# Result: Nodes/Joints/Supports

**Analysis Results**

## Functions

| | |
|---|---|
| afx_msg VARIANT | **OSOutputUI::GetNodeDisplacements** (const VARIANT FAR &nNodeNo, const VARIANT FAR &nLC, VARIANT FAR &pdDisps)<br>Returns nodal displacements for the node number and load case specified. |
| afx_msg VARIANT | **OSOutputUI::GetSupportReactions** (const VARIANT FAR &nNodeNo, const VARIANT FAR &nLC, VARIANT FAR &pdReactions)<br>Returns support reactions for the node number and load case specified. |
| afx_msg VARIANT | **OSOutputUI::GetMatInfluenceAreas** (VARIANT FAR &nNodes, VARIANT FAR &varYZAreas, VARIANT FAR &varZXAreas, VARIANT FAR &varXYAreas)<br>Returns the mat influence areas for nodes supported using ELASTIC MAT command. |
| afx_msg VARIANT | **OSOutputUI::GetBasePressures** (const VARIANT FAR &varLC, VARIANT FAR &nNodes, VARIANT FAR &varYZPress, VARIANT FAR &varZXPress, VARIANT FAR &varXYPress)<br>Returns base presure in X, Y and Z direction using Base Presure command. |

## Detailed Description

These functions are related to output analysis results in terms of nodes/joints and supports.

## Function Documentation

### ◆ GetBasePressures()

```
VARIANT OSOutputUI::GetBasePressures ( const VARIANT FAR & varLC,
                                       VARIANT FAR &       nNodes,
                                       VARIANT FAR &       varYZPress,
                                       VARIANT FAR &       varZXPress,
                                       VARIANT FAR &       varXYPress )
```

Returns base presure in X, Y and Z direction using Base Presure command.

**Parameters**

| | | |
|---|---|---|
| [in] | **nNodes** | Array containing the support node numbers. |
| [in] | **varLC** | Load Case reference ID. |
| [out] | **varYZPress** | Array containing the base pressures in X direction per load case reference ID |
| [out] | **varZXPress** | Array containing the base pressures in Y direction per load case reference ID |
| [out] | **varXYPress** | Array containing the base pressures in Z direction per load case reference ID |

**Return values**

**TRUE**

1 OK.

**Return values**

**FALSE**

0 Error.

**VBA Syntax**

```
'Get Base Pressures
Sub BasePressures()
  Dim RetVal As Variant
  Dim Lcase As Long
  Dim count As Integer
  Dim nNodes() As Long
  Dim varYZPress() As Double
  Dim varZXPress() As Double
  Dim varXYPress() As Double

'Launch OpenSTAAD Object
On Error GoTo ErrHandler
  Set objOpenSTAAD = GetObject(, "StaadPro.OpenSTAAD")

'Is Analysis Completed
  Cells(1, 2).Value = objOpenSTAAD.Output.AreResultsAvailable()

'Get Base Pressures
  Lcase = Cells(2, 2).Value     'LoadCase = 2
  count = objOpenSTAAD.Support.GetSupportCount
  ReDim nNodes(0 To (count - 1)) As Long
  ReDim varYZPress(0 To (count - 1)) As Double
  ReDim varZXPress(0 To (count - 1)) As Double
  ReDim varXYPress(0 To (count - 1)) As Double
  Dim nodeArrCount As Variant
  nodeArrCount = objOpenSTAAD.Support.GetSupportNodes(nNodes)
  RetVal = objOpenSTAAD.Output.GetBasePressures(Lcase, nNodes, varYZPress, varZXPress,
        varXYPress)
  For i = 1 To count
  Cells(2 + i, 1).Value = nNodes(i - 1)
  Next
  For i = 1 To count
  Cells(2 + i, 2).Value = varYZPress(i - 1)
  Next
  For i = 1 To count
  Cells(2 + i, 3).Value = varZXPress(i - 1)
  Next
  For i = 1 To count
  Cells(2 + i, 4).Value = varXYPress(i - 1)
  Next

  Set objOpenSTAAD = Nothing
Exit Sub

ErrHandler:
  MsgBox ("Run StaadPro First" & vbCrLf)
Resume Next
End Sub
```

## GetMatInfluenceAreas()

VARIANT OSOutputUI::GetMatInfluenceAreas ( VARIANT FAR & nNodes,
VARIANT FAR & varYZAreas,
VARIANT FAR & varZXAreas,
VARIANT FAR & varXYAreas )

Returns the mat influence areas for nodes supported using ELASTIC MAT command.

**Parameters**

`[in]` **nNodes** Array containing the support node numbers.

`[out]` **varYZAreas** Array containing the influence areas in YZ plase ordered per the node nos. in nodeNos array

`[out]` **varZXAreas** Array containing the influence areas in ZX plase ordered per the node nos. in nodeNos array

`[out]` **varXYAreas** Array containing the influence areas in XY plase ordered per the node nos. in nodeNos array

**Return values**

**0** OK.

**-107** Array of double expected.

**-108** Array size is smaller than expected.

**VBA Syntax**

```
'Get Mat Influence Areas
Sub MatInfluenceAreas()
  Dim RetVal As Variant
  Dim count As Integer
  Dim nNodes() As Long
  Dim varYZAreas() As Double
  Dim varZXAreas() As Double
  Dim varXYAreas() As Double

'Launch OpenSTAAD Object
On Error GoTo ErrHandler
  Set objOpenSTAAD = GetObject(, "StaadPro.OpenSTAAD")

'Is Analysis Completed
  Cells(1, 2).Value = objOpenSTAAD.Output.AreResultsAvailable()

'Get Mat Influence Areas
  count = objOpenSTAAD.Support.GetSupportCount
  ReDim nNodes(0 To (count - 1)) As Long
  ReDim varYZAreas(0 To (count - 1)) As Double
  ReDim varZXAreas(0 To (count - 1)) As Double
  ReDim varXYAreas(0 To (count - 1)) As Double
  Dim nodeArrCount As Variant
  nodeArrCount = objOpenSTAAD.Support.GetSupportNodes(nNodes)
  RetVal = objOpenSTAAD.Output.GetMatInfluenceAreas(nNodes, varYZAreas, varZXAreas,
        varXYAreas)
  For i = 1 To count
  Cells(2 + i, 1).Value = nNodes(i - 1)
  Next
  For i = 1 To count
  Cells(2 + i, 2).Value = varYZAreas(i - 1)
  Next
  For i = 1 To count
  Cells(2 + i, 3).Value = varZXAreas(i - 1)
  Next
  For i = 1 To count
  Cells(2 + i, 4).Value = varXYAreas(i - 1)
  Next

  Set objOpenSTAAD = Nothing
Exit Sub

ErrHandler:
  MsgBox ("Run StaadPro First" & vbCrLf)
Resume Next
End Sub
```

## GetNodeDisplacements()

VARIANT OSOutputUI::GetNodeDisplacements ( const VARIANT FAR & nNodeNo,
const VARIANT FAR & nLC,
VARIANT FAR & pdDisps )

Returns nodal displacements for the node number and load case specified.

**Parameters**

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

[in] **nLC** Load Case reference ID.

[out] **pdDisps** VARIANT array with nodal translational displacements in X, Y and Z directions, rotation about X, Y and Z directions, respectively.

**Returns**

Boolean (TRUE/FALSE) whether succeeded or not.

**VBA Syntax**

```
'Get Node Displacement at node #10 for Load case #1
Sub NodeDisplacements()
  Dim Node As Long
  Dim Lcase As Long
  Dim Disp(0 To 5) As Double
  Dim i As Integer
  Dim RetVal As Variant

'Launch OpenSTAAD Object
On Error GoTo ErrHandler
  Set objOpenSTAAD = GetObject(, "StaadPro.OpenSTAAD")

'Is Analysis Completed
  Cells(1, 2).Value = objOpenSTAAD.Output.AreResultsAvailable()

'Node Displacement
  Lcase = Cells(7, 2).Value  'Loadcase ID = 1
  Node = Cells(9, 1).Value  'Node ID = 10
  RetVal = objOpenSTAAD.Output.GetNodeDisplacements(Node, Lcase, Disp)
  For i = 1 To 6
  Cells(9, i + 1).Value = Disp(i - 1)
  Next

  Set objOpenSTAAD = Nothing
Exit Sub

ErrHandler:
  MsgBox ("Run StaadPro First" & vbCrLf)
Resume Next
End Sub
```

## ◆ GetSupportReactions()

VARIANT OSOutputUI::GetSupportReactions ( const VARIANT FAR & nNodeNo,
const VARIANT FAR & nLC,
VARIANT FAR & pdReactions )

Returns support reactions for the node number and load case specified.

**Parameters**

| | | |
|---|---|---|
| [in] | **nNodeNo** | Node number ID. |
| [in] | **nLC** | Load Case reference ID. |
| [out] | **pdReactions** | Support Reaction in GLOBAL direction:[ FX, FY, FZ, MX, MY, MZ]. |

**Returns**

Boolean (TRUE/FALSE) whether succeeded or not.

**VBA Syntax**

```
'Get Support Reactions at node #2 for Load case #1
Sub SupportReactions()
  Dim Lcase As Long
  Dim supp_nodes As Long
  Dim dReactionArray(0 To 5) As Double
  Dim count1 As Integer
  Dim RetVal As Variant

'Launch OpenSTAAD Object
On Error GoTo ErrHandler
  Set objOpenSTAAD = GetObject(, "StaadPro.OpenSTAAD")

'Is Analysis Completed
  Cells(1, 2).Value = objOpenSTAAD.Output.AreResultsAvailable()

'Support Reactions
  Lcase = Cells(11, 2).Value 'Loadcase ID = 1
  supp_nodes = Cells(13, 1).Value 'Node ID = 2
  RetVal = objOpenSTAAD.Output.GetSupportReactions(supp_nodes, Lcase, dReactionArray)
  For count1 = 1 To 6
  Cells(13, count1 + 1).Value = dReactionArray(count1 - 1)
  Next

  Set objOpenSTAAD = Nothing
Exit Sub

ErrHandler:
  MsgBox ("Run StaadPro First" & vbCrLf)
Resume Next
End Sub
```