Section: Assign Section to Element

Property

Functions

afx_msg VARIANT	**OSPropertyUI::CreateAssignProfileProperty** (const VARIANT FAR &varnAssignType) Create "Assign Profile" property.
afx_msg VARIANT	**OSPropertyUI::AssignBeamProperty** (const VARIANT FAR &nBeamNo, const VARIANT FAR &nProperty) Assign beam property.
afx_msg VARIANT	**OSPropertyUI::AssignMemberSpecToBeam** (const VARIANT FAR &varnBeamNo, const VARIANT FAR &varnSpecNo) Assign specifications to beam(s).
afx_msg VARIANT	**OSPropertyUI::AssignPlateThickness** (const VARIANT FAR &nPlateNo, const VARIANT FAR &nProperty) Assign thickness to a plate or a set of plates. The API will skip the plate numbers that are not found if the list contains a combination of valid and invalid plates.
afx_msg VARIANT	**OSPropertyUI::AssignParametricSurfaceThickness** (const VARIANT FAR &nPlateNo, const VARIANT FAR &nProperty) This function is reserved, do not use.
afx_msg VARIANT	**OSPropertyUI::AssignElementSpecToPlate** (const VARIANT FAR &varnPlateNo, const VARIANT FAR &varnSpecNo) Assign specifications to plate(s).
afx_msg VARIANT	**OSPropertyUI::UpdatePropertiesToDesignSection** () Updates all the section properties that have been designed with a SELECT MEMBER command.

Detailed Description

These functions are related to assign section to elements.

Function Documentation

◆ AssignBeamProperty()

Loading [MathJax]/extensions/MathZoom.js

VARIANT OSPropertyUI::AssignBeamProperty (const VARIANT FAR & nBeamNo,
const VARIANT FAR & nProperty)

Assign beam property.

Parameters

[in] **nBeamNo** The beam number IDs in VARIANT type.

[in] **nProperty** The number ID identifying a property.

Return values

0 OS: OK.

-106 nBeamNo array dimension error.

-3006 Invalid member number ID(s).

-6001 Invalid section The assigned section property ID.

Example

```
// BeamNo is a 3 * 1 array storing 3 number of beams.
long BeamNo[3][1] = { { 1 }, { 4 }, { 7 } };

// Create an VARIANT type variable to store the beam list.
VARIANT nBeamNo;
nBeamNo.vt = VT_ARRAY | VT_I4; // define nBeamNo to store array of long.

SAFEARRAYBOUND SAB[2];
SAB[0].lLbound = 0; SAB[0].cElements = 3;
nBeamNo.parray = SafeArrayCreate(VT_I4, 1, SAB); // create a 3 * 1 array of long.

for (long i = 0; i < 3; i++)
{
HRESULT hRet = SafeArrayPutElement(nBeamNo.parray, &i, &BeamNo[i]); // assign value for
       nBeamNo.
}

long nProperty = 2; // Property ID

// Assign property #2 to beams #1, 4, 7.
VARIANT RetVal = OSPropertyUI::AssignBeamProperty(nBeamNo, nProperty);
```

◆ AssignElementSpecToPlate()

VARIANT OSPropertyUI::AssignElementSpecToPlate (const VARIANT FAR & varnPlateNo,
const VARIANT FAR & varnSpecNo)

Assign specifications to plate(s).

Parameters

[in] **varnPlateNo** The plate number ID(s) in VARIANT array.

[in] **varnSpecNo** The specification number ID.

Return values

0 OK

-106 1 dimensional array of long expected.

-6017 Library Error: Unable to assign specification.

Example

```
// PlateNo is a 3 * 1 array storing 3 number of plates.
long PlateNo[3][1] = { { 1 }, { 2 }, { 3 } };

// Create an VARIANT type variable to store the plate list.
VARIANT varnPlateNo;
varnPlateNo.vt = VT_ARRAY | VT_I4; // define varnPlateNo to store array of long.

SAFEARRAYBOUND SAB[2];
SAB[0].lLbound = 0; SAB[0].cElements = 3;
varnPlateNo.parray = SafeArrayCreate(VT_I4, 1, SAB); // create a 3 * 1 array of long.

for (long i = 0; i < 3; i++)
{
HRESULT hRet = SafeArrayPutElement(varnPlateNo.parray, &i, &PlateNo[i]); // assign value
       for varnPlateNo.
}

long varnSpecNo = 3; // varnSpecNo ID

// Assign specification #3 to plates #1, 2, 3.
VARIANT RetVal = OSPropertyUI::AssignElementSpecToPlate(varnPlateNo, varnSpecNo);
```

◆ AssignMemberSpecToBeam()

VARIANT OSPropertyUI::AssignMemberSpecToBeam (const VARIANT FAR & varnBeamNo,
const VARIANT FAR & varnSpecNo)

Assign specifications to beam(s).

Parameters

[in] **varnBeamNo** The beam number ID(s) in VARIANT array.

[in] **varnSpecNo** The specification number ID.

Return values

0 OK

-106 1 dimensional array of long expected.

-6017 Library Error: Unable to assign specification.

Example

```
// BeamNo is a 3 * 1 array storing 3 number of beams.
long BeamNo[3][1] = { { 1 }, { 4 }, { 7 } };

// Create an VARIANT type variable to store the beam list.
VARIANT varnBeamNo;
varnBeamNo.vt = VT_ARRAY | VT_I4; // define varnBeamNo to store array of long.

SAFEARRAYBOUND SAB[2];
SAB[0].lLbound = 0; SAB[0].cElements = 3;
varnBeamNo.parray = SafeArrayCreate(VT_I4, 1, SAB); // create a 3 * 1 array of long.

for (long i = 0; i < 3; i++)
{
HRESULT hRet = SafeArrayPutElement(varnBeamNo.parray, &i, &BeamNo[i]); // assign value
       for varnBeamNo.
}

long varnSpecNo = 2; // varnSpecNo ID

// Assign specification #2 to beams #1, 4, 7.
VARIANT RetVal = OSPropertyUI::AssignMemberSpecToBeam(varnBeamNo, varnSpecNo);
```

See also

OSPropertyUI::CreateMemberTrussSpec

OSPropertyUI::CreateMemberTensionSpec

OSPropertyUI::CreateMemberCompressionSpec

OSPropertyUI::CreateMemberCableSpec

OSPropertyUI::GetMemberSpecCode

◆ AssignPlateThickness()

VARIANT OSPropertyUI::AssignPlateThickness (const VARIANT FAR & nPlateNo,
const VARIANT FAR & nProperty)

Assign thickness to a plate or a set of plates. The API will skip the plate numbers that are not found if the list contains a combination of valid and invalid plates.

Parameters

[in] **nPlateNo** The plate number ID/s. (VARIANT / VARIANT Array).

[in] **nProperty** The assigned section property ID. (Long)

Return values

0 OK.

-1 Error.

-106 nPlateNo array dimension error.

-113 nPlateNo type error (Long or Int Expected)

-4009 All the plate numbers are invalid.

-4008 Some of the plate numbers are invalid.

-6001 The assigned section property ID is invalid.

C++ Syntax

```
 // Assign property #10 to plate #3 and #5
long PlateNo[2] = { 3, 5 };
// Create an VARIANT type variable to store the beam list.

VARIANT nPlateNo;
nPlateNo.vt = VT_ARRAY | VT_I4; // define nBeamNo to store array of long.

SAFEARRAYBOUND SAB[2];
SAB[0].lLbound = 0; SAB[0].cElements = 2;
nPlateNo.parray = SafeArrayCreate(VT_I4, 1, SAB); // create a 2 array of long.

for (long i = 0; i < 2; i++)
{
HRESULT hRet = SafeArrayPutElement(nPlateNo.parray, &i, &BeamNo[i]); // assign value for
       nBeamNo.
}

long nProperty = 10; // Property ID

// Assign property #10 to plate #3 and #5
VARIANT RetVal = OSPropertyUI::AssignPlateThickness(nPlateNo, nProperty);
```

VBA Syntax

```
Option Explicit
    Sub Main
    Dim objOpenStaad As Object
    Dim PropertyId As Long
```

```
    Dim varReturnVal As Long
    Dim nPlates As Long
    Dim PlateArray() As Long

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

    Dim ThicknessArray(3) As Double
    ThicknessArray(0)=2
    ThicknessArray(1)=2
    ThicknessArray(2)=1.5
    ThicknessArray(3)=1.5
    PropertyId = objOpenStaad.Property.CreatePlateThicknessProperty(ThicknessArray)
    nPlates = objOpenStaad.Geometry.GetPlateCount()
    ReDim PlateArray(nPlates-1)
    varReturnVal = objOpenStaad.Geometry.GetPlateList(PlateArray)
    varReturnVal = objOpenStaad.Property.AssignPlateThickness(PlateArray, PropertyId)
    ' process the return value
       If (varReturnVal = 0) Then
           MsgBox"Successful"
       Else
           MsgBox"Unsuccessful"
       End If
    Set objOpenStaad = Nothing
End Sub
```

◆ CreateAssignProfileProperty()

Loading [MathJax]/extensions/MathZoom.js

VARIANT OSPropertyUI::CreateAssignProfileProperty (const VARIANT FAR & varnAssignType)

Create "Assign Profile" property.

Parameters

[in] **varnAssignType** Profile type number ID.

Type of Profile	Value
Angle	0
Double Angle	1
Beam	2
Column	3
Channel	4

Return values

<Val> The assigned section property ID.

0 Library Error: Unable to create property.

-6008 Invalid assign profile type.

C++ Syntax

```
// Create Assign Profile property.
VARIANT RetVal = OSPropertyUI::CreateAssignProfileProperty(2);
```

VBA Syntax

```
 ' Create Assign Profile property.
Dim RetVal As VARIANT = OSPropertyUI.CreateAssignProfileProperty(2)
```

◆ UpdatePropertiesToDesignSection()

VARIANT OSPropertyUI::UpdatePropertiesToDesignSection ( )

Updates all the section properties that have been designed with a SELECT MEMBER command.

Return values

1 if assignment is successful.

0 if assignment is unsuccessful.

> **Remarks**
> Assignment will fail if there are no members in the model or design results are not available. This API will not work with physical model

VBA Syntax

```
Option Explicit

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

       ' launch STAAD.Pro application and open "Tutorial 1 - Steel Portal Frame.std" file
        from the Tutorials folder

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

      ' perform analysis and design the model and wait for finish
       varReturnVal = objOpenStaad.AnalyzeEx(1, 0, 1)

       ' Check if results are available or not
       If objOpenStaad.Output.AreResultsAvailable = 0 Then
           MsgBox "Results Unavailable"
          Set objOpenStaad = Nothing
           Exit Sub
      End If

       ' now call method UpdatePropertiesToDesignSection
      varReturnVal = objOpenStaad.Property.UpdatePropertiesToDesignSection() ' section
       property of beam 2 & 3 will be updated with new design property

      ' process the return value
       If varReturnVal > 0 Then
         MsgBox"Assign is successful"
       Else
        MsgBox"Assign is unsuccessful"
       End If

Loading [MathJax]/extensions/MathZoom.js = Nothing
```

```
End Sub
```