Views

Functions

afx_msg void **OSViewUI::ShowFront** ()
This function displays the front view of the structure.

afx_msg void **OSViewUI::ShowBack** ()
This function displays the back view of the structure.

afx_msg void **OSViewUI::ShowRight** ()
This function displays the right view of the structure.

afx_msg void **OSViewUI::ShowLeft** ()
This function displays the left view of the structure.

afx_msg void **OSViewUI::ShowPlan** ()
This function displays the top (i.e., plan) view of the structure.

afx_msg void **OSViewUI::ShowBottom** ()
This function displays the bottom view of the structure.

afx_msg void **OSViewUI::ShowIsometric** ()
This function displays the isometric view of the structure.

afx_msg void **OSViewUI::RotateUp** (const VARIANT FAR &dDegrees)
Rotates the structure through Degrees about the Global X-Axis.

afx_msg void **OSViewUI::RotateDown** (const VARIANT FAR &dDegrees)
Rotates the structure through Degrees about the Global X-Axis.

afx_msg void **OSViewUI::RotateLeft** (const VARIANT FAR &dDegrees)
Rotates the structure through Degrees about the Global Y-Axis.

afx_msg void **OSViewUI::RotateRight** (const VARIANT FAR &dDegrees)
Rotates the structure through Degrees about the Global Y-Axis.

afx_msg void **OSViewUI::SpinLeft** (const VARIANT FAR &dDegrees)
Rotates the structure through Degrees about the Global Z-Axis.

afx_msg void **OSViewUI::SpinRight** (const VARIANT FAR &dDegrees)
Rotates the structure through Degrees about the Global Z-Axis.

afx_msg void **OSViewUI::ZoomAll** ()
zoom current view to Display the whole structure.

afx_msg void **OSViewUI::CreateNewViewForSelections** ()
Creates a new view in new window for the selected objects displayed in the active window.

afx_msg void **OSViewUI::SetLabel** (const VARIANT FAR &which, const VARIANT FAR &showFlag)
Sets the label on the structure diagram on or off.

afx_msg void	**OSViewUI::SetSectionView** (const VARIANT FAR &plane, const VARIANT FAR &minVal, const VARIANT FAR &maxVal) Creates a section view of the structure.
afx_msg void	**OSViewUI::SetDiagramMode** (const VARIANT FAR &which, const VARIANT FAR &showFlag, const VARIANT FAR &refreshFlag) Sets the label on the structure diagram on or off.
afx_msg void	**OSViewUI::RefreshView** () Refreshes the viewing window.
afx_msg void	**OSViewUI::SetNodeAnnotationMode** (const VARIANT FAR &dFlag, const VARIANT FAR &refreshFlag) Sets the node displacement annotation mode. This function works only in the post-processing mode of STAAD.Pro.
afx_msg void	**OSViewUI::SetReactionAnnotationMode** (const VARIANT FAR &dFlag, const VARIANT FAR &refreshFlag) Sets the node displacement annotation mode. This function works only in the post-processing mode of STAAD.Pro.
afx_msg VARIANT	**OSViewUI::GetInterfaceMode** () This function returns the current visual mode in the STAAD.Pro environment.
afx_msg VARIANT	**OSViewUI::SetInterfaceMode** (const VARIANT FAR &interfaceMode) This function sets the current visual mode in the STAAD.Pro environment.
afx_msg void	**OSViewUI::SetModeSectionPage** (const VARIANT FAR &interfaceMode, const VARIANT FAR &sectionNumber, const VARIANT FAR &pageNumber) This function sets the current page mode in the STAAD.Pro environment.
afx_msg void	**OSViewUI::SetBeamAnnotationMode** (const VARIANT FAR &Type, const VARIANT FAR &DWFlags, const VARIANT FAR &refreshFlag) This function sets the current page mode in the STAAD.Pro environment.
afx_msg void	**OSViewUI::ShowAllMembers** () Display all members of the current structure.
afx_msg void	**OSViewUI::HideAllMembers** () Hides all the members in the current structure.
afx_msg void	**OSViewUI::SetUnits** (const VARIANT FAR &uType, const VARIANT FAR &strUnit) Set viewing unit for the active view.
afx_msg void	**OSViewUI::ShowMembers** (const VARIANT FAR &nMembers, const VARIANT FAR &naMemberNos) Show the specified member.
afx_msg void	**OSViewUI::HideMembers** (const VARIANT FAR &nMembers, const VARIANT FAR &naMemberNos) Hide the specified member.
	owMember (const VARIANT FAR &nMember)

Show the specified member.

afx_msg void	**OSViewUI::HideMember** (const VARIANT FAR &nMember) Hide the specified member.
afx_msg void	**OSViewUI::HidePlate** (const VARIANT FAR &nPlate) Hide the specified plate.
afx_msg void	**OSViewUI::HideSolid** (const VARIANT FAR &nSolid) Hide the specified solid.
afx_msg void	**OSViewUI::HideSurface** (const VARIANT FAR &nSurface) Hide the specified surface.
afx_msg void	**OSViewUI::HideEntity** (const VARIANT FAR &nEntity) Hides the specified entity, which may be a Beam, Plate, Solid, or Surface.
afx_msg void	**OSViewUI::ZoomExtentsMainView** () Adjust viewing scale to zoom main view to the extents in new view.
afx_msg void	**OSViewUI::SelectMembersParallelTo** (const VARIANT FAR &bstrAxis) Select members parallel to the specified axis.
afx_msg VARIANT	**OSViewUI::SelectGroup** (const VARIANT FAR &bstrGroup) Select the relevant entities of the specified group.
afx_msg void	**OSViewUI::SelectInverse** (const VARIANT FAR &entityType) Inverse geometry selection for the specified entity.
afx_msg void	**OSViewUI::SelectByItemList** (const VARIANT FAR &entityType, const VARIANT FAR &nItems, const VARIANT FAR &itemList) Select entities as specified.
afx_msg void	**OSViewUI::SelectByMissingAttribute** (const VARIANT FAR &propCode) Select entity list for which specified entity is missing.
afx_msg void	**OSViewUI::SelectEntitiesConnectedToNode** (const VARIANT FAR &entityType, const VARIANT FAR &nodeNo) Select entities as specified in type and connected with the specified node.
afx_msg void	**OSViewUI::SelectEntitiesConnectedToMember** (const VARIANT FAR &entityType, const VARIANT FAR &memberNo) Select entities as specified in type and connected with the specified Member.
afx_msg void	**OSViewUI::SelectEntitiesConnectedToPlate** (const VARIANT FAR &entityType, const VARIANT FAR &plateNo) Select entities as specified in type and connected with the specified Plate.
afx_msg void	**OSViewUI::SelectEntitiesConnectedToSolid** (const VARIANT FAR &entityType, const VARIANT FAR &solidNo) Select entities as specified in type and connected with the specified Solid.
afx_msg VARIANT	**OSViewUI::GetNoOfBeamsInView** () Get No Of Beams In View.
afx_msg VARIANT	**OSViewUI::GetBeamsInView** (VARIANT FAR &nBeamList)

	Get Beams In View.
afx_msg VARIANT	**OSViewUI::CreateNewViewForSelectionsEx** (long windowOptions) Creates a new view for the selected objects displayed in the active window based on specified options.
afx_msg long	**OSViewUI::ExportView** (LPCTSTR fileLocation, LPCTSTR fileName, ImageExportTypes fileFormat, BOOL overwrite=TRUE) Used for exporting the information displayed in the active view window into a standard, graphical image format (e.g., Bitmap, JPEG, TIFF et.).
afx_msg VARIANT	**OSViewUI::CopyPicture** (long *xDim, long *yDim) Copy active view to clipbord and gives size of image in it's referance variables. It does not copy table views but works with Structure View, Graphs, 3d Rendered view.
afx_msg VARIANT	**OSViewUI::SetWindowPosition** (long xTop, long yTop, long xWindow, long yWindow) Resize and reposition active window. Note that, this function only sets windows such as they fully fit in the application desktop. See **OSViewUI::GetApplicationDesktopSize** to determine the size of the application desktop. This function resizes and changes position of windows in all state. if window is in minimized or maximized state it will restore it and reposition and resize it. API works with view window as well as the grid tables.
afx_msg VARIANT	**OSViewUI::GetApplicationDesktopSize** (long *xDim, long *yDim) This provides the dimension of the application desktop in pixels measured from the top left of view window to the bottom right. This is the grey zone on which all windows and dialogs in STAAD.Pro are placed. This should be used to determine the values to position the current application window using the function **OSViewUI::SetWindowPosition**.
afx_msg long	**OSViewUI::GetScaleValues** (VARIANT &ScalesArray) Obtain the current set of scales used for displaying loads and results shown in the Diagrams>Scales dialog. The values are provided in an array which should be sized using the value returned by the function **OSViewUI::GetScaleCount**. To obtain the value of only one specific scale item, see the function **OSViewUI::GetScaleValueByType**. Note that the values are returned in base units. See function OpenSTAADUI::GetBaseUnit.
afx_msg long	**OSViewUI::SetScaleValues** (const VARIANT &ScalesArray) Set the scales used for displaying loads and results as shown in the Diagrams>Scales dialog. The values should be provided in an array which should be sized using the value returned by the function **OSViewUI::GetScaleCount**. To set the value of only one specific scale item, see the function **OSViewUI::SetScaleValueByType**. Note that the values should be set in base units. See the function OpenSTAADUI::GetBaseUnit. To display/hide loads/results, see the function **OSViewUI::SetDiagramMode**. After setting the value, the display should be refreshed using the function **OSViewUI::RefreshView()**.
	ScaleValueByType (long scaleType, double *value)

	Obtain the value of the scale that is used to display a specified load or result diagram as defined in the Diagrams>Scales dialog depending upon the scale ID passed. Note that the values are returned in base units. See function OpenSTAADUI::GetBaseUnit. To obtain the value of all available scale items, see the function **OSViewUI::GetScaleValues**. .
afx_msg VARIANT	**OSViewUI::SetScaleValueByType** (long scaleType, double value) Set the scale used for displaying a chosen load or result diagram as shown in the Diagrams>Scales dialog. See also the function **OSViewUI::SetScaleValues** to set all scales in a single call. After setting the value, the display should be refreshed using the function **OSViewUI::RefreshView()**. To display/hide loads/results, see the function **OSViewUI::SetDiagramMode**. Note that the values are returned in base units. See function OpenSTAADUI::GetBaseUnit.
afx_msg long	**OSViewUI::GetScaleCount** () Returns the count of scales that are used in STAAD.Pro which can be read and set using the functions **OSViewUI::GetScaleValues** and **OSViewUI::SetScaleValues**.
afx_msg long	**OSViewUI::DetachView** () Remove a view from the collection of saved views. The view to be removed must be open and the active window. Once the view is detached, the active window becomes <untitled>. To set the active window see the function **OSViewUI::SetActiveWindow**. Note that the active window view cannot be Whole Structure or untitled.
afx_msg long	**OSViewUI::RenameView** (LPCTSTR viewName) Renames a saved view. The view should be open and be the active window. Note that the new name should not be the same as any other existing saved view. To rename as an existing saved view name, the existing view should first be detached. See the function **OSViewUI::DetachView**. Also the current window cannot be 'Whole Structure' or an untitled view. See the function **OSViewUI::SetActiveWindow** to ensure the view is set as the active window.
afx_msg long	**OSViewUI::OpenView** (LPCTSTR viewName, BOOL OpenInNewWindow) Open a previously saved view in either the active window or create a new window which becomes the active window.
afx_msg long	**OSViewUI::SaveView** (LPCTSTR viewName, BOOL overWriteIfExist) Save the active graphic view to the collection of saved views which can be opened using the function **OSViewUI::OpenView**.
afx_msg VARIANT	**OSViewUI::GetWindowTitle** (long id) Returns the Title of the Window.
afx_msg long	**OSViewUI::GetWindowCount** () Get the number of windows currently open. This includes both graphic windows and tables.
	seActiveWindow ()

	Closes the active graphic or table window, however there must be at least one graphic window remaining open. Note that window at position 1 can not be closed.
afx_msg VARIANT	**OSViewUI::SetActiveWindow** (long id) Set a given window (active graphic or table window) with the provided id as the active window. The indexing starts from 1. See the function **OSViewUI::GetWindowCount** to obtain the number of windows currently open.
afx_msg long	**OSViewUI::SetDesignResults** (long utilization, BOOL color, BOOL ShowValues) Sets Design Results to active view, this function replicates the setting of Design Results in the Diagrams>Design Results dialog.

Detailed Description

These functions are related to views.

Function Documentation

◆ CloseActiveWindow()

VARIANT OSViewUI::CloseActiveWindow ( )

protected

Closes the active graphic or table window, however there must be at least one graphic window remaining open. Note that window at position 1 can not be closed.

Return values

1/True Window closed.

0/FALSE Unsuccessful.

C++ Syntax

```
// Close active window.
OSViewUI::CloseActiveWindow();
```

C# Syntax

```
OpenSTAADUI.OpenSTAAD os = Marshal.GetActiveObject("StaadPro.OpenSTAAD") as
       OpenSTAADUI.OpenSTAAD;
OpenSTAADUI.OSViewUI output = os.View;
output.CloseActiveWindow();
```

VBA Syntax

```
Option Explicit
 Sub Main
     Dim objOpenStaad As Object
     Dim stdFile As String
     Dim result As VARIANT

     Set objOpenStaad = GetObject(,"StaadPro.OpenSTAAD")
     objOpenStaad.GetSTAADFile stdFile, "TRUE"
     If stdFile="" Then
         MsgBox"Bad"
         Set objOpenStaad = Nothing
         Exit Sub
     End If

     result = objOpenStaad.View.CloseActiveWindow()

     ' process the return value
        If (result) Then
            MsgBox "Window has been closed."
        Else
            MsgBox"unsuccessful"
        End If
     Set objOpenStaad = Nothing
 End Sub
```

See also

OSViewUI::OpenView , **OSViewUI::SetActiveWindow**.

- CopyPicture()

VARIANT OSViewUI::CopyPicture (long * xDim,
long * yDim)

protected

Copy active view to clipbord and gives size of image in it's referance variables. It does not copy table views but works with Structure View, Graphs, 3d Rendered view.

Parameters

[out] **xDim** size of image in x direction (Length in Pixel).

[out] **yDim** size of image in y direction (width in Pixel).

Return values

1/True Copy view is successful.

0/False Copy view is unsuccessful.

C++ Syntax

```
// Copy View to clipbord
OSViewUI::CopyPicture(&xDim, &yDim);
```

C# Syntax

```
OpenSTAADUI.OpenSTAAD os = Marshal.GetActiveObject("StaadPro.OpenSTAAD") as
       OpenSTAADUI.OpenSTAAD;
OpenSTAADUI.OSViewUI output = os.View;
output.CopyPicture(ref xDim, ref yDim);
```

VBA Syntax

```
Option Explicit
 Sub Main
     Dim objOpenStaad As Object
     Dim stdFile As String
     Dim varReturnVal As Boolean

     Set objOpenStaad = GetObject(,"StaadPro.OpenSTAAD")
     objOpenStaad.GetSTAADFile stdFile, "TRUE"
     If stdFile="" Then
         MsgBox"Bad"
         Set objOpenStaad = Nothing
         Exit Sub
     End If

     Dim xDim As Long, yDim As Long
     varReturnVal = objOpenStaad.View.CopyPicture(xDim, yDim)

     ' process the return value
        If (varReturnVal) Then
            MsgBox"Size of coppied diagram is '" & xDim & "' X '" & yDim & "'."
        Else
            MsgBox"Export view is unsuccessful"
        End If
        Set objOpenStaad = Nothing
```

```
End Sub
```

CreateNewViewForSelections()

void OSViewUI::CreateNewViewForSelections ( )

protected

Creates a new view in new window for the selected objects displayed in the active window.

C++ Syntax

```
OSViewUI::CreateNewViewForSelections();
```

VBA Syntax

```
Option Explicit

    Sub Main
        Dim objOpenStaad As Object
        Dim stdFile As String
        Dim varMembers(3) As Long

        ' launch STAAD.Pro application and open "US-1 Plane Frame with Steel Design.STD"
       file from the Samples folder and
        ' select any object(s) in the model, which will be shown in new view

        Set objOpenStaad = GetObject(,"StaadPro.OpenSTAAD")
        objOpenStaad.GetSTAADFile stdFile, "TRUE"
        If stdFile="" Then
            MsgBox"Bad"
            Set objOpenStaad = Nothing
            Exit Sub
        End If

        ' fill in the array with beam ids for selection
        varMembers(0) = 1
        varMembers(1) = 2
        varMembers(2) = 3
        varMembers(3) = 4

        ' select multiple beams.
        objOpenStaad.Geometry.SelectMultipleBeams(varMembers)

        ' call method CreateNewViewForSelections
        objOpenStaad.View.CreateNewViewForSelections()

        Set objOpenStaad = Nothing
    End Sub
```

See also

OSViewUI::CreateNewViewForSelectionsEx

- CreateNewViewForSelectionsEx()

VARIANT OSViewUI::CreateNewViewForSelectionsEx (long windowOptions) protected

Creates a new view for the selected objects displayed in the active window based on specified options.

Parameters

[in] **windowOptions** 0 = Creates a new window for the view, 1 = Display the view in the active window (type - Long).

Return values

1/True Creation of new view is successful.

0/False Creation of new view is unsuccessful.

C++ Syntax

```
OSViewUI::CreateNewViewForSelectionsEx(0 or 1);
```

VBA Syntax

```
Option Explicit

    Sub Main
        Dim objOpenStaad As Object
        Dim stdFile As String
        Dim varReturnVal As Boolean
        Dim varMembers(3) As Long

        ' launch STAAD.Pro application and open "US-1 Plane Frame with Steel Design.STD"
       file from the Samples folder and
        ' select any object(s) in the model, which will be shown in new view

        Set objOpenStaad = GetObject(,"StaadPro.OpenSTAAD")
        objOpenStaad.GetSTAADFile stdFile, "TRUE"
        If stdFile="" Then
            MsgBox"Bad"
            Set objOpenStaad = Nothing
            Exit Sub
        End If

        ' fill in the array with beam ids for selection
        varMembers(0) = 1
        varMembers(1) = 2
        varMembers(2) = 3
        varMembers(3) = 4

        ' select multiple beams.
        objOpenStaad.Geometry.SelectMultipleBeams(varMembers)

        ' call method CreateNewViewForSelectionsEx(0 or 1)
        varReturnVal = objOpenStaad.View.CreateNewViewForSelectionsEx(1)

        ' process the return value
        If (varReturnVal) Then
            MsgBox"Create new view is successful"
        Else
            MsgBox"Create new view is unsuccessful"
```

```
        End If

        Set objOpenStaad = Nothing
    End Sub
```

See also

OSViewUI::CreateNewViewForSelections

DetachView()

long OSViewUI::DetachView ( ) protected

Remove a view from the collection of saved views. The view to be removed must be open and the active window. Once the view is detached, the active window becomes <untitled>. To set the active window see the function **OSViewUI::SetActiveWindow**. Note that the active window view cannot be Whole Structure or untitled.

Return values

1 View successfully detached.

0 Unsuccessful

-1 Generic Error

C++ Syntax

```
OSViewUI::DetachView();
```

C# Syntax

```
OpenSTAADUI.OpenSTAAD os = Marshal.GetActiveObject("StaadPro.OpenSTAAD") as
        OpenSTAADUI.OpenSTAAD;
OpenSTAADUI.OSViewUI output = os.View;
output.DetachView();
```

VBA Syntax

```
Option Explicit

    Sub Main
        Dim objOpenStaad As Object
        Dim stdFile As String
        Dim varReturnVal As Long

        Set objOpenStaad = GetObject(,"StaadPro.OpenSTAAD")
        objOpenStaad.GetSTAADFile stdFile, "TRUE"

        varReturnVal = objOpenStaad.View.DetachView()

        ' process the return value
        If (varReturnVal = 1) Then
            MsgBox"Detach view is successful"
        Else
            MsgBox"Detach view is unsuccessful"
        End If

        Set objOpenStaad = Nothing
    End Sub
```

◆ ExportView()

```
long OSViewUI::ExportView ( LPCTSTR          fileLocation,
                            LPCTSTR          fileName,
                            ImageExportTypes fileFormat,
                            BOOL             overwrite = TRUE )
```

protected

Used for exporting the information displayed in the active view window into a standard, graphical image format (e.g., Bitmap, JPEG, TIFF et.).

Parameters

[in] **FileLocation** - Location of the saved view file (type - LPCTSTR, Folder need to be present otherwise it will return -100) .

[in] **FileName** - Name of the saved view file(type - LPCTSTR).

[in] **FileFormat** 0 = bmp, 1 = jpg, 2 = tga, 3 = tif - Create the view in the specific format (type - Long).

[in] **Overwrite** - Boolean for provide option to overwrite an existing file.
True - Allow Overwrite
False - No overwrite

Return values

1 Export view is successful.

-1 Generic Error.

-100 Invalid Argument.

-1003 File already exist.

C++ Syntax

```
OSViewUI::ExportView(fileLoaction, fileName, fileFormat, overwrite);
```

C# Syntax

```
OpenSTAADUI.OpenSTAAD os = Marshal.GetActiveObject("StaadPro.OpenSTAAD") as
        OpenSTAADUI.OpenSTAAD;
OpenSTAADUI.OSViewUI output = os.View;
output.ExportView(fileLoaction, fileName, OSViewUI.ImageExportTypes.bmp, overwrite);
```

VBA Syntax

```
Option Explicit

    Sub Main
        Dim objOpenStaad As Object
        Dim stdFile As String
        Dim varReturnVal As Long

        ' launch STAAD.Pro application and open "US-1 Plane Frame with Steel Design.STD"
        file from the Samples folder and
        iew
```

Loading [MathJax]/extensions/MathZoom.js

```
        Set objOpenStaad = GetObject(,"StaadPro.OpenSTAAD")
        objOpenStaad.GetSTAADFile stdFile, "TRUE"
        If stdFile="" Then
            MsgBox"Bad"
            Set objOpenStaad = Nothing
            Exit Sub
        End If

        Dim fileLoc As String
        Dim fileName As String
        Dim fileType As Long
        Dim overWrite As Boolean

        fileLoc = "D:\New Folder\"
        fileName = "US-1 Plane Frame with Steel Design"
        fileType = 1
        overWrite = True
        varReturnVal = objOpenStaad.View.ExportView(fileLoc, fileName, fileType,
       overWrite)

        ' process the return value
        If (varReturnVal = 1) Then
            MsgBox"Export view is successful"
        Else
            MsgBox"Export view is unsuccessful"
        End If

        Set objOpenStaad = Nothing
    End Sub
```

GetApplicationDesktopSize()

VARIANT OSViewUI::GetApplicationDesktopSize (long * xDim,
long * yDim) protected

This provides the dimension of the application desktop in pixels measured from the top left of view window to the bottom right. This is the grey zone on which all windows and dialogs in STAAD.Pro are placed. This should be used to determine the values to position the current application window using the function **OSViewUI::SetWindowPosition**.

Parameters

[out] **xDim** size of image in x direction (Length in pixel).

[out] **yDim** size of image in y direction (width in pixel).

Return values

1/True Successfully obtained the dimensions of the application desktop..

0/False Failed to obtain the dimensions of the application desktop.

C++ Syntax

```
// Get available size.
OSViewUI::GetApplicationDesktopSize(&xDim, &yDim);
```

C# Syntax

```
OpenSTAADUI.OpenSTAAD os = Marshal.GetActiveObject("StaadPro.OpenSTAAD") as
        OpenSTAADUI.OpenSTAAD;
OpenSTAADUI.OSViewUI output = os.View;
output.GetApplicationDesktopSize(ref xDim, ref yDim);
```

VBA Syntax

```
Option Explicit
 Sub Main
     Dim objOpenStaad As Object
     Dim stdFile As String
     Dim varReturnVal As Boolean

     Set objOpenStaad = GetObject(,"StaadPro.OpenSTAAD")
     objOpenStaad.GetSTAADFile stdFile, "TRUE"
     If stdFile="" Then
         MsgBox"Bad"
         Set objOpenStaad = Nothing
         Exit Sub
     End If

     Dim xDim As Long, yDim As Long
     varReturnVal = objOpenStaad.View.GetApplicationDesktopSize(xDim, yDim)

     ' process the return value
        If (varReturnVal) Then
                                of client area is '" & xDim & "' X '" & yDim & "'."
```

```
            MsgBox"Export view is unsuccessful"
        End If
    Set objOpenStaad = Nothing
End Sub
```

GetBeamsInView()

VARIANT OSViewUI::GetBeamsInView (VARIANT FAR & nBeamList) protected

Get Beams In View.

Parameters

[in] **nBeamList** Collection of beam

C++ Syntax

```
// Get Beams In View
VARIANT RetVal = OSViewUI::GetBeamsInView(nBeamList);
```

VBA Syntax

```
' Get Beams In View
Dim RetVal As VARIANT = OSViewUI.GetBeamsInView(nBeamList)
```

GetInterfaceMode()

VARIANT OSViewUI::GetInterfaceMode ( ) protected

This function returns the current visual mode in the STAAD.Pro environment.

Return values

0 Pre-processor or modeling mode.

1 Post-processing mode.

2 Interactive design mode for STAAD.etc interoperability.

4 Piping mode.

5 BEAVA (i.e., Bridge Deck) mode.

C++ Syntax

```
// Returns the current visual mode in the STAAD.Pro environment.
VARIANT RetVal = OSViewUI::GetInterfaceMode();
```

VBA Syntax

```
' Returns the current visual mode in the STAAD.Pro environment.
Dim RetVal As VARIANT = OSViewUI.GetInterfaceMode()
```

◆ GetNoOfBeamsInView()

VARIANT OSViewUI::GetNoOfBeamsInView ( ) protected

Get No Of Beams In View.

C++ Syntax

```
// Get No Of Beams In View
VARIANT RetVal = OSViewUI::GetNoOfBeamsInView();
```

VBA Syntax

```
' Get No Of Beams In View
Dim RetVal As VARIANT = OSViewUI.GetNoOfBeamsInView()
```

◆ GetScaleCount()

long OSViewUI::GetScaleCount ( )

protected

Returns the count of scales that are used in STAAD.Pro which can be read and set using the functions **OSViewUI::GetScaleValues** and **OSViewUI::SetScaleValues**.

Return values

Positive_Value Count of scales.

0/Negative_Value Unsuccessful.

C++ Syntax

```
// Get available size.
long sizeOfArray = OSViewUI::GetScaleCount();
```

C# Syntax

```
OpenSTAADUI.OpenSTAAD os = Marshal.GetActiveObject("StaadPro.OpenSTAAD") as
       OpenSTAADUI.OpenSTAAD;
OpenSTAADUI.OSViewUI output = os.View;
long sizeOfArray = output.GetScaleCount();
```

VBA Syntax

```
Option Explicit
 Sub Main
     Dim objOpenStaad As Object
     Dim stdFile As String
     Dim sizeOfArray As Long

     Set objOpenStaad = GetObject(,"StaadPro.OpenSTAAD")
     objOpenStaad.GetSTAADFile stdFile, "TRUE"
     If stdFile="" Then
         MsgBox"Bad"
         Set objOpenStaad = Nothing
         Exit Sub
     End If

     sizeOfArray = objOpenStaad.View.GetScaleCount()

     MsgBox "Size of Scales array is '" & sizeOfArray & "'."

     Set objOpenStaad = Nothing
 End Sub
```

See also

OSViewUI::GetScaleValues , **OSViewUI::SetScaleValues** , **OSViewUI::GetScaleValueByType** , **OSViewUI::SetScaleValueByType**

VARIANT OSViewUI::GetScaleValueByType (long scaleTypeId,
double * value) protected

Obtain the value of the scale that is used to display a specified load or result diagram as defined in the Diagrams>Scales dialog depending upon the scale ID passed.
Note that the values are returned in base units. See function OpenSTAADUI::GetBaseUnit.
To obtain the value of all available scale items, see the function **OSViewUI::GetScaleValues**.
.

Parameters

[in] **scaleTypeId** The index of the required load or result type (Type: Long). Refer to the function **OSViewUI::GetScaleValues** for the ID list.

[out] **value** Value of scale type listed, in Base Units (Type: Double).

Return values

1/TRUE Value was successfully retrieved.

0/FALSE Value could not be retrieved.

C++ Syntax

```
// Get available size.
OSViewUI::GetScaleValueByType(scaleTypeId, &value);
```

C# Syntax

```
OpenSTAADUI.OpenSTAAD os = Marshal.GetActiveObject("StaadPro.OpenSTAAD") as
       OpenSTAADUI.OpenSTAAD;
OpenSTAADUI.OSViewUI output = os.View;
output.GetScaleValueByType(scaleTypeId, ref value);
```

VBA Syntax

```
Option Explicit
 Sub Main
     Dim objOpenStaad As Object
     Dim stdFile As String
     Dim varReturnVal As Variant

     Set objOpenStaad = GetObject(,"StaadPro.OpenSTAAD")
     objOpenStaad.GetSTAADFile stdFile, "TRUE"
     If stdFile="" Then
         MsgBox"Bad"
         Set objOpenStaad = Nothing
         Exit Sub
     End If

     Dim value As Double
     Dim scaleTypeId As Long

                              penStaad.View.GetScaleValueByType(scaleTypeId, value)
```

```
    ' process the return value
       If (varReturnVal) Then
           MsgBox "Scale of Point Force is '" & value & "'."
       Else
           MsgBox "Unsuccessful."
       End If
    Set objOpenStaad = Nothing
End Sub
```

See also

OSViewUI::GetScaleValues , **OSViewUI::GetScaleCount** , **OSViewUI::SetScaleValues** , **OSViewUI::SetScaleValueByType**

GetScaleValues()

long OSViewUI::GetScaleValues (VARIANT & ScalesArray) protected

Obtain the current set of scales used for displaying loads and results shown in the Diagrams>Scales dialog. The values are provided in an array which should be sized using the value returned by the function **OSViewUI::GetScaleCount**.
To obtain the value of only one specific scale item, see the function **OSViewUI::GetScaleValueByType**.
Note that the values are returned in base units. See function OpenSTAADUI::GetBaseUnit.

Parameters

[out] **ScalesArray** Array of double type and size same as number of scale types. API gets all the values in this array. see **OSViewUI::GetScaleCount** for getsize of scalesArray. Scale type and ID in array will be as following.

ID	Type	Scale Items	Unit per length
0	Loads	Point Force	Force
1	Loads	Dist. Force	Force/length
2	Loads	Point Moment	Force*length
3	Loads	Dist. Moment	Force*length/length
4	Loads	Pressure	Force/length^2
5	Results	Bending Y	Force*length
6	Results	Bending Z	Force*length
7	Results	Shear Y	Force
8	Results	Shear Z	Force
9	Results	Axial	Force
10	Results	Torsion	Force*length
11	Results	Displacement	Length
12	Results	Beam Stress	Force/length^2
13	Results	Mode Shape	(none)

Return values

1 Values were successfully obtained.

0 Values could not be obtained.

-1 Error with defined array.

C++ Syntax

```
// Get available size.
OSViewUI::GetScaleValues(&ScalesArray);
```

```
OpenSTAADUI.OpenSTAAD os = Marshal.GetActiveObject("StaadPro.OpenSTAAD") as
       OpenSTAADUI.OpenSTAAD;
OpenSTAADUI.OSViewUI output = os.View;
int nScaleCount = output.GetScaleCount();
double[] ScalesArray = new double[nScaleCount]
int result = output.GetScaleValues(ref ScalesArray);
```

VBA Syntax

```
Option Explicit
 Sub Main
     Dim objOpenStaad As Object
     Dim stdFile As String
     Dim varReturnVal As Long
     Dim scaleSize As Long
     Dim ScalesArray() As Double

     Set objOpenStaad = GetObject(,"StaadPro.OpenSTAAD")
     objOpenStaad.GetSTAADFile stdFile, "TRUE"
     If stdFile="" Then
         MsgBox"Bad"
         Set objOpenStaad = Nothing
         Exit Sub
     End If

     scaleSize = objOpenStaad.View.GetScaleCount()
     ReDim ScalesArray(scaleSize)
     varReturnVal = objOpenStaad.View.GetScaleValues(ScalesArray)

     ' process the return value
        If (varReturnVal = 1) Then
            MsgBox"Scale of Point Force is '" & ScalesArray(0) & "'."
        Else
            MsgBox"Unsuccessful"
        End If
     Set objOpenStaad = Nothing
 End Sub
```

See also

OSViewUI::GetScaleValueByType , OSViewUI::GetScaleCount , OSViewUI::SetScaleValues , OSViewUI::SetScaleValueByType

◆ GetWindowCount()

long OSViewUI::GetWindowCount ( ) protected

Get the number of windows currently open. This includes both graphic windows and tables.

Return values

Positive_Number	The count of open Window.
0	Error.

C++ Syntax

```
// Get window count.
int count = OSViewUI::GetWindowCount();
```

C# Syntax

```
OpenSTAADUI.OpenSTAAD os = Marshal.GetActiveObject("StaadPro.OpenSTAAD") as
        OpenSTAADUI.OpenSTAAD;
OpenSTAADUI.OSViewUI output = os.View;
int count = output.GetWindowCount();
```

VBA Syntax

```
Option Explicit
 Sub Main
     Dim objOpenStaad As Object
     Dim stdFile As String
     Dim count As Long

     Set objOpenStaad = GetObject(,"StaadPro.OpenSTAAD")
     objOpenStaad.GetSTAADFile stdFile, "TRUE"
     If stdFile="" Then
         MsgBox"Bad"
         Set objOpenStaad = Nothing
         Exit Sub
     End If

     count = objOpenStaad.View.GetWindowCount()

     ' process the return value
        If (count) Then
            MsgBox "Number of window with opened in STAAD.Pro: '" & count & "'."
        Else
            MsgBox"unsuccessful"
        End If
     Set objOpenStaad = Nothing
 End Sub
```

See also

OSViewUI::GetWindowTitle , **OSViewUI::SetActiveWindow**.

◆ GetWindowTitle()

VARIANT OSViewUI::GetWindowTitle (long id)

protected

Returns the Title of the Window.

Parameters

[in] **ID** The index of the required Window (Type: Long). Note that IDs start from 1. Window IDs depend on the creation time; that is the last created window will have the last ID. Windows which are listed under in View tab > Windows Dropdown are supported in this API.

Return values

<VARIANT> The Window string title.

Empty_String Window **id** not found.

C++ Syntax

```
// Get available size.
name = OSViewUI::GetWindowTitle(Id);
```

C# Syntax

```
OpenSTAADUI.OpenSTAAD os = Marshal.GetActiveObject("StaadPro.OpenSTAAD") as
       OpenSTAADUI.OpenSTAAD;
OpenSTAADUI.OSViewUI output = os.View;
String name = output.GetWindowTitle(Id);
```

VBA Syntax

```
Option Explicit
 Sub Main
     Dim objOpenStaad As Object
     Dim stdFile As String
     Dim title As String
    Dim id As Long

     Set objOpenStaad = GetObject(,"StaadPro.OpenSTAAD")
     objOpenStaad.GetSTAADFile stdFile, "TRUE"
     If stdFile="" Then
         MsgBox"Bad"
         Set objOpenStaad = Nothing
         Exit Sub
     End If

    id = 1
     title = objOpenStaad.View.GetWindowTitle(id)

     ' process the return value
        If (title <> "") Then
            MsgBox "Title of the window with ID 1 is '" & title & "'."
        Else
            MsgBox"unsuccessful"
        End If
Nothing
```

```
End Sub
```

See also

OSViewUI::GetWindowCount , **OSViewUI::SetActiveWindow**.

◆ HideAllMembers()

void OSViewUI::HideAllMembers ( ) protected

Hides all the members in the current structure.

C++ Syntax

```
// Hides all the members in the current structure.
OSViewUI::HideAllMembers();
```

VBA Syntax

```
' Hides all the members in the current structure.
OSViewUI.HideAllMembers()
```

◆ HideEntity()

void OSViewUI::HideEntity (const VARIANT FAR & nEntity) protected

Hides the specified entity, which may be a Beam, Plate, Solid, or Surface.

Parameters

[in] **nEntity** Variable that holds an entity (i.e., Member, Plates etc.) number to be hidden.

C++ Syntax

```
// Hide Entity
OSViewUI::HideEntity(nEntity);
```

VBA Syntax

```
' Hide Entity
OSViewUI.HideEntity(nEntity)
```

void OSViewUI::HideMember (const VARIANT FAR & nMember) protected

Hide the specified member.

Parameters

[in] **nMember** Variable that holds member number to be hidden.

C++ Syntax

```
// Hide Member
OSViewUI::HideMember(5);
```

VBA Syntax

```
' Hide Member
OSViewUI.HideMember(5)
```

◆ HideMembers()

void OSViewUI::HideMembers (const VARIANT FAR & nMembers,
const VARIANT FAR & naMemberNos) protected

Hide the specified member.

Parameters

[in] **nMembers** Variable that holds number of members to hide.

[in] **naMemberNos** Variable array that holds the member nos, which need to be hidden.

C++ Syntax

```
// Hide Member
OSViewUI::HideMembers(5, naMemberNos);
```

VBA Syntax

```
' Hide Member
OSViewUI.HideMembers(5, naMemberNos)
```

◆ HidePlate()

void OSViewUI::HidePlate (const VARIANT FAR & nPlate) protected

Hide the specified plate.

Parameters

[in] **nPlate** Variable that holds plate number to be hidden.

C++ Syntax

```
// Hide Plate
OSViewUI::HidePlate(nPlate);
```

VBA Syntax

```
' Hide Plate
OSViewUI.HidePlate(nPlate)
```

◆ HideSolid()

void OSViewUI::HideSolid (const VARIANT FAR & nSolid) protected

Hide the specified solid.

Parameters

[in] **nSolid** Variable that holds solid number to be hidden.

C++ Syntax

```
// Hide Solid
OSViewUI::HideSolid(nSolid);
```

VBA Syntax

```
' Hide Solid
OSViewUI.HideSolid(nSolid)
```

◆ HideSurface()

void OSViewUI::HideSurface (const VARIANT FAR & nSurface) protected

Hide the specified surface.

Parameters

[in] **nSurface** Variable that holds surface number to be hidden.

C++ Syntax

```
// Hide Surface
OSViewUI::HideSurface(nSurface);
```

VBA Syntax

```
' Hide Surface
OSViewUI.HideSurface(nSurface)
```

◆ OpenView()

long OSViewUI::OpenView (LPCTSTR viewName,
BOOL windowOptions) protected

Open a previously saved view in either the active window or create a new window which becomes the active window.

Parameters

[in] **viewName** - New name of the saved view (type - LPCTSTR).

[in] **windowOptions** - False / 0 = Creates a new window for the view which becomes the active window,
True / 1 = Display the view in the current active window (type - BOOL).

Return values

1 View Successfully opened.

0 Unsuccessful

2 View name does not exist.

-1 Generic Error

-100 Invalid Argument

C++ Syntax

```
OSViewUI::OpenView(viewName, windowOptions);
```

C# Syntax

```
OpenSTAADUI.OpenSTAAD os = Marshal.GetActiveObject("StaadPro.OpenSTAAD") as
        OpenSTAADUI.OpenSTAAD;
OpenSTAADUI.OSViewUI output = os.View;
string newName = "view1";
BOOL windowOptions = TRUE;
output.OpenView(newName, windowOptions);
```

VBA Syntax

```
Option Explicit

    Sub Main
        Dim objOpenStaad As Object
        Dim stdFile As String
        Dim varReturnVal As Long
        Dim nName As String
        Dim windowOptions As Boolean

        Set objOpenStaad = GetObject(,"StaadPro.OpenSTAAD")
        objOpenStaad.GetSTAADFile stdFile, "TRUE"

        nName = "view1"
        windowOptions = True
        varReturnVal = objOpenStaad.View.OpenView(nName, windowOptions)
```

```
        ' process the return value
        If (varReturnVal = 1) Then
            MsgBox"Open view is successful"
        Else
            MsgBox"Open view is unsuccessful"
        End If

        Set objOpenStaad = Nothing
    End Sub
```

◆ RefreshView()

void OSViewUI::RefreshView ( )

protected

Refreshes the viewing window.

C++ Syntax

```
// Refresh viewing area of STAAD.Pro Window.
OSViewUI::RefreshView();
```

VBA Syntax

```
' Refresh viewing area of STAAD.Pro Window.
 Option Explicit

    Sub Main
        Dim objOpenStaad As Object
        Dim stdFile As String
        Dim Xord As Double
        Dim Yord As Double
        Dim Zord As Double
        Dim nNodeNo As Long
        Dim SetRet As Long

        Set objOpenStaad = GetObject(,"StaadPro.OpenSTAAD")
        objOpenStaad.GetSTAADFile stdFile, "TRUE"
        nNodeNo = 2
        Xord = 10
        Yord = 10
        Zord = 10
        SetRet = objOpenStaad.Geometry.SetNodeCoordinate(nNodeNo, Xord, Yord, Zord)
        objOpenStaad.View.RefreshView()
    End Sub
```

◆ RenameView()

long OSViewUI::RenameView (LPCTSTR viewName) protected

Renames a saved view. The view should be open and be the active window.
Note that the new name should not be the same as any other existing saved view. To rename as an existing saved view name, the existing view should first be detached. See the function **OSViewUI::DetachView**.
Also the current window cannot be 'Whole Structure' or an untitled view.
See the function **OSViewUI::SetActiveWindow** to ensure the view is set as the active window.

Parameters

[in] **viewName** - New name of the saved view (type - LPCTSTR).

Return values

1 Rename view is successful.

0 Unsuccessful

2 View name already used.

-1 Generic Error

-100 Invalid Argument

C++ Syntax

```
OSViewUI::RenameView(viewName);
```

C# Syntax

```
OpenSTAADUI.OpenSTAAD os = Marshal.GetActiveObject("StaadPro.OpenSTAAD") as
        OpenSTAADUI.OpenSTAAD;
OpenSTAADUI.OSViewUI output = os.View;
string newName = "view1";
output.RenameView(newName);
```

VBA Syntax

```
Option Explicit

    Sub Main
        Dim objOpenStaad As Object
        Dim stdFile As String
        Dim varReturnVal As Long
        Dim nName As String
        Dim oldName As String
        Dim windowOptions As Boolean

        Set objOpenStaad = GetObject(,"StaadPro.OpenSTAAD")
        objOpenStaad.GetSTAADFile stdFile, "TRUE"

        oldName = "v1"
        windowOptions = True
        varReturnVal = objOpenStaad.View.OpenView(oldName, windowOptions)

        varReturnVal = objOpenStaad.View.RenameView(nName)
```

```
        ' process the return value
        If (varReturnVal = 1) Then
            MsgBox"Rename view is successful"
        Else
            MsgBox"Rename view is unsuccessful"
        End If

        Set objOpenStaad = Nothing
    End Sub
```

◆ RotateDown()

void OSViewUI::RotateDown (const VARIANT FAR & dDegrees) protected

Rotates the structure through Degrees about the Global X-Axis.

Parameters

[in] **dDegrees** Variable providing the degree of rotation.

C++ Syntax

```
// Rotate the structure about X Axis through 30 degrees
OSViewUI::RotateDown(30);
```

VBA Syntax

```
' Rotate the structure about X Axis through 30 degrees
OSViewUI.RotateDown(30)
```

◆ RotateLeft()

void OSViewUI::RotateLeft (const VARIANT FAR & dDegrees) protected

Rotates the structure through Degrees about the Global Y-Axis.

Parameters

[in] **dDegrees** Variable providing the degree of rotation.

C++ Syntax

```
// Rotate the structure about Y Axis through 30 degrees
OSViewUI::RotateLeft(30);
```

VBA Syntax

```
' Rotate the structure about Y Axis through 30 degrees
OSViewUI.RotateLeft(30)
```

◆ RotateRight()

void OSViewUI::RotateRight (const VARIANT FAR & dDegrees) protected

Rotates the structure through Degrees about the Global Y-Axis.

Parameters

[in] **dDegrees** Variable providing the degree of rotation.

C++ Syntax

```
// Rotate the structure about Y Axis through 30 degrees
OSViewUI::RotateRight(30);
```

VBA Syntax

```
' Rotate the structure about Y Axis through 30 degrees
OSViewUI.RotateRight(30)
```

◆ RotateUp()

void OSViewUI::RotateUp (const VARIANT FAR & dDegrees) protected

Rotates the structure through Degrees about the Global X-Axis.

Parameters

[in] **dDegrees** Variable providing the degree of rotation.

C++ Syntax

```
// Rotate the structure about X Axis through 30 degrees
OSViewUI::RotateUp(30);
```

VBA Syntax

```
' Rotate the structure about X Axis through 30 degrees
OSViewUI.RotateUp(30)
```

◆ SaveView()

long OSViewUI::SaveView (LPCTSTR viewName,
BOOL overWrite) protected

Save the active graphic view to the collection of saved views which can be opened using the function **OSViewUI::OpenView**.

Parameters

[in] **viewName** - New name for the view (type - LPCTSTR).

[in] **overWrite** - Option to overwrite if the given viewName already exists. (type - BOOL).
0/FALSE = Do not overwrite.
1/TRUE = Overwrite viewName if it exists.

Return values

1 Save view is successful.

0 Unsuccessful

2 View name already exist and overWrite is false.

-1 Generic Error

-100 Invalid Argument

C++ Syntax

```
OSViewUI::SaveView(viewName, overWrite);
```

C# Syntax

```
OpenSTAADUI.OpenSTAAD os = Marshal.GetActiveObject("StaadPro.OpenSTAAD") as
        OpenSTAADUI.OpenSTAAD;
OpenSTAADUI.OSViewUI output = os.View;
string newName = "view1";
BOOL overWrite = TRUE;
output.SaveView(newName, overWrite);
```

VBA Syntax

```
Option Explicit

    Sub Main
        Dim objOpenStaad As Object
        Dim stdFile As String
        Dim varReturnVal As Long
        Dim nName As String
        Dim overWrite As Boolean
        Dim varMembers(3) As Long

        Set objOpenStaad = GetObject(,"StaadPro.OpenSTAAD")
        objOpenStaad.GetSTAADFile stdFile, "TRUE"

                                    ray with beam ids for selection
                                    1
```

```
        varMembers(1) = 2
        varMembers(2) = 3
        varMembers(3) = 4

        ' select multiple beams.
        objOpenStaad.Geometry.SelectMultipleBeams(varMembers)

        objOpenStaad.View.CreateNewViewForSelections()

        nName = "view1"
        overWrite = True
        varReturnVal = objOpenStaad.View.SaveView(nName, overWrite)

        ' process the return value
        If (varReturnVal = 1) Then
            MsgBox"Save view is successful"
        Else
            MsgBox"Save view is unsuccessful"
        End If

        Set objOpenStaad = Nothing
    End Sub
```

SelectByItemList()

void OSViewUI::SelectByItemList (const VARIANT FAR & entityType,
const VARIANT FAR & nItems,
const VARIANT FAR & itemList) protected

Select entities as specified.

Parameters

[in] **entityType** Variable that holds entity type. Values may be as follows:

ID	Entity Type
1	Node
2	Beam
3	Plate
4	Solid
5	Surface

[in] **nItems** Variable that holds total number of entities needs to be selected.

[in] **itemList** Array List holds the entity nos, which need to be selected.

C++ Syntax

```
// Select by list
OSViewUI::SelectByItemList(entityType, nItems, itemList);
```

VBA Syntax

```
' Select by list
OSViewUI.SelectByItemList(entityType, nItems, itemList)
```

◆ SelectByMissingAttribute()

void OSViewUI::SelectByMissingAttribute (const VARIANT FAR & attributeCode) protected

Select entity list for which specified entity is missing.

Parameters

[in] **attributeCode** Variable that holds attribute type. Values may be as follows:

ID	Entity Type
1	Missing Property
2	Missing Modulus of Elasticity
3	Missing Density of Material
4	Missing Alpha (Coefficient of Thermal Expansion)
5	Missing Poisson Ratio

C++ Syntax

```
// Select by Missing Attribute
OSViewUI::SelectByMissingAttribute(attributeCode);
```

VBA Syntax

```
' Select by Missing Attribute
OSViewUI.SelectByMissingAttribute(attributeCode)
```

◆ SelectEntitiesConnectedToMember()

void OSViewUI::SelectEntitiesConnectedToMember (const VARIANT FAR & entityType,
const VARIANT FAR & memberNo) protected

Select entities as specified in type and connected with the specified Member.

Parameters

[in] **entityType** Variable that holds entity type. Values may be as follows:

ID	Entity Type
0	Geometry
1	Beam
2	Plate
3	Solid

[in] **memberNo** Variable that holds Member numbers with which connected entities needs to be selected.

C++ Syntax

```
// Select Entities Connected To Member
OSViewUI::SelectEntitiesConnectedToMember(entityType, memberNo);
```

VBA Syntax

```
' Select Entities Connected To Member
OSViewUI.SelectEntitiesConnectedToMember(entityType, memberNo)
```

◆ SelectEntitiesConnectedToNode()

void OSViewUI::SelectEntitiesConnectedToNode (const VARIANT FAR & entityType,
const VARIANT FAR & nodeNo) protected

Select entities as specified in type and connected with the specified node.

Parameters

[in] **entityType** Variable that holds entity type. Values may be as follows:

ID	Entity Type
0	Geometry
1	Beam
2	Plate
3	Solid

[in] **nodeNo** Variable that holds node numbers with which connected entities needs to be selected.

C++ Syntax

```
// Select Entities Connected To Node
OSViewUI::SelectEntitiesConnectedToNode(entityType, nodeNo);
```

VBA Syntax

```
' Select Entities Connected To Node
OSViewUI.SelectEntitiesConnectedToNode(entityType, nodeNo)
```

◆ SelectEntitiesConnectedToPlate()

void OSViewUI::SelectEntitiesConnectedToPlate (const VARIANT FAR & entityType,
const VARIANT FAR & plateNo) protected

Select entities as specified in type and connected with the specified Plate.

Parameters

[in] **entityType** Variable that holds entity type. Values may be as follows:

ID	Entity Type
0	Geometry
1	Beam
2	Plate
3	Solid

[in] **plateNo** Variable that holds Plate numbers with which connected entities needs to be selected.

C++ Syntax

```
// Select Entities Connected To Plate
OSViewUI::SelectEntitiesConnectedToPlate(entityType, plateNo);
```

VBA Syntax

```
' Select Entities Connected To Plate
OSViewUI.SelectEntitiesConnectedToPlate(entityType, plateNo)
```

◆ SelectEntitiesConnectedToSolid()

void OSViewUI::SelectEntitiesConnectedToSolid (const VARIANT FAR & entityType,
const VARIANT FAR & solidNo) protected

Select entities as specified in type and connected with the specified Solid.

Parameters

[in] **entityType** Variable that holds entity type. Values may be as follows:

ID	Entity Type
0	Geometry
1	Beam
2	Plate
3	Solid

[in] **solidNo** Variable that holds Solid numbers with which connected entities needs to be selected.

C++ Syntax

```
// Select Entities Connected To Solid
OSViewUI::SelectEntitiesConnectedToSolid(entityType, solidNo);
```

VBA Syntax

```
' Select Entities Connected To Solid
OSViewUI.SelectEntitiesConnectedToSolid(entityType, solidNo)
```

◆ SelectGroup()

VARIANT OSViewUI::SelectGroup (const VARIANT FAR & bstrGroup) protected

Select the relevant entities of the specified group.

Parameters

[in] **bstrGroup** A string variable that holds the group name.

C++ Syntax

```
// Select Group
VARIANT RetVal = OSViewUI::SelectGroup(bstrGroup);
```

VBA Syntax

```
' Select Group
Dim RetVal As VARIANT = OSViewUI.SelectGroup(bstrGroup)
```

◆ SelectInverse()

void OSViewUI::SelectInverse (const VARIANT FAR & entityType) protected

Inverse geometry selection for the specified entity.

Parameters

[in] **entityType** Variable that holds entity type. Values may be as follows:

ID	Entity Type
1	Node
2	Beam
3	Plate
4	Solid
5	Surface

C++ Syntax

```
// Inverse geometry selection for the specified entity.
OSViewUI::SelectInverse(entityType);
```

VBA Syntax

```
' Inverse geometry selection for the specified entity.
OSViewUI.SelectInverse(entityType)
```

◆ SelectMembersParallelTo()

void OSViewUI::SelectMembersParallelTo (const VARIANT FAR & bstrAxis) protected

Select members parallel to the specified axis.

Parameters

[in] **bstrAxis** Variable that holds the Axis ID. It may have three values:

ID	Axis
X	X-Axis
Y	Y-Axis
Z	Z-Axis

C++ Syntax

```
// Select members parallel to the X axis.
OSViewUI::SelectMembersParallelTo(X);
```

VBA Syntax

```
' Select members parallel to the X axis.
OSViewUI.SelectMembersParallelTo(X)
```

◆ SetActiveWindow()

VARIANT OSViewUI::SetActiveWindow (long id) protected

Set a given window (active graphic or table window) with the provided id as the active window. The indexing starts from 1. See the function **OSViewUI::GetWindowCount** to obtain the number of windows currently open.

Parameters

[in] **id** - The id of the window to be made the active window (Type: Long).

Return values

1/TRUE Successful.

0/FALSE Unsuccessful.

C++ Syntax

```
// Set active window.
OSViewUI::SetActiveWindow(Id);
```

C# Syntax

```
OpenSTAADUI.OpenSTAAD os = Marshal.GetActiveObject("StaadPro.OpenSTAAD") as
       OpenSTAADUI.OpenSTAAD;
OpenSTAADUI.OSViewUI output = os.View;
output.SetActiveWindow(Id);
```

VBA Syntax

```
Option Explicit
 Sub Main
     Dim objOpenStaad As Object
     Dim stdFile As String
     Dim result As Boolean
    Dim id As Long

     Set objOpenStaad = GetObject(,"StaadPro.OpenSTAAD")
     objOpenStaad.GetSTAADFile stdFile, "TRUE"
     If stdFile="" Then
         MsgBox"Bad"
         Set objOpenStaad = Nothing
         Exit Sub
     End If

    id = 2
     result = objOpenStaad.View.SetActiveWindow(id)

     ' process the return value
        If (result) Then
            MsgBox "Window with ID 2 has been set active."
        Else
            MsgBox"unsuccessful"
        End If
     Set objOpenStaad = Nothing
```

See also

OSViewUI::OpenView , **OSViewUI::GetWindowCount**, **OSViewUI::GetWindowTitle**.

◆ SetBeamAnnotationMode()

void OSViewUI::SetBeamAnnotationMode (const VARIANT FAR & Type,
const VARIANT FAR & DWFlags,
const VARIANT FAR & refreshFlag) protected

This function sets the current page mode in the STAAD.Pro environment.

Parameters

[in] **Type** Variable controlling the annotation type. It may be one of the following values:

ID	Annotation Type
0	Axial Diagram
1	Torsion Diagram
2	Moment Diagram
3	Shear Diagram
4	Stress Diagram
5	Displacement Diagram

[in] **DWFlags** Variable controlling what values are to be shown for the annotationType. It may be one of the following values:

ID	Values
1	End Values
2	Max Absolute Values
3	Mid-span Values

[in] **refreshFlag** Boolean variable (True or False). If True, STAAD.Pro viewing windows refresh with the current annotation mode.

C++ Syntax

```
//Annotate the view with values of Moments at the end of beams for the active beam moment
       diagrams
//Make sure that Staad.Pro is in Postprocessing mode
OSViewUI::SetBeamAnnotationMode(2, 1, True);
```

VBA Syntax

```
'Annotate the view with values of Moments at the end of beams for the active beam moment
       diagrams
'Make sure that Staad.Pro is in Postprocessing mode
OSViewUI.SetBeamAnnotationMode(2, 1, True)
```

```
long OSViewUI::SetDesignResults ( long   utilization,
                                  BOOL color,
                                  BOOL ShowValues )
```

protected

Sets Design Results to active view, this function replicates the setting of Design Results in the Diagrams>Design Results dialog.

Parameters

[in] **utilization** - Value of type Long. (0 = None, 1 = Actual Ratio, 2 = Normalised Ratio.)

[in] **color** - Value of type Boolean. (False/0 = Basic Colored, True/1 = Detailed Colored.)

[in] **ShowValues** - Value of type Boolean. (False/0 = Do Not Show Values, True/1 = Show Values.)

Return values

1 Set Design Results is successful.

0 Unsuccessful.

-1 Generic Error.

-2 Design Results not Loaded.

-100 Invalid Argument.

C++ Syntax

```
OSViewUI::SetDesignResults(utilization, color, ShowValues);
```

C# Syntax

```
OpenSTAADUI.OpenSTAAD os = Marshal.GetActiveObject("StaadPro.OpenSTAAD") as
       OpenSTAADUI.OpenSTAAD;
OpenSTAADUI.OSViewUI output = os.View;
long utilization = 1;
BOOL color = TRUE;
BOOL ShowValues = TRUE;
output.SetDesignResults(utilization, color, ShowValues);
```

VBA Syntax

```
Option Explicit

    Sub Main
        Dim objOpenStaad As Object
        Dim stdFile As String
        Dim varReturnVal As Long
        Dim utilization As Long
        Dim color As Boolean
        Dim ShowValues As Boolean
        Dim varMembers(3) As Long

        Set objOpenStaad = GetObject(,"StaadPro.OpenSTAAD")
        varReturnVal = objOpenStaad.GetSTAADFile stdFile, "TRUE"
        If stdFile="" Then
```

```
            MsgBox"Bad"
            Set objOpenStaad = Nothing
        Exit Sub
        End If

        Dim nRetVal As Long
        nRetVal = objOpenStaad.AnalyzeEx(1, 0, 1)

        utilization = 1
        color = True
        ShowValues = True
        varReturnVal = objOpenStaad.View.SetDesignResults(utilization, color, ShowValues)

        ' process the return value
        If (varReturnVal = 1) Then
            MsgBox"Set Design Results is successful."
        Else
            MsgBox"Set Design Results is unsuccessful."
        End If

        Set objOpenStaad = Nothing
    End Sub
```

SetDiagramMode()

```
void OSViewUI::SetDiagramMode ( const VARIANT FAR & which,
                                const VARIANT FAR & showFlag,
                                const VARIANT FAR & refreshFlag )
```

protected

Sets the label on the structure diagram on or off.

Parameters

[in] **which** Variable identifying the diagram type. It may be one of the following values:

ID	Diagram Type
0	Load
1	Displacement
2	MY
3	MZ
4	FY
5	FZ
6	AX
7	TR
8	Structure
9	Full Section
10	Section Outline
11	Stress
12	Shrink
13	Perspective
14	Hide Structure
15	Fill Plates & Solids
16	Hide Plates & Solids
18	Hide Piping
19	Sort Geometry
20	Sort Nodes
21	Plate Stress
22	Solid Stress
23	Mode Shape
24	Stress Animation
25	Plate reinforcement
26	Deck Influence Diagram*

27	Deck Carriageways*
28	Deck Triangulation*
29	Deck Loads*
30	Deck Vehicles*
	*Requires the STAAD.beava component

[in] **showFlag** Variable to set label mode on (True) or off (False).

[in] **refreshFlag** Variable (True or False). If True, STAAD.Pro viewing windows refresh.

C++ Syntax

```
// Turn on the MZ diagram.
OSViewUI::SetDiagramMode(1, True, True);
```

VBA Syntax

```
' Turn on the MZ diagram.
OSViewUI.SetDiagramMode(1, True, True)
```

◆ SetInterfaceMode()

VARIANT OSViewUI::SetInterfaceMode (const VARIANT FAR & interfaceMode) protected

This function sets the current visual mode in the STAAD.Pro environment.

Parameters

[in] **interfaceMode** Variable to set the current visual mode in STAAD.Pro environment. Followings are the valid values for mode:

ID	Mode Type
0	Pre-processor or modeling mode
1	Physical modeling mode
2	Building planner mode
3	Piping mode
5	Post Processing mode
6	FoundationDesign mode
7	ConnectionDesign mode
9	AdvancedConcreteDesign mode
10	AdvancedSlabDesign mode
11	Earthquake mode
12	SteelAutoDrafter mode
13	ChineseSteelDesign mode

Return values

1 / true Successful.

0 / false Specified mode is not visible in the workflow panel

C++ Syntax

```
// Make sure that Staad.Pro is in Postprocessing mode
VARIANT RetVal = OSViewUI::SetInterfaceMode(1);
```

VBA Syntax

```
 ' Make sure that Staad.Pro is in Postprocessing mode
Option Explicit

    Sub Main
       Dim objOpenStaad As Object
       Dim stdFile As String

       ' launch STAAD.Pro application and open "US-1 Plane Frame with Steel Design.STD"
        file from the Samples folder

        = GetObject(,"StaadPro.OpenSTAAD")
       objOpenStaad.GetSTAADFile stdFile, "TRUE"
```

```
        If stdFile="" Then
          MsgBox"Bad"
          Set objOpenStaad = Nothing
          Exit Sub
        End If

         ' declare variables
        Dim RetVal As Variant

        ' call method SetInterfaceMode
        RetVal = objOpenStaad.View.SetInterfaceMode(3)

        Set objOpenStaad = Nothing
    End Sub
```

SetLabel()

void OSViewUI::SetLabel (const VARIANT FAR & which,
const VARIANT FAR & showFlag) protected

Sets the label on the structure diagram on or off.

Parameters

[in] **which** Variable identifying the label type. It may be one of the following values:

ID	Label Type
0	Node number label
1	Member number label
2	Member property reference label
3	Material property reference label
4	Support label
5	Member release label
6	Member orientation label
7	Member section label
8	Load value label
9	Axes label
10	Node position label
11	Member specification label
12	Member ends
13	Plate element number label
14	Plate element orientation label
15	Solid element number label
16	Dimension label
17	Floor load label
18	Floor load distribution diagram label
19	Wind load label
20	Wind load influence area diagram label
21	Diagram Info

[in] **showFlag** Variable to set label mode on (True) or off (False).

C++ Syntax

```
// Label the member numbers
OSViewUI::SetLabel(1, True);
```

VBA Syntax

```
' Label the member numbers
OSViewUI.SetLabel(1, True)
```

SetModeSectionPage()

void OSViewUI::SetModeSectionPage (const VARIANT FAR & interfaceMode,
const VARIANT FAR & sectionNumber,
const VARIANT FAR & pageNumber) protected

This function sets the current page mode in the STAAD.Pro environment.

Parameters

[in] **interfaceMode** Variable to set the current visual mode in STAAD.Pro environment. Followings are the valid values for mode:

ID	Interface Mode
0	Pre-processor or modeling mode
1	Post-processing mode
2	Interactive design mode for STAAD.etc interoperability
4	Piping mode
5	BEAVA (i.e., Bridge Deck) mode

[in] **sectionNumber** Variable to set the current main page (the tabs on the left-hand side of the screen) in the STAAD.Pro environment. The following are valid values for section:

ID	Main Page
1	Setup page
2	Geometry page
3	General page
5	Node Results page
6	Beam Result page
7	Plate Results page
8	Solid Results page

[in] **pageNumber** Variable to set the current sub page (within a particular main page - the tabs on the left-hand side of the screen) in the STAAD.Pro environment. The following are valid values for modeSubPage:

ID	Page Number
0	Job Info page
1	Beam page
4	Plate page
5	Solid page
6	Property page
7	Constant page
8	Material page

9	Support page
10	Member Specifications page
11	Load page
17	Reaction page
18	Displacement page
19	Failure page
20	Forces page
21	Beam Stress page
22	Plate Stress page
23	Solid Stress page

C++ Syntax

```
// Sets the current page mode in the STAAD.Pro environment.
OSViewUI::SetModeSectionPage(1,6,20);
```

VBA Syntax

```
' Sets the current page mode in the STAAD.Pro environment.
OSViewUI.SetModeSectionPage(1,6,20)
```

SetNodeAnnotationMode()

void OSViewUI::SetNodeAnnotationMode (const VARIANT FAR & dFlag,
const VARIANT FAR & refreshFlag) protected

Sets the node displacement annotation mode. This function works only in the post-processing mode of STAAD.Pro.

Parameters

[in] **dFlag** Variable controlling the annotation type. It may be one of the following values:

ID	Annotation Type
1	X Displacement
2	Y Displacement
3	Z Displacement
4	Resultant Displacement

[in] **refreshFlag** Variable (True or False). If True, STAAD.Pro viewing windows refresh with the current annotation mode.

C++ Syntax

```
// Annotate the view with X displacement labels
// Make sure that Staad.Pro is in Postprocessing mode
VARIANT RetVal = OSViewUI::SetInterfaceMode(1)
OSViewUI::SetNodeAnnotationMode(1, True);
```

VBA Syntax

```
' Annotate the view with X displacement labels
' Make sure that Staad.Pro is in Postprocessing mode
Dim RetVal As VARIANT = OSViewUI.SetInterfaceMode(1)
OSViewUI.SetNodeAnnotationMode(1, True)
```

◆ SetReactionAnnotationMode()

void OSViewUI::SetReactionAnnotationMode (const VARIANT FAR & dFlag,
const VARIANT FAR & refreshFlag) protected

Sets the node displacement annotation mode. This function works only in the post-processing mode of STAAD.Pro.

Parameters

[in] **dFlag** Variable controlling the annotation type. It may be one of the following values:

ID	Annotation Type
1	X Reaction
2	Y Reaction
3	Z Reaction
4	X Rotation
5	Y Rotation
6	Z Rotation
7	Reaction Value Only

[in] **refreshFlag** Variable (True or False). If True, STAAD.Pro viewing windows refresh with the current annotation mode.

C++ Syntax

```
// Annotate the view with X displacement labels
// Make sure that Staad.Pro is in Postprocessing mode
VARIANT RetVal = OSViewUI::SetInterfaceMode(1)
OSViewUI::SetReactionAnnotationMode(1, True);
```

VBA Syntax

```
' Annotate the view with X displacement labels
' Make sure that Staad.Pro is in Postprocessing mode
Dim RetVal As VARIANT = OSViewUI.SetInterfaceMode(1)
OSViewUI.SetReactionAnnotationMode(1, True)
```

◆ SetScaleValueByType()

VARIANT OSViewUI::SetScaleValueByType (long scaleTypeId,
double value) protected

Set the scale used for displaying a chosen load or result diagram as shown in the Diagrams>Scales dialog.
See also the function **OSViewUI::SetScaleValues** to set all scales in a single call.
After setting the value, the display should be refreshed using the function **OSViewUI::RefreshView()**.
To display/hide loads/results, see the function **OSViewUI::SetDiagramMode**.
Note that the values are returned in base units. See function OpenSTAADUI::GetBaseUnit.

Parameters

[in] **scaleTypeId** The index of the required load or result type to be set (Type: Long). Refer to the function **OSViewUI::GetScaleValues** for the ID list.

[in] **value** Value of scale type to be used (Type: Double), defined in Base Units.

Return values

1/TRUE Value was successfully updated.

0/FALSE Value could not be updated.

C++ Syntax

```
// Get available size.
OSViewUI::SetScaleValueByType(scaleTypeId, value);
```

C# Syntax

```
OpenSTAADUI.OpenSTAAD os = Marshal.GetActiveObject("StaadPro.OpenSTAAD") as
       OpenSTAADUI.OpenSTAAD;
OpenSTAADUI.OSViewUI output = os.View;
output.SetScaleValueByType(scaleTypeId, value);
```

VBA Syntax

```
Option Explicit
 Sub Main
     Dim objOpenStaad As Object
     Dim stdFile As String
     Dim varReturnVal As Variant

     Set objOpenStaad = GetObject(,"StaadPro.OpenSTAAD")
     objOpenStaad.GetSTAADFile stdFile, "TRUE"
     If stdFile="" Then
         MsgBox"Bad"
         Set objOpenStaad = Nothing
         Exit Sub
     End If

     Dim value As Double
     Dim scaleTypeId As Long

     varReturnVal = objOpenStaad.View.SetScaleValueByType(scaleTypeId, value)
```

```
    ' process the return value
       If (varReturnVal) Then
           MsgBox"Successful"
       Else
           MsgBox"unsuccessful"
       End If
    Set objOpenStaad = Nothing
End Sub
```

See also

OSViewUI::SetScaleValues , **OSViewUI::GetScaleCount** , **OSViewUI::GetScaleValues** , **OSViewUI::GetScaleValueByType**

◆ SetScaleValues()

long OSViewUI::SetScaleValues (const VARIANT & ScalesArray) protected

Set the scales used for displaying loads and results as shown in the Diagrams>Scales dialog. The values should be provided in an array which should be sized using the value returned by the function **OSViewUI::GetScaleCount**.
To set the value of only one specific scale item, see the function **OSViewUI::SetScaleValueByType**.
Note that the values should be set in base units. See the function OpenSTAADUI::GetBaseUnit.
To display/hide loads/results, see the function **OSViewUI::SetDiagramMode**.
After setting the value, the display should be refreshed using the function **OSViewUI::RefreshView()**.

Parameters

[in] **ScalesArray** - Array of double type and size same as number of scale types. API sets the values in this array to scale types see **OSViewUI::GetScaleCount** for getsize of scalesArray. For Scale type and ID in array see **OSViewUI::GetScaleValues**.

Return values

1 Values were successfully updated.

0 Values could not be updated.

-1 Error with defined array.

C++ Syntax

```
// Get available size.
OSViewUI::SetScaleValues(ScalesArray);
```

C# Syntax

```
OpenSTAADUI.OpenSTAAD os = Marshal.GetActiveObject("StaadPro.OpenSTAAD") as
        OpenSTAADUI.OpenSTAAD;
OpenSTAADUI.OSViewUI output = os.View;
int nScaleCount = output.GetScaleCount();
double[] ScalesArray = new double[nScaleCount]
int result = output.GetScaleValues(ref ScalesArray);
if(result == 1)
  {
  ScalesArray[0] = 5;
  int setResult = output.SetScaleValues(ScalesArray);
  }
```

VBA Syntax

```
Option Explicit
 Sub Main
     Dim objOpenStaad As Object
     Dim stdFile As String
     Dim varReturnVal As Long
     Dim scales() As Double
     Dim scaleSize As Long

     Set objOpenStaad = GetObject(,"StaadPro.OpenSTAAD")
```

```
    objOpenStaad.GetSTAADFile stdFile, "TRUE"
    If stdFile="" Then
        MsgBox"Bad"
        Set objOpenStaad = Nothing
        Exit Sub
    End If

    scaleSize = objOpenStaad.View.GetScaleCount()
    ReDim scales(scaleSize)
    varReturnVal = objOpenStaad.View.GetScaleValues(scales)
    scales(0) = 5
    scales(1) = 8
    varReturnVal = objOpenStaad.View.SetScaleValues(scales)

    ' process the return value
       If (varReturnVal = 1) Then
           MsgBox"Successful"
       Else
           MsgBox"unsuccessful"
       End If
    Set objOpenStaad = Nothing
End Sub
```

See also

OSViewUI::SetScaleValueByType , **OSViewUI::GetScaleCount** , **OSViewUI::GetScaleValues** , **OSViewUI::GetScaleValueByType**

◆ SetSectionView()

void OSViewUI::SetSectionView (const VARIANT FAR & plane,
const VARIANT FAR & minVal,
const VARIANT FAR & maxVal) protected

Creates a section view of the structure.

Parameters

[in] **plane** Variable identifying the section plane. It may be one of the following values:

ID	Values for plane
0	XY Plane
1	YZ Plane
2	XZ Plane

[in] **minVal** Minimum range of the cutting plane.

[in] **maxVal** Maximum range of the cutting plane.

C++ Syntax

```
// The following call will create a section view in the YZ plane between values X = 0.4
       and X = 0.6 in the current view units:
OSViewUI::SetSectionView(1, 0.4, 0.6);
```

VBA Syntax

```
' The following call will create a section view in the YZ plane between values X = 0.4
       and X = 0.6 in the current view units:
OSViewUI.SetSectionView(1, 0.4, 0.6)
```

◆ SetUnits()

void OSViewUI::SetUnits (const VARIANT FAR & uType,
const VARIANT FAR & strUnit) protected

Set viewing unit for the active view.

Parameters

[in] **uType** Variable that holds unit type. Values are as follows:

ID	Unit Type
0	Dimension
1	Displacement
2	SectionDimension
3	SectionArea
4	Inertia
5	Force
6	Moment
7	DistributedForce
8	DistributedMoment
9	Density
10	Acceleration
11	Spring
12	RotSpring
13	MaterialModulus
14	Stress
15	Alpha
16	Temperature
17	Mass
18	SectionModulus
19	RotationalDisplacement
20	SubgradeModulus
-1	NoUnit

[in] **strUnit** Variable array that holds the unit for the specified type. Like "cm", "kns", "feet", "kn/cm" etc.

C++ Syntax

```
// Set viewing unit for the active view.
OSViewUI::SetUnits(0, "cm");
```

VBA Syntax

```
' Set viewing unit for the active view.
OSViewUI.SetUnits(0, "cm")
```

SetWindowPosition()

VARIANT OSViewUI::SetWindowPosition (long xTop,
long yTop,
long xWindow,
long yWindow) protected

Resize and reposition active window. Note that, this function only sets windows such as they fully fit in the application desktop. See **OSViewUI::GetApplicationDesktopSize** to determine the size of the application desktop. This function resizes and changes position of windows in all state. if window is in minimized or maximized state it will restore it and reposition and resize it. API works with view window as well as the grid tables.

Parameters

[in] **xTop** - Horizontal distance in pixels from top left corner of the desktop to the top left corner of the active window.

[in] **yTop** - Vertical distance in pixels down from top left corner of the desktop to the top left corner of the active window.

[in] **xWindow** - Width of window in pixels. xTop + xWindow must be less than the application desktop width.

[in] **yWindow** - Height of window in pixels. yTop + yWindow must be less than the application desktop height.

Return values

1/True is successful.

0/False is unsuccessful.

C++ Syntax

```
OSViewUI::SetWindowPosition(xTop, yTop, xWindow, yWindow);
```

VBA Syntax

```
Option Explicit
 Sub Main
     Dim objOpenStaad As Object
     Dim stdFile As String
     Dim varReturnVal As Boolean
     Dim varSizeOk As Boolean

     Set objOpenStaad = GetObject(,"StaadPro.OpenSTAAD")
     objOpenStaad.GetSTAADFile stdFile, "TRUE"
     If stdFile="" Then
         MsgBox"Bad"
         Set objOpenStaad = Nothing
         Exit Sub
     End If
```



```
     Dim yTop As Long
```

```
    Dim xWindow As Long
    Dim yWindow As Long
    Dim availableLength As Long, availableWidth As Long
    varSizeOk = objOpenStaad.View.GetApplicationDesktopSize(availableLength,
      availableWidth)
    xTop = 10
    yTop = 10
    If (varSizeOk) Then
        xWindow = availableLength/2
        yWindow = availableWidth/2
    Else
        xWindow = 100
        yWindow = 100
    End If

    varReturnVal = objOpenStaad.View.SetWindowPosition(xTop, yTop, xWindow, yWindow)

    ' process the return value
    If (varReturnVal) Then
        MsgBox"Successful"
    Else
        MsgBox"Unsuccessful"
    End If
    Set objOpenStaad = Nothing
End Sub
```

See also

OSViewUI::GetApplicationDesktopSize

◆ ShowAllMembers()

void OSViewUI::ShowAllMembers ( ) protected

Display all members of the current structure.

C++ Syntax

```
// Display the whole structure.
OSViewUI::ShowAllMembers();
```

VBA Syntax

```
' Display the whole structure.
OSViewUI.ShowAllMembers()
```

◆ ShowBack()

void OSViewUI::ShowBack ( ) protected

This function displays the back view of the structure.

C++ Syntax

```
// This function displays the back view of the structure.
OSViewUI::ShowBack();
```

VBA Syntax

```
' This function displays the back view of the structure.
OSViewUI.ShowBack()
```

◆ ShowBottom()

void OSViewUI::ShowBottom ( ) protected

This function displays the bottom view of the structure.

C++ Syntax

```
// This function displays the bottom view of the structure.
OSViewUI::ShowBottom();
```

VBA Syntax

```
' This function displays the bottom view of the structure.
OSViewUI.ShowBottom()
```

◆ ShowFront()

void OSViewUI::ShowFront ( ) protected

This function displays the front view of the structure.

C++ Syntax

```
// This function displays the front view of the structure.
OSViewUI::ShowFront();
```

VBA Syntax

```
' This function displays the front view of the structure.
OSViewUI.ShowFront()
```

◆ ShowIsometric()

void OSViewUI::ShowIsometric ( ) protected

This function displays the isometric view of the structure.

C++ Syntax

```
// This function displays the isometric view of the structure.
OSViewUI::ShowIsometric();
```

VBA Syntax

```
' This function displays the isometric view of the structure.
OSViewUI.ShowIsometric()
```

◆ ShowLeft()

void OSViewUI::ShowLeft ( ) protected

This function displays the left view of the structure.

C++ Syntax

```
// This function displays the left view of the structure.
OSViewUI::ShowLeft();
```

VBA Syntax

```
' This function displays the left view of the structure.
OSViewUI.ShowLeft()
```

◆ ShowMember()

void OSViewUI::ShowMember (const VARIANT FAR & nMember) protected

Show the specified member.

Parameters

[in] **nMember** Variable that holds member number to be shown.

C++ Syntax

```
// Show Member
OSViewUI::ShowMember(5);
```

VBA Syntax

```
' Show Member
OSViewUI.ShowMember(5)
```

◆ ShowMembers()

void OSViewUI::ShowMembers (const VARIANT FAR & nMembers,
const VARIANT FAR & naMemberNos) protected

Show the specified member.

Parameters

[in] **nMembers** Variable that holds member number to be shown.

[in] **naMemberNos** Variable array that holds the member nos, which need to be shown.

C++ Syntax

```
// Show Member
OSViewUI::ShowMembers(5, naMemberNos);
```

VBA Syntax

```
' Show Member
OSViewUI.ShowMembers(5, naMemberNos)
```

◆ ShowPlan()

void OSViewUI::ShowPlan ( ) protected

This function displays the top (i.e., plan) view of the structure.

C++ Syntax

```
// This function displays the top (i.e., plan) view of the structure.
OSViewUI::ShowPlan();
```

VBA Syntax

```
' This function displays the top (i.e., plan) view of the structure.
OSViewUI.ShowPlan()
```

◆ ShowRight()

void OSViewUI::ShowRight ( ) protected

This function displays the right view of the structure.

C++ Syntax

```
// This function displays the right view of the structure.
OSViewUI::ShowRight();
```

VBA Syntax

```
' This function displays the right view of the structure.
OSViewUI.ShowRight()
```

◆ SpinLeft()

void OSViewUI::SpinLeft (const VARIANT FAR & dDegrees) protected

Rotates the structure through Degrees about the Global Z-Axis.

Parameters

[in] **dDegrees** Variable providing the degree of rotation.

C++ Syntax

```
// Rotate the structure about Z Axis through 30 degrees
OSViewUI::SpinLeft(30);
```

VBA Syntax

```
' Rotate the structure about Z Axis through 30 degrees
OSViewUI.SpinLeft(30)
```

◆ SpinRight()

void OSViewUI::SpinRight (const VARIANT FAR & dDegrees) protected

Rotates the structure through Degrees about the Global Z-Axis.

Parameters

[in] **dDegrees** Variable providing the degree of rotation.

C++ Syntax

```
// Rotate the structure about Z Axis through 30 degrees
OSViewUI::SpinRight(30);
```

VBA Syntax

```
' Rotate the structure about Z Axis through 30 degrees
OSViewUI.SpinRight(30)
```

◆ ZoomAll()

void OSViewUI::ZoomAll ( ) protected

zoom current view to Display the whole structure.

C++ Syntax

```
// Display the whole structure.
OSViewUI::ZoomAll();
```

VBA Syntax

```
' Display the whole structure.
OSViewUI.ZoomAll()
```

◆ ZoomExtentsMainView()

void OSViewUI::ZoomExtentsMainView ( ) protected

Adjust viewing scale to zoom main view to the extents in new view.

C++ Syntax

```
// Adjust viewing scale to zoom main view to the extents.
OSViewUI::ZoomExtentsMainView();
```

VBA Syntax

```
' Adjust viewing scale to zoom main view to the extents.
OSViewUI.ZoomExtentsMainView()
```