Geometry: Beam

Geometry

Functions

afx_msg void	**OSGeometryUI::CreateBeam** (const VARIANT FAR &nBeamNo, const VARIANT FAR &nNodeA, const VARIANT FAR &nNodeB) Creates a beam/member with specified nodes in current model.
afx_msg VARIANT	**OSGeometryUI::AddBeam** (const VARIANT FAR &nNodeA, const VARIANT FAR &nNodeB) Adds a beam/member with specified nodes in current model, and returns the member number ID automatically assigned with.
afx_msg void	**OSGeometryUI::DeleteBeam** (const VARIANT FAR &nBeamNo) Delete a specified member.
afx_msg void	**OSGeometryUI::SplitBeam** (const VARIANT FAR &nBeamNo, const VARIANT FAR &nNodes, const VARIANT FAR &faDistToNodes) Split a specified beam into several beams by specified node(s).
afx_msg void	**OSGeometryUI::SplitBeamInEqlParts** (const VARIANT FAR &nBeamNo, const VARIANT FAR &nParts) Split a specified beam into several EQUAL beams by specified number of node(s)
afx_msg VARIANT	**OSGeometryUI::GetLastBeamNo** () Returns the member number ID of the last beam in the model.
afx_msg VARIANT	**OSGeometryUI::GetNoOfSelectedBeams** () Returns the number of selected member(s).
afx_msg void	**OSGeometryUI::GetSelectedBeams** (VARIANT FAR &naBeamNos, VARIANT FAR &nIsSorted) Returns a list of selected member(s).
afx_msg VARIANT	**OSGeometryUI::GetBeamLength** (const VARIANT FAR &nBeamNo) Returns the length for specified member.
afx_msg VARIANT	**OSGeometryUI::SelectMultipleBeams** (const VARIANT FAR &naBeamNos) Selects multiple member(s) in current model.
afx_msg VARIANT	**OSGeometryUI::SelectBeam** (const VARIANT FAR &nBeamNo) Selects the specified member in current model.
afx_msg void	**OSGeometryUI::AddMultipleBeams** (const VARIANT FAR &naIncidences) Add multiple beams with specified end node number ID(s).
afx_msg VARIANT	**OSGeometryUI::SetCheckForIdenticalEntity** (long entityType, long bEnable) This API will set whether to enable checking for existing identical entities (beam, plate, node etc.) or not. If set is enabled, time taken by the corresponding add/create multiple entities corresponding APIs will be comparatively less. Please refer to the code snippet below.

afx_msg void	**OSGeometryUI::CreateMultipleBeams** (const VARIANT FAR &nBeamIdArray, const VARIANT FAR &nBeamIncidenceArray) Create multiple beams with specified beam number ID(s).
afx_msg VARIANT	**OSGeometryUI::GetMemberCount** () Returns the total number of members in the current model.
afx_msg void	**OSGeometryUI::GetBeamList** (VARIANT FAR &nBeamList) Returns a list of all the member ID(s) the current model.
afx_msg VARIANT	**OSGeometryUI::GetMemberIncidence** (const VARIANT FAR &nBeamNo, VARIANT FAR &nNodeA, VARIANT FAR &nNodeB) Returns the number ID(s) of connecting node(s) for specified member.
afx_msg void	**OSGeometryUI::ClearMemberSelection** () Unselect all the member item(s).
afx_msg void	**OSGeometryUI::SetMemberUniqueID** (const VARIANT FAR &nMembNo, const VARIANT FAR &szName) Assigns an unique string ID (GUID) to specified member.
afx_msg VARIANT	**OSGeometryUI::GetMemberUniqueID** (const VARIANT FAR &nBeamNo) Returns the unique string ID (GUID) for specified member.
afx_msg VARIANT	**OSGeometryUI::GetMemberIncidence_CIS2** (const VARIANT FAR &nBeamNo, VARIANT FAR &szName, VARIANT FAR &nNodeA, VARIANT FAR &nNodeB) Returns the number ID(s) of connecting node(s) for specified member.
afx_msg VARIANT	**OSGeometryUI::IsZUp** () Returns if Z-axis is in upward direction?
afx_msg VARIANT	**OSGeometryUI::IsBeam** (const VARIANT FAR &nMemberNo, const VARIANT FAR &dTolAngle) Returns if the angle of inclination for specified BEAM member is not more than given tolerance angle (for small angle only).
afx_msg VARIANT	**OSGeometryUI::IsColumn** (const VARIANT FAR &nMemberNo, const VARIANT FAR &dTolAngle) Returns if the angle of inclination for specified COLUMN member is not more than given tolerance angle (for small angle only).
afx_msg VARIANT	**OSGeometryUI::GetNoOfBeamsConnectedAtNode** (const VARIANT FAR &nNodeNo) Returns no of beams connected at a specified node.
afx_msg VARIANT	**OSGeometryUI::GetBeamsConnectedAtNode** (const VARIANT FAR &nNodeNo, VARIANT FAR &nBeamList) Returns a list of all the beams connected to the specified node.
afx_msg VARIANT	**OSGeometryUI::RenumberBeam** (const VARIANT FAR &varBeamNoOld, const VARIANT FAR &varBeamNoNew) Renumbers the existing beam id with the specified id.

y, const VARIANT FAR &varBeamNo, const VARIANT FAR &varPropertyNo, const VARIANT FAR &varBetaAngle,

	const VARIANT FAR &varMaterialName) Merges multiple collinear and connected beams to a single beam with specified id, property, material and beta angle.
afx_msg VARIANT	**OSGeometryUI::IntersectBeams** (const VARIANT FAR &Method, const VARIANT FAR &BeamNosArray, const VARIANT FAR &varTolerance, VARIANT FAR &NewBeamNosArray) A function that takes a list of beam numbers and either identify those that would be split/connected due to overlapping each other and highlighting them on the model or simply performing the intersection routine and returning a list of members resulting from running the intersection routine that have been either modified or added to the model.
afx_msg VARIANT	**OSGeometryUI::GetCountOfBreakableBeamsAtSpecificNodes** (const VARIANT FAR &nNodeIdArray) Returns number of beams that can be broken based on the list of node Ids. This method can be used to get the count of beams that can be used to define the size of output array while using the API BreakBeamsAtSpecificNodes.
afx_msg VARIANT	**OSGeometryUI::BreakBeamsAtSpecificNodes** (const VARIANT FAR &nNodeIdArray, VARIANT FAR &nBrokenBeamIdArray, VARIANT FAR &nNewBeamIdArray) Breaks beams that passes through the specified list of nodes and assigns same attributes to the newly added beams. This method can be used in conjunction with API GetCountOfBreakableBeamsAtSpecificNodes, which will return the number of beams that can be broken.
afx_msg VARIANT	**OSGeometryUI::GetIntersectBeamsCount** (const VARIANT FAR &BeamNosArray, const VARIANT FAR &varTolerance) Returns number of new beams that will be created if the specified list of beams are intersected. This method can be used to get the count of beams that can be used to define the size of output array while using the API **OSGeometryUI::IntersectBeams**.

Detailed Description

These functions are related to operations of creating, adding, getting and deleting member(s).

Function Documentation

◆ AddBeam()

VARIANT OSGeometryUI::AddBeam (const VARIANT FAR & nNodeA,
const VARIANT FAR & nNodeB) private

Adds a beam/member with specified nodes in current model, and returns the member number ID automatically assigned with.

Parameters

[in] **nNodeA** Number ID of the STARTING end node (**node$_A$**).

[in] **nNodeB** Number ID of the ENDING end node (**node$_B$**).

Return values

<Val> Member number ID assigned to this created member.

-1 Unable to add member.

-2001 Cannot find Node < **nNodeA** or **nNodeB** >.

C++ Syntax

```
long nNodeA = 2;
long nNodeB = 4;
// Add a beam  starts from node #2 to node #4.
VARIANT nBeamNo = OSGeometryUI::AddBeam(nNodeA, nNodeB);
```

VBA Syntax

```
Dim nNodeA As Long = 2
Dim nNodeB As Long = 4
' Add a beam  starts from node #2 to node #4.
Dim nBeamNo As VARIANT = OSGeometryUI.AddBeam(nNodeA, nNodeB)
```

Remarks

The difference between **OSGeometryUI::CreateBeam** and **OSGeometryUI::AddBeam** is the former has an option to label the node with any user-defined number.

See also

OSGeometryUI::CreateBeam

OSGeometryUI::DeleteBeam

OSGeometryUI::AddMultipleBeams

OSGeometryUI::CreatePMember

◆ AddMultipleBeams()

void OSGeometryUI::AddMultipleBeams (const VARIANT FAR & naIncidences) private

Add multiple beams with specified end node number ID(s).

Parameters

[in] **naIncidences** VARIANT array of m * 2 dimension containing member starting and ending end nodes: [$NodeA_i$, $NodeB_i$].

C++ Syntax

```
// Add multiple members.
OSGeometryUI::AddMultipleBeams(naIncidences);
```

VBA Syntax

```
' Add multiple members.
OSGeometryUI.AddMultipleBeams(naIncidences)
```

See also

OSGeometryUI::CreateBeam

OSGeometryUI::AddBeam

OSGeometryUI::DeleteBeam

◆ BreakBeamsAtSpecificNodes()

VARIANT OSGeometryUI::BreakBeamsAtSpecificNodes (const VARIANT FAR & nNodeIdArray,
VARIANT FAR & nBrokenBeamIdArray,
VARIANT FAR & nNewBeamIdArray) private

Breaks beams that passes through the specified list of nodes and assigns same attributes to the newly added beams. This method can be used in conjunction with API GetCountOfBreakableBeamsAtSpecificNodes, which will return the number of beams that can be broken.

Parameters

[in]	**nNodeIdArray**	array of node numbers to be used to find the number of beams that can be split (type - Long/Integer).
[out]	**nBrokenBeamIdArray**	array of existing beam numbers that are broken (type - Long/Integer).
[out]	**nNewBeamIdArray**	array of new beam numbers that are added (type - Long/Integer).

Return values

1 if the method is successful.

0 if the method is unsuccessful.

VBA Syntax

```
Option Explicit

    Sub Main
       Dim objOpenStaad As Object
       Dim stdFile As String

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

       ' declare variables
       Dim varNodeIds(0) As Long
       Dim varCountReturnVal As Long
       Dim varBreakReturnVal As Long
       Dim varExistingBeamIds() As Long
       Dim varNewBeamIds() As Long

       ' initialize variables
       ' fill in the array. note node 2 lies between node 1 and node 3, node 1 and node 3
        are the two ends of member 1
       varNodeIds(0) = 2

       ' call method GetCountOfBreakableBeamsAtSpecificNodes first to get the number of
        beams that can be broken
       varCountReturnVal =
        objOpenStaad.Geometry.GetCountOfBreakableBeamsAtSpecificNodes(varNodeIds)

       If varCountReturnVal > 0 Then
```

```
        ' redim the output arrays
        Redim varExistingBeamIds(varCountReturnVal-1)
        Redim varNewBeamIds(varCountReturnVal-1)

        varBreakReturnVal = objOpenStaad.Geometry.BreakBeamsAtSpecificNodes(varNodeIds,
      varExistingBeamIds, varNewBeamIds)
        If varBreakReturnVal = 1 Then
           MsgBox"Break beam at selected node is successful."
        Else
           MsgBox"Break beam at selected node is unsuccessful."
        End If
      Else
        MsgBox"Breakable beams not found."
      End If

      Set objOpenStaad = Nothing
   End Sub
```

Remarks

Method will fail if the specified nodes are not lying on any beam. Method will also fail if there is data type mismatch.

See also

OSGeometryUI::GetCountOfBreakableBeamsAtSpecificNodes

◆ ClearMemberSelection()

void OSGeometryUI::ClearMemberSelection ( ) private

Unselect all the member item(s).

C++ Syntax

```
// Clear Member Selection
OSGeometryUI::ClearMemberSelection();
```

VBA Syntax

```
' Clear Member Selection
OSGeometryUI.ClearMemberSelection()
```

See also

OSGeometryUI::SelectBeam

OSGeometryUI::SelectMultipleBeams

OSGeometryUI::GetSelectedBeams

◆ CreateBeam()

void OSGeometryUI::CreateBeam (const VARIANT FAR & nBeamNo,
const VARIANT FAR & nNodeA,
const VARIANT FAR & nNodeB) private

Creates a beam/member with specified nodes in current model.

Parameters

[in] **nBeamNo** Member number ID to be assigned to the newly created member.

[in] **nNodeA** Number ID of the STARTING end node (**node$_A$**).

[in] **nNodeB** Number ID of the ENDING end node (**node$_B$**).

C++ Syntax

```
long nNodeA = 2;
long nNodeB = 4;
// Create a beam starts from node #2 to node #4, call it beam # 77.
OSGeometryUI::CreateBeam(77, nNodeA, nNodeB);
```

VBA Syntax

```
Dim nNodeA As Long = 2
Dim nNodeB As Long = 4
' Create a beam starts from node #2 to node #4, call it beam # 77.
OSGeometryUI.CreateBeam(77, nNodeA, nNodeB)
```

Remarks

The difference between **OSGeometryUI::CreateBeam** and **OSGeometryUI::AddBeam** is the former has an option to label the node with any user-defined number.

Note

If geometry update flag is set, this function will update the beam/member incidence using the input node numbers.

See also

OSGeometryUI::AddBeam

OSGeometryUI::DeleteBeam

OSGeometryUI::AddMultipleBeams

OSGeometryUI::CreatePMember

- CreateMultipleBeams()

void OSGeometryUI::CreateMultipleBeams (const VARIANT FAR & nBeamIdArray,
const VARIANT FAR & nBeamIncidenceArray) private

Create multiple beams with specified beam number ID(s).

Parameters

[in] **nBeamIdArray** Integer array of 1 dimension containing m node IDs:[ID_i]

[in] **nBeamIncidenceArray** int VARIANT array of m * 2 dimension containing member starting and ending end nodes: [$NODEA_i$, $NODEB_i$].

C++ Syntax

```
// Create Multiple Beams - Setting Identical check for Beam Entity as 0 (=FALSE). So,
       while creating multiple beams in the subsequent API call, it will not check for
// existing identical nodes while creating beams resulting in faster operation. If, check
       is set to 1 (=TRUE), it will check for identical beams
// in the subsequent add multiple beam API which will take longer time.

int nBeamIdArray[] = {13, 14, 15};
int nBeamIncidenceArray[][2] = {{3, 5}, {15, 25}, {47, 27}};
OSGeometryUI::SetCheckForIdenticalEntity(STAADEntityType.MemberEntity, 0);
OSGeometryUI::CreateMultipleBeams(nBeamIdArray, nBeamIncidenceArray);
```

C# Syntax

```
// Create Multiple Beams - Setting Identical check for Beam Entity as 0 (=FALSE). So,
       while creating multiple beams in the subsequent API call, it will not check for
// existing identical nodes while creating beams resulting in faster operation. If, check
       is set to 1 (=TRUE), it will check for identical beams
// in the subsequent add multiple beam API which will take longer time.

int[] nBeamIdArray = new int[3] {13, 14, 15};
int[,] nBeamIncidenceArray = new int[3, 2] {{3, 5}, {15, 25}, {47, 27}};
object objectBeamIncidenceArray = nBeamIncidenceArray as object;
object objectBeamIdArray = nNodeIdArray as object;
m_OStd.Geometry.SetCheckForIdenticalEntity(STAADEntityType.MemberEnity, 1);
int RetValue =
       m_OStd.Geometry.CreateMultipleBeams(objectBeamIdArray,objectBeamIncidenceArray);
```

VBA Syntax

```
// Create Multiple Beams - Setting Identical check for Beam Entity as 0 (=FALSE). So,
       while creating multiple beams in the subsequent API call, it will not check for
// existing identical nodes while creating beams resulting in faster operation. If, check
       is set to 1 (=TRUE), it will check for identical beams
// in the subsequent add multiple beam API which will take longer time.

Dim nBeamIdArray(1) As Integer
nBeamIdArray(0) = 13
nBeamIdArray(1) = 14
Dim nBeamIncidenceArray(1,1) As Integer
```

```
nBeamIncidenceArray(1,0) = 15
nBeamIncidenceArray(1,1) = 25
objOpenStaad.Geometry.SetCheckForIdenticalEntity(1, 1);
objOpenStaad.Geometry.CreateMultipleBeams(nBeamIdArray,nBeamIncidenceArray)
```

See also

COSGeometry::SetCheckForIdenticalEntity

COSGeometry::CreateNewBeam

◆ DeleteBeam()

void OSGeometryUI::DeleteBeam (const VARIANT FAR & nBeamNo) private

Delete a specified member.

Parameters

[in] **nBeamNo** Member number ID.

C++ Syntax

```
long nBeamNo = 25;
//Delete beam #25
OSGeometryUI::DeleteBeam(nBeamNo);
```

VBA Syntax

```
Dim nBeamNo As Long = 25
' Delete beam #25.
OSGeometryUI.DeleteBeam(nBeamNo)
```

See also

OSGeometryUI::CreateBeam

OSGeometryUI::AddBeam

OSGeometryUI::AddMultipleBeams

◆ GetBeamLength()

VARIANT OSGeometryUI::GetBeamLength (const VARIANT FAR & nBeamNo) private

Returns the length for specified member.

Parameters

[in] **nBeamNo** Member number ID for which the length is to be retrieved

Return values

<Val> The length of specified member in double.

0 Cannot find member < **nBeamNo** >.

C++ Syntax

```
// Get length of the beam 10.
VARIANT BeamLen = OSGeometryUI::GetBeamLength(10);
```

VBA Syntax

```
' Get length of the beam 10.
Dim BeamLen As VARIANT
BeamLen = objOpenStaad.Geometry.GetBeamLength(10)
```

◆ GetBeamList()

void OSGeometryUI::GetBeamList (VARIANT FAR & nBeamList) private

Returns a list of all the member ID(s) the current model.

Parameters

[out] **nBeamList** VARIANT array of LONG type, for storing returned member number ID(s).

C++ Syntax

```
// Get beams list
OSGeometryUI::GetBeamList(&nBeamList);
```

VBA Syntax

```
' Get beams list.
OSGeometryUI.GetBeamList(&nBeamList)
```

See also

OSGeometryUI::GetMemberCount

◆ GetBeamsConnectedAtNode()

VARIANT OSGeometryUI::GetBeamsConnectedAtNode (const VARIANT FAR & nNodeNo,
VARIANT FAR & nBeamList) private

Returns a list of all the beams connected to the specified node.

Parameters

[in] **nNodeNo** Node number ID.

[out] **nBeamList** VARIANT array of LONG type, for storing returned member number ID(s).

C++ Syntax

```
// Get beams list of node #3
VARIANT noOfBeams = OSGeometryUI::GetBeamsConnectedAtNode(3, &nBeamList);
```

VBA Syntax

```
' Get beams list of node #3
Dim noOfBeams As VARIANT = OSGeometryUI.GetBeamsConnectedAtNode(3, &nBeamList)
```

See also

OSGeometryUI::GetMemberCount

OSGeometryUI::GetNoOfBeamsConnectedAtNode

◆ GetCountOfBreakableBeamsAtSpecificNodes()

VARIANT
OSGeometryUI::GetCountOfBreakableBeamsAtSpecificNodes (const VARIANT FAR & nNodeIdArray) private

Returns number of beams that can be broken based on the list of node Ids. This method can be used to get the count of beams that can be used to define the size of output array while using the API BreakBeamsAtSpecificNodes.

Parameters

[in] **nNodeIdArray** array of node numbers to be used to find the number of beams that can be split (type - Long/Integer).

Return values

returns the number of beams that satifies the criteria.

VBA Syntax

```
Option Explicit

    Sub Main
       Dim objOpenStaad As Object
       Dim stdFile As String

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

       ' declare variables
       Dim varNodeIds(0) As Long
       Dim varReturnVal As Long

       ' initialize variables
       ' fill in the array. note node 2 lies between node 1 and node 3, node 1 and node 3
        are the two ends of member 1
       varNodeIds(0) = 2

       ' call method GetCountOfBreakableBeamsAtSpecificNodes
       varReturnVal =
        objOpenStaad.Geometry.GetCountOfBreakableBeamsAtSpecificNodes(varNodeIds)

       ' process the return value
       If varReturnVal > 0 Then
         MsgBox"Breakable beams found"
       Else
         MsgBox"Breakable beams not found"
       End If

       Set objOpenStaad = Nothing
    End Sub
```


Remarks

Method will fail if the specified nodes are not lying on any beam. Method will also fail if there is data type mismatch.

See also

OSGeometryUI::BreakBeamsAtSpecificNodes

GetIntersectBeamsCount()

afx_msg VARIANT OSGeometryUI::GetIntersectBeamsCount (const VARIANT FAR & BeamNosArray, const VARIANT FAR & varTolerance) private

Returns number of new beams that will be created if the specified list of beams are intersected. This method can be used to get the count of beams that can be used to define the size of output array while using the API **OSGeometryUI::IntersectBeams**.

Parameters

[in] **BeamNosArray** Array of Beams numbers. If the array is either null or empty then all members in current model will be considered (Long)

[in] **dTolerance** Tolerance to be used for finding beam intersection, should not be negative value, meter for Metric and inch for English in Base Unit(float/double)

Return values

returns the number of beams that satifies the criteria.

VBA Syntax

```
Option Explicit

    Sub Main
       Dim objOpenStaad As Object
       Dim stdFile As String

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

       Dim beams() As Long
       ReDim beams(0) As Long
       Dim newBeamCount As Long
       newBeamCount= objOpenStaad.Geometry.GetIntersectBeamsCount(beams, 0.01)

       ' process the return value
       If newBeamCount > 0 Then
         MsgBox"Intesection beams found"
       Else
         MsgBox"Intesection beams not found"
       End If

       Set objOpenStaad = Nothing
    End Sub
```

Remarks

Method will fail if there is no beams intersected. Method will also fail if there is data type mismatch.

OSGeometryUI::IntersectBeams

GetLastBeamNo()

VARIANT OSGeometryUI::GetLastBeamNo ( ) private

Returns the member number ID of the last beam in the model.

Return values

<Val> The number of the highest beam number ID in the model (Type: Long)

-1 General error.

C++ Syntax

```
// Get last beam #.
VARIANT LastBeamNo = OSGeometryUI::GetLastBeamNo();
```

VBA Syntax

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

    Set objOpenStaad = GetObject(,"StaadPro.OpenSTAAD")
    objOpenStaad.GetSTAADFile stdFile, "TRUE"
    If stdFile="" Then
        MsgBox"Bad"
         Set objOpenStaad = Nothing
         Exit Sub
     End If
    LastBeamNo = objOpenStaad.Geometry.GetLastBeamNo()
    MsgBox"Beam Number: '" & LastBeamNo & "'."
    Set objOpenStaad = Nothing
End Sub
```

GetMemberCount()

VARIANT OSGeometryUI::GetMemberCount ( ) private

Returns the total number of members in the current model.

Return values

<Val> The total number of member(s).

C++ Syntax

```
// Count for the members.
VARIANT lMemberCount = OSGeometryUI::GetMemberCount();
```

VBA Syntax

```
' Count for the members.
Dim lMemberCount As VARIANT = OSGeometryUI.GetMemberCount()
```

See also

OSGeometryUI::GetBeamList

◆ GetMemberIncidence()

VARIANT OSGeometryUI::GetMemberIncidence (const VARIANT FAR & nBeamNo,
VARIANT FAR & nNodeA,
VARIANT FAR & nNodeB) private

Returns the number ID(s) of connecting node(s) for specified member.

Parameters

[in] **nBeamNo** Member number ID.

[out] **nNodeA** Number ID of the starting end node (**node$_A$**).

[out] **nNodeB** Number ID of the ending end node(**node$_B$**).

Return values

0 OK.

-1 General error.

-3001 Cannot find Node < **nBeamNo** >.

C++ Syntax

```
VARIANT nBeamNo = 5;
VARIANT nNodeA, nNodeB;
// Get joints of beam # 5.
VARIANT RetVal = OSGeometryUI::GetMemberIncidence(5, &nNodeA, &nNodeB);
```

VBA Syntax

```
Private nBeamNo As VARIANT = 5
Private nNodeA, nNodeB As VARIANT
' Get joints of beam # 5.
Dim RetVal As VARIANT = OSGeometryUI.GetMemberIncidence(5, &nNodeA, &nNodeB)
```

See also

OSGeometryUI::GetBeamListAll

◆ GetMemberIncidence_CIS2()

VARIANT OSGeometryUI::GetMemberIncidence_CIS2 (const VARIANT FAR & nBeamNo,
VARIANT FAR & szName,
VARIANT FAR & nNodeA,
VARIANT FAR & nNodeB) private

Returns the number ID(s) of connecting node(s) for specified member.

Parameters

[in] **nBeamNo** Member number ID.

[out] **szName** (LPCTSTR) unique string ID.

[out] **nNodeA** Number ID of the starting end node (**node$_A$**).

[out] **nNodeB** Number ID of the ending end node(**node$_B$**).

Return values

0 OK.

-1 General error.

-3001 Cannot find Node < **nBeamNo** >.

C++ Syntax

```
VARIANT nBeamNo = 5;
VARIANT nNodeA, nNodeB;
// Get joints of beam # 5.
VARIANT RetVal = OSGeometryUI::GetMemberIncidence_CIS2(5, &szName, &nNodeA, &nNodeB);
```

VBA Syntax

```
' Get joints of beam # 5.
Dim RetVal As VARIANT = OSGeometryUI.GetMemberIncidence_CIS2(5, &szName, &nNodeA,
       &nNodeB)
```

See also

OSGeometryUI::GetBeamListAll

◆ GetMemberUniqueID()

VARIANT OSGeometryUI::GetMemberUniqueID (const VARIANT FAR & nMembNo) private

Returns the unique string ID (GUID) for specified member.

Parameters

[in] **nMembNo** Member number ID.

Return values

<VARIANT> Unique string ID for specified member.

The API would return an empty string if specified member < **nMembNo** > is *not* found

C++ Syntax

```
// Get the unique ID of member #3.
VARIANT szName = OSGeometryUI::GetMemberUniqueID(3);
```

VBA Syntax

```
' Get the unique ID of member #3.
Dim szName As VARIANT = OSGeometryUI.GetMemberUniqueID(3)
```

See also

OSGeometryUI::SetMemberUniqueID

◆ GetNoOfBeamsConnectedAtNode()

VARIANT OSGeometryUI::GetNoOfBeamsConnectedAtNode (const VARIANT FAR & nNodeNo) private

Returns no of beams connected at a specified node.

Parameters

[in] **nNodeNo** Node number ID.

C++ Syntax

```
// Get the unique ID of node #3.
VARIANT noOfBeams = OSGeometryUI::GetNoOfBeamsConnectedAtNode(3);
```

VBA Syntax

```
' Get the unique ID of node #3.
Dim noOfBeams As VARIANT = OSGeometryUI.GetNoOfBeamsConnectedAtNode(3)
```

See also

OSGeometryUI::GetMemberCount

OSGeometryUI::GetBeamsConnectedAtNode

◆ GetNoOfSelectedBeams()

VARIANT OSGeometryUI::GetNoOfSelectedBeams ( ) private

Returns the number of selected member(s).

Return values

The number of selected member(s).

C++ Syntax

```
// Counts for the total number of member(s) selected.
VARIANT NoOfSelectedBeams = OSGeometryUI::GetNoOfSelectedBeams();
```

VBA Syntax

```
' Counts for the total number of member(s) selected.
Dim NoOfSelectedBeams As VARIANT = OSGeometryUI.GetNoOfSelectedBeams()
```

See also

OSGeometryUI::SelectBeam

OSGeometryUI::ClearMemberSelection

OSGeometryUI::SelectMultipleBeams

OSGeometryUI::GetSelectedBeams

◆ GetSelectedBeams()

void OSGeometryUI::GetSelectedBeams (VARIANT FAR & naBeamNos,
VARIANT FAR & nIsSorted) private

Returns a list of selected member(s).

Parameters

[out] **naBeamNos** Returned selected member number ID(s) VARIANAT array.

[in] **nIsSorted** The order of the selection(s): in sorted order (= 1), in the order of selection (= 0).

C++ Syntax

```
// Get selected beam list in the order of selection.
OSGeometryUI::GetSelectedBeams(&naBeamNos, 0);
```

VBA Syntax

```
' Get selected beam list in the order of selection.
OSGeometryUI.GetSelectedBeams(&naBeamNos, 0)
```

See also

OSGeometryUI::SelectBeam

OSGeometryUI::ClearMemberSelection

OSGeometryUI::SelectMultipleBeams

OSGeometryUI::GetNoOfSelectedBeams

◆ IntersectBeams()

VARIANT OSGeometryUI::IntersectBeams (const VARIANT FAR & Method,
const VARIANT FAR & BeamNosArray,
const VARIANT FAR & varTolerance,
VARIANT FAR & NewBeamNosArray) private

A function that takes a list of beam numbers and either identify those that would be split/connected due to overlapping each other and highlighting them on the model or simply performing the intersection routine and returning a list of members resulting from running the intersection routine that have been either modified or added to the model.

Parameters

[in]	**Method**	Pass 1 to highlight the member(s) or 2 to intersect the member(s) (Long/Integer)
[in]	**BeamNosArray**	Array of Beams numbers. If the array is either null or empty then all members in current model will be considered (Long)
[in]	**dTolerance**	Tolerance to be used for finding beam intersection, should not be negative value, meter for Metric and inch for English in Base Unit (float/double)
[out]	**NewBeamNosArray**	The ids of the beams that have been changed and added, only used for intersect method.(type - array of Long)

Return values

0 Failed

1 Succeeded

VBA Syntax

```
Sub Main
    Dim objOpenStaad As Object
    Dim stdFile As String
    'launch STAAD.Pro application and open "Verification Models\02 Trusses\Roof Truss
       Axial Forces.STD" file from the Samples folder
    Set objOpenStaad = GetObject(,"StaadPro.OpenSTAAD")
    objOpenStaad.GetSTAADFile stdFile, "TRUE"
    If stdFile="" Then
        MsgBox"Bad"
        Set objOpenStaad = Nothing
        Exit Sub
    End If
    Dim beams() As Long
    Dim retval As Long
    Dim method As Integer
    ReDim beams(0) As Long
    Dim newBeamCount As Long
    newBeamCount= objOpenStaad.Geometry.GetIntersectBeamsCount(beams, 0.01)
    Dim newBeams() As Long
    ReDim newBeams(newBeamCount-1) As Long
    method =1
    retval = objOpenStaad.Geometry.IntersectBeams(method, beams, 0.01, newBeams)
        MsgBox "Intersect Beams is successful"
```

```
    Else
       MsgBox "Intersect Beams is unsuccessful"
  End If
    Set objOpenStaad = Nothing
End Sub
```

Remarks

Intersect will fail if the beams are not Intersected, Or NewBeamNosArray size is small than the size of beams will be changed and added.

See also

OSGeometryUI::GetIntersectBeamsCount

IsBeam()

VARIANT OSGeometryUI::IsBeam (const VARIANT FAR & nMemberNo,
const VARIANT FAR & dTolAngle) private

Returns if the angle of inclination for specified BEAM member is not more than given tolerance angle (for small angle only).

Parameters

[in] **nMemberNo** Beam member number ID (Type: Long).

[in] **dTolAngle** Tolerance inclination angle (Type: Double).

Return values

1 True

0 False

-3001 Member number ID is not found

C++ Syntax

```
// Is the inclination of beam member #5 not more than 5.0 Deg?
VARIANT GetRes = OSGeometryUI::IsBeam(5, 5.0);
```

VBA Syntax

```
' Is the inclination of beam member #5 not more than 5.0 Deg?
Option Explicit

Sub Main
    Dim objOpenStaad As Object
    Dim stdFile As String
    Dim nMemberNo As Long
    Dim dTolAngle As Double
    Dim GetRes As Long

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

    nMemberNo = 5
    dTolAngle = 5.0
    GetRes = objOpenStaad.Geometry.IsBeam(nMemberNo, dTolAngle)
    If GetRes = 1 Then
         MsgBox"True"
    ElseIf GetRes = 0 Then
         MsgBox"False"
    Else
         MsgBox"Error Code:"&GetRes
    End If
```

```
End Sub
```

Remarks

Do not use this function for column member, because the angle of inclination is considered to be the angle between specified member to horizontal plate. Also, it is not recommended to use this function for inclination of beam larger than 10 Deg.

See also

OSGeometryUI::IsColumn

◆ IsColumn()

VARIANT OSGeometryUI::IsColumn (const VARIANT FAR & nMemberNo,
const VARIANT FAR & dTolAngle) private

Returns if the angle of inclination for specified COLUMN member is not more than given tolerance angle (for small angle only).

Parameters

[in] **nMemberNo** Column member number ID (Type: Long).

[in] **dTolAngle** Tolerance inclination angle (Type: Double).

Return values

1 True

0 False

-3001 Member number ID is not found

C++ Syntax

```
// Is the inclination of column member #5 not more than 5.0 Deg?
VARIANT GetRes = OSGeometryUI::IsColumn(5, 5.0);
```

VBA Syntax

```
' Is the inclination of column member #5 not more than 5.0 Deg?
Option Explicit

Sub Main
    Dim objOpenStaad As Object
    Dim stdFile As String
    Dim nMemberNo As Long
    Dim dTolAngle As Double
    Dim GetRes As Long

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

    nMemberNo = 5
    dTolAngle = 5.0
    GetRes = objOpenStaad.Geometry.IsColumn(nMemberNo, dTolAngle)
    If GetRes = 1 Then
         MsgBox"True"
    ElseIf GetRes = 0 Then
         MsgBox"False"
    Else
         MsgBox"Error Code:"&GetRes
    End If
```

```
End Sub
```

Remarks

Do not use this function for beam member, because the angle of inclination is considered to be the angle between specified member to horizontal plate. Also, it is not recommended to use this function for inclination of column larger than 10 Deg.

See also

OSGeometryUI::IsBeam

◆ IsZUp()

VARIANT OSGeometryUI::IsZUp ( ) private

Returns if Z-axis is in upward direction?

Return values

1 True;

0 False;

C++ Syntax

```
// Z axis going up?
VARIANT RetVal = OSGeometryUI::IsZUp();
```

VBA Syntax

```
' Z axis going up?
Dim RetVal As VARIANT = OSGeometryUI.IsZUp()
```

◆ MergeBeams()

VARIANT OSGeometryUI::MergeBeams (const VARIANT FAR & nBeamIdArray,
const VARIANT FAR & varBeamNo,
const VARIANT FAR & varPropertyNo,
const VARIANT FAR & varBetaAngle,
const VARIANT FAR & varMaterialName) private

Merges multiple collinear and connected beams to a single beam with specified id, property, material and beta angle.

Parameters

[in] **nBeamIdArray** array of beam numbers to be merged (must have more than one beam) (type - Long/Integer).

[in] **varBeamNo** beam number to be assigned to the merged beam (must be present in nBeamIdArray) (type - Long/Integer).

[in] **varPropertyNo** property reference number to be assigned to the merged beam (type - Long/Integer).

[in] **varBetaAngle** beta angle (in degrees) to be assigned to the merged beam (type - float/double).

[in] **varMaterialName** material name to be assigned to the merged beam (type - String).

Return values

1 if merging is successful.

0 if merging is unsuccessful.

VBA Syntax

```
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

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

       ' declare variables
       Dim varMembers(2) As Long
       Dim varMemberToKeep As Long
       Dim varPropertyToKeep As Long
       Dim varBetaToKeep As Double
```

```
        ' initialize variables
        ' fill in the array. note 5, 6 & 7 are collinear and connected
        varMembers(0) = 5
        varMembers(1) = 6
        varMembers(2) = 7

        varMemberToKeep = 6 ' beam id should be present in the array of beams to be merged
        varPropertyToKeep = 4 'property id should exist in the property list
        varBetaToKeep = 0.0 ' beta angle should exist in the beta command list, use 0.0
         for none
        varMaterialToKeep = "Steel" 'material name should exist in the material definition
         block

        ' call method MergeBeams
        varReturnVal = objOpenStaad.Geometry.MergeBeams(varMembers, varMemberToKeep,
         varPropertyToKeep, varBetaToKeep, varMaterialToKeep)

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

        Set objOpenStaad = Nothing
    End Sub
```

Remarks

Merging will fail if the beams are not collinear or disconnected. If array contains only one beam or the merged beam number is not present in the array. Merging will be unsuccessful if property, material and beta angle are not defined in the model. Merging will also fail if there is data type mismatch.

◆ RenumberBeam()

VARIANT OSGeometryUI::RenumberBeam (const VARIANT FAR & varBeamNoOld,
const VARIANT FAR & varBeamNoNew) private

Renumbers the existing beam id with the specified id.

Parameters

[in] **varBeamNoOld** Old beam ID.

[in] **varBeamNoNew** New beam ID.

Return values

1 True;

0 False;

C++ Syntax

```
// Renumber old beam ID from #5 to a new beam ID #3.
VARIANT RetVal = OSGeometryUI::RenumberBeam(5, 3);
```

VBA Syntax

```
' Renumber old beam ID from #5 to a new beam ID #3.
Dim RetVal As VARIANT = OSGeometryUI.RenumberBeam(5, 3)
```

See also

OSGeometryUI::GetMemberCount

OSGeometryUI::GetBeamsConnectedAtNode

◆ SelectBeam()

VARIANT OSGeometryUI::SelectBeam (const VARIANT FAR & nBeamNo) private

Selects the specified member in current model.

Parameters

[in] **nBeamNo** Member number ID.

C++ Syntax

```
//Select member # 3.
OSGeometryUI::SelectBeam(3);
```

VBA Syntax

```
' Select member # 3.
OSGeometryUI.SelectBeam(3)
```

See also

OSGeometryUI::SelectMultipleBeams

OSGeometryUI::ClearMemberSelection

OSGeometryUI::GetNoOfSelectedBeams

OSGeometryUI::GetSelectedBeams

◆ SelectMultipleBeams()

VARIANT OSGeometryUI::SelectMultipleBeams (const VARIANT FAR & naBeamNos) private

Selects multiple member(s) in current model.

Parameters

[in] **naBeamNos** Member number ID(s) VARIANT array (type - Long/Integer).

Return values

0 Failed

1 Succeeded

C++ Syntax

```
// Select multiple beams.
OSGeometryUI::SelectMultipleBeams(naBeamNos);
```

VBA Syntax

```
Option Explicit
    Sub Main
        Dim objOpenStaad As Object
        Dim stdFile As String
        'launch STAAD.Pro application and open "Verification Models\02 Trusses\Roof Truss
       Axial Forces.STD" file from the Samples folder
        Set objOpenStaad = GetObject(,"StaadPro.OpenSTAAD")
        objOpenStaad.GetSTAADFile stdFile, "TRUE"
        If stdFile="" Then
            MsgBox"Bad"
            Set objOpenStaad = Nothing
            Exit Sub
        End If
        Dim Objs(1) As Long
        Objs(0)=5
        Objs(1)=10
        Dim bRes As Boolean
        bRes = objOpenStaad.Geometry.SelectMultipleBeams( Objs)
        If bRes Then
            MsgBox"Success"
        Else
            MsgBox"Failed"
        End If
    End Sub
```

See also

OSGeometryUI::SelectBeam

OSGeometryUI::ClearMemberSelection

OSGeometryUI::GetNoOfSelectedBeams

OSGeometryUI::GetSelectedBeams

- SetCheckForIdenticalEntity()

VARIANT OSGeometryUI::SetCheckForIdenticalEntity (long entityType,
long bEnable) private

This API will set whether to enable checking for existing identical entities (beam, plate, node etc.) or not. If set is enabled, time taken by the corresponding add/create multiple entities APIs will take longer time, otherwise if set is disabled, time taken by corresponding APIs will be comparatively less. Please refer to the code snippet below.

Parameters

[in] **entityType** Identification for the entity type(enum STAADEntityType).

value	entity Type
0	NodeEntity
1	MemberEntity
2	PlateEntity
3	SolidEntity
4	SurfaceEntity
5	PhysicalMemberEntity
6	ParametricSurface

[in] **bEnable** Whether to enable identical enitity check or not. (0 = FALSE, 1 = TRUE)

Return values

0 (=FALSE) or 1 (=TRUE)

C++ Syntax

```
// Create Multiple Plates - Setting Identical check for Plate Entity as 0 (=FALSE). So,
       while creating multiple plates in the subsequent API call, it will not check for
// existing identical plates while creating plates resulting in faster operation. If,
       check is set to 1 (=TRUE), it will check for identical plates
// in the subsequent add multiple plate API which will take longer time.
int nPlateIds[] = {91, 92, 93}
int nPlateIncidences[][4] = {{3, 5, 15, 13}, {15, 25, 27, 17}, {47, 27, 29, 49}}
OSGeometryUI::SetCheckForIdenticalEntity(STAADEntityType.PlateEntity, 0);
OSGeometryUI::CreateMultiplePlates(nPlateIds, nPlateIncidences);
```

C# Syntax

```
// Create Multiple Plates - Setting Identical check for Plate Entity as 0 (=FALSE). So,
       while creating multiple plates in the subsequent API call, it will not check for
// existing identical plates while creating plates resulting in faster operation. If,
       check is set to 1 (=TRUE), it will check for identical plates
// in the subsequent add multiple plate API which will take longer time.

int[] nPlateIdArray = new int[3] {91, 92, 93};
int[,] nPlateIncidenceArray = new int[3, 4] {{3, 5, 15, 13 }, {15, 25, 27, 17 }, {47, 27,
```



```
Object objectPlateIdArray = nPlateIdArray as Object;
```

```
int rValue1 = m_OStd.Geometry.SetCheckForIdenticalEntity(STAADEntityType.PlateEntity, 0);
rValue1 = m_OStd.Geometry.CreateMultiplePlates(objectPlateIdArray,
        objectPlateIncidenceArray);
```

VBA Syntax

```
// Create Multiple Plates - Setting Identical check for Plate Entity as 0 (=FALSE). So,
        while creating multiple plates in the subsequent API call, it will not check for
// existing identical plates while creating plates resulting in faster operation. If,
        check is set to 1 (=TRUE), it will check for identical plates
// in the subsequent add multiple plate API which will take longer time.

Dim nPlateIds(1) As Integer
nPlateIds(0) = 91
nPlateIds(1) = 92
Dim nPlateIncidences(1,3) As Integer
nPlateIncidences(0,0) = 3
nPlateIncidences(0,1) = 5
nPlateIncidences(0,2) = 15
nPlateIncidences(0,3) = 13
nPlateIncidences(1,0) = 15
nPlateIncidences(1,1) = 25
nPlateIncidences(1,2) = 27
nPlateIncidences(1,3) = 17
objOpenStaad.Geometry.SetCheckForIdenticalEntity(STAADEntityType.PlateEntity, 0);
objOpenStaad.Geometry.CreateMultiplePlates(nPlateIds,nPlateIncidences)
```

See also

OSGeometryUI::CreateMultiplePlates

SetMemberUniqueID()

void OSGeometryUI::SetMemberUniqueID (const VARIANT FAR & nMembNo,
const VARIANT FAR & szName) private

Assigns an unique string ID (GUID) to specified member.

Parameters

[in] **nMembNo** Member number ID.

[in] **szName** (LPCTSTR) unique string ID.

C++ Syntax

```
// Set "MEMBERSP" to member #3.
OSGeometryUI::SetMemberUniqueID(3, (LPCTSTR)"MEMBERSP");
```

VBA Syntax

```
' Set "MEMBERSP" to member #3.
OSGeometryUI.SetMemberUniqueID(3, "MEMBERSP")
```

See also

OSGeometryUI::GetMemberUniqueID

◆ SplitBeam()

void OSGeometryUI::SplitBeam (const VARIANT FAR & nBeamNo,
const VARIANT FAR & nNodes,
const VARIANT FAR & faDistToNodes) private

Split a specified beam into several beams by specified node(s).

Parameters

[in] **nBeamNo** Number ID of the beam to split.

[in] **nNodes** The number of node(s) to be inserted in the beam.

[in] **faDistToNodes** VARIANT array of distance(s) in length from the starting end node of member.

C# Syntax

```
OpenSTAADUI.OpenSTAAD os = Marshal.GetActiveObject("StaadPro.OpenSTAAD") as
       OpenSTAADUI.OpenSTAAD;
OpenSTAADUI.OSGeometryUI geom = os.Geometry;
long nBeamNo = 10;
long nNodes = 3;
double[] faDistToNodes = {1.0, 4.0, 3.0};
// Add split Beam # 10 into three unequal parts.
Object objectNodeDistArr = faDistToNodes as Object;
geom.SplitBeam(nBeamNo, nNodes, objectNodeDistArr);
```

VBA Syntax

```
Option Explicit

Sub Main
    Dim objOpenStaad As Object
    Dim stdFile As String
    Dim nBeamNo As Long
    Dim nNodes As Long
    Dim faDistToNodes(2) As Double

    Set objOpenStaad = GetObject(,"StaadPro.OpenSTAAD")
    objOpenStaad.GetSTAADFile stdFile, "TRUE"
    nBeamNo = 10
    nNodes = 3
    faDistToNodes(0) = 1.0
    faDistToNodes(1) = 2.0
    faDistToNodes(2) = 3.0
' Add split Beam # 10 into three unequal parts.
    objOpenStaad.Geometry.SplitBeam(nBeamNo, nNodes, faDistToNodes)
End Sub
```

See also

OSGeometryUI::SplitBeamInEqlParts

void OSGeometryUI::SplitBeamInEqlParts (const VARIANT FAR & nBeamNo,
const VARIANT FAR & nParts) private

Split a specified beam into several EQUAL beams by specified number of node(s)

Parameters

[in] **nBeamNo** Number ID of the beam to split.

[in] **nParts** The number of parts into which the beam is to be split.

C++ Syntax

```
long nBeamNo = 10;
long nParts = 3;
// Split Beam # 10 into three equal parts.
OSGeometryUI::SplitBeamInEqlParts(nBeamNo, nParts)
```

VBA Syntax

```
Dim nBeamNo As Long = 10
Dim nParts As Long = 3
' Split Beam # 10 into three equal parts.
OSGeometryUI.SplitBeamInEqlParts(nBeamNo, nParts)
```

See also

OSGeometryUI::SplitBeam