise dynamic", 3
Canvas1_Component2.SetParameterMode "Convert noise bins", 3
Canvas1_Component2.SetParameterScript "Convert noise bins",
"Convert noise bins"
Canvas1_Component2.SetParameterMode "Generate random seed", 0
Canvas1_Component2.SetParameterValue "Generate random seed", TRUE
Canvas1_Component2.SetParameterMode "Random seed index", 0
Canvas1_Component2.SetParameterValue "Random seed index", 0
'SCRIPT for component Dual Port WDM Analyzer.
Dim Canvas1_Component3
Set Canvas1_Component3 = Canvas1.CreateComponent("Dual Port WDM
Analyzer","{F11D0C25-3C7D-11D4-93F0-0050DAB7C5D6}",200,50, 40,
34,0)
Canvas1_Component3.Name  = "Dual Port WDM Analyzer"
'Set Dual Port WDM Analyzer parameters.
Canvas1_Component3.SetParameterMode "Lower frequency limit", 0
Canvas1_Component3.SetParameterUnit "Lower frequency limit", "THz"
Canvas1_Component3.SetParameterValue "Lower frequency limit", 185
Canvas1_Component3.SetParameterMode "Upper frequency limit", 0
Canvas1_Component3.SetParameterUnit "Upper frequency limit", "THz"
Canvas1_Component3.SetParameterValue "Upper frequency limit", 200
Canvas1_Component3.SetParameterMode "Resolution bandwidth", 0
Canvas1_Component3.SetParameterValue "Resolution bandwidth", 0.1
Canvas1_Component3.SetParameterMode "Minimum value", 0
Canvas1_Component3.SetParameterValue "Minimum value", -100
Canvas1_Component3.SetParameterMode "Noise interpolation", 0
Canvas1_Component3.SetParameterValue "Noise interpolation", "Off"
Canvas1_Component3.SetParameterMode "Interpolation offset", 0
Canvas1_Component3.SetParameterValue "Interpolation offset", 0.1
Canvas1_Component3.SetParameterMode "Frequency unit", 0
Canvas1_Component3.SetParameterValue "Frequency unit", "nm"
Canvas1_Component3.SetParameterMode "Enabled", 0
Canvas1_Component3.SetParameterValue "Enabled", TRUE
Canvas1_Component3.SetParameterMode "Dynamic update", 0
Canvas1_Component3.SetParameterValue "Dynamic update", TRUE
 'Set Total Sweep Iterations
```


```
Layout1.SetTotalSweepIterations(1)
 'Set Current Sweep Iteration
Layout1.SetCurrentSweepIteration(1)
'Attach Monitors.
Canvas1_Component1.GetOutputPort(1).CreateMonitor
Canvas1_Component2.GetOutputPort(1).CreateMonitor
'Connecting components.
Canvas1_Component1.GetOutputPort(1).Connect(Canvas1_Component2.Get
InputPort(1))
Canvas1_Component1.GetOutputPort(1).ConnectVisualizer(Canvas1_Comp
onent3.GetInputPort(1))
Canvas1_Component2.GetOutputPort(1).ConnectVisualizer(Canvas1_Comp
onent3.GetInputPort(2))
'-------------------------------------------------------------------
------------------'
'OptiSystem Script: Nested loops with 3 parameters and 3 results
'                       Parameters and Results can be exported to
Excel or as Text
'-------------------------------------------------------------------
------------------'
'-------------------------------------------------------------------
------------------'
'Global parameters - Parameter and Result names
'-------------------------------------------------------------------
------------------'
ComponentName1 = "CW Laser"
ComponentName2 = "CW Laser"
ComponentName3 = "EDFA"
ParameterName1 = "Power"
ParameterName2 = "Frequency"
ParameterName3 = "Length"
VisualizerName1 = "Dual Port WDM Analyzer"
VisualizerName2 = "Dual Port WDM Analyzer"
VisualizerName3 = "Dual Port WDM Analyzer"
ResultName1 = "Max. Gain (dB)"
ResultName2 = "Min. Noise Figure (dB)"
ResultName3 = "Output : Max. OSNR (dB)"
'-------------------------------------------------------------------
------------------'
'Global parameters - Sweep ranges for parameters
'-------------------------------------------------------------------
------------------'
NumberOfSweepIterations1 = 3
ParameterStart1 = -20
ParameterEnd1 = 10
NumberOfSweepIterations2 = 3
ParameterStart2 = 190
ParameterEnd2 = 193
NumberOfSweepIterations3 = 3
ParameterStart3 = 5
ParameterEnd3 = 20
```

```
'-------------------------------------------------------------------
------------------'
'Internal parameters - step for each sweep
'-------------------------------------------------------------------
------------------'
ParameterStep1 = ( ParameterEnd1 - ParameterStart1 ) / (
NumberOfSweepIterations1 - 1 )
ParameterStep2 = ( ParameterEnd2 - ParameterStart2 ) / (
NumberOfSweepIterations2 - 1 )
ParameterStep3 = ( ParameterEnd3 - ParameterStart3 ) / (
NumberOfSweepIterations3 - 1 )
'-------------------------------------------------------------------
------------------'
'Create Excel application - visible must be FALSE
'-------------------------------------------------------------------
------------------'
Dim Excel
Set Excel = CreateObject("Excel.Application")

Excel.Visible = false
Excel.Workbooks.Add
'-------------------------------------------------------------------
------------------'
' OptiSystem SDK specifics - access components and visualizers
'-------------------------------------------------------------------
------------------'
Dim Component1
Set Component1 = Canvas1.GetComponentByName(ComponentName1)
Dim Component2
Set Component2 = Canvas1.GetComponentByName(ComponentName2)
Dim Component3
Set Component3 = Canvas1.GetComponentByName(ComponentName3)
Dim Visualizer1
Set Visualizer1 = Canvas1.GetComponentByName(VisualizerName1)
Dim Visualizer2
Set Visualizer2 = Canvas1.GetComponentByName(VisualizerName2)
Dim Visualizer3
Set Visualizer3 = Canvas1.GetComponentByName(VisualizerName3)
'-------------------------------------------------------------------
------------------'
' Calculation loop - access parameters and results
'-------------------------------------------------------------------
------------------'
Count = 1
For i = 0 to NumberOfSweepIterations1 - 1
     For j = 0 to NumberOfSweepIterations2 - 1
        For k = 0 to NumberOfSweepIterations3 - 1
            'Parameter Values
            ParameterValue1 = ( ParameterStart1 + i * ParameterStep1
)
             ParameterValue2 = ( ParameterStart2 + j * ParameterStep2
)
```


```
            ParameterValue3 = ( ParameterStart3 + k * ParameterStep3
)
            'Set component parameters
            Component1.SetParameterValue ParameterName1,
ParameterValue1 * 1.0
             Component2.SetParameterValue ParameterName2,
ParameterValue2 * 1.0
             Component3.SetParameterValue ParameterName3,
ParameterValue3 * 1.0
'Calculate
             Document.CalculateProject TRUE, TRUE
'Access visualizer results
             Set Result1 = Visualizer1.GetResult( ResultName1 )
             Set Result2 = Visualizer2.GetResult( ResultName2 )
          Set Result3 = Visualizer3.GetResult( ResultName3 )
'Access result values
             ResultValue1 = Result1.GetValue( 1 )
             ResultValue2 = Result2.GetValue( 1 )
             ResultValue3 = Result3.GetValue( 1 )
'Send parameters and results to Excel
          Excel.ActiveWorkbook.Worksheets("sheet1").Cells(Count,1)
= ParameterValue1
          Excel.ActiveWorkbook.Worksheets("sheet1").Cells(Count,2)
= ParameterValue2
          Excel.ActiveWorkbook.Worksheets("sheet1").Cells(Count,3)
= ParameterValue3
        Excel.ActiveWorkbook.Worksheets("sheet1").Cells(Count,4) =
ResultValue1
        Excel.ActiveWorkbook.Worksheets("sheet1").Cells(Count,5) =
ResultValue2
        Excel.ActiveWorkbook.Worksheets("sheet1").Cells(Count,6) =
ResultValue3

        Count = Count + 1
    Next
 Next
Next
'-----------------------------------------------------------------
------------------'
'Enable Excel application - visible must be TRUE
'-----------------------------------------------------------------
------------------'
Excel.Visible = true
Nested Sweeps using MatLab
'Get Layout Manager.
Dim Lm
Set Lm = Document.GetLayoutMgr
'SCRIPT for Layout 1
'Get Current Layout.
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Layout1.Name  = "Layout 1"
```

```
'Set Total Sweep Iterations
Layout1.SetTotalSweepIterations(1)
'Set Current Sweep Iteration
Layout1.SetCurrentSweepIteration(1)
'Get Current Canvas.
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
'SCRIPT for Layout global parameters.
Layout1.SetParameterMode "Simulation window", 0
Layout1.SetParameterValue "Simulation window", "Set bit rate"
Layout1.SetParameterMode "Reference bit rate", 0
Layout1.SetParameterValue "Reference bit rate", TRUE
Layout1.SetParameterMode "Bit rate", 0
Layout1.SetParameterValue "Bit rate", 2.5e+009
Layout1.SetParameterMode "Time window", 0
Layout1.SetParameterValue "Time window", 5.12e-008
Layout1.SetParameterMode "Sample rate", 0
Layout1.SetParameterValue "Sample rate", 1.6e+011
Layout1.SetParameterMode "Sequence length", 0
Layout1.SetParameterValue "Sequence length", 128
Layout1.SetParameterMode "Samples per bit", 0
Layout1.SetParameterValue "Samples per bit", 64
Layout1.SetParameterMode "Number of samples", 0
Layout1.SetParameterValue "Number of samples", 8192
Layout1.SetParameterMode "Iterations", 0
Layout1.SetParameterValue "Iterations", 1
Layout1.SetParameterMode "Parameterized", 0
Layout1.SetParameterValue "Parameterized", FALSE
Layout1.SetParameterMode "Convert noise bins", 0
Layout1.SetParameterValue "Convert noise bins", FALSE
Layout1.SetParameterMode "Calculate signal tracing", 0
Layout1.SetParameterValue "Calculate signal tracing", TRUE
Layout1.SetParameterMode "Power unit", 0
Layout1.SetParameterValue "Power unit", "dBm"
Layout1.SetParameterMode "Frequency unit", 0
Layout1.SetParameterValue "Frequency unit", "THz"
Layout1.SetParameterMode "Decimal places", 0
Layout1.SetParameterValue "Decimal places", 4
Layout1.SetParameterMode "Sensitivity", 0
Layout1.SetParameterValue "Sensitivity", -100
Layout1.SetParameterMode "Resolution", 0
Layout1.SetParameterValue "Resolution", 0.1
Layout1.SetParameterMode "Calculate noise floor", 0
Layout1.SetParameterValue "Calculate noise floor", FALSE
Layout1.SetParameterMode "Interpolation offset", 0
Layout1.SetParameterValue "Interpolation offset", 0.5
'SCRIPT for each component in the Layout.
'SCRIPT for component CW Laser.
Dim Canvas1_Component1
Set Canvas1_Component1 = Canvas1.CreateComponent("CW
Laser","{6DA31CEE-058F-11D4-93BD-0050DAB7C5D6}",60,90, 34, 34,0)
```


```
Canvas1_Component1.Name  = "CW Laser"
'Set CW Laser parameters.
Canvas1_Component1.SetParameterMode "Frequency", 0
Canvas1_Component1.SetParameterUnit "Frequency", "THz"
Canvas1_Component1.SetParameterValue "Frequency", 193
Canvas1_Component1.SetParameterMode "Power", 0
Canvas1_Component1.SetParameterUnit "Power", "dBm"
Canvas1_Component1.SetParameterValue "Power", 10
Canvas1_Component1.SetParameterMode "Linewidth", 0
Canvas1_Component1.SetParameterValue "Linewidth", 10
Canvas1_Component1.SetParameterMode "Initial phase", 0
Canvas1_Component1.SetParameterValue "Initial phase", 0
Canvas1_Component1.SetParameterMode "Azimuth", 0
Canvas1_Component1.SetParameterValue "Azimuth", 0
Canvas1_Component1.SetParameterMode "Ellipticity", 0
Canvas1_Component1.SetParameterValue "Ellipticity", 0
Canvas1_Component1.SetParameterMode "Enabled", 0
Canvas1_Component1.SetParameterValue "Enabled", TRUE
Canvas1_Component1.SetParameterMode "Iterations", 3
Canvas1_Component1.SetParameterScript "Iterations", "Iterations"
Canvas1_Component1.SetParameterMode "Parameterized", 3
Canvas1_Component1.SetParameterScript "Parameterized",
"Parameterized"
Canvas1_Component1.SetParameterMode "Sample rate", 3
Canvas1_Component1.SetParameterUnit "Sample rate", "Hz"
Canvas1_Component1.SetParameterScript "Sample rate", "Sample rate"
Canvas1_Component1.SetParameterMode "Noise bandwidth", 0
Canvas1_Component1.SetParameterUnit "Noise bandwidth", "THz"
Canvas1_Component1.SetParameterValue "Noise bandwidth", 0
Canvas1_Component1.SetParameterMode "Noise threshold", 0
Canvas1_Component1.SetParameterValue "Noise threshold", -100
Canvas1_Component1.SetParameterMode "Noise dynamic", 0
Canvas1_Component1.SetParameterValue "Noise dynamic", 3
Canvas1_Component1.SetParameterMode "Generate random seed", 0
Canvas1_Component1.SetParameterValue "Generate random seed", TRUE
Canvas1_Component1.SetParameterMode "Random seed index", 0
Canvas1_Component1.SetParameterValue "Random seed index", 0
'SCRIPT for component EDFA.
Dim Canvas1_Component2
Set Canvas1_Component2 = Canvas1.CreateComponent("EDFA","{255EDC8F-
37E4-11D4-93EC-0050DAB7C5D6}",210,170, 32, 32,0)
Canvas1_Component2.Name  = "EDFA"
'Set EDFA parameters.
Canvas1_Component2.SetParameterMode "Core radius", 0
Canvas1_Component2.SetParameterValue "Core radius", 2.2
Canvas1_Component2.SetParameterMode "Er doping radius", 0
Canvas1_Component2.SetParameterValue "Er doping radius", 2.2
Canvas1_Component2.SetParameterMode "Er metastable lifetime", 0
Canvas1_Component2.SetParameterValue "Er metastable lifetime", 10
Canvas1_Component2.SetParameterMode "Numerical aperture", 0
Canvas1_Component2.SetParameterValue "Numerical aperture", 0.24
```

```
Canvas1_Component2.SetParameterMode "Er ion density", 0
Canvas1_Component2.SetParameterUnit "Er ion density", "m^-3"
Canvas1_Component2.SetParameterValue "Er ion density", 1e+025
Canvas1_Component2.SetParameterMode "Loss at 1550 nm", 0
Canvas1_Component2.SetParameterValue "Loss at 1550 nm", 0.1
Canvas1_Component2.SetParameterMode "Loss at 980 nm", 0
Canvas1_Component2.SetParameterValue "Loss at 980 nm", 0.15
Canvas1_Component2.SetParameterMode "Length", 0
Canvas1_Component2.SetParameterValue "Length", 20
Canvas1_Component2.SetParameterMode "Forward pump power", 0
Canvas1_Component2.SetParameterUnit "Forward pump power", "mW"
Canvas1_Component2.SetParameterValue "Forward pump power", 100
Canvas1_Component2.SetParameterMode "Backward pump power", 0
Canvas1_Component2.SetParameterUnit "Backward pump power", "mW"
Canvas1_Component2.SetParameterValue "Backward pump power", 0
Canvas1_Component2.SetParameterMode "Forward pump wavelength", 0
Canvas1_Component2.SetParameterValue "Forward pump wavelength", 980
Canvas1_Component2.SetParameterMode "Backward pump wavelength", 0
Canvas1_Component2.SetParameterValue "Backward pump wavelength",
980
Canvas1_Component2.SetParameterMode "File frequency unit", 0
Canvas1_Component2.SetParameterValue "File frequency unit", "nm"
Canvas1_Component2.SetParameterMode "EDFA_Design format", 0
Canvas1_Component2.SetParameterValue "EDFA_Design format", FALSE
Canvas1_Component2.SetParameterMode "Cross section file name", 0
Canvas1_Component2.SetParameterValue "Cross section file name",
"Erbium.dat"
Canvas1_Component2.SetParameterMode "Relative error", 0
Canvas1_Component2.SetParameterValue "Relative error", 0.1
Canvas1_Component2.SetParameterMode "Max. number of iterations", 0
Canvas1_Component2.SetParameterValue "Max. number of iterations", 50
Canvas1_Component2.SetParameterMode "Longitudinal steps", 0
Canvas1_Component2.SetParameterValue "Longitudinal steps", 50
Canvas1_Component2.SetParameterMode "Polarization filter", 0
Canvas1_Component2.SetParameterValue "Polarization filter", "None"
Canvas1_Component2.SetParameterMode "Enabled", 0
Canvas1_Component2.SetParameterValue "Enabled", TRUE
Canvas1_Component2.SetParameterMode "Noise center frequency", 0
Canvas1_Component2.SetParameterUnit "Noise center frequency", "THz"
Canvas1_Component2.SetParameterValue "Noise center frequency",
193.4
Canvas1_Component2.SetParameterMode "Noise bandwidth", 0
Canvas1_Component2.SetParameterUnit "Noise bandwidth", "THz"
Canvas1_Component2.SetParameterValue "Noise bandwidth", 13
Canvas1_Component2.SetParameterMode "Noise bins spacing", 0
Canvas1_Component2.SetParameterUnit "Noise bins spacing", "GHz"
Canvas1_Component2.SetParameterValue "Noise bins spacing", 125
Canvas1_Component2.SetParameterMode "Noise threshold", 0
Canvas1_Component2.SetParameterValue "Noise threshold", -100
Canvas1_Component2.SetParameterMode "Noise dynamic", 0
Canvas1_Component2.SetParameterValue "Noise dynamic", 3
```


```
Canvas1_Component2.SetParameterMode "Convert noise bins", 3
Canvas1_Component2.SetParameterScript "Convert noise bins",
"Convert noise bins"
Canvas1_Component2.SetParameterMode "Generate random seed", 0
Canvas1_Component2.SetParameterValue "Generate random seed", TRUE
Canvas1_Component2.SetParameterMode "Random seed index", 0
Canvas1_Component2.SetParameterValue "Random seed index", 0
'SCRIPT for component Dual Port WDM Analyzer.
Dim Canvas1_Component3
Set Canvas1_Component3 = Canvas1.CreateComponent("Dual Port WDM
Analyzer","{F11D0C25-3C7D-11D4-93F0-0050DAB7C5D6}",200,50, 40,
34,0)
Canvas1_Component3.Name  = "Dual Port WDM Analyzer"
'Set Dual Port WDM Analyzer parameters.
Canvas1_Component3.SetParameterMode "Lower frequency limit", 0
Canvas1_Component3.SetParameterUnit "Lower frequency limit", "THz"
Canvas1_Component3.SetParameterValue "Lower frequency limit", 185
Canvas1_Component3.SetParameterMode "Upper frequency limit", 0
Canvas1_Component3.SetParameterUnit "Upper frequency limit", "THz"
Canvas1_Component3.SetParameterValue "Upper frequency limit", 200
Canvas1_Component3.SetParameterMode "Resolution bandwidth", 0
Canvas1_Component3.SetParameterValue "Resolution bandwidth", 0.1
Canvas1_Component3.SetParameterMode "Minimum value", 0
Canvas1_Component3.SetParameterValue "Minimum value", -100
Canvas1_Component3.SetParameterMode "Noise interpolation", 0
Canvas1_Component3.SetParameterValue "Noise interpolation", "Off"
Canvas1_Component3.SetParameterMode "Interpolation offset", 0
Canvas1_Component3.SetParameterValue "Interpolation offset", 0.1
Canvas1_Component3.SetParameterMode "Frequency unit", 0
Canvas1_Component3.SetParameterValue "Frequency unit", "nm"
Canvas1_Component3.SetParameterMode "Enabled", 0
Canvas1_Component3.SetParameterValue "Enabled", TRUE
Canvas1_Component3.SetParameterMode "Dynamic update", 0
Canvas1_Component3.SetParameterValue "Dynamic update", TRUE
 'Set Total Sweep Iterations
Layout1.SetTotalSweepIterations(1)
 'Set Current Sweep Iteration
Layout1.SetCurrentSweepIteration(1)
'Attach Monitors.
Canvas1_Component1.GetOutputPort(1).CreateMonitor
Canvas1_Component2.GetOutputPort(1).CreateMonitor
'Connecting components.
Canvas1_Component1.GetOutputPort(1).Connect(Canvas1_Component2.Get
InputPort(1))
Canvas1_Component1.GetOutputPort(1).ConnectVisualizer(Canvas1_Comp
onent3.GetInputPort(1))
Canvas1_Component2.GetOutputPort(1).ConnectVisualizer(Canvas1_Comp
onent3.GetInputPort(2))
'-----------------------------------------------------------------
-------------------'
'OptiSystem Script: Nested loops with 2 parameters and 3 results
```

```
'                    Parameters and Results can be exported to Matlab
'-------------------------------------------------------------------
-------------------'
'-------------------------------------------------------------------
-------------------'
'Global parameters - Parameter and Result names
'-------------------------------------------------------------------
-------------------'
ComponentName1 = "CW Laser"
ComponentName2 = "EDFA"
ParameterName1 = "Power"
ParameterName2 = "Length"
VisualizerName1 = "Dual Port WDM Analyzer"
VisualizerName2 = "Dual Port WDM Analyzer"
VisualizerName3 = "Dual Port WDM Analyzer"
ResultName1 = "Max. Gain (dB)"
ResultName2 = "Min. Noise Figure (dB)"
ResultName3 = "Output : Max. OSNR (dB)"
'-------------------------------------------------------------------
-------------------'
'Global parameters - Sweep ranges for parameters
'-------------------------------------------------------------------
-------------------'
NumberOfSweepIterations1 = 10
ParameterStart1 = -20
ParameterEnd1 = 10
NumberOfSweepIterations2 = 10
ParameterStart2 = 1
ParameterEnd2 = 20
'-------------------------------------------------------------------
-------------------'
'Internal parameters - step for each sweep
'-------------------------------------------------------------------
-------------------'
ParameterStep1 = ( ParameterEnd1 - ParameterStart1 ) / (
NumberOfSweepIterations1 - 1 )
ParameterStep2 = ( ParameterEnd2 - ParameterStart2 ) / (
NumberOfSweepIterations2 - 1 )
'-------------------------------------------------------------------
-------------------'
'Create Matlab application - visible must be FALSE
'-------------------------------------------------------------------
-------------------'
Dim Matlab
Set Matlab = CreateObject("Matlab.Application")

Matlab.Visible = false
'-------------------------------------------------------------------
-------------------'
' OptiSystem SDK specifics - access components and visualizers
'-------------------------------------------------------------------
-------------------'
```


```
Dim Component1
Set Component1 = Canvas1.GetComponentByName(ComponentName1)
Dim Component2
Set Component2 = Canvas1.GetComponentByName(ComponentName2)
Dim Visualizer1
Set Visualizer1 = Canvas1.GetComponentByName(VisualizerName1)
Dim Visualizer2
Set Visualizer2 = Canvas1.GetComponentByName(VisualizerName2)
Dim Visualizer3
Set Visualizer3 = Canvas1.GetComponentByName(VisualizerName3)
'-------------------------------------------------------------------
------------------'
' Calculation loop - access parameters and results
'-------------------------------------------------------------------
------------------'
For i = 0 to NumberOfSweepIterations1 - 1
    For j = 0 to NumberOfSweepIterations2 - 1
        'Parameter Values
        ParameterValue1 = ( ParameterStart1 + i * ParameterStep1 )
        ParameterValue2 = ( ParameterStart2 + j * ParameterStep2 )
        'Set component parameters
        Component1.SetParameterValue ParameterName1,
ParameterValue1 * 1.0
        Component2.SetParameterValue ParameterName2,
ParameterValue2 * 1.0
        'Calculate
        Document.CalculateProject TRUE , TRUE
        'Access visualizer results
        Set Result1 = Visualizer1.GetResult( ResultName1 )
        Set Result2 = Visualizer2.GetResult( ResultName2 )
        Set Result3 = Visualizer3.GetResult( ResultName3 )
        'Access result values
        ResultValue1 = Result1.GetValue( 1 )
        ResultValue2 = Result2.GetValue( 1 )
        ResultValue3 = Result3.GetValue( 1 )
        'Send parameters and results to Matlab
        Matlab.Execute( "P1( " + CStr( i + 1 ) + "," + CStr( j + 1
) + ")=" + CStr( ParameterValue1 ) + ";" )
        Matlab.Execute( "P2( " + CStr( i + 1 ) + "," + CStr( j + 1
) + ")=" + CStr( ParameterValue2 ) + ";" )
        Matlab.Execute( "R1( " + CStr( i + 1 ) + "," + CStr( j + 1
) + ")=" + CStr( ResultValue1 ) + ";" )
        Matlab.Execute( "R2( " + CStr( i + 1 ) + "," + CStr( j + 1
) + ")=" + CStr( ResultValue2 ) + ";" )
        Matlab.Execute( "R3( " + CStr( i + 1 ) + "," + CStr( j + 1
) + ")=" + CStr( ResultValue3 ) + ";" )
        Matlab.Execute("surf(P1,P2,R1)")
        Count = Count + 1
    Next
Next
'-------------------------------------------------------------------
------------------'
```

```
'Enable Matlab application - visible must be TRUE
'-----------------------------------------------------------------
-------------------'
Matlab.Visible = true
MsgBox("Close Matlab ?")
Random distributions using MatLab
'Get Layout Manager.
Dim Lm
Set Lm = Document.GetLayoutMgr
'SCRIPT for Layout 1
'Get Current Layout.
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Layout1.Name  = "Layout 1"
'Set Total Sweep Iterations
Layout1.SetTotalSweepIterations(1)
'Set Current Sweep Iteration
Layout1.SetCurrentSweepIteration(1)
'Get Current Canvas.
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
'SCRIPT for Layout global parameters.
Layout1.SetParameterMode "Simulation window", 0
Layout1.SetParameterValue "Simulation window", "Set bit rate"
Layout1.SetParameterMode "Reference bit rate", 0
Layout1.SetParameterValue "Reference bit rate", TRUE
Layout1.SetParameterMode "Bit rate", 0
Layout1.SetParameterValue "Bit rate", 1e+010
Layout1.SetParameterMode "Time window", 0
Layout1.SetParameterValue "Time window", 1.28e-008
Layout1.SetParameterMode "Sample rate", 0
Layout1.SetParameterValue "Sample rate", 6.4e+011
Layout1.SetParameterMode "Sequence length", 0
Layout1.SetParameterValue "Sequence length", 128
Layout1.SetParameterMode "Samples per bit", 0
Layout1.SetParameterValue "Samples per bit", 64
Layout1.SetParameterMode "Number of samples", 0
Layout1.SetParameterValue "Number of samples", 8192
Layout1.SetParameterMode "Iterations", 0
Layout1.SetParameterValue "Iterations", 1
Layout1.SetParameterMode "Parameterized", 0
Layout1.SetParameterValue "Parameterized", FALSE
Layout1.SetParameterMode "Convert noise bins", 0
Layout1.SetParameterValue "Convert noise bins", FALSE
Layout1.SetParameterMode "Calculate signal tracing", 0
Layout1.SetParameterValue "Calculate signal tracing", TRUE
Layout1.SetParameterMode "Power unit", 0
Layout1.SetParameterValue "Power unit", "dBm"
Layout1.SetParameterMode "Frequency unit", 0
Layout1.SetParameterValue "Frequency unit", "THz"
Layout1.SetParameterMode "Decimal places", 0
```


```
Layout1.SetParameterValue "Decimal places", 4
Layout1.SetParameterMode "Sensitivity", 0
Layout1.SetParameterValue "Sensitivity", -100
Layout1.SetParameterMode "Resolution", 0
Layout1.SetParameterValue "Resolution", 0.1
Layout1.SetParameterMode "Calculate noise floor", 0
Layout1.SetParameterValue "Calculate noise floor", FALSE
Layout1.SetParameterMode "Interpolation offset", 0
Layout1.SetParameterValue "Interpolation offset", 0.5
'SCRIPT for each component in the Layout.
'SCRIPT for component WDM Transmitter.
Dim Canvas1_Component1
Set Canvas1_Component1 = Canvas1.CreateComponent("WDM
Transmitter","{2A9D9567-99DE-4C83-8F05-720F1480B57E}",40,170, 34,
48,0)
Canvas1_Component1.Name  = "WDM Transmitter"
'Set WDM Transmitter parameters.
Canvas1_Component1.SetParameterMode "Number of output ports", 0
Canvas1_Component1.SetParameterValue "Number of output ports", 1
Canvas1_Component1.SetParameterMode "Frequency", 0
Canvas1_Component1.SetParameterUnit "Frequency", "THz"
Canvas1_Component1.SetParameterValue "Frequency", 193.1
Canvas1_Component1.SetParameterMode "Frequency spacing", 0
Canvas1_Component1.SetParameterUnit "Frequency spacing", "GHz"
Canvas1_Component1.SetParameterValue "Frequency spacing", 100
Canvas1_Component1.SetParameterMode "Power", 0
Canvas1_Component1.SetParameterUnit "Power", "dBm"
Canvas1_Component1.SetParameterValue "Power", 0
Canvas1_Component1.SetParameterMode "Extinction ratio", 0
Canvas1_Component1.SetParameterValue "Extinction ratio", 10
Canvas1_Component1.SetParameterMode "Linewidth", 0
Canvas1_Component1.SetParameterValue "Linewidth", 10
Canvas1_Component1.SetParameterMode "Initial phase", 0
Canvas1_Component1.SetParameterValue "Initial phase", 0
Canvas1_Component1.SetParameterMode "Bit rate", 3
Canvas1_Component1.SetParameterUnit "Bit rate", "Bits/s"
Canvas1_Component1.SetParameterScript "Bit rate", "Bit rate"
Canvas1_Component1.SetParameterMode "Order", 3
Canvas1_Component1.SetParameterScript "Order", "log(Sequence
length)/log(2)"
Canvas1_Component1.SetParameterMode "Number of leading zeros", 0
Canvas1_Component1.SetParameterValue "Number of leading zeros", 1
Canvas1_Component1.SetParameterMode "Number of trailing zeros", 0
Canvas1_Component1.SetParameterValue "Number of trailing zeros", 1
Canvas1_Component1.SetParameterMode "Modulation type", 0
Canvas1_Component1.SetParameterValue "Modulation type", "NRZ"
Canvas1_Component1.SetParameterMode "Duty cycle", 0
Canvas1_Component1.SetParameterValue "Duty cycle", 0.5
Canvas1_Component1.SetParameterMode "Position", 0
Canvas1_Component1.SetParameterValue "Position", 0
Canvas1_Component1.SetParameterMode "Rise time", 3
```

```
Canvas1_Component1.SetParameterUnit "Rise time", "s"
Canvas1_Component1.SetParameterScript "Rise time", "1 / (Bit rate )
* 0.05"
Canvas1_Component1.SetParameterMode "Fall time", 3
Canvas1_Component1.SetParameterUnit "Fall time", "s"
Canvas1_Component1.SetParameterScript "Fall time", "1 / (Bit rate )
* 0.05"
Canvas1_Component1.SetParameterMode "Transmitter type", 0
Canvas1_Component1.SetParameterValue "Transmitter type", "EML"
Canvas1_Component1.SetParameterMode "Overshoot", 0
Canvas1_Component1.SetParameterValue "Overshoot", 30
Canvas1_Component1.SetParameterMode "Undershoot", 0
Canvas1_Component1.SetParameterValue "Undershoot", 30
Canvas1_Component1.SetParameterMode "Damping time leading edge", 3
Canvas1_Component1.SetParameterUnit "Damping time leading edge", "s"
Canvas1_Component1.SetParameterScript "Damping time leading edge",
"1 / (Bit rate ) * 0.5"
Canvas1_Component1.SetParameterMode "Damping time trailing edge", 3
Canvas1_Component1.SetParameterUnit "Damping time trailing edge",
"s"
Canvas1_Component1.SetParameterScript "Damping time trailing edge",
"1 / (Bit rate ) * 0.5"
Canvas1_Component1.SetParameterMode "Resonant frequency leading
edge", 3
Canvas1_Component1.SetParameterUnit "Resonant frequency leading
edge", "Hz"
Canvas1_Component1.SetParameterScript "Resonant frequency leading
edge", "(Bit rate ) * 5"
Canvas1_Component1.SetParameterMode "Resonant frequency trailing
edge", 3
Canvas1_Component1.SetParameterUnit "Resonant frequency trailing
edge", "Hz"
Canvas1_Component1.SetParameterScript "Resonant frequency trailing
edge", "(Bit rate ) * 5"
Canvas1_Component1.SetParameterMode "Calculate side mode", 0
Canvas1_Component1.SetParameterValue "Calculate side mode", FALSE
Canvas1_Component1.SetParameterMode "Separation", 0
Canvas1_Component1.SetParameterUnit "Separation", "GHz"
Canvas1_Component1.SetParameterValue "Separation", 75
Canvas1_Component1.SetParameterMode "Suppression ratio", 0
Canvas1_Component1.SetParameterValue "Suppression ratio", 30
Canvas1_Component1.SetParameterMode "Include RIN", 0
Canvas1_Component1.SetParameterValue "Include RIN", FALSE
Canvas1_Component1.SetParameterMode "RIN", 0
Canvas1_Component1.SetParameterValue "RIN", -130
Canvas1_Component1.SetParameterMode "Measured power", 0
Canvas1_Component1.SetParameterUnit "Measured power", "dBm"
Canvas1_Component1.SetParameterValue "Measured power", 10
Canvas1_Component1.SetParameterMode "Alpha parameter", 0
Canvas1_Component1.SetParameterValue "Alpha parameter", 0
Canvas1_Component1.SetParameterMode "Adiabatic chirp", 0
Canvas1_Component1.SetParameterValue "Adiabatic chirp", 0
```


```
Canvas1_Component1.SetParameterMode "Azimuth", 0
Canvas1_Component1.SetParameterValue "Azimuth", 10
Canvas1_Component1.SetParameterMode "Ellipticity", 0
Canvas1_Component1.SetParameterValue "Ellipticity", 20
Canvas1_Component1.SetParameterMode "Polarization filter", 0
Canvas1_Component1.SetParameterValue "Polarization filter", "None"
Canvas1_Component1.SetParameterMode "Enabled", 0
Canvas1_Component1.SetParameterValue "Enabled", TRUE
Canvas1_Component1.SetParameterMode "Iterations", 3
Canvas1_Component1.SetParameterScript "Iterations", "Iterations"
Canvas1_Component1.SetParameterMode "Parameterized", 3
Canvas1_Component1.SetParameterScript "Parameterized",
"Parameterized"
Canvas1_Component1.SetParameterMode "Sample rate", 3
Canvas1_Component1.SetParameterUnit "Sample rate", "Hz"
Canvas1_Component1.SetParameterScript "Sample rate", "Sample rate"
Canvas1_Component1.SetParameterMode "Noise bandwidth", 3
Canvas1_Component1.SetParameterUnit "Noise bandwidth", "Hz"
Canvas1_Component1.SetParameterScript "Noise bandwidth", "Sample
rate"
Canvas1_Component1.SetParameterMode "Noise bins spacing", 3
Canvas1_Component1.SetParameterUnit "Noise bins spacing", "Hz"
Canvas1_Component1.SetParameterScript "Noise bins spacing", "Sample
rate"
Canvas1_Component1.SetParameterMode "Convert noise bins", 3
Canvas1_Component1.SetParameterScript "Convert noise bins",
"Convert noise bins"
Canvas1_Component1.SetParameterMode "Generate random seed", 0
Canvas1_Component1.SetParameterValue "Generate random seed", TRUE
Canvas1_Component1.SetParameterMode "Random seed index", 0
Canvas1_Component1.SetParameterValue "Random seed index", 0
'SCRIPT for component Photodetector PIN.
Dim Canvas1_Component6
Set Canvas1_Component6 = Canvas1.CreateComponent("Photodetector
PIN","{0B8011BF-3C6B-11D4-93EF-0050DAB7C5D6}",390,420, 34, 34,0)
Canvas1_Component6.Name  = "Photodetector PIN"
'Set Photodetector PIN parameters.
Canvas1_Component6.SetParameterMode "Responsivity", 0
Canvas1_Component6.SetParameterValue "Responsivity", 1
Canvas1_Component6.SetParameterMode "Dark current", 0
Canvas1_Component6.SetParameterValue "Dark current", 10
Canvas1_Component6.SetParameterMode "Centered at max power", 0
Canvas1_Component6.SetParameterValue "Centered at max power", TRUE
Canvas1_Component6.SetParameterMode "Center frequency", 0
Canvas1_Component6.SetParameterUnit "Center frequency", "THz"
Canvas1_Component6.SetParameterValue "Center frequency", 193.1
Canvas1_Component6.SetParameterMode "Sample rate", 3
Canvas1_Component6.SetParameterUnit "Sample rate", "Hz"
Canvas1_Component6.SetParameterScript "Sample rate", "5 * ( Sample
rate )"
Canvas1_Component6.SetParameterMode "Noise calculation type", 0
```

```
Canvas1_Component6.SetParameterValue "Noise calculation type",
"Numerical"
Canvas1_Component6.SetParameterMode "Add signal-ASE noise", 0
Canvas1_Component6.SetParameterValue "Add signal-ASE noise", TRUE
Canvas1_Component6.SetParameterMode "Add ASE-ASE noise", 0
Canvas1_Component6.SetParameterValue "Add ASE-ASE noise", TRUE
Canvas1_Component6.SetParameterMode "Add thermal noise", 0
Canvas1_Component6.SetParameterValue "Add thermal noise", TRUE
Canvas1_Component6.SetParameterMode "Thermal noise", 0
Canvas1_Component6.SetSweepParameterValue "Thermal noise",1,
6.48767e-023
Canvas1_Component6.SetParameterMode "Add shot noise", 0
Canvas1_Component6.SetParameterValue "Add shot noise", TRUE
Canvas1_Component6.SetParameterMode "Shot noise distribution", 0
Canvas1_Component6.SetParameterValue "Shot noise distribution",
"Gaussian"
Canvas1_Component6.SetParameterMode "Generate random seed", 0
Canvas1_Component6.SetParameterValue "Generate random seed", TRUE
Canvas1_Component6.SetParameterMode "Random seed index", 0
Canvas1_Component6.SetParameterValue "Random seed index", 0
'SCRIPT for component Low Pass Bessel Filter.
Dim Canvas1_Component8
Set Canvas1_Component8 = Canvas1.CreateComponent("Low Pass Bessel
Filter","{161B94D1-3BA4-11D4-93EE-0050DAB7C5D6}",500,420, 34, 34,0)
Canvas1_Component8.Name  = "Low Pass Bessel Filter"
'Set Low Pass Bessel Filter parameters.
Canvas1_Component8.SetParameterMode "Cutoff frequency", 3
Canvas1_Component8.SetParameterUnit "Cutoff frequency", "Hz"
Canvas1_Component8.SetParameterScript "Cutoff frequency", "0.75 *
Bit rate"
Canvas1_Component8.SetParameterMode "Insertion loss", 0
Canvas1_Component8.SetParameterValue "Insertion loss", 0
Canvas1_Component8.SetParameterMode "Depth", 0
Canvas1_Component8.SetParameterValue "Depth", 100
Canvas1_Component8.SetParameterMode "Order", 0
Canvas1_Component8.SetParameterValue "Order", 4
Canvas1_Component8.SetParameterMode "Enabled", 0
Canvas1_Component8.SetParameterValue "Enabled", TRUE
'SCRIPT for component 3R Regenerator.
Dim Canvas1_Component9
Set Canvas1_Component9 = Canvas1.CreateComponent("3R
Regenerator","{0B8011BF-3C6B-11D4-93EF-0050DAB7C5D6}",630,420, 34,
34,0)
Canvas1_Component9.Name  = "3R Regenerator"
'Set 3R Regenerator parameters.
Canvas1_Component9.SetParameterMode "Reference bit rate", 3
Canvas1_Component9.SetParameterUnit "Reference bit rate", "Bits/s"
Canvas1_Component9.SetParameterScript "Reference bit rate", "Bit
rate"
Canvas1_Component9.SetParameterMode "User defined delay", 0
Canvas1_Component9.SetParameterValue "User defined delay", FALSE
```


```
Canvas1_Component9.SetParameterMode "Delay compensation", 0
Canvas1_Component9.SetParameterUnit "Delay compensation", "s"
Canvas1_Component9.SetParameterValue "Delay compensation", 0
Canvas1_Component9.SetParameterMode "User defined decision", 0
Canvas1_Component9.SetParameterValue "User defined decision", FALSE
Canvas1_Component9.SetParameterMode "Decision instant", 0
Canvas1_Component9.SetParameterValue "Decision instant", 0.5
Canvas1_Component9.SetParameterMode "User defined threshold", 0
Canvas1_Component9.SetParameterValue "User defined threshold",
FALSE
Canvas1_Component9.SetParameterMode "Absolute threshold", 0
Canvas1_Component9.SetParameterValue "Absolute threshold", 0.5
'SCRIPT for component BER Analyzer.
Dim Canvas1_Component10
Set Canvas1_Component10 = Canvas1.CreateComponent("BER
Analyzer","{F11D0C25-3C7D-11D4-93F0-0050DAB7C5D6}",760,420, 40,
34,0)
Canvas1_Component10.Name  = "BER Analyzer"
'Set BER Analyzer parameters.
Canvas1_Component10.SetParameterMode "Algorithm", 0
Canvas1_Component10.SetParameterValue "Algorithm", "Gaussian"
Canvas1_Component10.SetParameterMode "Time window", 0
Canvas1_Component10.SetParameterValue "Time window", 1.5
Canvas1_Component10.SetParameterMode "Ignore start bits", 0
Canvas1_Component10.SetParameterValue "Ignore start bits", 1
Canvas1_Component10.SetParameterMode "Ignore end bits", 0
Canvas1_Component10.SetParameterValue "Ignore end bits", 1
Canvas1_Component10.SetParameterMode "Lower calculation limit", 0
Canvas1_Component10.SetParameterValue "Lower calculation limit", 0
Canvas1_Component10.SetParameterMode "Upper calculation limit", 0
Canvas1_Component10.SetParameterValue "Upper calculation limit", 1
Canvas1_Component10.SetParameterMode "Clock recovery", 0
Canvas1_Component10.SetParameterValue "Clock recovery", "On"
Canvas1_Component10.SetParameterMode "Enabled FEC gain estimation",
0
Canvas1_Component10.SetParameterValue "Enabled FEC gain
estimation", FALSE
Canvas1_Component10.SetParameterMode "Threshold mode", 0
Canvas1_Component10.SetParameterValue "Threshold mode", "Relative"
Canvas1_Component10.SetParameterMode "Absolute threshold", 0
Canvas1_Component10.SetParameterValue "Absolute threshold", 0
Canvas1_Component10.SetParameterMode "Relative threshold", 0
Canvas1_Component10.SetParameterValue "Relative threshold", 50
Canvas1_Component10.SetParameterMode "Decision instant", 0
Canvas1_Component10.SetParameterValue "Decision instant", 0.5
Canvas1_Component10.SetParameterMode "Load threshold from file", 0
Canvas1_Component10.SetParameterValue "Load threshold from file",
FALSE
Canvas1_Component10.SetParameterMode "Measured threshold filename",
0
```

```
Canvas1_Component10.SetParameterValue "Measured threshold
filename", "Threshold.dat"
Canvas1_Component10.SetParameterMode "Reload before calculation", 0
Canvas1_Component10.SetParameterValue "Reload before calculation",
FALSE
Canvas1_Component10.SetParameterMode "Time unit", 0
Canvas1_Component10.SetParameterValue "Time unit", "Bit period"
Canvas1_Component10.SetParameterMode "Ratio unit", 0
Canvas1_Component10.SetParameterValue "Ratio unit", "none"
Canvas1_Component10.SetParameterMode "Limit number of points", 0
Canvas1_Component10.SetParameterValue "Limit number of points", TRUE
Canvas1_Component10.SetParameterMode "Max. number of points", 0
Canvas1_Component10.SetParameterValue "Max. number of points",
128000
Canvas1_Component10.SetParameterMode "Calculate patterns", 0
Canvas1_Component10.SetParameterValue "Calculate patterns", FALSE
Canvas1_Component10.SetParameterMode "Number of points", 0
Canvas1_Component10.SetParameterValue "Number of points", 16
Canvas1_Component10.SetParameterMode "BER for pattern 1", 0
Canvas1_Component10.SetParameterValue "BER for pattern 1", 1e-012
Canvas1_Component10.SetParameterMode "BER for pattern 2", 0
Canvas1_Component10.SetParameterValue "BER for pattern 2", 1e-011
Canvas1_Component10.SetParameterMode "BER for pattern 3", 0
Canvas1_Component10.SetParameterValue "BER for pattern 3", 1e-010
Canvas1_Component10.SetParameterMode "BER for pattern 4", 0
Canvas1_Component10.SetParameterValue "BER for pattern 4", 1e-009
Canvas1_Component10.SetParameterMode "BER for pattern 5", 0
Canvas1_Component10.SetParameterValue "BER for pattern 5", 1e-008
Canvas1_Component10.SetParameterMode "Calculate 3D graph", 0
Canvas1_Component10.SetParameterValue "Calculate 3D graph", FALSE
Canvas1_Component10.SetParameterMode "Reference values setup", 0
Canvas1_Component10.SetParameterValue "Reference values setup",
"User defined"
Canvas1_Component10.SetParameterMode "Total power", 0
Canvas1_Component10.SetParameterValue "Total power", -1000
Canvas1_Component10.SetParameterMode "Signal power", 0
Canvas1_Component10.SetParameterValue "Signal power", -1000
Canvas1_Component10.SetParameterMode "Noise power", 0
Canvas1_Component10.SetParameterValue "Noise power", -1000
Canvas1_Component10.SetParameterMode "Min. BER", 0
Canvas1_Component10.SetParameterValue "Min. BER", 1
Canvas1_Component10.SetParameterMode "Q factor from min. BER", 0
Canvas1_Component10.SetParameterValue "Q factor from min. BER", 0
Canvas1_Component10.SetParameterMode "Max. Q factor", 0
Canvas1_Component10.SetParameterValue "Max. Q factor", 0
Canvas1_Component10.SetParameterMode "Max. eye height", 0
Canvas1_Component10.SetParameterValue "Max. eye height", 0
Canvas1_Component10.SetParameterMode "Max. eye amplitude", 0
Canvas1_Component10.SetParameterValue "Max. eye amplitude", 0
Canvas1_Component10.SetParameterMode "Max. eye closure", 0
Canvas1_Component10.SetParameterValue "Max. eye closure", 0
```


```
Canvas1_Component10.SetParameterMode "Max. eye opening factor", 0
Canvas1_Component10.SetParameterValue "Max. eye opening factor", 0
Canvas1_Component10.SetParameterMode "Extinction ratio at min. BER",
0
Canvas1_Component10.SetParameterValue "Extinction ratio at min.
BER", 0
Canvas1_Component10.SetParameterMode "Min. BER at user defined
decision instant", 0
Canvas1_Component10.SetParameterValue "Min. BER at user defined
decision instant", 1
Canvas1_Component10.SetParameterMode "Q factor from min. BER at user
defined decision instant", 0
Canvas1_Component10.SetParameterValue "Q factor from min. BER at
user defined decision instant", 0
Canvas1_Component10.SetParameterMode "Q factor at user defined
decision instant", 0
Canvas1_Component10.SetParameterValue "Q factor at user defined
decision instant", 0
Canvas1_Component10.SetParameterMode "BER at user defined
threshold", 0
Canvas1_Component10.SetParameterValue "BER at user defined
threshold", 1
Canvas1_Component10.SetParameterMode "Q factor from BER at user
defined threshold", 0
Canvas1_Component10.SetParameterValue "Q factor from BER at user
defined threshold", 0
Canvas1_Component10.SetParameterMode "BER at user defined decision
instant and threshold", 0
Canvas1_Component10.SetParameterValue "BER at user defined decision
instant and threshold", 1
Canvas1_Component10.SetParameterMode "Q factor from BER at user
defined decision instant and threshold", 0
Canvas1_Component10.SetParameterValue "Q factor from BER at user
defined decision instant and threshold", 0
Canvas1_Component10.SetParameterMode "Eye height at user defined
decision instant", 0
Canvas1_Component10.SetParameterValue "Eye height at user defined
decision instant", 0
Canvas1_Component10.SetParameterMode "Eye amplitude at user defined
decision instant", 0
Canvas1_Component10.SetParameterValue "Eye amplitude at user defined
decision instant", 0
Canvas1_Component10.SetParameterMode "Eye closure  at user defined
decision instant", 0
Canvas1_Component10.SetParameterValue "Eye closure  at user defined
decision instant", 0
Canvas1_Component10.SetParameterMode "Eye opening factor at user
defined decision instant", 0
Canvas1_Component10.SetParameterValue "Eye opening factor at user
defined decision instant", 0
Canvas1_Component10.SetParameterMode "Extinction ratio at user
defined decision instant", 0
```

```
Canvas1_Component10.SetParameterValue "Extinction ratio at user
defined decision instant", 0
Canvas1_Component10.SetParameterMode "Enabled", 0
Canvas1_Component10.SetParameterValue "Enabled", TRUE
Canvas1_Component10.SetParameterMode "Dynamic update", 0
Canvas1_Component10.SetParameterValue "Dynamic update", TRUE
Canvas1_Component10.SetParameterMode "Add noise to signal", 0
Canvas1_Component10.SetParameterValue "Add noise to signal", TRUE
Canvas1_Component10.SetParameterMode "Generate random seed", 0
Canvas1_Component10.SetParameterValue "Generate random seed", TRUE
Canvas1_Component10.SetParameterMode "Random seed index", 0
Canvas1_Component10.SetParameterValue "Random seed index", 0
'SCRIPT for component Optical Attenuator 1.
Dim Canvas1_Component2
Set Canvas1_Component2 = Canvas1.CreateComponent("Optical
Attenuator","{F11D0C07-3C7D-11D4-93F0-0050DAB7C5D6}",160,170, 20,
20,0)
Canvas1_Component2.Name  = "Optical Attenuator 1"
'Set Optical Attenuator 1 parameters.
Canvas1_Component2.SetParameterMode "Attenuation", 0
Canvas1_Component2.SetParameterValue "Attenuation", 2.9901
Canvas1_Component2.SetParameterMode "Enabled", 0
Canvas1_Component2.SetParameterValue "Enabled", TRUE
'SCRIPT for component Optical Fiber.
Dim Canvas1_Component3
Set Canvas1_Component3 = Canvas1.CreateComponent("Optical
Fiber","{416EC6F1-529F-11D4-9403-0050DAB7C5D6}",300,190, 32, 32,0)
Canvas1_Component3.Name  = "Optical Fiber"
'Set Optical Fiber parameters.
Canvas1_Component3.SetParameterMode "User defined reference
wavelength", 0
Canvas1_Component3.SetParameterValue "User defined reference
wavelength", FALSE
Canvas1_Component3.SetParameterMode "Reference wavelength", 0
Canvas1_Component3.SetParameterValue "Reference wavelength", 1550
Canvas1_Component3.SetParameterMode "Length", 0
Canvas1_Component3.SetParameterValue "Length", 50
Canvas1_Component3.SetParameterMode "Attenuation effect", 0
Canvas1_Component3.SetParameterValue "Attenuation effect", TRUE
Canvas1_Component3.SetParameterMode "Attenuation data type", 0
Canvas1_Component3.SetParameterValue "Attenuation data type",
"Constant"
Canvas1_Component3.SetParameterMode "Attenuation", 0
Canvas1_Component3.SetParameterValue "Attenuation", 0.2
Canvas1_Component3.SetParameterMode "Attenuation vs. wavelength", 0
Canvas1_Component3.SetParameterValue "Attenuation vs. wavelength",
"Attenuation.dat"
Canvas1_Component3.SetParameterMode "Group velocity dispersion", 0
Canvas1_Component3.SetParameterValue "Group velocity dispersion",
TRUE
Canvas1_Component3.SetParameterMode "Third-order dispersion", 0
Canvas1_Component3.SetParameterValue "Third-order dispersion", TRUE
```


```
Canvas1_Component3.SetParameterMode "Dispersion data type", 0
Canvas1_Component3.SetParameterValue "Dispersion data type",
"Constant"
Canvas1_Component3.SetParameterMode "Frequency domain parameters",
0
Canvas1_Component3.SetParameterValue "Frequency domain parameters",
FALSE
Canvas1_Component3.SetParameterMode "Dispersion", 0
Canvas1_Component3.SetParameterValue "Dispersion", 16.75
Canvas1_Component3.SetParameterMode "Dispersion slope", 0
Canvas1_Component3.SetParameterValue "Dispersion slope", 0.075
Canvas1_Component3.SetParameterMode "Beta 2", 0
Canvas1_Component3.SetParameterValue "Beta 2", -20
Canvas1_Component3.SetParameterMode "Beta 3", 0
Canvas1_Component3.SetParameterValue "Beta 3", 0
Canvas1_Component3.SetParameterMode "Dispersion file format", 0
Canvas1_Component3.SetParameterValue "Dispersion file format",
"Dispersion vs. wavelength"
Canvas1_Component3.SetParameterMode "Dispersion file name", 0
Canvas1_Component3.SetParameterValue "Dispersion file name",
"Dispersion.dat"
Canvas1_Component3.SetParameterMode "Birefringence type", 0
Canvas1_Component3.SetParameterValue "Birefringence type",
"Deterministic"
Canvas1_Component3.SetParameterMode "Differential group delay", 0
Canvas1_Component3.SetParameterValue "Differential group delay",
0.2
Canvas1_Component3.SetParameterMode "PMD coefficient", 0
Canvas1_Component3.SetParameterValue "PMD coefficient", 0.5
Canvas1_Component3.SetParameterMode "Mean scattering section
length", 0
Canvas1_Component3.SetParameterValue "Mean scattering section
length", 500
Canvas1_Component3.SetParameterMode "Scattering section
dispersion", 0
Canvas1_Component3.SetParameterValue "Scattering section
dispersion", 100
Canvas1_Component3.SetParameterMode "Self-phase modulation", 0
Canvas1_Component3.SetParameterValue "Self-phase modulation", FALSE
Canvas1_Component3.SetParameterMode "Effective area data type", 0
Canvas1_Component3.SetParameterValue "Effective area data type",
"Constant"
Canvas1_Component3.SetParameterMode "Effective area", 0
Canvas1_Component3.SetParameterValue "Effective area", 80
Canvas1_Component3.SetParameterMode "Effective area vs.
wavelength", 0
Canvas1_Component3.SetParameterValue "Effective area vs.
wavelength", "EffectiveAra.dat"
Canvas1_Component3.SetParameterMode "n2 data type", 0
Canvas1_Component3.SetParameterValue "n2 data type", "Constant"
Canvas1_Component3.SetParameterMode "n2", 0
Canvas1_Component3.SetParameterValue "n2", 2.6e-020
```

```
Canvas1_Component3.SetParameterMode "n2 vs. wavelength", 0
Canvas1_Component3.SetParameterValue "n2 vs. wavelength", "n2.dat"
Canvas1_Component3.SetParameterMode "Self-steepening", 0
Canvas1_Component3.SetParameterValue "Self-steepening", FALSE
Canvas1_Component3.SetParameterMode "Full Raman Response", 0
Canvas1_Component3.SetParameterValue "Full Raman Response", FALSE
Canvas1_Component3.SetParameterMode "Intrapulse Raman Scatt.", 0
Canvas1_Component3.SetParameterValue "Intrapulse Raman Scatt.",
FALSE
Canvas1_Component3.SetParameterMode "Raman self-shift time1", 0
Canvas1_Component3.SetParameterValue "Raman self-shift time1", 14.2
Canvas1_Component3.SetParameterMode "Raman self-shift time2", 0
Canvas1_Component3.SetParameterValue "Raman self-shift time2", 3
Canvas1_Component3.SetParameterMode "Fract. Raman contribution", 0
Canvas1_Component3.SetParameterValue "Fract. Raman contribution",
0.18
Canvas1_Component3.SetParameterMode "Orthogonal Raman factor", 0
Canvas1_Component3.SetParameterValue "Orthogonal Raman factor",
0.75
Canvas1_Component3.SetParameterMode "Model type", 0
Canvas1_Component3.SetParameterValue "Model type", "Scalar"
Canvas1_Component3.SetParameterMode "Propagator type", 0
Canvas1_Component3.SetParameterValue "Propagator type",
"Exponential"
Canvas1_Component3.SetParameterMode "Calculation type", 0
Canvas1_Component3.SetParameterValue "Calculation type",
"Noniterative"
Canvas1_Component3.SetParameterMode "Number of iterations", 0
Canvas1_Component3.SetParameterValue "Number of iterations", 2
Canvas1_Component3.SetParameterMode "Step size", 0
Canvas1_Component3.SetParameterValue "Step size", "Variable"
Canvas1_Component3.SetParameterMode "Max. nonlinear phase shift", 0
Canvas1_Component3.SetParameterValue "Max. nonlinear phase shift", 3
Canvas1_Component3.SetParameterMode "Boundary conditions", 0
Canvas1_Component3.SetParameterValue "Boundary conditions",
"Periodic"
Canvas1_Component3.SetParameterMode "Filter steepness", 0
Canvas1_Component3.SetParameterValue "Filter steepness", 0.05
Canvas1_Component3.SetParameterMode "Lower calculation limit", 0
Canvas1_Component3.SetParameterValue "Lower calculation limit",
1400
Canvas1_Component3.SetParameterMode "Upper calculation limit", 0
Canvas1_Component3.SetParameterValue "Upper calculation limit",
1700
Canvas1_Component3.SetParameterMode "Calculate graphs", 0
Canvas1_Component3.SetParameterValue "Calculate graphs", FALSE
Canvas1_Component3.SetParameterMode "Number of distance steps", 0
Canvas1_Component3.SetParameterValue "Number of distance steps", 200
Canvas1_Component3.SetParameterMode "Number of wavelength/time
steps", 0
Canvas1_Component3.SetParameterValue "Number of wavelength/time
steps", 200
```


```
Canvas1_Component3.SetParameterMode "Linear scale", 0
Canvas1_Component3.SetParameterValue "Linear scale", TRUE
Canvas1_Component3.SetParameterMode "Minimum value", 0
Canvas1_Component3.SetParameterValue "Minimum value", -100
Canvas1_Component3.SetParameterMode "Spectrum (total power) graph",
0
Canvas1_Component3.SetParameterValue "Spectrum (total power)
graph", FALSE
Canvas1_Component3.SetParameterMode "Spectrum (X component) graph",
0
Canvas1_Component3.SetParameterValue "Spectrum (X component)
graph", FALSE
Canvas1_Component3.SetParameterMode "Spectrum (Y component) graph",
0
Canvas1_Component3.SetParameterValue "Spectrum (Y component)
graph", FALSE
Canvas1_Component3.SetParameterMode "Waveform (total power) graph",
0
Canvas1_Component3.SetParameterValue "Waveform (total power)
graph", FALSE
Canvas1_Component3.SetParameterMode "Waveform (X component) graph",
0
Canvas1_Component3.SetParameterValue "Waveform (X component)
graph", FALSE
Canvas1_Component3.SetParameterMode "Waveform (Y component) graph",
0
Canvas1_Component3.SetParameterValue "Waveform (Y component)
graph", FALSE
Canvas1_Component3.SetParameterMode "Enabled", 0
Canvas1_Component3.SetParameterValue "Enabled", TRUE
Canvas1_Component3.SetParameterMode "Convert noise bins", 0
Canvas1_Component3.SetParameterValue "Convert noise bins", FALSE
Canvas1_Component3.SetParameterMode "Generate random seed", 0
Canvas1_Component3.SetParameterValue "Generate random seed", TRUE
Canvas1_Component3.SetParameterMode "Random seed index", 0
Canvas1_Component3.SetParameterValue "Random seed index", 0
'SCRIPT for component Optical Attenuator 2.
Dim Canvas1_Component4
Set Canvas1_Component4 = Canvas1.CreateComponent("Optical
Attenuator","{F11D0C07-3C7D-11D4-93F0-0050DAB7C5D6}",350,310, 20,
20,0)
Canvas1_Component4.Name  = "Optical Attenuator 2"
'Set Optical Attenuator 2 parameters.
Canvas1_Component4.SetParameterMode "Attenuation", 0
Canvas1_Component4.SetParameterValue "Attenuation", 2.9216
Canvas1_Component4.SetParameterMode "Enabled", 0
Canvas1_Component4.SetParameterValue "Enabled", TRUE
'SCRIPT for component Optical Power Meter Visualizer.
Dim Canvas1_Component5
Set Canvas1_Component5 = Canvas1.CreateComponent("Optical Power
Meter Visualizer","{F11D0C25-3C7D-11D4-93F0-0050DAB7C5D6}",520,270,
34, 38,0)
```

```
Canvas1_Component5.Name  = "Optical Power Meter Visualizer"
'Set Optical Power Meter Visualizer parameters.
Canvas1_Component5.SetParameterMode "Minimum value", 0
Canvas1_Component5.SetParameterValue "Minimum value", -100
Canvas1_Component5.SetParameterMode "Enabled", 0
Canvas1_Component5.SetParameterValue "Enabled", TRUE
Canvas1_Component5.SetParameterMode "Dynamic update", 0
Canvas1_Component5.SetParameterValue "Dynamic update", TRUE
'Set Total Sweep Iterations
Layout1.SetTotalSweepIterations(1)
'Set Current Sweep Iteration
Layout1.SetCurrentSweepIteration(1)
'Attach Monitors.
Canvas1_Component9.GetOutputPort(1).CreateMonitor
Canvas1_Component9.GetOutputPort(2).CreateMonitor
Canvas1_Component9.GetOutputPort(3).CreateMonitor
Canvas1_Component4.GetOutputPort(1).CreateMonitor
'Connecting components.
Canvas1_Component1.GetOutputPort(1).Connect(Canvas1_Component2.Get
InputPort(1))
Canvas1_Component6.GetOutputPort(1).Connect(Canvas1_Component8.Get
InputPort(1))
Canvas1_Component8.GetOutputPort(1).Connect(Canvas1_Component9.Get
InputPort(1))
Canvas1_Component9.GetOutputPort(1).ConnectVisualizer(Canvas1_Comp
onent10.GetInputPort(1))
Canvas1_Component9.GetOutputPort(2).ConnectVisualizer(Canvas1_Comp
onent10.GetInputPort(2))
Canvas1_Component9.GetOutputPort(3).ConnectVisualizer(Canvas1_Comp
onent10.GetInputPort(3))
Canvas1_Component2.GetOutputPort(1).Connect(Canvas1_Component3.Get
InputPort(1))
Canvas1_Component3.GetOutputPort(1).Connect(Canvas1_Component4.Get
InputPort(1))
Canvas1_Component4.GetOutputPort(1).Connect(Canvas1_Component6.Get
InputPort(1))
Canvas1_Component4.GetOutputPort(1).ConnectVisualizer(Canvas1_Comp
onent5.GetInputPort(1))
'------------------------------------------------------------------
-------------------'
'OptiSystem Script: Monte Carlo with 2 parameters and 3 results
'                        Parameter distribution from Matlab
'------------------------------------------------------------------
-------------------'
'------------------------------------------------------------------
-------------------'
'Global parameters - Parameter and Result names
'------------------------------------------------------------------
-------------------'
ComponentName1 = "Optical Attenuator 1"
ComponentName2 = "Optical Attenuator 2"
ParameterName1 = "Attenuation"
```


```
ParameterName2 = "Attenuation"
VisualizerName1 = "BER Analyzer"
VisualizerName2 = "BER Analyzer"
VisualizerName3 = "BER Analyzer"
ResultName1 = "Max. Q Factor"
ResultName2 = "Min. log of BER"
ResultName3 = "Max. Eye Closure (a.u.)"
'-------------------------------------------------------------------
-------------------'
'Global parameters - Number of runs and statistics of parameters
'-------------------------------------------------------------------
-------------------'
NumberOfRuns = 20
ParameterAverage1 = 3
ParameterSigma1 = 0.5
ParameterAverage2 = 3
ParameterSigma2 = 0.5
'-------------------------------------------------------------------
-------------------'
'Create Matlab application - visible must be FALSE
'-------------------------------------------------------------------
-------------------'
Dim Matlab
Set Matlab = CreateObject("Matlab.Application")
Matlab.Visible = false
'-------------------------------------------------------------------
-------------------'
' OptiSystem SDK specifics - access components and visualizers
'-------------------------------------------------------------------
-------------------'
Dim Component1
Set Component1 = Canvas1.GetComponentByName(ComponentName1)
Dim Component2
Set Component2 = Canvas1.GetComponentByName(ComponentName2)
Dim Visualizer1
Set Visualizer1 = Canvas1.GetComponentByName(VisualizerName1)
Dim Visualizer2
Set Visualizer2 = Canvas1.GetComponentByName(VisualizerName2)
Dim Visualizer3
Set Visualizer3 = Canvas1.GetComponentByName(VisualizerName3)
'-------------------------------------------------------------------
-------------------'
' Calculation loop - access parameters and results
'-------------------------------------------------------------------
-------------------'
For i = 0 to NumberOfRuns - 1
    Parameter Values
    ParameterValue1 = NormRnd( ParameterAverage1 , ParameterSigma1
)
    ParameterValue2 = NormRnd( ParameterAverage2 , ParameterSigma2
)
    'Set component parameters
```

```
    Component1.SetParameterValue ParameterName1, ParameterValue1 *
1.0
    Component2.SetParameterValue ParameterName2, ParameterValue2 *
1.0
    'Calculate
    Document.CalculateProject TRUE , TRUE
    'Access visualizer results
    Set Result1 = Visualizer1.GetResult( ResultName1 )
    Set Result2 = Visualizer2.GetResult( ResultName2 )
    Set Result3 = Visualizer3.GetResult( ResultName3 )
    'Access resuLt values
    ResultValue1 = Result1.GetValue( 1 )
    ResultValue2 = Result2.GetValue( 1 )
    ResultValue3 = Result3.GetValue( 1 )
    'Send parameters and results to Matlab
    Matlab.Execute( "P1( " + CStr( i + 1 ) + ")=" + CStr(
ParameterValue1 ) + ";" )
    Matlab.Execute( "P2( " + CStr( i + 1 ) + ")=" + CStr(
ParameterValue2 ) + ";" )
    Matlab.Execute( "R1( " + CStr( i + 1 ) + ")=" + CStr(
ResultValue1 ) + ";" )
    Matlab.Execute( "R2( " + CStr( i + 1 ) + ")=" + CStr(
ResultValue2 ) + ";" )
    Matlab.Execute( "R3( " + CStr( i + 1 ) + ")=" + CStr(
ResultValue3 ) + ";" )
Next

Matlab.Execute("hist(R1)")
'-------------------------------------------------------------------
------------------'
' Matlab function - generate a random number from a normal dist.
' NormRnd( Average , Sigma )
'-------------------------------------------------------------------
------------------'
Function NormRnd(Average,Sigma)
    MatlabReturn = Matlab.Execute( "normrnd(" + CStr( Average ) +
"," + CStr( Sigma ) + ")" )
    Pos = InStrRev( MatlabReturn , "=") + 1
    NormRnd = CDbl(Mid(MatlabReturn,Pos))
End Function
'-------------------------------------------------------------------
------------------'
'Enable Matlab application - visible must be TRUE
'-------------------------------------------------------------------
------------------'
Matlab.Visible = true
MsgBox("Close Matlab ?")
```


Exporting visualizer graphs to a file

Project **Nested Sweep Export Graph.osd** demonstrates how to export graphs from a visualizer to a text file. The OptiSystem script is provided bellow.

```
'--------------------------------------------------------------------'
'OptiSystem Script: Nested loops with 2 parameters and 1 graph
'                   Graphs can be exported to Text (multiple files)
'--------------------------------------------------------------------'

FolderName = "C:\Tests"

'--------------------------------------------------------------------'
'Global parameters - Parameter and Result names
'--------------------------------------------------------------------'
ComponentName1 = "NRZ Pulse Generator"
ComponentName2 = "Low Pass Bessel Filter"

ParameterName1 = "Amplitude"
ParameterName2 = "Insertion loss"

VisualizerName1 = "Oscilloscope Visualizer"

GraphName1 = "Signal + Noise Amplitude"

'--------------------------------------------------------------------'
'Global parameters - Sweep ranges for parameters
'--------------------------------------------------------------------'
NumberOfSweepIterations1 = 3
ParameterStart1 = 1
ParameterEnd1 = 10

NumberOfSweepIterations2 = 3
ParameterStart2 = 0
ParameterEnd2 = 20

'--------------------------------------------------------------------'
'Internal parameters - step for each sweep
'--------------------------------------------------------------------'

ParameterStep1 = ( ParameterEnd1 - ParameterStart1 ) / (
NumberOfSweepIterations1 - 1 )
ParameterStep2 = ( ParameterEnd2 - ParameterStart2 ) / (
NumberOfSweepIterations2 - 1 )

'--------------------------------------------------------------------'
'Create FileSystemObject
'--------------------------------------------------------------------'

Dim FileSystemObject
```

```
Dim OutF

Set FileSystemObject = CreateObject("Scripting.FileSystemObject")

'-------------------------------------------------------------------'
' OptiSystem SDK specifics - access components and visualizers
'-------------------------------------------------------------------'

Dim LayoutMgr
Set LayoutMgr = Document.GetLayoutMgr

Dim Layout
Set Layout = LayoutMgr.GetCurrentLayout

Dim Canvas
Set Canvas  = Layout.GetCurrentCanvas

Dim Component1
Set Component1 = Canvas.GetComponentByName(ComponentName1)

Dim Component2
Set Component2 = Canvas.GetComponentByName(ComponentName2)

Dim Visualizer1
Set Visualizer1 = Canvas.GetComponentByName(VisualizerName1)

'-------------------------------------------------------------------'
' Calculation loop - access parameters and results
'-------------------------------------------------------------------'

For i = 0 to NumberOfSweepIterations1 - 1

    For j = 0 to NumberOfSweepIterations2 - 1

            'Parameter Values
          ParameterValue1 = ( ParameterStart1 + i * ParameterStep1 )
        ParameterValue2 = ( ParameterStart2 + j * ParameterStep2 )

            'Set component parameters
            Component1.SetParameterValue ParameterName1,
ParameterValue1 * 1.0
            Component2.SetParameterValue ParameterName2,
ParameterValue2 * 1.0

  'Calculate
            Document.CalculateProject TRUE , TRUE

            'Create file name
            Filename = FolderName + "\graphs_" + CStr( i ) + "_" +
CStr( j ) + ".txt"
```


```
                'Create file
            Set OutF = FileSystemObject.CreateTextFile(Filename, True)

                'Acces visualizer graphs
                Dim Graph1
                Set Graph1 = Visualizer1.GetGraph( GraphName1 )

                For k = 0 to Graph1.GetNrOfPoints - 1

                    X = Graph1.GetXDataAt(k,1)
                    Y = Graph1.GetYDataAt(k,1)

                    'Send graph values to file

                    TextLine = ""
                    TextLine = TextLine + CStr( X ) + " "
                  TextLine = TextLine + CStr( Y )

                    OutF.WriteLine( TextLine )
                Next

                ' Close file
                OutF.Close
    Next
Next
Dim Lm
Set Lm = Document.GetLayoutMgr
Dim Layout1
Set Layout1 = Lm.GetCurrentLayout
Layout1.Name  = "Layout 1"
Layout1.SetTotalSweepIterations(1)
Layout1.SetCurrentSweepIteration(1)
Dim Canvas1
Set Canvas1 = Layout1.GetCurrentCanvas
Layout1.SetParameterMode "Simulation window", 0
Layout1.SetParameterValue "Simulation window", "Set bit rate"
Layout1.SetParameterMode "
```

Monte-Carlo simulations using MATLAB

Project **Random distributions using Matlab Monte Carlo.osd** demonstrates how to generate random numbers using MATLAB and how to export simulation results to MATLAB.

Nested sweeps using MATLAB

Project **Nested sweeps using Matlab Nested Sweep Matlab.osd** demonstrates how to simulate nested sweeps and export parameters and results to MATLAB.

Optiwave
7 Capella Court
Ottawa, Ontario, K2E 7X1, Canada

Tel.: 1.613.224.4700
Fax: 1.613.224.4706

E-mail: support@optiwave.com
URL: www.optiwave.com