Result: Dynamic

Analysis Results

Functions

afx_msg VARIANT	**OSOutputUI::GetNLLoadStep** (const VARIANT FAR &nLC) Returns the Load Step value used for nonlinear analysis for specified Load Case.
afx_msg VARIANT	**OSOutputUI::GetNLNodeDisplacements** (const VARIANT FAR &nNodeNo, const VARIANT FAR &nLC, const VARIANT FAR &loadStep, VARIANT FAR &loadLevel, VARIANT FAR &pdDisps) Returns the Load Level value and nodal displacements for specified node, Load Case, and Load Step.
afx_msg VARIANT	**OSOutputUI::GetNoOfModesExtracted** () Returns the number of modes extracted by a dynamic analysis.
afx_msg VARIANT	**OSOutputUI::GetModeFrequency** (const VARIANT FAR &varMode, VARIANT FAR &varFreq) Returns the natural frequency (Hz) for a specified mode.
afx_msg VARIANT	**OSOutputUI::GetModalDisplacementAtNode** (const VARIANT FAR &varMode, const VARIANT FAR &varNode, VARIANT FAR &varModalDisps) Returns the modal displacement at a specified node number and mode.
afx_msg VARIANT	**OSOutputUI::GetModalMassParticipationFactors** (const VARIANT FAR &varMode, VARIANT FAR &varfactorX, VARIANT FAR &varfactorY, VARIANT FAR &varfactorZ) Returns the modal participation factors for a specified mode number.
afx_msg VARIANT	**OSOutputUI::GetMissingMassParticipationFactors** (const VARIANT FAR &varLC, VARIANT FAR &varfactorX, VARIANT FAR &varfactorY, VARIANT FAR &varfactorZ) Returns the participation factors for missing mass mode for specified Load Case.
afx_msg VARIANT	**OSOutputUI::GetTimeHistoryIntegrationStepInfo** (double *deltaT) Returns the time-step (secs) used for time-history integration.
afx_msg VARIANT	**OSOutputUI::GetTimeHistoryResponseAtTime** (long lCase, long nodeNo, long dofNo, long responseType, double atTime, double *response) Returns the response at a spefictime within the integration time span at a specified node for a given DOF.
afx_msg VARIANT	**OSOutputUI::GetTimeHistoryResponse** (long lCase, long nodeNo, long dofNo, long responseType, VARIANT FAR *responses) Returns the time-history responses of DOF at specified node in the VARIANT array responses.
afx_msg VARIANT	**OSOutputUI::GetTimeHistoryResponseMinMax** (long lCase, long nodeNo, long dofNo, long responseType, double *responseMax, double *timeMax, double *responseMin, double *timeMin)

	Returns the min/max time-history responses of DOF at specified node.
afx_msg long	**OSOutputUI::GetMaxBeamStresses** (long nBeamNo, long nLC, double *pdCompStress, long *nCompCorner, double *pdTensileStress, long *nTensileCorner) Gets maximum beam Stresses for Beam.

Detailed Description

These functions are related to output dynamic analysis.

Function Documentation

◆ GetMaxBeamStresses()

```
long OSOutputUI::GetMaxBeamStresses ( long     nBeamNo,
                                      long     nLC,
                                      double * pdCompStress,
                                      long *   nCompCorner,
                                      double * pdTensileStress,
                                      long *   nTensileCorner )
```

Gets maximum beam Stresses for Beam.

Parameters

[in]	**nBeamNo**	Beam ID.
[in]	**nLC**	Load Case reference ID.
[out]	**pdCompStress**	Value of maximum compressive stress
[out]	**nCompCorner**	0 for non-corner-section material (eg. prismatic circle and pipe), 1 to 4 for section corner where maximum compressive stress was found.
[out]	**pdTensileStress**	Value of maximum tensile stress
[out]	**nTensileCorner**	0 for non-corner-section material (eg. prismatic circle and pipe), 1 to 4 for section corner where maximum tensile stress was found.

Return values

0	OK.
-3001	Cannot find member **nBeamNo**.
-8002	Load Case **nLC** not found.
-9004	Beam Forces results not loaded.

C++ Syntax

```
// Get maximum beam stresses for Beam #2, Load case #1.
long RetVal = OSOutputUI::GetMaxBeamStresses(2, 1, &pdCompStress, &nCompCorner,
       &pdTensileStress, &nTensileCorner);
```

VB 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
' Get Max Beam Stress for Beam #2, Load case #1
         Dim nBeamNo As Long
         Dim nLC As Long
```

```
          Dim pdCompStress As Double
          Dim nCompCorner As Long
          Dim pdTensileStress As Double
          Dim nTensileCorner As Long
          Dim nReturn As Long
          Dim MessageText As String

          nBeamNo = 2
          nLC= 1
          nReturn = objOpenStaad.Output.GetMaxBeamStresses(nBeamNo, nLC, pdCompStress,
        nCompCorner, pdTensileStress, nTensileCorner)
          If (nReturn = 0) Then
            MessageText = "Value of maximum compressive stress," &Str(pdCompStress) &
        Chr(10)
            MessageText = MessageText & "nCompCorner, " & Str(nCompCorner) & Chr(10)
            MessageText = MessageText & "Value of maximum tensile stress, " &
        Str(pdTensileStress) & Chr(10)
            MessageText = MessageText & "nTensileCorner, " &Str(nTensileCorner) & Chr(10)
            MsgBox(MessageText)
         Else
            MsgBox("Unsuccessful!!")
          End If
End Sub
```

GetMissingMassParticipationFactors()

VARIANT OSOutputUI::GetMissingMassParticipationFactors (const VARIANT FAR & varLC,
VARIANT FAR & varfactorX,
VARIANT FAR & varfactorY,
VARIANT FAR & varfactorZ)

Returns the participation factors for missing mass mode for specified Load Case.

Parameters

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

[out] **varfactorX** Missing mass modal participation factor for X direction.

[out] **varfactorY** Missing mass modal participation factor for Y direction.

[out] **varfactorZ** Missing mass modal participation factor for Z direction.

Returns

Boolean (TRUE/FALSE) whether succeeded or not.

C++ Syntax

```
// Get participation factors for missing mass mode in Load Case #100
VARIANT RetVal = OSOutputUI::GetMissingMassParticipationFactors(100, &varfactorX,
       &varfactorY, &varfactorZ);
```

VBA Syntax

```
' Get modal participation factors for missing mass mode in Load Case #100
Dim RetVal As VARIANT = OSOutputUI.GetMissingMassParticipationFactors(100, &varfactorX,
       &varfactorY, &varfactorZ)
```

◆ GetModalDisplacementAtNode()

VARIANT OSOutputUI::GetModalDisplacementAtNode (const VARIANT FAR & varMode,
const VARIANT FAR & varNode,
VARIANT FAR & varModalDisps)

Returns the modal displacement at a specified node number and mode.

Parameters

[in]	**varMode**	Variant variable containing the mode number.
[in]	**varNode**	Variant variable containing the node number.
[out]	**varModalDisps**	Variant array of dimension six, which returns nodal displacements.

Returns

Boolean (TRUE/FALSE) whether succeeded or not.

VBA Syntax

```
'Get Modal Displacement At Node
Sub ModalDisplacementAtNode()
  Dim RetVal As Variant
  Dim Disp(0 To 5) As Double
  Dim varNode As Long
  Dim varMode As Long

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

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

'Get Modal Displacement At Node
  varMode = Cells(96, 2).Value        'Mode = 4
  varNode = Cells(98, 1).Value        'Node = 7
  RetVal = objOpenSTAAD.Output.GetModalDisplacementAtNode(varMode, varNode, Disp)
  'Get Modal Displacement At Node #7 and Mode #4
  For count = 1 To 6
  Cells(98, count + 1).Value = Disp(count - 1)
  Next

  Set objOpenSTAAD = Nothing
Exit Sub

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

◆ GetModalMassParticipationFactors()

VARIANT OSOutputUI::GetModalMassParticipationFactors (const VARIANT FAR & varMode,
VARIANT FAR & varfactorX,
VARIANT FAR & varfactorY,
VARIANT FAR & varfactorZ)

Returns the modal participation factors for a specified mode number.

Parameters

[in] **varMode** Mode index.

[out] **varfactorX** Modal mass participation factor for X direction.

[out] **varfactorY** Modal mass participation factor for Y direction.

[out] **varfactorZ** Modal mass participation factor for Z direction.

Returns

Boolean (TRUE/FALSE) whether succeeded or not.

VBA Syntax

```
'Get Modal Mass Participation Factors
Sub ModalMassParticipationFactors()
  Dim RetVal As Variant
  Dim varMode As Long
  Dim patX As Double
  Dim patY As Double
  Dim patZ 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 Modal Mass Participation Factors
  varMode = Cells(101, 1).Value        'Mode = 2
  RetVal = objOpenSTAAD.Output.GetModalMassParticipationFactors(varMode, patX, patY,
       patZ)
  'Get Modal Mass Participation Factors for Mode #2.
  Cells(101, 2).Value = patX
  Cells(101, 3).Value = patY
  Cells(101, 4).Value = patZ

  Set objOpenSTAAD = Nothing
Exit Sub

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

GetModeFrequency()

VARIANT OSOutputUI::GetModeFrequency (const VARIANT FAR & varMode,
VARIANT FAR & varFreq)

Returns the natural frequency (Hz) for a specified mode.

Parameters

[in] **varMode** Variant variable containing the mode number.

[out] **varFreq** Variant variable for storing the frequency value.

Returns

Boolean (TRUE/FALSE) whether succeeded or not.

VBA Syntax

```
'Get Mode Frequency
Sub ModeFrequency()
  Dim RetVal As Variant
  Dim varFreq As Double
  Dim varMode As Long

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

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

'Get Mode Frequency
  varMode = Cells(93, 2).Value        'Mode = 6
  RetVal = objOpenSTAAD.Output.GetModeFrequency(varMode, varFreq)
  'Get Mode Frequency for Mode #6
  Cells(94, 2).Value = varFreq

  Set objOpenSTAAD = Nothing
Exit Sub

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

GetNLLoadStep()

VARIANT OSOutputUI::GetNLLoadStep (const VARIANT FAR & nLC)

Returns the Load Step value used for nonlinear analysis for specified Load Case.

Parameters

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

Return values

<Val> Load Step value.

-1 General error.

-8002 Load Case **nLC** not found.

-9911 Nonlinear result set is not available.

VBA Syntax

```
 ' Get Load Step value for Load Case #1
Sub NLLoadStep()
  Dim Lcase As Long
  Dim RetValNL As Variant

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

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

'Get Load Step value
  Lcase = Cells(20, 2).Value 'LoadCase = 1
  RetValNL = objOpenSTAAD.Output.GetNLLoadStep(Lcase)
  MsgBox (RetValNL & vbCrLf)

Set objOpenSTAAD = Nothing
Exit Sub

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

◆ GetNLNodeDisplacements()

```
VARIANT OSOutputUI::GetNLNodeDisplacements ( const VARIANT FAR & nNodeNo,
                                             const VARIANT FAR & nLC,
                                             const VARIANT FAR & loadStep,
                                             VARIANT FAR &       loadLevel,
                                             VARIANT FAR &       pdDisps )
```

Returns the Load Level value and nodal displacements for specified node, Load Case, and Load Step.

Parameters

[in]	**nNodeNo**	Node number ID.
[in]	**nLC**	Load Case reference ID.
[in]	**loadStep**	Load Step index.
[out]	**loadLevel**	Value of Load Level.
[out]	**pdDisps**	Nodal displacement in GLOBAL [X, Y, Z, rX, rY, rZ].

Returns

Boolean (TRUE/FALSE) whether succeeded or not.

C++ Syntax

```
// Get Load Level and nodal displacements for node #1, Load Case #100, Load Step #30.
VARIANT RetVal = OSOutputUI::GetNLNodeDisplacements(1, 100, 30, &loadLevel, &pdDisps);
```

VBA Syntax

```
' Get Load Level and nodal displacements for node #1, Load Case #100, Load Step #30.
Dim RetVal As VARIANT = OSOutputUI.GetNLNodeDisplacements(1, 100, 30, &loadLevel,
       &pdDisps)
```

◆ GetNoOfModesExtracted()

VARIANT OSOutputUI::GetNoOfModesExtracted ( )

Returns the number of modes extracted by a dynamic analysis.

Returns

The number of mode(s) extracted by a dynamic analysis.

VBA Syntax

```
'Get the number of modes.
Sub NoOfModesExtracted()
  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()

'Get No Of Modes Extracted
  RetVal = objOpenSTAAD.Output.GetNoOfModesExtracted
  MsgBox (RetVal & vbCrLf)

  Set objOpenSTAAD = Nothing
Exit Sub

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

◆ GetTimeHistoryIntegrationStepInfo()

VARIANT OSOutputUI::GetTimeHistoryIntegrationStepInfo (double * deltaT)

Returns the time-step (secs) used for time-history integration.

Parameters

[out] **deltaT** - Time step used for the integration in seconds.

Return values

0 No time-history analysis is done.

<val> Number of time-step responses are computed.

-1 General error.

C# Syntax

```
// Get the number of integration steps and time increment between the steps
double dT = 0.0;
int nSteps = OStd.Output.GetTimeHistoryIntegrationStepInfo(ref dT);
```

C++ Syntax

```
// Get the number of integration steps and time increment between the steps
double dT = 0.0;
int nSteps = OStd.Output.GetTimeHistoryIntegrationStepInfo(&dT);
```

VB Syntax

```
// Get the number of integration steps and time increment between the steps
Dim dT as Double
Dim nSteps as Integer
dT = 0.0
nSteps = 0
nSteps = objOpenStaad.Output.GetTimeHistoryIntegrationStepInfo(dT);
```

\

See also

OSOutputUI::GetTimeHistoryResponseAtTime

OSOutputUI::GetTimeHistoryResponse

OSOutputUI::GetTimeHistoryResponseMinMax

◆ GetTimeHistoryResponse()

```
VARIANT OSOutputUI::GetTimeHistoryResponse ( long          lCase,
                                             long          nodeNo,
                                             long          dofNo,
                                             long          responseType,
                                             VARIANT FAR * responses )
```

Returns the time-history responses of DOF at specified node in the VARIANT array responses.

Parameters

`[in]` **lCase** Load case number for future use. Use 0 at present.

`[in]` **nodeNo** Node number where the response is sought.

`[in]` **dofNo** Degrees of freedom define as DegreesOfFreedom enum.

Value	Degrees Of Freedom
Fx = 1	DegreesOfFreedom.Fx
Fy = 2	DegreesOfFreedom.Fy
Fz = 3	DegreesOfFreedom.Fz
Mx = 4	DegreesOfFreedom.Mx
My = 5	DegreesOfFreedom.My
Mz = 6	DegreesOfFreedom.Mz

`[in]` **responseType** Response type, i.e., displacement, velocity, or acceleration defined in TimeHistoryResponseType enum.

Value	Response Type
displacement = 0	TimeHistoryResponseType.dispResponse
velocity = 1	TimeHistoryResponseType.velResponse
acceleration = 2	TimeHistoryResponseType.acclResponse

`[out]` **responses** VARIANT double array returning the responses at the integration steps. The size of the array is (no of integration steps), the first location is for the response at time = 0.0.

Return values

0 No time-history analysis is done.

-1 General error.

1 Results successfuly returned.

C# Syntax

```
// Get get the Y displacement responses at node 7
int targetNode = 7;
double dT = 0.0;
int nSteps = OStd.Output.GetTimeHistoryIntegrationStepInfo(ref dT);
```

```
if(nSteps > 0)
{
 double[] response = new double[nSteps+1];
 Object objResponse = response as Object;
 nReturn = OStd.Output.GetTimeHistoryResponse(0, targetNode, (long)DegreesOfFreedom.Fy,
       (long)TimeHistoryResponseType.dispResponse, ref objResponse);
 response = objResponse as double[];
}
```

C++ Syntax

```
// Get get the Y displacement responses at node 7
double dT=0.0;
int targetNode = 7;
int nSteps = OStd.Output.GetTimeHistoryIntegrationStepInfo(&dT);
if(nSteps > 0)
{
 double atTime = 2.31;
 double response[] = new double[nSteps+1];
 Object objResponse = response as Object;
 int nReturn = OStd.GetTimeHistoryResponse(0, targetNode, (long)DegreesOfFreedom.Fy,
       (long)TimeHistoryResponseType.dispResponse, objResponse);
}
```

VB Syntax

```
// Get get the Y displacement responses at node 7
Dim dT as Double
Dim targetNode as Integer
Dim nSteps as Integer
dT = 0.0
targetNode = 7
nSteps = OStd.Output.GetTimeHistoryIntegrationStepInfo(dT)
if(nSteps > 0)
{
 Dim response() as Double
 Dim nReturn as Integer
 ReDim response(nSteps)
 nReturn = OStd.GetTimeHistoryResponse(0, targetNode, (long)DegreesOfFreedom.Fy,
       (long)TimeHistoryResponseType.dispResponse, response);
}
```

See also

OSOutputUI::GetTimeHistoryIntegrationStepInfo

OSOutputUI::GetTimeHistoryResponseAtTime

OSOutputUI::GetTimeHistoryResponseMinMax

◆ GetTimeHistoryResponseAtTime()

```
VARIANT OSOutputUI::GetTimeHistoryResponseAtTime ( long     lCase,
                                                   long     nodeNo,
                                                   long     dofNo,
                                                   long     responseType,
                                                   double   atTime,
                                                   double * response )
```

Returns the response at a spefictime within the integration time span at a specified node for a given DOF.

Parameters

`[in]` **lCase** Load case number for future use. Use 0 at present.

`[in]` **nodeNo** Node number where the response is sought.

`[in]` **dofNo** Degrees of freedom define as DegreesOfFreedom enum.

Value	Degrees Of Freedom
Fx = 1	DegreesOfFreedom.Fx
Fy = 2	DegreesOfFreedom.Fy
Fz = 3	DegreesOfFreedom.Fz
Mx = 4	DegreesOfFreedom.Mx
My = 5	DegreesOfFreedom.My
Mz = 6	DegreesOfFreedom.Mz

`[in]` **responseType** Response type, i.e., displacement, velocity, or acceleration defined in TimeHistoryResponseType enum.

Value	Response Type
displacement = 0	TimeHistoryResponseType.dispResponse
velocity = 1	TimeHistoryResponseType.velResponse
acceleration = 2	TimeHistoryResponseType.acclResponse

`[in]` **atTime** Time in seconds for which the respose is sought.

`[out]` **response** The response.

Return values

0 No time-history analysis is done.

-1 General error.

1 Results successfuly returned.

C# Syntax

```
// Get get the Y displacement responses at node 7
double dT=0.0;
int targetNode = 7;
```

```
int nSteps = OStd.Output.GetTimeHistoryIntegrationStepInfo(ref dT);
if(nSteps > 0)
{
 double atTime = 2.31;
 double response = 0.0;
 int nReturn = OStd.GetTimeHistoryResponseAtTime(0, targetNode,
       (long)DegreesOfFreedom.Fy, (long)TimeHistoryResponseType.dispResponse, atTime,
       ref response);
}
```

C++ Syntax

```
// Get get the Y displacement responses at node 7
double dT=0.0;
int targetNode = 7;
int nSteps = OStd.Output.GetTimeHistoryIntegrationStepInfo(&dT);
if(nSteps > 0)
{
 double atTime = 2.31;
 double response = 0.0;
 int nReturn = OStd.GetTimeHistoryResponseAtTime(0, targetNode,
       (long)DegreesOfFreedom.Fy, (long)TimeHistoryResponseType.dispResponse, atTime,
       &response);
}
```

VB Syntax

```
// Get get the Y displacement responses at node 7
Dim dT as Double
Dim targetNode as Integer
Dim nSteps as Integer
dT = 0.0
targetNode = 7
nSteps = OStd.Output.GetTimeHistoryIntegrationStepInfo(dT)
if(nSteps > 0)
{
 Dim atTime as Double
 Dim response as Double
 Dim nReturn as Integer
 atTime = 2.31;
 response = 0.0;
 nReturn = OStd.GetTimeHistoryResponseAtTime(0, targetNode, (long)DegreesOfFreedom.Fy,
       (long)TimeHistoryResponseType.dispResponse, atTime, response);
}
```

See also

OSOutputUI::GetTimeHistoryIntegrationStepInfo

OSOutputUI::GetTimeHistoryResponse

OSOutputUI::GetTimeHistoryResponseMinMax

◆ GetTimeHistoryResponseMinMax()

```
VARIANT OSOutputUI::GetTimeHistoryResponseMinMax ( long     ICase,
                                                   long     nodeNo,
                                                   long     dofNo,
                                                   long     responseType,
                                                   double * responseMax,
                                                   double * timeMax,
                                                   double * responseMin,
                                                   double * timeMin )
```

Returns the min/max time-history responses of DOF at specified node.

Parameters

`[in]` **ICase** Use 0.

`[in]` **nodeNo** Node number where the response is sought.

`[in]` **dofNo** Degrees of freedom.

Degrees Of Freedom	Value
Fx	1
Fy	2
Fz	3

`[in]` **responseType** Response type, i.e., displacement, velocity, or acceleration.

Response Type	Value
displacement	0
velocity	1
acceleration	2

`[out]` **responseMax** Maximum response.

`[out]` **timeMax** The time when the maximum response occurs.

`[out]` **responseMin** Minimum response.

`[out]` **timeMin** The time when the minimum response occurs.

Return values

0 No time-history analysis is done.

-1 General error.

-2001 Node not found

-8002 Load Case **ICase** not found.

-9915 Result is not found or could not be loaded.

C# Syntax

```
// Get get the Y displacement responses at node 7
long targetNode = 7;
double dT = 0.0;
long DegreesOfFreedom = 2;
long TimeHistoryResponseType = 0;
int nSteps = OStd.Output.GetTimeHistoryIntegrationStepInfo(ref dT);
if(nSteps > 0)
{
 double responseMax = 0.0;
 double responseMin = 0.0;
 double timeMax = 0.0;
 double timeMin = 0.0;
 long nReturn = OStd.Output.GetTimeHistoryResponseMinMax(0, targetNode, DegreesOfFreedom,
       TimeHistoryResponseType,
                                                         ref responseMax, ref timeMax, ref
       responseMin, ref timeMin);
}
```

C++ Syntax

```
// Get get the Y displacement responses at node 7
double dT=0.0;
long targetNode = 7;
long DegreesOfFreedom = 2;
long TimeHistoryResponseType = 0;
int nSteps = OStd.Output.GetTimeHistoryIntegrationStepInfo(&dT);
if(nSteps > 0)
{
 double responseMax = 0.0;
 double responseMin = 0.0;
 double timeMax = 0.0;
 double timeMin = 0.0;
 long nReturn = OStd.Output.GetTimeHistoryResponseMinMax(0, targetNode, DegreesOfFreedom,
       TimeHistoryResponseType,
                                                         &responseMax, &timeMax, &responseMin,
       &timeMin);
}
```

VB 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


' Get the Y displacement responses at node 2
     Dim dT As Double
     Dim targetNode As Long
     Dim nSteps As Long
```

```
    Dim nReturn As Long
    Dim MessageText As String

    dT = 0.0
    targetNode = 2
    nSteps = objOpenStaad.Output.GetTimeHistoryIntegrationStepInfo(dT)

    If(nSteps > 0) Then

        Dim responseMax As Double
        Dim responseMin As Double
        Dim timeMax As Double
        Dim timeMin As Double
        responseMax = 0.0
        responseMin = 0.0
        timeMax = 0.0
        timeMin = 0.0

        Dim DegreesOfFreedom As Long
        Dim TimeHistoryResponseType As Long
        'Fx = 1
        DegreesOfFreedom = 1
        'acceleration = 2
        TimeHistoryResponseType = 2

        nReturn = objOpenStaad.Output.GetTimeHistoryResponseMinMax(0, targetNode,
      DegreesOfFreedom, TimeHistoryResponseType, responseMax, timeMax, responseMin,
      timeMin)

        MessageText = "DegreesOfFreedom," &Str(DegreesOfFreedom) & Chr(10)
        MessageText = MessageText & "TimeHistoryResponseType, " &
      Str(TimeHistoryResponseType) & Chr(10)
        MessageText = MessageText & "ResponseMax, " & Str(responseMax) & Chr(10)
        MessageText = MessageText & "TimeMax, " &Str(timeMax) & Chr(10)
        MessageText = MessageText & "ResponseMin, " &Str(responseMin) & Chr(10)
        MessageText = MessageText & "TimeMin, " & Str(timeMin)

        MsgBox(MessageText,vbOkOnly,"Node:" & Str(targetNode))

    End If


    MsgBox"Macro Ending"
    Set objOpenStaad = Nothing
End Sub
```

See also

- **OSOutputUI::GetTimeHistoryIntegrationStepInfo**
- **OSOutputUI::GetTimeHistoryResponse**
- **OSOutputUI::GetTimeHistoryResponseAtTime**